Real–Time
Model Checking

Patricia Bouyer–Decitre
Kim G. Larsen
Nicolas Markey
Timed Automata
.. and Prices and Games

Patricia Bouyer-Decitre
Kim G. Larsen
Nicolas Markey
Model Checking

System Description

Requirement

A\(\square ( \text{req} \Rightarrow \text{A}\diamond ) \)

A\(\square ( \text{req} \Rightarrow \text{A}\diamond_{t<30s} \text{grant} ) \)

A\(\square ( \text{req} \Rightarrow \text{A}\diamond_{t<30s,c<5} \text{grant} ) \)

A\(\square ( \text{req} \Rightarrow \text{A}\diamond_{t<30s,p>0.90} \text{grant} ) \)
Synthesis

System Description

Debugging Information

Time Cost Probability

? TOOL

Yes

Debugging Information

No!

Yes

Control Strategy

Requirement

A\square(\text{req} \Rightarrow A\lozenge_{t<30s} \text{grant})

A\square(\text{req} \Rightarrow A\lozenge_{t<30s, c<5} \text{grant})

A\square(\text{req} \Rightarrow A\lozenge_{t<30s, p>0.90} \text{grant})

QMC, PhD School, March 3, 2010

Kim Larsen [4]
Overview

- Introduction to Timed Automata
- Decidability and undecidability results
- Timed Temporal Logics
- UPPAAL .. (hands-on)
- Timed Games
- Priced Timed Automata
- Open Problems
Timed Automata
UPPAAL (contributors)

@UPPsala
- Wang Yi
- Paul Pettersson
- John Håkansson
- Anders Hessel
- Pavel Krcal
- Leonid Mokrushin
- Shi Xiaochun

@AALborg
- Kim G Larsen
- Gerd Behrman
- Arne Skou
- Brian Nielsen
- Alexandre David
- Jacob I. Rasmussen
- Marius Mikucionis
- Thomas Chatain

@Elsewhere
Real Time Systems

- Plant: Continuous
- Controller Program: Discrete

Eg.: Realtime Protocols
- Pump Control
- Air Bags
- Robots
- Cruise Control
- ABS
- CD Players
- Production Lines

Real Time System
A system where correctness not only depends on the logical order of events but also on their timing!!

QMC, PhD School, March 3, 2010
Kim Larsen [8]
A Dumb Light Controller

QMC, PhD School, March 3, 2010

Kim Larsen [9]
Timed Automata

[Alur & Dill’89]

Synchronizing action

Reset

Clock Guard

Conjunctions of $n$

ADD a clock $x$

$x$: real-valued clock

press?  $x=0$

press?  $x \leq 3$

press?  $x > 3$

Off

Light

Bright

QMC, PhD School, March 3, 2010

Kim Larsen [10]
A Timed Automata (Semantics)

States: 
( location, x=v) where v ∈ R

Transitions: 
( Off, x=0 )
delay 4.32 → ( Off, x=4.32 )
press? → ( Light, x=0 )
delay 2.51 → ( Light, x=2.51 )
press? → ( Bright, x=2.51 )
Intelligent Light Controller

Invariant (Henzinger)

QMC, PhD School, March 3, 2010 Kim Larsen [12]
Intelligent Light Controller

Transitions:

( Off , x=0 )
delay 4.32  ( Off , x=4.32 )
press?  ( Light , x=0 )
delay 4.51  ( Light , x=4.51 )
press?  ( Light , x=0 )
delay 100  ( Light , x=100 )
τ  ( Off , x=0 )

Note:
( Light , x=0 ) delay 103  ( Light , x=100 )

Invariants
ensures QMC, PhD School, March 3, 2010 Kim Larsen [13]
**Constraints**

**Definition**
Let $X$ be a set of clock variables. The set $B(X)$ of *clock constraints* $\phi$ is given by the grammar:

$$
\phi ::= x \leq c \mid c \leq x \mid x < c \mid c < x \mid \phi_1 \land \phi_2
$$

where $c \in \mathbb{N}$ (or $\mathbb{Q}$).
Timed Automata (formally)

Clock Valuations and Notation

**Definition**
The set of clock valuations, $\mathbb{R}^C$ is the set of functions $C \rightarrow \mathbb{R}_{\geq 0}$ ranged over by $u, v, w, \ldots$.

**Notation**
Let $u \in \mathbb{R}^C$, $r \subseteq C$, $d \in \mathbb{R}_{\geq 0}$, and $g \in \mathcal{B}(X)$ then:

- $u + d \in \mathbb{R}^C$ is defined by $(u + d)(x) = u(x) + d$ for any clock $x$.
- $u[r] \in \mathbb{R}^C$ is defined by $u[r](x) = 0$ when $x \in r$ and $u[r](x) = u(x)$ for $x \not\in r$.
- $u \models g$ denotes that $g$ is satisfied by $u$. 
Timed Automata (formally)

**Definition**
A timed automaton $A$ over clocks $C$ and actions $Act$ is a tuple $(L, l_0, E, I)$, where:

- $L$ is a finite set of locations
- $l_0 \in L$ is the initial location
- $E \subseteq L \times B(X) \times Act \times P(C) \times L$ is the set of edges
- $I : L \rightarrow B(X)$ assigns to each location an invariant
Timed Automata (formally)

Semantics

Definition
The semantics of a timed automaton $A$ is a labelled transition system with state space $L \times \mathbb{R}^C$ with initial state $(l_0, u_0)^*$ and with the following transitions:

- $(l, u) \xrightarrow{\epsilon(d)} (l, u + d)$ iff $u \in I(l)$ and $u + d \in I(l)$,
- $(l, u) \xrightarrow{a} (l', u')$ iff there exists $(l, g, a, r, l') \in E$ such that
  - $u \models g$,
  - $u' = u[r]$, and
  - $u' \in I(l')$

*$u_0(x) = 0$ for all $x \in C$
Example

QMC, PhD School, March 3, 2010

Kim Larsen [18]
Example

\( y := 0 \)

\( y \leq 2 \)

\( x := 0 \)

\( x \leq 2 \)

\( y \leq 2, x = 4 \)

\( \ell_0, x = 0, y = 0 \)
Example

\[
\begin{align*}
y &= 0 \\
y &\leq 2 \\
y &\leq 2, \ x = 4 \\
x &= 0 \\
x &\leq 2 \\
\end{align*}
\]

\[
\begin{align*}
(\ell_0, x = 0, y = 0) \\
\rightarrow (\ell_0, x = 1.4, y = 1.4)
\end{align*}
\]
Example

\[(\ell_0, x = 0, y = 0)\]
\[\xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4)\]
\[\xrightarrow{a} (\ell_0, x = 1.4, y = 0)\]
Example

QMC, PhD School, March 3, 2010

\[(\ell_0, x = 0, y = 0) \xrightarrow{1.4} (\ell_0, x = 1.4, y = 1.4) \]
\[(\ell_0, x = 1.4, y = 0) \xrightarrow{a} (\ell_0, x = 1.4, y = 1.4) \xrightarrow{1.6} (\ell_0, x = 3.0, y = 1.6) \]
\[(\ell_0, x = 3.0, y = 0) \xrightarrow{a} (\ell_0, x = 3.0, y = 0) \]
Light Control Interface
Light Control Interface

- press? d release? → touch! 0.5 ≤ d ≤ 1
- press? 1 → starthold!
- press? d release? → endhold!  d > 1

User

Interface

Light

Control Program

QMC, PhD School, March 3, 2010
Light Control Interface

User

QMC, PhD School, March 3, 2010
Light Control Network

![Diagram of Light Control Network]

QMC, PhD School, March 3, 2010
Task Graph Scheduling
Resources & Tasks & Composition

Semantics:

\[( \text{Idle}, \text{Init}, B=0, x=0) \]
\[d(3.1415) \rightarrow ( \text{Idle}, \text{Init}, B=0, x=3.1415)\]
\[\text{use} \rightarrow ( \text{InUse}, \text{Using}, B=6, x=0)\]
\[d(6) \rightarrow ( \text{InUse}, \text{Using}, B=6, x=6)\]
\[\text{done} \rightarrow ( \text{Idle}, \text{Done}, B=6, x=6)\]
Task Graph Scheduling – Example

Compute:
\[(D \times (C \times (A + B)) + ((A + B) + (C \times D))\]

using 2 processors

P1 (fast)

P2 (slow)

13 pico-sec !!
Compute:

\[(D \times (C \times (A + B)) + ((A + B) + (C \times D)))\]

using 2 processors

**P1** (fast)

\[
\begin{align*}
+ & 2ps \\
\ast & 3ps
\end{align*}
\]

**P2** (slow)

\[
\begin{align*}
+ & 5ps \\
\ast & 7ps
\end{align*}
\]

**OPTIMAL!!**

12 pico-sec
Task Graph Scheduling

\[ M = \{M_1, M_2\} \]
Task Graph Scheduling

E<> (Task1.End and ... and Task7.End)

QMC, PhD School, March 3, 2010

Kim Guldstrand Larsen [32]
### Experimental Results

<table>
<thead>
<tr>
<th>Name</th>
<th>#Tasks</th>
<th>#Chains</th>
<th>#Machines</th>
<th>Optimal</th>
<th>TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>437</td>
<td>125</td>
<td>4</td>
<td>1178</td>
<td>1182</td>
</tr>
<tr>
<td>000</td>
<td>452</td>
<td>43</td>
<td>20</td>
<td>537</td>
<td>537</td>
</tr>
<tr>
<td>018</td>
<td>730</td>
<td>175</td>
<td>10</td>
<td>700</td>
<td>704</td>
</tr>
<tr>
<td>074</td>
<td>1007</td>
<td>66</td>
<td>12</td>
<td>891</td>
<td>894</td>
</tr>
<tr>
<td>021</td>
<td>1145</td>
<td>88</td>
<td>20</td>
<td>605</td>
<td>612</td>
</tr>
<tr>
<td>228</td>
<td>1187</td>
<td>293</td>
<td>8</td>
<td>1570</td>
<td>1574</td>
</tr>
<tr>
<td>071</td>
<td>1193</td>
<td>124</td>
<td>20</td>
<td>629</td>
<td>634</td>
</tr>
<tr>
<td>271</td>
<td>1348</td>
<td>127</td>
<td>12</td>
<td>1163</td>
<td>1164</td>
</tr>
<tr>
<td>237</td>
<td>1566</td>
<td>152</td>
<td>12</td>
<td>1340</td>
<td>1342</td>
</tr>
<tr>
<td>231</td>
<td>1664</td>
<td>101</td>
<td>16</td>
<td>t.o.</td>
<td>1137</td>
</tr>
<tr>
<td>235</td>
<td>1782</td>
<td>218</td>
<td>16</td>
<td>t.o.</td>
<td>1150</td>
</tr>
<tr>
<td>233</td>
<td>1980</td>
<td>207</td>
<td>19</td>
<td>1118</td>
<td>1121</td>
</tr>
<tr>
<td>294</td>
<td>2014</td>
<td>141</td>
<td>17</td>
<td>1257</td>
<td>1261</td>
</tr>
<tr>
<td>295</td>
<td>2168</td>
<td>965</td>
<td>18</td>
<td>1318</td>
<td>1322</td>
</tr>
<tr>
<td>292</td>
<td>2333</td>
<td>318</td>
<td>3</td>
<td>8009</td>
<td>8009</td>
</tr>
<tr>
<td>298</td>
<td>2399</td>
<td>303</td>
<td>10</td>
<td>2471</td>
<td>2473</td>
</tr>
</tbody>
</table>

Abdeddaïm, Kerbaa, Maler
Brick Sorting
LEGO Mindstorms/RCX

- **Sensors:** temperature, light, rotation, pressure.
- **Actuators:** motors, lamps,
- **Virtual machine:**
  - 10 tasks, 4 timers, 16 integers.
- **Several Programming Languages:**
  - NotQuiteC, Mindstorm, Robotics, legOS, etc.
A Real Real Timed System

The Plant
Conveyor Belt
&
Bricks

Controller
Program
LEGO MINDSTORM

QMC, PhD School, March 3, 2010
First UPPAAL model

Sorting of Lego Boxes

Exercise: Design Controller so that black boxes are being pushed out
int active;
int DELAY;
int LIGHT_LEVEL;

task MAIN{
    DELAY=75;
    LIGHT_LEVEL=35;
    active=0;
    Sensor(IN_1, IN_LIGHT);
    Fwd(OUT_A,1);
    Display(1);
    start PUSH;
    while(true){
        wait(IN_1<=LIGHT_LEVEL);
        ClearTimer(1);
        active=1;
        PlaySound(1);
        wait(IN_1>LIGHT_LEVEL);
    }
}

task PUSH{
    while(true){
        wait(Timer(1)>DELAY && active==1);
        active=0;
        Rev(OUT_C,1);
        Sleep(8);
        Fwd(OUT_C,1);
        Sleep(12);
        Off(OUT_C);
    }
}
A Black Brick
Control Tasks & Piston

GLOBAL DECLARATIONS:
const int ctime = 75;

int[0,1] active;
clock x, time;

chan eject, ok;
urgent chan blk, red, remove, go;
Case Studies: Controllers

- Gearbox Controller [TACAS'98]
- Bang & Olufsen Power Controller [RTPS'99, FTRTFT'2k]
- SIDMAR Steel Production Plant [RTCSA'99, DSVV'2k]
- Real-Time RCX Control-Programs [ECRTS'2k]
- Terma, Verification of Memory Management for Radar (2001)
- Scheduling Lacquer Production (2005)
- Memory Arbiter Synthesis and Verification for a Radar Memory Interface Card [NJC'05]

- Adapting the UPPAAL Model of a Distributed Lift System, 2007
- Analyzing a χ model of a turntable system using Spin, CADP and Uppaal, 2006
- **Designing, Modelling and Verifying a Container Terminal System Using UPPAAL, 2008**
- Model-based system analysis using Chi and Uppaal: An industrial case study, 2008
- Climate Controller for Pig Stables, 2008
- Optimal and Robust Controller for Hydraulic Pump, 2009

QMC, PhD School, March 3, 2010

Kim Larsen [41]
Case Studies: Protocols

- Philips Audio Protocol [HS’95, CAV’95, RTSS’95, CAV’96]
- Bounded Retransmission Protocol [TACAS’97]
- Bang & Olufsen Audio/Video Protocol [RTSS’97]
- TDMA Protocol [PRFTS’97]
- Lip-Synchronization Protocol [FMICS’97]
- ATM ABR Protocol [CAV’99]
- ABB Fieldbus Protocol [ECRTS’2k]
- Distributed Agreement Protocol [Formats05]
- Leader Election for Mobile Ad Hoc Networks [Charme05]

- Analysis of a protocol for dynamic configuration of IPv4 link local addresses using Uppaal, 2006
- Formalizing SHIM6, a Proposed Internet Standard in UPPAAL, 2007
- Verifying the distributed real-time network protocol RTnet using Uppaal, 2007
- Analysis of the Zeroconf protocol using UPPAAL, 2009

QMC, PhD School, March 3, 2010

Kim Larsen [42]
Using UPPAAL as Back–end

- Vooduu: verification of object–oriented designs using Uppaal, 2004
- Formalising the ARTS MPSOC Model in UPPAAL, 2007
- Timed automata translator for Uppaal to PVS
- Component–Based Design and Analysis of Embedded Systems with UPPAAL PORT, 2008
- Verification of COMDES–II Systems Using UPPAAL with Model Transformation, 2008
Timed automata – Decidability issues

Patricia Bouyer-Decitre

LSV, CNRS & ENS Cachan, France
An example of a timed automaton

This run reads the timed word
((problem, 23)(delayed, 38.6)(repair, 40.9), (done, 63).
An example of a timed automaton

This run reads the timed word \((\text{problem, } x := 0)(\text{delayed, } 38.6)(\text{repair, } 40.9), (\text{done, } y \in [22, 25])\).
An example of a timed automaton

This run reads the timed word
(problem, 23)(delayed, 38.6)(repair, 40.9), (done, 63).
An example of a timed automaton:

- **Safe**
  - **Problem, x:=0**
  - **Repair, y:=0**
  - **Delayed, y:=0**
- **Alarm**
- **Repairing**
- **Failsafe**

---

This run reads the timed word:

```
(problem, 23)
(delayed, 38.6)
(repair, 40.9)
(done, 63).
```

---

<table>
<thead>
<tr>
<th>State</th>
<th>X</th>
<th>Y</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe</td>
<td>0</td>
<td>23</td>
<td>Safe</td>
</tr>
<tr>
<td>Safe</td>
<td>23</td>
<td>0</td>
<td>Problem</td>
</tr>
<tr>
<td>Safe</td>
<td>0</td>
<td>23</td>
<td>Repair</td>
</tr>
<tr>
<td>Repair</td>
<td>23</td>
<td>0</td>
<td>Failsafe</td>
</tr>
<tr>
<td>Repair</td>
<td>0</td>
<td>23</td>
<td>Done</td>
</tr>
</tbody>
</table>

---

The run evolves through the following states and actions:

1. **Safe** to **Alarm**
2. **Alarm** to **Repairing**
3. **Repairing** to **Failsafe**
4. **Failsafe** to **Safe**

---

The transitions are labeled with conditions and time delays, ensuring the system's safety and functionality.
An example of a timed automaton

---

This run reads the timed word

\[(\text{problem}, 23)(\text{delayed}, 38.6)(\text{repair}, 40.9), (\text{done}, 63)\].

---

\[x := 0, x \leq 15\]
\[y := 0, 15 \leq x \leq 16\]
\[2 \leq y \land x \leq 56\]
An example of a timed automaton

This run reads the timed word
(problem, 23)(delayed, 38.6)(repair, 40.9), (done, 63).

| safe  | 23 | safe  | problem | alarm  | 15.6 | alarm  | delayed | failsafe | 15.6  | ...
|-------|----|-------|---------|-------|------|-------|---------|---------|------|-----
| x     | 0  | 23    | 0       | 15.6  | 15.6 |       |         |         |      |     
| y     | 0  | 23    | 23      | 38.6  | 0    |       |         |         |      |     

failsafe

... 15.6

0
An example of a timed automaton

This run reads the timed word

(problem, 23)(delayed, 38.6)(repair, 40.9), (done, 63).
An example of a timed automaton

This run reads the timed word
(problem, 23)(delayed, 38.6)(repair, 40.9), (done, 63).
An example of a timed automaton

This run reads the timed word (problem, 23)(delayed, 38.6)(repair, 40.9), (done, 63).
An example of a timed automaton

This run reads the timed word
(problem, 23)(delayed, 38.6)(repair, 40.9), (done, 63).

X  0  23  0  15.6  15.6  ...
y  0  23  23  38.6  0

... 15.6 17.9 17.9 40 40
... 0 2.3 0 22.1 22.1

safe  23  safe  problem  alarm  15.6  alarm  delayed  failsafe

done, 22 \leq y \leq 25
repair, x \leq 15
15 \leq x \leq 16
delayed, y = 0
An example of a timed automaton

This run reads the timed word
(problem, 23)(delayed, 38.6)(repair, 40.9), (done, 63).
Outline

1. Decidability of basic properties

2. Equivalence (or preorder) checking

3. Some extensions of timed automata
Verification

Emptiness problem

Is the language accepted by a timed automaton empty?

- basic reachability/safety properties (final states)
- basic liveness properties ($\omega$-regular conditions)

Theorem [AD90,AD94]
The emptiness problem for timed automata is decidable and PSPACE-complete.
Method: construct a finite abstraction
Verification

Emptiness problem

Is the language accepted by a timed automaton empty?

- **Problem:** the set of configurations is infinite
  
  \[\Rightarrow\] classical methods for finite-state systems cannot be applied
Verification

Emptiness problem

Is the language accepted by a timed automaton empty?

- **Problem:** the set of configurations is infinite
  \[\leadsto\] classical methods for finite-state systems cannot be applied

- **Positive key point:** variables (clocks) increase at the same speed

Theorem \[AD90,AD94\]
The emptiness problem for timed automata is decidable and PSPACE-complete.
Method: construct a finite abstraction
Verification

Emptiness problem
Is the language accepted by a timed automaton empty?

- **Problem:** the set of configurations is infinite
  $\leadsto$ classical methods for finite-state systems cannot be applied

- **Positive key point:** variables (clocks) increase at the same speed

**Theorem** [AD90, AD94]
The emptiness problem for timed automata is decidable and PSPACE-complete.

[AD90] Alur, Dill. Automata for modeling real-time systems (ICALP’90).
Verification

Emptiness problem

Is the language accepted by a timed automaton empty?

- **Problem:** the set of configurations is infinite
  $\Rightarrow$ classical methods for finite-state systems cannot be applied

- **Positive key point:** variables (clocks) increase at the same speed

**Theorem** [AD90,AD94]

The emptiness problem for timed automata is decidable and PSPACE-complete.

**Method:** construct a finite abstraction

[AD90] Alur, Dill. Automata for modeling real-time systems (ICALP’90).

The region abstraction

clock $y$

clock $x$

only constraints: $x \sim c$ with $c \in \{0, 1, 2\}$

$y \sim c$ with $c \in \{0, 1, 2\}$

The path $x = 1$ $y = 1$ can be fired from

"compatibility" between regions and constraints

$\Rightarrow$ an equivalence of finite index

a time-abstract bisimulation
The region abstraction

only constraints: $x \sim c$ with $c \in \{0, 1, 2\}$
$y \sim c$ with $c \in \{0, 1, 2\}$

"compatibility" between regions and constraints
The region abstraction

- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

The path $x=1 \rightarrow y=1$
- can be fired from
- cannot be fired from
The region abstraction

- “compatibility” between regions and constraints
- “compatibility” between regions and time elapsing
The region abstraction

“compatibility” between regions and constraints
“compatibility” between regions and time elapsing
\[\rightsquigarrow\text{ an equivalence of finite index}\]
The region abstraction

```
<table>
<thead>
<tr>
<th>clock x</th>
<th>clock y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
```

- “compatibility” between regions and constraints
- “compatibility” between regions and time elapsing

\[ \leadsto \text{an equivalence of finite index} \]
\[ \text{a time-abstract bisimulation} \]
Time-abstract bisimulation

This is a relation between ● and ○ such that:

∀a ∃u1D6FF(d) ∀d > 0 ∃d′ > 0 /u1D6FF(d′) ... and vice-versa (swap ● and ○).
Time-abstract bisimulation

This is a relation between \( \bullet \) and \( \circ \) such that:

\[
\forall a \exists a \forall d > 0 \exists d' > 0
\]

This is a relation between \( \bullet \) and \( \circ \) such that:

\[
\forall \exists a \forall d > 0 \exists d' > 0
\]... and vice-versa (swap \( \bullet \) and \( \circ \)).
Time-abstract bisimulation

This is a relation between $\bullet$ and $\bullet$ such that:

$\forall \exists a \in\mathbb{R} \quad \exists d > 0 \quad \forall d' > 0 \quad a \in\mathbb{R} \quad \exists d'' > 0 \quad \forall d''' > 0 \quad a \in\mathbb{R}$

... and vice-versa (swap $\bullet$ and $\bullet$).
Decidability of basic properties

Time-abstract bisimulation

This is a relation between \( \bullet \) and \( \bullet \) such that:

\[
\forall a \exists a /u1D6FF(d) \forall d > 0 \exists d' > 0 /u1D6FF(d') \ldots \text{and vice-versa (swap } \bullet \text{ and } \bullet).\]
Time-abstract bisimulation

This is a relation between ● and ● such that:

\[ \forall \exists \frac{a}{u1D6FF(d)} \quad \forall d > 0 \exists d' > 0 \frac{\delta(d')}{u1D6FF(d')} \]

... and vice-versa (swap ● and ●).

6/22
Time-abstract bisimulation

This is a relation between \( \bullet \) and \( \bullet \) such that:

\[
\forall \quad \exists \quad a
\]

\[
\forall d > 0 \quad \exists d' > 0
\]

... and vice-versa (swap \( \bullet \) and \( \bullet \)).
The region abstraction (2)

- region $R$ defined by:
  
  \[
  \begin{aligned}
  0 < x < 1 \\
  0 < y < 1 \\
  y < x
  \end{aligned}
  \]
The region abstraction (2)

- region $R$ defined by:
  \[
  \begin{cases}
  0 < x < 1 \\
  0 < y < 1 \\
  y < x
  \end{cases}
  \]

- time successors of $R$

Decidability of basic properties
The region abstraction (2)

- region $R$ defined by:
  $$\begin{align*}
  0 &< x < 1 \\
  0 &< y < 1 \\
  y &< x
  \end{align*}$$

- time successors of $R$

image of $R$ when resetting clock $x$
The construction of the region graph

It “mimicks” the behaviours of the clocks.
Region automaton $\equiv$ finite bisimulation quotient

Timed automaton

Region graph
Region automaton $\equiv$ finite bisimulation quotient

timed automaton

region graph

region automaton

$\ell_1 \quad y < 1, a, x := 0 \quad \ell_2$
Region automaton ≡ finite bisimulation quotient

Region automaton

\[ \mathcal{L}(\text{reg. aut.}) = \text{UNTIME}(\mathcal{L}(\text{timed aut.})) \]
An example [AD94]

Decidability of basic properties

s0 \rightarrow x>0,a, y:=0

s1 \rightarrow y=1,b

s2 \rightarrow x<1,c

s3 \rightarrow x<1,c

\rightarrow x>1,d

y<1,a,y:=0
Decidability of basic properties

An example [AD94]
An example [AD94]
Decidability of basic properties

Timed automaton

Finite bisimulation quotient

Large (but finite) automaton (region automaton)

- Reachability/safety properties
- Liveness properties (like Büchi properties)
- \(\Rightarrow\) problems with Zeno behaviours? (infinitely many actions in bounded time)

NB: standard problem in timed automata...

11/22
Decidability of basic properties

timed automaton
finite bisimulation
quotient
large (but finite) automaton (region automaton)

- **large**: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is:

\[
\prod_{x \in X} (2M_x + 2) \cdot |X|! \cdot 2^{|X|}
\]

It can be used to check for:
- reachability/safety properties
- liveness properties (like Büchi properties)

\[\Rightarrow\] problems with Zeno behaviours? (infinitely many actions in bounded time)

NB: standard problem in timed automata...
large: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is:

\[
\prod_{x \in X} (2M_x + 2) \cdot |X|! \cdot 2^{|X|}
\]

It can be used to check for:

- reachability/safety properties
- liveness properties (like Büchi properties)
- \(\Rightarrow\) problems with Zeno behaviours?
  (infinitely many actions in bounded time)

NB: standard problem in timed automata...
Decidability of basic properties

- **large**: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is:

  \[ \prod_{x \in X} (2M_x + 2) \cdot |X|! \cdot 2^{|X|} \]

- It can be used to check for:
  - reachability/safety properties

NB: standard problem in timed automata...
Decidability of basic properties

- **large**: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is:

  \[
  \prod_{x \in X} \left(2M_x + 2\right) \cdot |X|! \cdot 2^{|X|}
  \]

- It can be used to check for:
  - reachability/safety properties
  - liveness properties (like Büchi properties)
Decidability of basic properties

- **large**: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is:
\[
\prod_{x \in X} (2M_x + 2) \cdot |X|! \cdot 2^{|X|}
\]

- It can be used to check for:
  - reachability/safety properties
  - liveness properties (like Büchi properties)
  - problems with Zeno behaviours?
    (infinitely many actions in bounded time)
Decidability of basic properties

- **large**: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is:

\[
\prod_{x \in X} (2M_x + 2) \cdot |X|! \cdot 2^{|X|}
\]

- It can be used to check for:
  - reachability/safety properties
  - liveness properties (like Büchi properties)

  \[\leadsto\] problems with Zeno behaviours?

  (infinitely many actions in bounded time)

NB: standard problem in timed automata...
Decidability of basic properties

Back to the example

\[ s_0 \xrightarrow{x>0,a} s_1 \]
\[ s_1 \xrightarrow{y=1,b} s_2 \]
\[ s_1 \xrightarrow{x<1,c} s_3 \]
\[ s_3 \xrightarrow{x>1,d} s_1 \]

\[ s_0 \xrightarrow{y:=0} s_1 \]
\[ s_1 \xrightarrow{y<1,a,y:=0} s_3 \]
\[ s_2 \xrightarrow{x<1,c} s_3 \]

The states and transitions are as follows:
- \( s_0 \) with input \( x>0,a \) and self-loop on \( y:=0 \)
- \( s_1 \) with transitions \( y=1,b \) to \( s_2 \), \( x<1,c \) to \( s_3 \), and \( y<1,a,y:=0 \) to itself
- \( s_2 \) with transition \( x<1,c \) to \( s_3 \)
- \( s_3 \) with transition \( x>1,d \) to \( s_1 \)
Back to the example

Decidability of basic properties
Decidability of basic properties

Back to the example

\[ s_0 \xrightarrow{x>0,a} s_1 \xrightarrow{y:=0} s_3 \xrightarrow{x>1,d} \]

\[ s_0 \xleftarrow{s_0} \]

\[ s_1 \]

\[ s_1 \xrightarrow{y=0,x=1} s_1 \xrightarrow{y=0,x>1} s_2 \]

\[ s_2 \xleftarrow{y=1,b} \]

\[ s_3 \]

\[ s_3 \xrightarrow{0<y<x<1} s_3 \xrightarrow{0<y<1<x} s_3 \xrightarrow{1=y<x} s_3 \xrightarrow{x>1,y>1} \]
Back to the example

Decidability of basic properties

Zeno cycles
Back to the example

Decidability of basic properties

Cycles with non-Zeno behaviours
Outline

1. Decidability of basic properties

2. Equivalence (or preorder) checking

3. Some extensions of timed automata
Strong timed (bi)simulation

This is a relation between $\cdot$ and $\cdot$ such that:

\[
\forall a \exists \exists a/u_1D6FF (d)\quad \forall d > 0 \exists \exists /u_1D6FF (d)
\]

... and vice-versa (swap $\cdot$ and $\cdot$) for the bisimulation relation.

Theorem

Strong timed (bi)simulation between timed automata is decidable and EXPTIME-complete. (see later for a simple proof of the upper bound)
Equivalence (or preorder) checking

Strong timed (bi)simulation

This is a relation between $\bullet$ and $\bullet$ such that:

\[ \forall d > 0 \exists \text{ ... and vice-versa (swap $\bullet$ and $\bullet$) for the bisimulation relation.} \]

Theorem
Strong timed (bi)simulation between timed automata is decidable and EXPTIME-complete.
(see later for a simple proof of the upper bound)

14/22
Strong timed (bi)simulation

This is a relation between $\bullet$ and $\bullet$ such that:

\[ \forall \exists \quad a \leq 0 \exists \forall \quad a \geq 0 \]

... and vice-versa (swap $\bullet$ and $\bullet$) for the bisimulation relation.

Theorem

Strong timed (bi)simulation between timed automata is decidable and EXPTIME-complete.

(see later for a simple proof of the upper bound)
Strong timed (bi)simulation

This is a relation between $\bullet$ and $\bullet$ such that:

\[
\forall a \exists \delta(d) \quad \forall d > 0
\]

Theorem: Strong timed (bi)simulation between timed automata is decidable and \text{EXPTIME}-complete. (see later for a simple proof of the upper bound)
Strong timed (bi)simulation

This is a relation between $\bullet$ and $\bullet$ such that:

$$\forall a \exists (d) \quad \forall d > 0 \exists \delta(d)$$

and vice-versa (swap $\bullet$ and $\bullet$), for the bisimulation relation.

Theorem

Strong timed (bi)simulation between timed automata is decidable and EXPTIME-complete.

(see later for a simple proof of the upper bound)
Strong timed (bi)simulation

This is a relation between ∙ and ∙ such that:

∀ ∙ → ∙

∃ ∙ → ∙

∀d > 0

∃ ∙ → ∙

δ(d)

... and vice-versa (swap ∙ and ∙) for the bisimulation relation.
Equivalence (or preorder) checking

Strong timed (bi)simulation

This is a relation between \( \bullet \) and \( \bullet \) such that:

\[
\forall \quad \exists \quad a
\]

\[
\forall d > 0 \quad \exists \quad \delta(d)
\]

... and vice-versa (swap \( \bullet \) and \( \bullet \)) for the bisimulation relation.

Theorem

Strong timed (bi)simulation between timed automata is decidable and EXPTIME-complete.

(see later for a simple proof of the upper bound)
Language (or trace) equivalence and inclusion

Question

Given two timed automata $A$ and $B$, is $L(A) = L(B)$ (resp. $L(A) \subseteq L(B)$)?

Theorem

Language equivalence and language inclusion are undecidable in timed automata.
Language (or trace) equivalence and inclusion

**Question**

Given two timed automata $\mathcal{A}$ and $\mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$ (resp. $L(\mathcal{A}) \subseteq L(\mathcal{B})$)?

**Theorem** [AD90,AD94]

Language equivalence and language inclusion are undecidable in timed automata.

... as a special case of the universality problem (are all timed words accepted by the automaton?).

Language (or trace) equivalence and inclusion

Question
Given two timed automata $\mathcal{A}$ and $\mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$ (resp. $L(\mathcal{A}) \subseteq L(\mathcal{B})$)?

Theorem [AD90, AD94]
Language equivalence and language inclusion are undecidable in timed automata.

... as a special case of the universality problem (are all timed words accepted by the automaton?).

$\Rightarrow$ Proof by reduction from the recurring problem of a two-counter machine

[AD90] Alur, Dill. Automata for modeling real-time systems (ICALP’90).
Undecidability of universality

Theorem [AD90,AD94]

Universality of timed automata is undecidable.
Undecidability of universality

**Theorem [AD90,AD94]**

Universality of timed automata is undecidable.

- one configuration is encoded in one time unit
- number of $c$’s: value of counter $c$
- number of $d$’s: value of counter $d$
- one time unit between two corresponding $c$’s (resp. $d$’s)
Undecidability of universality

**Theorem [AD90,AD94]**

Universality of timed automata is undecidable.

- one configuration is encoded in one time unit
- number of $c$’s: value of counter $c$
- number of $d$’s: value of counter $d$
- one time unit between two corresponding $c$’s (resp. $d$’s)

⇝ We encode “non-behaviours” of a two-counter machine
Example
Module to check that if instruction \( i \) does not decrease counter \( c \), then all actions \( c \) appearing less than 1 t.u. after \( b_i \) has to be followed by another \( c \) 1 t.u. later.

\[
\begin{align*}
S_0 & \quad b_i, x := 0, \\
S_1 & \quad x < 1, c, x := 0, \\
S_2 & \quad x = 1, \neg c
\end{align*}
\]
Example

Module to check that if instruction $i$ does not decrease counter $c$, then all actions $c$ appearing less than 1 t.u. after $b_i$ has to be followed by an other $c$ 1 t.u. later.

The union of all small modules is not universal iff

The two-counter machine has a recurring computation
Bad news

- Language inclusion is **undecidable**
  (Bad news for the application to verification)
- Complementability is **undecidable**
- ...

[AD90, AD94] Finkel. Undecidable problems about timed automata (FORMATS’06)
[Tri03, Fin06] Tripakis. Folk theorems on the determinization and minimization of timed automata (FORMATS’03).
[Fin06] Finkel. Undecidable problems about timed automata (FORMATS’06).
Bad news

- Language inclusion is undecidable
  (Bad news for the application to verification)
- Complementability is undecidable
- ...

An example of non-determinizable/non-complementable timed aut.:
Bad news

- Language inclusion is **undecidable**
  (Bad news for the application to verification)
- Complementability is **undecidable**
- ...

**An example of non-determinizable/non-complementable aut.:**

![Diagram showing a non-determinizable/non-complementable automaton](image)

---

[AD90,AD94] Tripakis. Folk theorems on the determinization and minimization of timed automata (*FORMATS’03*).

[Tri03] Tripakis. Folk theorems on the determinization and minimization of timed automata (*FORMATS’03*).

[Fin06] Finkel. Undecidable problems about timed automata (*FORMATS’06*).

Bad news

- Language inclusion is *undecidable*
  (Bad news for the application to verification)
- Complementability is *undecidable*
- ...

An example of non-determinizable/non-complementable aut.:  

\[
\begin{array}{c}
S_0 \\
\quad \xymatrix{ & S_1 \ar[rr] & & S_0 } \\
\end{array}
\]

\[a, b\quad x \neq 1, a, b\]

\[a, x := 0\]

\[\text{UNTIME } \left( L \cap \{(a^*b^*, \tau) \mid \text{all } a's \text{ happen before } 1 \text{ and no two } a's \text{ simultaneously}\} \right) \text{ is not regular (exercise!)}\]

---

[Tri03] Tripakis. Folk theorems on the determinization and minimization of timed automata (*FORMATS’03*).
[Fin06] Finkel. Undecidable problems about timed automata (*FORMATS’06*).
Outline

1. Decidability of basic properties

2. Equivalence (or preorder) checking

3. Some extensions of timed automata
What if we extend the clock constraints?

- Diagonal constraints (i.e. $x - y \leq 2$)
What if we extend the clock constraints?

- Diagonal constraints \((i.e. \ x - y \leq 2)\)
  - decidable (with the same complexity)
What if we extend the clock constraints?

- Diagonal constraints \((i.e. \ x - y \leq 2)\)
  - **decidable** (with the same complexity)

is also a time-abstract bisimulation!
What if we extend the clock constraints?

- **Diagonal constraints** (i.e. $x - y \leq 2$)
  - **decidable** (with the same complexity)

  ![Diagonal constraints graph]

  is also a time-abstract bisimulation!

- **Linear constraints** (i.e. $2x + 3y \sim 5$)
What if we extend the clock constraints?

- **Diagonal constraints** (*i.e.* $x - y \leq 2$)
  - *decidable* (with the same complexity)

  ![Diagram of diagonal constraints](image)

  is also a time-abstract bisimulation!

- **Linear constraints** (*i.e.* $2x + 3y \sim 5$)
  - *undecidable* in general
What if we extend the clock constraints?

- **Diagonal constraints** (i.e. $x - y \leq 2$)
  - Decidable (with the same complexity)

  ![Diagram of diagonal constraints]

  is also a time-abstract bisimulation!

- **Linear constraints** (i.e. $2x + 3y \sim 5$)
  - Undecidable in general
  - Only decidable in few cases
What if we extend the clock constraints?

- **Diagonal constraints** *(i.e. $x - y \leq 2$)*
  - decidable (with the same complexity)
  - is also a time-abstract bisimulation!

- **Linear constraints** *(i.e. $2x + 3y \sim 5$)*
  - undecidable in general
  - only decidable in few cases
  - is a time-abstract bisimulation (when two clocks $x$ and $y$ and constraints $x + y \sim c$)!
What if we allow more operations on clocks?

- that can be transfer operations (i.e. $x := y$), or reinitialization operations (i.e. $x := 4$), or ...

What if we allow more operations on clocks?

- that can be **transfer operations** (*i.e.* $x := y$), or **reinitialization operations** (*i.e.* $x := 4$), or ...

[BDFP04] Bouyer, Dufourd, Fleury, Petit. Updatable Timed Automata (*Theoretical Computer Science*).

<table>
<thead>
<tr>
<th></th>
<th>simple constraints</th>
<th>+ diagonal constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x := c$, $x := y$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x := x + 1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x := y + c$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x := x - 1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x := c$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x &gt; c$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x \sim y + c$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y + c &lt;: x :&lt; y + d$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y + c &lt;: x :&lt; z + d$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

need of being very careful when using more operations on clocks!
What if we allow more operations on clocks?

- that can be transfer operations (*i.e.* \( x := y \)), or reinitialization operations (*i.e.* \( x := 4 \)), or ...

[BDFP04] Bouyer, Dufourd, Fleury, Petit. Updatable Timed Automata (*Theoretical Computer Science*).

<table>
<thead>
<tr>
<th></th>
<th>simple constraints</th>
<th>+ diagonal constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x := c, x := y )</td>
<td>decidable</td>
<td></td>
</tr>
<tr>
<td>( x := x + 1 )</td>
<td>decidable</td>
<td></td>
</tr>
<tr>
<td>( x := y + c )</td>
<td></td>
<td>undecidable</td>
</tr>
<tr>
<td>( x := x - 1 )</td>
<td></td>
<td>undecidable</td>
</tr>
<tr>
<td>( x :\leq c )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x :&gt; c )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x :\sim y + c )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( y + c :\leq x :\leq y + d )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( y + c :\leq x :\leq z + d )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( \Rightarrow \) need of being very careful when using more operations on clocks!
What if we allow more operations on clocks?

- that can be transfer operations \((i.e. x := y)\), or reinitialization operations \((i.e. x := 4)\), or ...

<table>
<thead>
<tr>
<th></th>
<th>simple constraints</th>
<th>+ diagonal constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x := c, x := y)</td>
<td>decidable</td>
<td>decidable</td>
</tr>
<tr>
<td>(x := x + 1)</td>
<td>decidable</td>
<td></td>
</tr>
<tr>
<td>(x := y + c)</td>
<td></td>
<td>undecidable</td>
</tr>
<tr>
<td>(x := x - 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x \leq c)</td>
<td></td>
<td>decidable</td>
</tr>
<tr>
<td>(x \geq c)</td>
<td></td>
<td>decidable</td>
</tr>
<tr>
<td>(x \sim y + c)</td>
<td></td>
<td>undecidable</td>
</tr>
<tr>
<td>(y + c \leq x \leq y + d)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(y + c \leq x \leq z + d)</td>
<td></td>
<td>undecidable</td>
</tr>
</tbody>
</table>

\(\Rightarrow\) need of being very careful when using more operations on clocks!

A note on hybrid automata (see more on Friday)

- a discrete control (the mode of the system)
- continuous evolution of the variables within a mode

A note on hybrid automata (see more on Friday)

- a discrete control (the mode of the system)
- continuous evolution of the variables within a mode

The thermostat example

\[
\begin{align*}
\text{Off} & : \dot{T} = -0.5T \\
& (T \geq 18) \\
\text{On} & : \dot{T} = 2.25 - 0.5T \\
& (T \leq 22)
\end{align*}
\]

Theorem [HKPV95]
The reachability problem is undecidable in hybrid automata, even for stopwatch automata.

(stopwatch automata: timed automata in which clocks can be stopped)

A relevant question: Is there something between timed automata and hybrid automata which is decidable? ⇝ See Nicolas' afternoon lecture

A note on hybrid automata (see more on Friday)

- A discrete control (the mode of the system)
- Continuous evolution of the variables within a mode

The thermostat example

\[ \begin{align*}
\text{Off} &; \quad \dot{T} = -0.5T \\
&; \quad (T \geq 18) \\
\text{On} &; \quad \dot{T} = 2.25 - 0.5T \\
&; \quad (T \leq 22)
\end{align*} \]

Theorem [HKPV95] The reachability problem is undecidable in hybrid automata, even for stopwatch automata. (Stopwatch automata: timed automata in which clocks can be stopped)

A relevant question: Is there something between timed automata and hybrid automata which is decidable? ⇝ See Nicolas' afternoon lecture

A note on hybrid automata (see more on Friday)

- A discrete control (the mode of the system)
- Continuous evolution of the variables within a mode

The thermostat example

Theorem [HKPV95]

The reachability problem is **undecidable** in hybrid automata, even for stopwatch automata.

(stopwatch automata: timed automata in which clocks can be stopped)

**The thermostat example**

\[ \dot{T} = \begin{cases} -0.5T & (T \geq 18) \\ 2.25 - 0.5T & (T \leq 22) \end{cases} \]

**Theorem** [HKPV95]

The reachability problem is **undecidable** in hybrid automata, even for stopwatch automata.

(stopwatch automata: timed automata in which clocks can be stopped)

A note on hybrid automata (see more on Friday)

a discrete control (the mode of the system)
+ continuous evolution of the variables within a mode

The thermostat example

Off
\( \dot{T} = -0.5T \)  
\((T \geq 18)\)

On
\( \dot{T} = 2.25 - 0.5T \)  
\((T \leq 22)\)

Theorem [HKPV95]
The reachability problem is **undecidable** in hybrid automata, even for stopwatch automata.

(stopwatch automata: timed automata in which clocks can be stopped)

A relevant question
Is there something between timed automata and hybrid automata which is decidable?
A note on hybrid automata (see more on Friday)

a discrete control (the mode of the system) + continuous evolution of the variables within a mode

The thermostat example

Theorem [HKPV95]

The reachability problem is undecidable in hybrid automata, even for stopwatch automata.

(stopwatch automata: timed automata in which clocks can be stopped)

A relevant question

Is there something between timed automata and hybrid automata which is decidable?  ⇝ See Nicolas’ afternoon lecture
Real-time Model Checking
— Timed Temporal Logics —

Nicolas MARKEY

Lav. Spécification & Vérification
CNRS & ENS Cachan – France

March 3, 2010
(Quantitative) Model checking

system:

⇒

property:

Always(safe)

model-checking algorithm

yes/no
(Quantitative) Model checking

system:

⇒

property:

Always(safe)

model-checking

algorithm

yes/no
timed automata

→

Always(safe)

→

yes/no

→
(Quantitative) Model checking

system:

⇒

property:

⇒

Always (safe) model-checking algorithm yes/no timed automata reachability via regions

Always (safe)

yes/no
(Quantitative) Model checking

system:
⇒
property:
Always(safe) model-checking algorithm yes/no
timed automata reachability via regions
quantitative temporal logics

timed automata
⇒
reachability via regions
⇒
yes/no
Quick reminder on untimed temporal logics

\[ \begin{align*}
\text{LTL } &\models \varphi \quad \bigcirc \mid \neg \varphi \mid \varphi \land \varphi \mid X \varphi \mid \varphi U \varphi \\
\text{CTL } &\models \varphi \quad \bigcirc \mid \neg \varphi \mid \varphi \land \varphi \mid E\psi \mid A\psi \\
\psi &\models X \varphi \mid \varphi U \varphi
\end{align*} \]

Quick reminder on untimed temporal logics

LTL \( \exists \varphi \quad \bigcirc \mid \neg \varphi \mid \varphi \land \varphi \mid X \varphi \mid \varphi U \varphi \)

CTL \( \exists \varphi \quad \bigcirc \mid \neg \varphi \mid \varphi \land \varphi \mid E\psi \mid A\psi \)

\[\psi \mid X \varphi \mid \varphi U \varphi\]

Quick reminder on untimed temporal logics

LTL \( \exists \varphi \) \( \bigcirc \ \mid \neg \varphi \mid \varphi \land \varphi \mid X \varphi \mid \varphi \ U \varphi \)

CTL \( \exists \varphi \) \( \bigcirc \ \mid \neg \varphi \mid \varphi \land \varphi \mid E\psi \mid A\psi \)
\( \psi \) \( X \varphi \mid \varphi \ U \varphi \)

Quick reminder on untimed temporal logics

LTL ⊨ ϕ ::= | ¬ϕ | ϕ ∧ ϕ | X ϕ | ϕ U ϕ

CTL ⊨ ϕ ::= | ¬ϕ | ϕ ∧ ϕ | Eψ | Aψ

ψ ::= X ϕ | ϕ U ...

Quick reminder on untimed temporal logics

LTL ∋ ϕ ::= |¬ϕ |ϕ ∧ϕ |X ϕ |ϕ U ϕ

CTL ∋ ϕ ::= |¬ϕ |ϕ ∧ϕ |Eψ |Aψ

ψ ::= X ϕ |ϕ U ...

Quick reminder on untimed temporal logics

LTL \( \exists \varphi \): \( \varnothing \mid \neg \varphi \mid \varphi \land \varphi \mid X \varphi \mid \varphi U \varphi \)

CTL \( \exists \varphi \): \( \varnothing \mid \neg \varphi \mid \varphi \land \varphi \mid E \psi \mid A \psi \)

\[ \psi \mid X \varphi \mid \varphi U \varphi \]

Quick reminder on untimed temporal logics

**LTL** \( \exists \varphi \)  
\[ \bigcirc | \neg \varphi | \varphi \land \varphi | X \varphi | \varphi \mathbin{U} \varphi \]

**CTL** \( \exists \varphi \)  
\[ \bigcirc | \neg \varphi | \varphi \land \varphi | E\psi | A\psi \]

\[ \psi \quad X \varphi | \varphi \mathbin{U} \varphi \]

Refs:  
Quick reminder on untimed temporal logics

LTL $\exists \varphi$ $\bigcirc | \neg \varphi | \varphi \land \varphi | \mathbf{X} \varphi | \varphi \mathbf{U} \varphi$

CTL $\exists \varphi$ $\bigcirc | \neg \varphi | \varphi \land \varphi | \mathbf{E} \psi | \mathbf{A} \psi$

$\psi$ $\mathbf{X} \varphi | \varphi \mathbf{U} \varphi$

Quick reminder on untimed temporal logics

LTL $\exists \varphi ::= \Box \neg \varphi | \varphi \land \varphi | X \varphi | \varphi U \varphi$

CTL $\exists \varphi ::= \Box \neg \varphi | \varphi \land \varphi | E\psi | A\psi$

Example

- $\bigvee \Box U \bigcirc \lor \bigcirc G \bigcirc$: weak until

Quick reminder on untimed temporal logics

LTL $\exists \varphi \mid \neg \varphi \mid \varphi \land \varphi \mid X \varphi \mid \varphi U \varphi$

CTL $\exists \varphi \mid \neg \varphi \mid \varphi \land \varphi \mid E\psi \mid A\psi$

$\psi \mid X \varphi \mid \varphi U \varphi$

Example

- $\bigcirc U \bigcirc \lor G \bigcirc$: weak until
- $G F \bigcirc$: “infinitely often”

Quick reminder on untimed temporal logics

LTL $\exists \varphi = \circ \ | \neg \varphi | \varphi \land \varphi | X \varphi | \varphi U \varphi$

CTL $\exists \varphi = \circ \ | \neg \varphi | \varphi \land \varphi | E\psi | A\psi$

$\psi = \varphi | X \varphi | \varphi U \varphi$

Example

- $\circ U \circ \lor G \circ$: weak until
- $G F \circ$: “infinitely often”
- $A G \circ \Rightarrow A F \circ$: response property

Quick reminder on untimed temporal logics

LTL $\exists \varphi ::= |\neg \varphi | \varphi \land \varphi | X \varphi | \varphi U \varphi$

CTL $\exists \varphi ::= |\neg \varphi | \varphi \land \varphi | E\psi | A\psi$

\[
\psi ::= \varphi | X \varphi | \varphi U \varphi
\]

Example

- $\bigcirc \bigcirc U \bigcirc \lor G \bigcirc$: weak until
- $G F \bigcirc$: “infinitely often”
- $A G \bigcirc \Rightarrow A F \bigcirc$: response property
- $A G F \bigcirc \Rightarrow G \bigcirc$: fair runs are safe (not a CTL formula)

Outline of the talk

1. Introduction

2. Extending temporal logics with real-time constraints
   - Continuous and pointwise semantics
   - Expressiveness issues

3. Model checking timed linear-time logics
   - Undecidability of MTL and TPTL
   - Decidable fragments

4. Model checking timed branching-time logics

5. Conclusions and open problems
Outline of the talk

1 Introduction

2 Extending temporal logics with real-time constraints
   - Continuous and pointwise semantics
   - Expressiveness issues

3 Model checking timed linear-time logics
   - Undecidability of MTL and TPTL
   - Decidable fragments

4 Model checking timed branching-time logics

5 Conclusions and open problems
Extending temporal modalities with time

- decorating modalities with timing constraints:

\[ | x := 0 \quad x \leq 5 \quad | \quad F(\land x. \quad G(x \leq 5 \implies \neg)) \]

Extending temporal modalities with time

- decorating modalities with timing constraints:

\[
\begin{align*}
& 1.4 & 3.4 & 0.2 & 1.3 & 1.2 \\
\end{align*}
\]

\[
\begin{array}{c}
\downarrow \\
\end{array}
\]

Extending temporal modalities with time

- decorating modalities with timing constraints:

\[
1.4 \rightarrow 3.4 \rightarrow 0.2 \rightarrow 1.3 \rightarrow 1.2 \rightarrow \cdots \quad | \quad U \geq 5
\]

\[
1.4 \rightarrow 3.5 \rightarrow 1.8 \rightarrow 3.6 \rightarrow 0.9 \rightarrow \cdots \quad | \quad F_{\geq 6} \equiv \top \quad U_{\geq 6}
\]

Extending temporal modalities with time

- decorating modalities with timing constraints:

\[
\begin{align*}
&\quad | \quad \textcolor{blue}{\mathbf{U}} \quad 5 \quad \textcolor{red}{\mathbf{U}} \\
&\quad | \quad \mathbf{F} \quad \mathbf{U} \quad \mathbf{U} \\
\end{align*}
\]

Extending temporal modalities with time
decorating modalities with timing constraints:

\[ \begin{array}{c}
1.4 \quad 3.4 \quad 0.2 \quad 1.3 \quad 1.2 \\
\hline
\end{array} \]

\[ \begin{array}{c}
U \quad 5
\hline
\end{array} \]

Extending temporal modalities with time

- decorating modalities with timing constraints:

<table>
<thead>
<tr>
<th>1.4</th>
<th>3.4</th>
<th>0.2</th>
<th>1.3</th>
<th>1.2</th>
</tr>
</thead>
</table>

\[ | U \geq 5 \]

<table>
<thead>
<tr>
<th>1.4</th>
<th>3.5</th>
<th>1.8</th>
<th>3.6</th>
<th>0.9</th>
</tr>
</thead>
</table>

\[ | F_{\geq 6} \equiv U_{\geq 6} \]

<table>
<thead>
<tr>
<th>1.4</th>
<th>1.7</th>
<th>2.5</th>
<th>0.7</th>
<th>1.2</th>
</tr>
</thead>
</table>

\[ | G_{\leq 7} \equiv \neg F_{\leq 7} \]

Extending temporal modalities with time

- decorating modalities with timing constraints:

- using formula clocks

Extending temporal modalities with time

- decorating modalities with timing constraints:

- using formula clocks

Extending temporal modalities with time

- decorating modalities with timing constraints:

- using formula clocks

Timed words vs. timed state sequences

Example

- **Continuous semantics**
  - $a, x \leq 2$
  - $y = 0$
  - $b, y > 0$
  - $x = 0$
  - $c, y \leq 2$
  - $x = 0$

- **Pointwise semantics**
  - $x = 0$
  - $y = 0$
  - $x = 1.5$
  - $y = 0$
  - $x = 0$
  - $y = 1.3$
  - $x = 2.6$
  - $y = 0$
  - $x = 0$
  - $y = 1.3$
Timed words vs. timed state sequences

Example

- **pointwise** semantics
  - $a, x \leq 2, y = 0$
  - $b, y > 0, x = 0$
  - $c, y \leq 2, x = 0$
  - $a, x \geq 2, y = 0$

- **continuous** semantics
  - $x = 0$
  - $y = 0$

Diagram:

- Nodes labeled with conditions:
  - $a$, $x \leq 2$, $y = 0$
  - $b$, $y > 0$, $x = 0$
  - $c$, $y \leq 2$, $x = 0$
  - $a$, $x \geq 2$, $y = 0$

- Edges connecting the nodes with the corresponding conditions.
Timed words vs. timed state sequences

Example

\[ a, \quad x \leq 2 \]
\[ y := 0 \]

\[ b, \quad y > 0 \]
\[ x := 0 \]

\[ a, \quad x \geq 2 \]
\[ y := 0 \]

\[ c, \quad y \leq 2 \]
\[ x := 0 \]

pointwise semantics

\[ x = 0 \]
\[ y = 0 \]
\[ x = 1.5 \]
\[ y = 0 \]

\[ x = 0 \]
\[ y = 1.3 \]

continuous semantics

\[ x = 0 \]
\[ y = 1.3 \]
\[ x = 2.6 \]
\[ y = 0 \]

\[ x = 0 \]
\[ y = 1.3 \]
Timed words vs. timed state sequences

Example

\[
\begin{align*}
    &a, \quad x\leq 2 \quad y = 0 \\
    &b, \quad y > 0 \quad x = 0
\end{align*}
\]

Continuous semantics

Pointwise semantics
Timed words vs. timed state sequences

Example

\[ a, \quad x \leq 2 \quad y := 0 \]
\[ b, \quad y > 0 \quad x := 0 \]
\[ a, \quad x \geq 2 \quad y := 0 \]
\[ c, \quad y \leq 2 \quad x := 0 \]

pointwise semantics

\[ x = 0 \]
\[ y = 0 \]
\[ x = 1.5 \]
\[ y = 0 \]
\[ x = 0 \]
\[ y = 1.3 \]
\[ x = 2.6 \]
\[ y = 0 \]

continuous semantics

\[ x = 2.6 \]
\[ y = 0 \]

\[ a \]
\[ 1.5 \]
\[ b \]
\[ 2.8 \]
\[ a \]
\[ 5.4 \]
Timed words vs. timed state sequences

Example

\begin{align*}
  &a, \quad x \leq 2 \\
  &y := 0 \\
  &b, \quad y > 0 \\
  &x := 0 \\
  &a, \quad x \geq 2 \\
  &y := 0 \\
  &c, \quad y \leq 2 \\
  &x := 0
\end{align*}

\textit{pointwise} semantics
\begin{align*}
  &x = 0 \\
  &y = 0 \\
  &x = 1.5 \\
  &y = 0 \\
  &x = 2.6 \\
  &y = 0
\end{align*}

\textit{continuous} semantics
\begin{align*}
  &x = 0 \\
  &y = 1.3
\end{align*}

\begin{align*}
  &a_{1.5} \\
  &b_{2.8} \\
  &a_{5.4} \\
  &c_{6.7}
\end{align*}
Timed logics in the pointwise framework

Definition

\[
\text{MTL} \ni \varphi \quad \circ \quad \neg \varphi \quad \varphi \lor \varphi \quad \varphi \mathbin{\bigvee} I \varphi
\]

where \( \circ \) ranges over \( \{\circ, \bigcirc, \ldots\} \) and \( I \) is an interval with bounds in \( \mathbb{Q}^+ \cup \{+\infty\} \).
Timed logics in the pointwise framework

Definition

Pointwise semantics of MTL: over $\pi = (w_i, t_i)_i$ with $t_0 = 0$:

- $\pi, i \models \varphi U_I \psi$ iff there exists some $j > 0$ s.t.
  - $\pi, i + j \models \psi$,
  - $\pi, i + k \models \varphi$ for all $0 < k < j$,
  - $t_{i+j} - t_i \in I$. 
Timed logics in the pointwise framework

Definition

Pointwise semantics of MTL: over $\pi = (w_i, t_i)_i$ with $t_0 = 0$:

- $\pi, i \models \varphi U I \psi$ iff there exists some $j > 0$ s.t.
  - $\pi, i + j \models \psi$,
  - $\pi, i + k \models \varphi$ for all $0 < k < j$,
  - $t_{i + j} - t_i \in I$.

Example

\begin{enumerate}
  \item $\text{init,0}$
  \item $a,0.6$
  \item $a,1.2$
  \item $c,2.1$
\end{enumerate}

$a \mathbf{U}_{2,3} c$
Timed logics in the pointwise framework

Definition

Pointwise semantics of MTL: over $\pi = (w_i, t_i)_i$ with $t_0 = 0$:

- $\pi, i | \varphi \mathcal{U} j \psi$ iff there exists some $j > 0$ s.t.
  - $\pi, i + j | \psi$,
  - $\pi, i + k | \varphi$ for all $0 < k < j$,
  - $t_i + j - t_i \in I$.

Example

0 1 2
(init,0) (b,0.8) (b,1.3) (a,2.3)
F(b \land \bot \mathcal{U}[1,1] a)
Timed logics in the pointwise framework

**Definition**

Pointwise semantics of MTL: over $\pi = (w_i, t_i)_i$ with $t_0 = 0$:

- $\pi, i | \varphi \mathbf{U}_1 \psi$ iff there exists some $j > 0$ s.t.
  - $\pi, i+j | \psi$,
  - $\pi, i+k | \varphi$ for all $0 < k < j$,
  - $t_{i+j} - t_i \in I$.

**Example**

```
0 1 2
(init,0) (b,0.9) (c,2)
F[2,2] c
```

$\not\equiv F=1 F=1 c$
Definition

Pointwise semantics of MTL: over \( \pi = (w_i, t_i)_i \) with \( t_0 = 0 \):

\[ \pi, i \models \varphi U I \psi \] iff there exists some \( j > 0 \) s.t.

- \( \pi, i \models j \models \psi \),
- \( \pi, i \models k \models \varphi \) for all \( 0 < k < j \),
- \( t_i \models j - t_i \in I \).

Example

\( \begin{align*}
0 & \quad 1 & \quad 2 \\
\text{init,0} & \quad \text{b,0.9} & \quad \text{c,2} \\
\text{F}_{2,2} \quad \text{c} & \quad \text{def} \quad \text{F}_{2} \quad \text{c}
\end{align*} \)
Timed logics in the pointwise framework

Definition

Pointwise semantics of MTL: over $\pi = (w_i, t_i)_i$ with $t_0 = 0$:

- $\pi, i \models \varphi \mathbf{U} I \psi$ iff there exists some $j > 0$ s.t.
  - $\pi, i \models \psi$,
  - $\pi, i \models \varphi$ for all $0 < k < j$,
  - $t_i + j - t_i \in I$.

Example

\[
\begin{array}{c}
\text{init,0} & \text{b,0.9} & \text{c,2} \\
\hline
0 & 1 & 2
\end{array}
\]

$F_{[2,2]} c \not\equiv F=1 F=1 c$
Timed logics in the pointwise framework

Definition

\[
\text{TPTL } \exists \varphi \quad \bigcirc | x \sim c | \neg \varphi | \varphi \lor \varphi | \varphi \lor \varphi | x. \varphi
\]

where \( \bigcirc \) ranges over \( \{\bigcirc, \bigcirc, \ldots\} \), \( x \) ranges over a set of formula clocks, \( c \in Q^+ \) and \( \sim \in \{<, \leq, =, \geq, >\} \).
## Timed logics in the pointwise framework

### Definition

Pointwise semantics of TPTL: over $\pi = (w_i, t_i)_i$ with $t_0 = 0$, under some clock valuation $\tau$: :

- $\pi, i, \tau \models x \sim c$ iff $\tau x \sim c$
Timed logics in the pointwise framework

Definition

Pointwise semantics of TPTL: over \( \pi = (w_i, t_i) \) with \( t_0 = 0 \), under some clock valuation \( \tau : \)

- \( \pi, i, \tau | x \sim c \iff \tau x \sim c \)
- \( \pi, i, \tau | x. \varphi \iff \pi, i, \tau x \leftarrow 0 | \varphi \)
Definition

Pointwise semantics of TPTL: over $\pi = (w_i, t_i)_i$ with $t_0 = 0$, under some clock valuation $\tau$: :

- $\pi, i, \tau \models x \sim c$ iff $\tau x \sim c$
- $\pi, i, \tau \models x. \varphi$ iff $\pi, i, \tau x \leftarrow 0 \models \varphi$
- $\pi, i, \tau \models \varphi U \psi$ iff there exists some $j > 0$ s.t.
  - $\pi, i, j, \tau \models t_i j - t_i \models \psi$,
  - $\pi, i, k, \tau \models t_i k - t_i \models \varphi$ for all $0 < k < j$. 
Timed logics in the pointwise framework

Definition

Pointwise semantics of TPTL: over $\pi = (w_i, t_i)_i$ with $t_0 = 0$, under some clock valuation $\tau$:

- $\pi, i, \tau \models x \sim c$ iff $\tau x \sim c$
- $\pi, i, \tau \models x. \varphi$ iff $\pi, i, \tau x \leftarrow 0 \models \varphi$
- $\pi, i, \tau \models \varphi U \psi$ iff there exists some $j > 0$ s.t.
  - $\pi, i, j, \tau \models t_i \cdot j - t_i \models \psi$,
  - $\pi, i, k, \tau \models t_i \cdot k - t_i \models \varphi$ for all $0 < k < j$.

Example

$0 \quad 1 \quad 2$

(init,0) (a,0.6) (a,1.2) (c,2.1)

$x. a U (c \land x \in [2,3])$
Timed logics in the pointwise framework

Definition

Pointwise semantics of TPTL: over $\pi = (w_i, t_i)_i$ with $t_0 = 0$, under some clock valuation $\tau$: :

- $\pi, i, \tau \models x \sim c$ iff $\tau x \sim c$
- $\pi, i, \tau \models x. \varphi$ iff $\pi, i, \tau x \leftarrow 0 \models \varphi$
- $\pi, i, \tau \models \varphi U \psi$ iff there exists some $j > 0$ s.t.
  - $\pi, i, j, \tau \models t_i \cdot j - t_i \models \psi,$
  - $\pi, i, k, \tau \models t_i \cdot k - t_i \models \varphi$ for all $0 < k < j$.

Example

0 1 2
(init,0) (a,0.6) (b,1.1) (a,2.1)

$F (b \land x. \perp U (a \land x = 1))$
Definition

Pointwise semantics of TPTL: over $\pi = (w_i, t_i)_i$ with $t_0 = 0$, under some clock valuation $\tau$: :

- $\pi, i, \tau \models x \sim c$ iff $\tau x \sim c$
- $\pi, i, \tau \models x. \varphi$ iff $\pi, i, \tau x \leftarrow 0 \models \varphi$
- $\pi, i, \tau \models \varphi U \psi$ iff there exists some $j > 0$ s.t.
  - $\pi, i, j, \tau \models t_i j - t_i \models \psi,$
  - $\pi, i, k, \tau \models t_i k - t_i \models \varphi$ for all $0 < k < j$.

Example

$x. F(a \land F(b \land x \leq 1))$
Timed logics in the continuous framework

Definition

Continuous semantics of MTL: over $\pi : \mathbb{R}^+ \rightarrow \{\top, \bot, \ldots\}$:
- $\pi, t \models \varphi U I \psi$ iff there exists some $u > 0$ s.t.
  - $\pi, t + u \models \psi$,
  - $\pi, t + v \models \varphi$ for all $0 < v < u$,
  - $u \in I$.

$\pi, t \models p$ iff $p \in \pi(t)$.
**Definition**

Continuous semantics of MTL: over $\pi: \mathbb{R}^+ \rightarrow \{\bigcirc, \lozenge, \ldots\}$:

- $\pi, t \models \varphi U I \psi$ iff there exists some $u > 0$ s.t.
  - $\pi, t + u \models \psi$,
  - $\pi, t + v \models \varphi$ for all $0 < v < u$,
  - $u \in I$.

- $\pi, t \models p$ iff $p \in \pi(t)$
Timed logics in the continuous framework

**Definition**

Continuous semantics of MTL: over \( \pi \rightarrow \{ \circ, \bullet, ... \} \):

- \( \pi, t \models \varphi \mathbin{\mathbf{U}_I} \psi \) iff there exists some \( u > 0 \) s.t.
  - \( \pi, t + u \models \psi \),
  - \( \pi, t + v \models \varphi \) for all \( 0 < v < u \),
  - \( u \in I \).

- \( \pi, t \models p \) iff \( p \in \pi(t) \)

**Example**

0 1 2 ( \lor \mathbin{\mathbf{U} \leq 2} )
### Definition

Continuous semantics of MTL: over $\pi: \mathbb{R}^+ \rightarrow \{\bigcirc, \bigcirc, \ldots\}$:

\[
\quad \pi, t \models \varphi \bigcirc I \psi \quad \text{iff} \quad \text{there exists some } u > 0 \text{ s.t.}
\]
\[
\quad - \quad \pi, t + u \models \psi,
\]
\[
\quad - \quad \pi, t + v \models \varphi \text{ for all } 0 < v < u,
\]
\[
\quad - \quad u \in I.
\]

\[
\pi, t \models p \quad \text{iff} \quad p \in \pi(t)
\]

### Example

0 1 2
\[F = 2 \equiv F = 1(F = 1)\]

\[F_2 \bigcirc\]
Timed logics in the continuous framework

Definition

Continuous semantics of MTL: over $\pi: \mathbb{R}^+ \rightarrow \{\bigcirc, \triangle, \ldots\}$:

- $\pi, t | \varphi \mathrel{U} \psi$ iff there exists some $u > 0$ s.t.
  - $\pi, t + u | \psi$,
  - $\pi, t + v | \varphi$ for all $0 < v < u$,
  - $u \in I$.

- $\pi, t | p$ iff $p \in \pi(t)$

Example

$0 \quad 1 \quad 2 \quad \text{F} \quad 2 \bigcirc \equiv \text{F} \quad 1 \quad \text{F} \quad 1 \bigcirc$
Timed logics in the continuous framework

Definition

Continuous semantics of TPTL: over \( \pi \):

\[ \pi, t, \tau | x \sim c \quad \text{iff} \quad \tau x \sim c \]

\[ \pi, t, \tau | \varphi \quad \text{iff} \quad \pi, i, \tau [x \leftarrow 0] | \varphi \]

\[ \pi, t, \tau | \varphi U \psi \quad \text{iff} \quad \exists u > 0 \text{ s.t.} \]

- \[ \pi, t + u, \tau + u - t | \psi \]
- \[ \pi, i + k, \tau + v - t | \varphi \quad \text{for all} \quad 0 < v < u. \]
Definition

Continuous semantics of TPTL: over $\pi: \mathbb{R}^+ \rightarrow \{\bigcirc, \bigcirc, \ldots\}$:

- $\pi, t, \tau \models x \sim c \iff \tau \times x \sim c$

- $\pi, t, \tau \models x. \varphi \iff \pi, i, \tau \times \leftarrow 0 \models \varphi$
Timed logics in the continuous framework

Definition

Continuous semantics of TPTL: over \( \pi \rightarrow \{\bigcirc, \bigcirc, \ldots\} \):

- \( \pi, t, \tau | x \sim c \iff \tau x \sim c \)
- \( \pi, t, \tau | x. \varphi \iff \pi, i, \tau x^0 | \varphi \)
- \( \pi, t, \tau | \varphi U \psi \iff \) there exists some \( u > 0 \) s.t.
  - \( \pi, t | u, \tau | u - t | \psi \),
  - \( \pi, i | k, \tau | v - t | \varphi \) for all \( 0 < v < u \).
Timed logics in the continuous framework

Definition

Continuous semantics of TPTL: over $\pi : \mathbb{R}^+ \to \{\circ, \cdot, \ldots\}$:

- $\pi, t, \tau \models x \sim c$ iff $\tau x \sim c$
- $\pi, t, \tau \models x. \varphi$ iff $\pi, i, \tau \leftarrow 0 \models \varphi$
- $\pi, t, \tau \models \varphi \mathcal{U} \psi$ iff there exists some $u > 0$ s.t.
  - $\pi, t + u, \tau + u - t \models \psi$,
  - $\pi, i + k, \tau + v - t \models \varphi$ for all $0 < v < u$.

Example

$0 \quad 1 \quad 2 \quad x. \circ \lor \circ \mathcal{U} \circ \land x \leq 2$
Timed logics in the continuous framework

**Definition**

Continuous semantics of TPTL: over $\pi: \mathbb{R}^+ \rightarrow \{0, 0, ...\}$:

- $\pi, t, \tau \models x \sim c \iff \tau x \sim c$
- $\pi, t, \tau \models x.\varphi \iff \pi, i, \tau \models x \leftarrow 0 \models \varphi$
- $\pi, t, \tau \models \varphi \mathbf{U} \psi \iff$ there exists some $u > 0$ s.t.
  - $\pi, t \models u, \tau \models u - t \models \psi$,
  - $\pi, i \models k, \tau \models v - t \models \varphi$ for all $0 < v < u$.

**Example**

0 1 2 x.$\mathbf{F}(\mathbf{x} \leq 2)$
Relative expressiveness of TPTL and MTL

Lemma

**MTL can be translated into TPTL.**

Proof.

\[ \varphi \mathbf{U}_I \psi \equiv x. \varphi \mathbf{U} \psi \land x \in I \]
Relative expressiveness of TPTL and MTL

Lemma

MTL can be translated into TPTL.

Proof.

\[ \varphi \mathbf{U}_I \psi \equiv x. \varphi \mathbf{U} \psi \land x \in I \, . \]

Conversely, consider the following TPTL formula:

\[ \mathbf{G} \left[ \mathbf{G} \Rightarrow x. \mathbf{F} \mathbf{G} \land \mathbf{F} \mathbf{G} \land x \leq 2 \right] \, . \]

It characterizes the following pattern:

0 1 2
green red blue
Relative expressiveness of TPTL and MTL

\[ G[\bigcirc \Rightarrow x. F \bigcirc \land F \bigcirc \land x \leq 2 ] . \]
Relative expressiveness of TPTL and MTL

\[ G \Rightarrow x. F \land F \land x \leq 2 \]
Relative expressiveness of TPTL and MTL

\[ \mathbf{G} \Rightarrow x. \mathbf{F} \land \mathbf{F} \land x \leq 2 \]

Remark
This translation is only valid in the continuous semantics.
Relative expressiveness of TPTL and MTL

\[ G \Rightarrow x. F \land F \land x \leq 2 \]

Remark
This translation is only valid in the continuous semantics
Relative expressiveness of TPTL and MTL

\[ G \Rightarrow x. F \land F \land x \leq 2 \]

Remark: This translation is only valid in the continuous semantics.
Relative expressiveness of TPTL and MTL

\[ G \Rightarrow x. F \land F \land x \leq 2 \]

\begin{align*}
G \Rightarrow & \\
F_{0,1} \land F_{1,2} & \\
\lor & \\
F_{0,1} \land F_{0,1} & \\
\lor & \\
F_{0,1} \land F_{0,1} & \\
\lor & \\
F_{0,1} \land F & \\
\end{align*}

Remark
This translation is only valid in the continuous semantics
Relative expressiveness of TPTL and MTL

Theorem

*TPTL is strictly more expressive than MTL.*

Theorem

*TPTL is strictly more expressive than MTL.*

Proof.

- In the pointwise semantics:

\[ G \left[ \text{□} \Rightarrow x. \ F \land F \land x \leq 2 \right] \]

cannot be expressed in MTL.

- In both semantics:

\[ \varphi \ x. \ F \land x \leq 1 \land G \ x \leq 1 \Rightarrow \neg \triangle \]

cannot be expressed in MTL.

Outline of the talk

1. Introduction

2. Extending temporal logics with real-time constraints
   - Continuous and pointwise semantics
   - Expressiveness issues

3. Model checking timed linear-time logics
   - Undecidability of MTL and TPTL
   - Decidable fragments

4. Model checking timed branching-time logics

5. Conclusions and open problems
MTL model-checking

Theorem

MTL model-checking and satisfiability are undecidable under the continuous semantics.

MTL model-checking

Theorem

MTL model-checking and satisfiability are undecidable under the continuous semantics.

Proof.

Encode the halting problem of a Turing machine:

One time-unit = one configuration of the Turing machine

MTL model-checking

Theorem

MTL model-checking and satisfiability are undecidable under the continuous semantics.

Proof.

Encode the halting problem of a Turing machine:

One time-unit = one configuration of the Turing machine

MTL model-checking

**Theorem**

*MTL model-checking and satisfiability are *undecidable* under the continuous semantics.*

**Proof.**

Encode the halting problem of a Turing machine:

One time-unit = one configuration of the Turing machine

MTL model-checking

**Theorem**

MTL model-checking and satisfiability are *undecidable* under the continuous semantics.

**Proof.**

Encode the halting problem of a Turing machine:

One time-unit = one configuration of the Turing machine

MTL model-checking

Remark

This reduction requires continuous semantics, or the use of past-time modalities:

\[ n \quad n+1 \quad \ldots \]

References:

MTL model-checking

Remark

This reduction requires continuous semantics, or the use of past-time modalities:

\[ n \quad n+1 \quad n+2 \]

Theorem

Under pointwise semantics, MTL model-checking and satisfiability are undecidable over infinite timed words; are decidable (with non-primitive recursive complexity) over finite timed words.

MTL model-checking

Remark

This reduction requires continuous semantics, or the use of past-time modalities:

$$n \rightarrow n+1 \is \rightarrow 1$$

“insertion errors”

Remark
This reduction requires continuous semantics, or the use of past-time modalities:

Theorem
Under pointwise semantics, MTL model-checking and satisfiability
- are undecidable over infinite timed words;
- are decidable (with non-primitive recursive complexity) over finite timed words.

Metric Interval Temporal Logic

Definition

MITL is the fragment of MTL where punctuality is not allowed:

\[ \text{MITL} \ni \varphi ::= \bigcirc | \neg \varphi | \varphi \lor \varphi | \varphi \cup I \varphi \]

where \( \bigcirc \) ranges over \( \{\bigcirc, \bigcirc, \ldots\} \) and \( I \) is a non-punctual interval with bounds in \( \mathbb{U}\{\infty\} \).

Metric Interval Temporal Logic

Definition

MITL is the fragment of MTL where punctuality is not allowed:

\[
\text{MITL} \ni \varphi ::= \varphi \lor \varphi \lor \varphi U I \varphi
\]

where \( I \) ranges over \{\( \), \( \), \( \), \( \), \( \), \( \)\} and \( I \) is a non-punctual interval with bounds in \( \bigcup \{ \infty \} \).

Example

- \( \text{G (\(\), \(\)) \Rightarrow F \(1,2\) (\(\))} \) is an MITL formula;
- \( \text{G (\(\), \(\)) \Rightarrow F \(1\) (\(\))} \) is not.

Metric Interval Temporal Logic

Definition

MITL is the fragment of MTL where punctuality is not allowed:

\[ \text{MITL} \ni \varphi := \circ \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \cup I \varphi \]

where \( \circ \) ranges over \( \{\circ_{\text{lo}}, \circ_{\text{hi}}, \ldots\} \) and \( I \) is a non-punctual interval with bounds in \( \mathbb{U} \{\infty\} \).

Example

- \( G \circ_{\text{lo}} \Rightarrow F_{1,2} \circ_{\text{hi}} \) is an MITL formula;
- \( G \circ_{\text{lo}} \Rightarrow F_{1} \circ_{\text{hi}} \) is not.

Theorem

**MITL** model checking and satisfiability are **EXPSPACE-complete**.

(Co)Flat MTL

Definition

CoFlatMTL is the fragment of MTL defined as:

\[
\text{CoFlatMTL} \ni \varphi ::= |\neg | \varphi \lor \varphi | \varphi \land \varphi |
\varphi \U_I \varphi | \varphi \U_J \psi | \varphi \R_I \varphi | \psi \R_J \varphi
\]

where

- \(\circlearrowleft\) ranges over \(\{\circlearrowleft, \circlearrowright, \ldots\}\),
- \(I\) ranges over bounded intervals with bounds in \(\mathbb{Q}\),
- \(J\) ranges over intervals with bounds in \(\mathbb{Q} \cup \{+\infty\}\), and
- \(\psi\) ranges over MITL.

(Co)Flat MTL

Definition

CoFlatMTL is the fragment of MTL defined as:

$$\text{CoFlatMTL } \ni \phi \equiv \bigcirc \mid \neg \bigcirc \mid \phi \lor \phi \mid \phi \land \phi \mid \phi U_I \phi \mid \phi U_J \psi \mid \phi R_I \phi \mid \psi R_J \phi$$

Remark

CoFlatMTL is not closed under negation.

(Co)Flat MTL

Definition

CoFlatMTL is the fragment of MTL defined as:

\[ \text{CoFlatMTL} \ni \varphi ::= \top \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \varphi \mathsf{U}_I \varphi \mid \varphi \mathsf{U}_J \psi \mid \varphi \mathsf{R}_I \varphi \mid \psi \mathsf{R}_J \varphi \]

Remark

CoFlatMTL is not closed under negation.

Example

- \( G \top \Rightarrow F \top \) is in CoFlatMTL.
- \( F \top \land G \top \) is in FlatMTL, but not in CoFlatMTL.

(Co)Flat MTL

**Definition**

CoFlatMTL is the fragment of MTL defined as:

\[
\text{CoFlatMTL} \ni \phi ::= | \neg | \phi \lor \phi | \phi \land \phi | \phi \mathcal{U} I \phi | \phi \mathcal{U} J \psi | \phi \mathcal{R} I \phi | \psi \mathcal{R} J \phi
\]

**Remark**

CoFlatMTL is not closed under negation.

**Theorem**

CoFlatMTL model-checking is EXPSPACE-complete. CoFlatMTL satisfiability is undecidable.

Outline of the talk

1. Introduction

2. Extending temporal logics with real-time constraints
   - Continuous and pointwise semantics
   - Expressiveness issues

3. Model checking timed linear-time logics
   - Undecidability of MTL and TPTL
   - Decidable fragments

4. Model checking timed branching-time logics

5. Conclusions and open problems
### Definition

<table>
<thead>
<tr>
<th>TCTL</th>
<th>ϕ ∈ {¬ϕ, ϕ ∧ ϕ, Eϕ U≤c ϕ, Aϕ U&lt; c ϕ, Aϕ U= c ϕ, Aϕ U&gt; c ϕ, Aϕ U≥ c ϕ}</th>
</tr>
</thead>
</table>

where ⊙ ∈ {⊙, ⊙, ⊙, ...}, ~ ∈ {≤, <, =, >, ≥} and c ∈ N.

Branching-time logics with timing constraints – syntax

Definition

\[
\text{TCTL } \equiv \varphi \mid \neg \varphi \mid \varphi \land \varphi \mid E\varphi \ U_{\sim c} \varphi \mid A\varphi \ U_{\sim c} \varphi
\]

where \( \in \{\text{ }, \ldots \} \), \( \sim \in \{\leq, <, =, >, \geq} \) and \( c \in \mathbb{N} \).

Example

\[\cdot \ A \ G \ \bullet \ \Rightarrow \ E \ F_{\leq 5} \ \bullet\]

Branching-time logics with timing constraints – syntax

Definition

\[ TCTL \ni \varphi ::= | \neg \varphi | \varphi \land \varphi | E \varphi \ U \sim c \ \varphi | A \varphi \ U \sim c \ \varphi \]

where \( \in \{ , \ldots \} \), \( \sim \in \{ \leq, <, =, >, \geq \} \) and \( c \in \) .

Example

- \( A G \ \in \Rightarrow \ E F \leq 5 \)
- \( A F \ A G \leq 5 \)

**Definition**

The semantics of TCTL is defined as follows: let $\bullet$ be a location and $v$ be a clock valuation.

- $E \bullet U_{\sim c} \bullet$ iff there is a run from $\bullet, v$ such that $v \sim_c v'$.

We could also define a pointwise semantics:

$$v + c v' + c' \text{ delay} = c \text{ action delay} = c'$$
The semantics of TCTL is defined as follows: let $\bigcirc$ be a location and $v$ be a clock valuation.

- $\bigcirc, v \models E \bigcirc U_{\sim c} \bigcirc$ iff there is a run from $\bigcirc, v$ such that $v \sim c, v \models A \bigcirc U_{\sim c} \bigcirc$ is defined similarly.

We could also define a pointwise semantics:

- $v, v + c, v' + c'$ delay = $c$, action delay = $c'$
Branching-time logics with timing constraints – semantics

Example

\[ x \leq 2 \]
\[ y := 0 \]
\[ y \leq 2 \]
\[ x \geq 3 \]
\[ y \leq 2, x := 0 \]
\[ x \leq 3, y := 0 \]

\[ (x = 1.2, y = 0.4) \]
\[ | E U_{\geq 1} \]

\[ (x = 1.2, y = 0.4) \]
\[ | A G \neg x = 1 \]

\[ x := 0 \]
\[ y = 3 \]
\[ (x = 0, y = 0) \]
\[ | E (E F = 1) U = 3 \]
Branching-time logics with timing constraints – semantics

Example

\[ x \leq 2 \]
\[ y := 0 \]
\[ y \leq 2 \]
\[ x \geq 3 \]
\[ x \leq 2, x \leq 3, y := 0 \]
\[ (x = 1.2, y = 0.4) \]
\[ E \]
\[ U \geq 1 \]
\[ A \]
\[ G \]
\[ \neg x = 1 \]
\[ x := 0 \]
\[ x = 0 \]
\[ y = 3 \]
\[ (x = 0, y = 0) \]
\[ E \]
\[ E \]
\[ F \]
\[ 1 \]
\[ U \]
\[ 3 \]

\[ \bullet \quad \bigcirc, \left( \begin{array}{c} x \ 1.2 \\ y \ 0.4 \end{array} \right) | \ E \quad \bigcirc \quad U \geq 1 \quad \bigcirc \]

\[ \bullet \quad \bigcirc, \left( \begin{array}{c} x \ 1.2 \\ y \ 0.4 \end{array} \right) | \ A \quad \bigcirc \quad G \quad \neg \bigcirc \]

\[ \bullet \quad \bigcirc, \left( \begin{array}{c} x \ 0 \\ y \ 0 \end{array} \right) | \ E \quad \bigcirc \quad E \quad \bigcirc \quad F \quad 1 \quad \bigcirc \quad U \quad 3 \bigcirc \]
TCTL model checking

Lemma

Let $\bigcirc$ be a location and $\varphi$ be a TCTL formula. For any two valuations $v$ and $v'$ that belong to the same region,

$$
\bigcirc, v \models \varphi \iff \bigcirc, v' \models \varphi.
$$

TCTL model checking

Lemma

Let \( \circ \) be a location and \( \varphi \) be a TCTL formula. For any two valuations \( v \) and \( v' \) that belong to the same region,

\[
\circ, v \models \varphi \iff \circ, v' \models \varphi.
\]

Proof.

By induction on \( \varphi \).
TCTL model checking

Lemma

Let $\bigcirc$ be a location and $\varphi$ be a TCTL formula. For any two valuations $v$ and $v'$ that belong to the same region,

$$\bigcirc, v \models \varphi \iff \bigcirc, v' \models \varphi.$$  

Proof.

By induction on $\varphi$.

Theorem

TCTL model-checking is PSPACE-complete.

Lemma

Let be a location and be a TCTL formula. For any two valuations and that belong to the same region,

\[ \bigcirc, v \models \varphi \iff \bigcirc, v' \models \varphi. \]

Proof.

By induction on .

Theorem

TCTL model-checking is PSPACE-complete.

Proof.

Space-efficient CTL labelling algorithm on the region graph.

Outline of the talk

1 Introduction

2 Extending temporal logics with real-time constraints
   - Continuous and pointwise semantics
   - Expressiveness issues

3 Model checking timed linear-time logics
   - Undecidability of MTL and TPTL
   - Decidable fragments

4 Model checking timed branching-time logics

5 Conclusions and open problems
Conclusions and perspectives

Real-time temporal logics have been much studied:

- Linear-time: natural extensions of LTL are undecidable; several restrictions lead to decidability; however, model-checking linear-time logics is hard; no implementation exists.

- Branching-time: TCTL model-checking is in PSPACE; can be made efficient in practice; implemented in several tools (Uppaal, Kronos, ...)

Hot topics in real-time temporal logic model-checking: symbolic algorithms for linear-time temporal logics; robust model-checking.
Conclusions and perspectives

Real-time temporal logics have been much studied:

- **linear-time:**
  - natural extensions of LTL are *undecidable*;
  - several restrictions lead to *decidability*;
  - however, model-checking linear-time logics is *hard*;
  - no implementation exists.

- **branching-time:**
  - TCTL model-checking is in PSPACE;
  - can be made efficient in practice;
  - implemented in several tools (Uppaal, Kronos, ...)

Hot topics in real-time temporal logic model-checking:
- symbolic algorithms for linear-time temporal logics;
- robust model-checking.
Conclusions and perspectives

Real-time temporal logics have been much studied:

- **linear-time:**
  - natural extensions of LTL are undecidable;
  - several restrictions lead to decidability;
  - however, model-checking linear-time logics is hard;
  - no implementation exists.

- **branching-time:**
  - TCTL model-checking is in PSPACE;
  - can be made efficient in practice;
  - implemented in several tools (Uppaal, Kronos, ...)

Hot topics in real-time temporal logic model-checking:
- symbolic algorithms for linear-time temporal logics;
- robust model-checking.
Conclusions and perspectives

Real-time temporal logics have been much studied:

- **linear-time:**
  - natural extensions of LTL are undecidable;
  - several restrictions lead to decidability;
  - however, model-checking linear-time logics is hard;
  - no implementation exists.

- **branching-time:**
  - TCTL model-checking is in PSPACE;
  - can be made efficient in practice;
  - implemented in several tools (Uppaal, Kronos, ...)

**Hot topics** in real-time temporal logic model-checking:

- symbolic algorithms for linear-time temporal logics;
- robust model-checking.
UPPAAL 4.0.8
Engine & Formalism

Kim G. Larsen
Outline

- UPPAAL Models & Specifications
- UPPAAL Engine
  - Zones, CDDs
- UPPAAL Options
- LAB Exercises
UPPAAL

Modeling & Specification
Train Crossing

Stopable
Area

[10, 20]

[3, 5]

River

Gate

QMC, PhD School, March 3, 2010
Declarations

Constants
Bounded integers
Channels
Clocks
Arrays
Types
Functions
Templates
Processes
Systems

6QMC, PhD School, March 3, 2010
UPPAAL Help

UPPAAL is a tool for modeling, validation and verification of real-time systems. It is appropriate for systems that can be modeled as a collection of non-deterministic processes with finite control structure and real-valued clocks (i.e. timed automata), communicating through channels and (or) shared data structures. Typical application areas include real-time controllers, communication protocols, and other systems in which timing aspects are critical.

The UPPAAL tool consists of three main parts:

- a graphical user interface (GUI),
- a verification server, and
- a command line tool.

The GUI is used for modelling, simulation, and verification. For both simulation and verification, the GUI uses the verification server. In simulation, the server is used to compute successor states. The command line tool is a stand-alone verifier, appropriate for e.g. batch verifications.

More information can be found at the UPPAAL web site: http://www.uppaal.com.
Logical Specifications

- **Validation Properties**
  - Possibly: \( E <> P \)

- **Safety Properties**
  - Invariant: \( A[] P \)
  - Pos. Inv.: \( E[] P \)

- **Liveness Properties**
  - Eventually: \( A <> P \)
  - Leadsto: \( P \rightarrow Q \)

- **Bounded Liveness**
  - Leads to within: \( P \rightarrow_{\leq t} Q \)

The expressions \( P \) and \( Q \) must be type safe, side effect free, and evaluate to a boolean.

Only references to integer variables, constants, clocks, and locations are allowed (and arrays of these).
UPPAAL ENGINE
Regions – From Infinite to Finite

\[ y < 1, \ x := 0 \]

**Theorem**

The number of regions is \( n! \cdot 2^n \cdot \prod_{x \in C} (2c_x + 2) \).
A zone $Z$:

$1 \leq x \leq 2 \land$

$0 \leq y \leq 2 \land$

$x - y \geq 0$
Zones – Operations

- Delay
  - \((n, 2 \leq x \leq 4 \land 1 \leq y \leq 3 \land y-x \leq 0)\)
  - \((n, 2 \leq x \land 1 \leq y \land -3 \leq y-x \leq 0)\)
  - \((n, 2 \leq x \land 1 \leq y \leq 3 \land y-x \leq 0)\)

- Delay (stopwatch)
  - \((n, x=0 \land 1 \leq y \leq 3)\)

- Reset
  - \((n, 2 \leq x \leq 4 \land 1 \leq y)\)

- Extrapolation
  - \((n, 2 \leq x \leq 4 \land 1 \leq y)\)

- Convex Hull

QMC, PhD School, March 3, 2010
Kim Guldstrand Larsen [12]
Symbolic Exploration

Reachable?
Symbolic Exploration

\begin{itemize}
  \item $y := 0$
  \item $y \leq 2$
  \item $x := 0$
  \item $x \leq 2$
  \item $y \leq 2, \ x = 4$
  \item $L_0$
  \item $L_1$
\end{itemize}
Symbolic Exploration

L0

y:=0

x:=0

y<=2

x<=2

y<=2, x>=4

L1

Reachable?

Left

y

x
Symbolic Exploration

\[ y := 0 \]
\[ y <= 2 \]
\[ x := 0 \]
\[ x <= 2 \]
\[ y <= 2, x >= 4 \]

Reachable?

Left
Symbolic Exploration

\( y := 0 \)  
\( x := 0 \)  
\( y <= 2 \)  
\( x <= 2 \)  
\( y <= 2, x >= 4 \)  

Reachable?
Symbolic Exploration

y := 0
y <= 2
y <= 2, x >= 4

x := 0
x <= 2

L0
L1

Reachable?

Left
Symbolic Exploration

L0
y=0
y<=2
x:=0
x<=2

L1
y<=2, x>=4

Reachable?

Left
Symbolic Exploration

\[
\begin{align*}
y &= 0 \\
y &\leq 2 \\
x &\leq 2 \\
y &\leq 2, x \geq 4 \\
\end{align*}
\]

\begin{align*}
L_0 &\rightarrow & &x = 0 \\
L_0 &\rightarrow & &y = 0 \\
L_0 &\rightarrow & &x = 2 \\
L_0 &\rightarrow & &y = 2 \\
L_1 &\rightarrow & &x = 4 \\
\end{align*}

Reachable? 

Delay
Symbolic Exploration

L0

L1

y:=0
y<=2
y<=2, x>=4
x:=0
x<=2

Reachable?

Down
Datastructures for Zones

- Difference Bounded Matrices (DBMs)
- Minimal Constraint Form
  \([RTSS97]\)
- Clock Difference Diagrams
  \([CAV99]\)
Inclusion Checking (DBMs)

Inclusion

D1

\[
\begin{align*}
x & \leq 1 \\
y-x & \leq 2 \\
z-y & \leq 2 \\
z & \leq 9
\end{align*}
\]

Graph

\[
\begin{array}{ccc}
0 & \rightarrow & x \\
9 & \rightarrow & z \\
2 & \rightarrow & y
\end{array}
\]

Shortest Path Closure

\[
\begin{array}{ccc}
1 & \rightarrow & x \\
4 & \rightarrow & y \\
3 & \rightarrow & z
\end{array}
\]

D2

\[
\begin{align*}
x & \leq 2 \\
y-x & \leq 3 \\
y & \leq 3 \\
z-y & \leq 3 \\
z & \leq 7
\end{align*}
\]

Graph

\[
\begin{array}{ccc}
0 & \rightarrow & x \\
3 & \rightarrow & z \\
7 & \rightarrow & y
\end{array}
\]

Shortest Path Closure

\[
\begin{array}{ccc}
2 & \rightarrow & x \\
6 & \rightarrow & y \\
3 & \rightarrow & z
\end{array}
\]
Future (DBMs)

\[1 \leq x \leq 4\]
\[1 \leq y \leq 3\]

Shortest Path Closure

Remove upper bounds on clocks

QMC, PhD School, March 3, 2010
Kim Larsen [24]
Reset (DBMs)

\[ \begin{align*}
1 \leq x, & \quad 1 \leq y \\
-2 \leq x - y & \leq 3
\end{align*} \]

Remove all bounds involving \( y \) and set \( y \) to 0.

QMC, PhD School, March 3, 2010

Kim Larsen [25]
UPPAAL

Verification Options
Verification Options

Search Order
- Depth First
- Breadth First

State Space Reduction
- None
- Conservative
- Aggressive

State Space Representation
- DBM
- Compact Form
- Under Approximation
- Over Approximation

Diagnostic Trace
- Some
- Shortest
- Fastest

Extrapolation
Hash Table size
- Reuse
State Space Reduction

However, Passed list useful for efficiency

No Cycles: Passed list not needed for termination
State Space Reduction

Cycles:
Only symbolic states involving loop-entry points need to be saved on Passed list.
To Store or Not To Store

Behrmann, Larsen, Pelanek 2003

117 states\textsubscript{total} →
81 states\textsubscript{endpoint} →
9 states

Time OH less than 10%

Audio Protocol
Question:
\[ G \in R \quad ? \]

How to use:

\[ G \in O \quad ? \]
\[ G \in U \quad ? \]

Declared State Space

\[ G \in U \Rightarrow G \in R \]
\[ \neg (G \in O) \Rightarrow \neg (G \in R) \]
Over-approximation

Convex Hull

TACAS04: An EXACT method performing as well as Convex Hull has been developed based on abstractions taking max constants into account distinguishing between clocks, locations and $\leq$ & $\geq$
Under-approximation

Bitstate Hashing
Under-approximation

Bitstate Hashing

Passed = Bitarray

UPPAAL
4 - 512 Mbits

QMC, PhD School, March 3, 2010
LAB–Exercises

http://www.cs.aau.dk/~kgl/QMC2010/exercises/

Exercise 19
Exercise 2
Exercise 1
Timed games

Patricia Bouyer-Decitre

LSV, CNRS & ENS Cachan, France
Why (timed) games?

- to model uncertainty

Example of a processor in the taskgraph example
Why (timed) games?

- to model uncertainty

Example of a processor in the taskgraph example

![Diagram of a processor with states and transitions]

- \( x \leq 2 \)
- \( x \geq 1 \)
- \( x := 0 \)
Why (timed) games?

- to model uncertainty

Example of a processor in the taskgraph example

\[ (x \leq 2) \]
\[ + \]
\[ x \geq 1 \quad \text{done} \]
\[ \text{add} \]
\[ x \leftarrow 0 \]
\[ \text{idle} \]
\[ x \leftarrow 0 \]
\[ \times \]
\[ (x \leq 3) \]

- to model an interaction with an environment

Example of the gate in the train/gate example
Why (timed) games?

- to model uncertainty

Example of a processor in the taskgraph example

- to model an interaction with an environment

Example of the gate in the train/gate example
Why (timed) games?

- to model uncertainty

Example of a processor in the taskgraph example

- to model an interaction with an environment

Example of the gate in the train/gate example
An example of a timed game

Rule of the game

- **Aim:** avoid 😞 and reach 😊

How do we play? According to a strategy:

\[ f : \text{history} \mapsto \text{char} \rightarrow (\text{delay}, \text{cont. transition}) \]
An example of a timed game

Rule of the game

- **Aim:** avoid 😞 and reach 😊
- **How do we play?** According to a strategy:

```latex
\begin{align*}
\ell_0 & \xrightarrow{(x \leq 2)} \ell_1 \\
\ell_1 & \xrightarrow{x \geq 1, u_3} \ell_0 \\
\ell_1 & \xrightarrow{x \geq 2, c_4} \ell_1 \\
\ell_1 & \xrightarrow{x \leq 1, c_3} \ell_2 \\
\ell_2 & \xrightarrow{c_2} \ell_1 \\
\ell_2 & \xrightarrow{x < 1, u_1} \ell_1 \\
\ell_3 & \xrightarrow{x < 1, u_2, x := 0} \ell_2 \\
\end{align*}
```
An example of a timed game

Rule of the game

- **Aim:** avoid 🙁 and reach 🙂
- **How do we play?** According to a strategy:

  \[ f : \text{history} \mapsto \text{char} \rightarrow (\text{delay, cont. transition}) \]
An example of a timed game

Rule of the game

- **Aim:** avoid 😞 and reach 😊
- **How do we play?** According to a strategy:

  \[ f : \text{history} \mapsto \text{(delay, cont. transition)} \]

A (memoryless) winning strategy

- from \((\ell_0, 0)\), play \((0.5, c_1)\)
An example of a timed game

Rule of the game

- **Aim:** avoid 😞 and reach 😊
- **How do we play?** According to a strategy:
  
  \[ f : \text{history} \mapsto (\text{delay, cont. transition}) \]

A (memoryless) winning strategy

- from \((l_0, 0)\), play \((0.5, c_1)\)
  
  \[ \sim \] can be preempted by \(u_2\)
An example of a timed game

Rule of the game

- **Aim:** avoid 😞 and reach 😊
- **How do we play?** According to a strategy:

\[
\begin{align*}
&f : \text{history} \mapsto \text{(delay, cont. transition)}
\end{align*}
\]

A (memoryless) winning strategy

- from \((\ell_0, 0)\), play \((0.5, c_1)\)
  \(\leadsto\) can be preempted by \(u_2\)
- from \((\ell_2, \star)\), play \((1 - \star, c_2)\)
An example of a timed game

Rule of the game

- **Aim:** avoid 🙁 and reach 🙂
- **How do we play?** According to a strategy:

\[ f : \text{history} \mapsto (\text{delay, cont. transition}) \]

A (memoryless) winning strategy

- from \((\ell_0, 0), \) play \((0.5, c_1)\)
  - can be preempted by \(u_2\)
- from \((\ell_2, \star), \) play \((1 - \star, c_2)\)
- from \((\ell_3, 1), \) play \((0, c_3)\)
An example of a timed game

Rule of the game
- **Aim:** avoid 😞 and reach 😊
- **How do we play?** According to a strategy:

\[ f : \text{history} \mapsto (\text{delay, cont. transition}) \]

A (memoryless) winning strategy
- from \((\ell_0, 0)\), play \((0.5, c_1)\)
  \(\leadsto\) can be preempted by \(u_2\)
- from \((\ell_2, \ast)\), play \((1 - \ast, c_2)\)
- from \((\ell_3, 1)\), play \((0, c_3)\)
- from \((\ell_1, 1)\), play \((1, c_4)\)
An example of a timed game

Rule of the game
- **Aim**: avoid 😞 and reach 😊
- **How do we play?** According to a strategy:
  
  $f : \text{history} \mapsto (\text{delay, cont. transition})$

Problems to be considered
An example of a timed game

Rule of the game

- **Aim:** avoid 🙁 and reach 🙂
- **How do we play?** According to a strategy:

\[ f : \text{history} \mapsto \text{(delay, cont. transition)} \]

Problems to be considered

- Does there exist a winning strategy?
An example of a timed game

Rule of the game

- **Aim:** avoid 😞 and reach 😊
- **How do we play?** According to a strategy:
  
  \[ f : \text{history} \mapsto \text{char} \rightarrow (\text{delay, cont. transition}) \]

Problems to be considered

- Does there exist a winning strategy?
- If yes, compute one (as simple as possible).
Decidability of timed games

**Theorem [AMPS98,HK99]**

Reachability and safety timed games are decidable and \textsc{EXPTIME}-complete. Furthermore memoryless and “region-based” strategies are sufficient.

---


Decidability of timed games

Theorem [AMPS98,HK99]

Reachability and safety timed games are decidable and EXPTIME-complete. Furthermore memoryless and “region-based” strategies are sufficient.

\[ \leadsto \] classical regions are sufficient for solving such problems
Decidability of timed games

Theorem [AMPS98,HK99]

Reachability and safety timed games are decidable and EXPTIME-complete. Furthermore memoryless and “region-based” strategies are sufficient.

\[ \leadsto \text{classical regions are sufficient for solving such problems} \]

Theorem [AM99,BHPR07,JT07]

Optimal-time reachability timed games are decidable and EXPTIME-complete.

Back to the example: computing winning states

\( \ell_0 \) \( (x \leq 2) \) \( x \leq 1, c_1 \)

\( \ell_1 \) \( x < 1, u_2, x := 0 \) \( x < 1, u_1 \)

\( \ell_2 \) \( c_2 \)

\( \ell_3 \) \( x \leq 1, c_3 \)

\( \ell_0 \) \( x \geq 1, u_3 \)

\( \ell_1 \) \( x \geq 2, c_4 \)
Back to the example: computing winning states

\begin{align*}
\ell_0 \quad & (x \leq 2) \\
\ell_1 \quad & x \leq 1, c_1 \\
\ell_2 \quad & x < 1, u_2, x := 0 \\
\ell_3 \quad & x < 1, u_1, x \geq 2, c_3 \\
\end{align*}

Attrac

Winning states

Losing states
Back to the example: computing winning states
Back to the example: computing winning states

\( \ell_0 \)  
\( x \leq 2 \)

\( \ell_1 \)
\( x \leq 1, c_1 \)
\( x < 1, u_1 \)
\( x < 1, u_2, x := 0 \)
\( x \geq 2, c_4 \)

\( \ell_2 \)
\( c_2 \)

\( \ell_3 \)

\( \ell_0 \)
\( x \geq 1, u_3 \)

Winning states

Losing states

\( \ell_0 \)
\( 0 \quad 1 \quad 2 \quad 3 \)

\( \ell_1 \)
\( 0 \quad 1 \quad 2 \quad 3 \)
\( \ell_2 \)
\( 0 \quad 1 \quad 2 \quad 3 \)
\( \ell_3 \)
\( 0 \quad 1 \quad 2 \quad 3 \)
Back to the example: computing winning states

\[ (x \leq 2) \]

\[ x \leq 1, c_1 \]

\[ x < 1, u_1 \]

\[ x < 1, u_2, x := 0 \]

\[ x \geq 1, u_3 \]

\[ x \geq 2, c_4 \]

\[ c_2 \]

\[ c_3 \]

\[ c_4 \]

\[ x \geq 1, u_3 \]

\[ x < 1, u_1 \]

\[ c_2 \]

\[ x \leq 1, c_3 \]

\[ x \leq 1, c_1 \]

\[ x \leq 2 \]

\[ x \leq 1, c_1 \]

\[ x < 1, u_1 \]

\[ x < 1, u_2, x := 0 \]

\[ x \geq 1, u_3 \]

\[ x \geq 2, c_4 \]

\[ c_2 \]

\[ c_3 \]

\[ c_4 \]
Back to the example: computing winning states

\[ (x \leq 2) \] 

\[ x \geq 1, u_3 \] 

\[ x \leq 1, c_1 \] 

\[ x < 1, u_1 \] 

\[ x < 1, u_2, x := 0 \] 

\[ x \geq 2, c_4 \] 

\[ c_2 \] 

\[ x \leq 1, c_3 \] 

\[ \ell_0 \] 

\[ \ell_1 \] 

\[ \ell_2 \] 

\[ \ell_3 \] 

Winning states Losing states

\[ \ell_0 \] 

\[ \ell_1 \] 

\[ \ell_2 \] 

\[ \ell_3 \]
Back to the example: computing winning states

\[ \ell_0 \] \( (x \leq 2) \)

\[ \ell_1 \] \( x\leq 1, c_1 \)

\[ \ell_2 \] \( x<1, u_2, x:=0 \)

\[ \ell_3 \] \( c_2 \)

\[ \ell_0 \] \( x\geq 1, u_3 \)

\[ \ell_1 \] \( x\geq 2, c_4 \)

\[ \ell_2 \] \( x\leq 1, c_3 \)
Back to the example: computing winning states

\( \ell_0 \) (\( x \leq 2 \))

\( x \geq 1, u_3 \)

\( x \leq 1, c_1 \)

\( x < 1, u_2, x := 0 \)

\( x < 1, u_1 \)

\( c_2 \)

\( \ell_1 \)

\( x \geq 2, c_4 \)

\( x \leq 1, c_3 \)

\( \ell_2 \)

\( \ell_3 \)

Winning states

Losing states

\( \ell_0 \)

0 1 2 3

\( \ell_1 \)

0 1 2 3

\( \ell_2 \)

0 1 2 3

\( \ell_3 \)

0 1 2 3
Decidability via attractors

\[
\text{Pred}_a(X) = \{ \cdot \mid \cdot \cdot a \rightarrow \cdot \in X \}
\]

controllable and uncontrollable discrete predecessors:

\[
\text{cPred}(X) = \text{Pred}_\text{cont.}(X)
\]

\[
\text{uPred}(X) = \text{Pred}_\text{uncont.}(X)
\]

time controllable predecessors:

\[
\text{Pred}_t(u)(X) = \{ \cdot \mid \exists t \geq 0, \cdot \cdot u(t) \rightarrow \cdot \text{ and } \forall 0 \leq t' \leq t, \cdot \cdot u(t') \rightarrow \cdot \in \text{Safe} \}
\]
Decidability via attractors

\[ \text{Pred}^a(X) = \{ \bullet \mid \bullet \xrightarrow{a} \bullet \in X \} \]
Decidability via attractors

$$\bullet \text{Pred}^a(X) = \{ \bullet | \bullet \xrightarrow{a} \bullet \in X \}$$

$$\bullet \text{controllable and uncontrollable discrete predecessors:}$$

$$\text{cPred}(X) = \bigcup_{a \text{ cont.}} \text{Pred}^a(X)$$

$$\text{uPred}(X) = \bigcup_{a \text{ uncont.}} \text{Pred}^a(X)$$

$$\text{uPred}(X) = \bigcup_{a \text{ uncont.}} \text{Pred}^a(X)$$
Decidability via attractors

- $\text{Pred}^a(X) = \{ \bullet | \bullet \xrightarrow{a} \bullet \in X \}$

- controllable and uncontrollable discrete predecessors:

  $$\text{cPred}(X) = \bigcup_{a \text{ cont.}} \text{Pred}^a(X) \quad \text{and} \quad \text{uPred}(X) = \bigcup_{a \text{ uncont.}} \text{Pred}^a(X)$$

- time controllable predecessors:

  delay $t$ t.u.

  should be safe
Decidability via attractors

- \( \text{Pred}^a(X) = \{ \bullet | \bullet \xrightarrow{a} \bullet \in X \} \)

- controllable and uncontrollable discrete predecessors:

\[
\text{cPred}(X) = \bigcup_{a \text{ cont.}} \text{Pred}^a(X) \quad \text{and} \quad \text{uPred}(X) = \bigcup_{a \text{ uncont.}} \text{Pred}^a(X)
\]

- time controllable predecessors:

\[
\text{Pred}_{\delta}(X, \text{Safe}) = \{ \bullet | \exists t \geq 0, \bullet \xrightarrow{\delta(t)} \bullet \text{ and } \forall 0 \leq t' \leq t, \bullet \xrightarrow{\delta(t')} \bullet \in \text{Safe} \}
\]
Timed games with a reachability objective

We write:

\[ \pi(X) = X \cup \text{Pred}_\delta(\text{cPred}(X), \neg \text{uPred}(\neg X)) \]
Timed games with a reachability objective

We write:
\[
\pi(X) = X \cup \text{Pred}_\delta(c\text{Pred}(X), \neg u\text{Pred}(\neg X))
\]

- The states from which one can ensure 😊 in no more than 1 step is:

\[
\text{Attr}_1(😊) = \pi(😊)
\]
Timed games with a reachability objective

We write:

$$\pi(X) = X \cup \text{Pred}_\delta(c\text{Pred}(X), \neg u\text{Pred}(\neg X))$$

- The states from which one can ensure 😊 in no more than 1 step is:

  $$\text{Attr}_1(😊) = \pi(😊)$$

- The states from which one can ensure 😊 in no more than 2 steps is:

  $$\text{Attr}_2(😊) = \pi(\text{Attr}_1(😊))$$
Timed games with a reachability objective

We write:

\[ \pi(X) = X \cup \text{Pred}_\delta(c\text{Pred}(X), \neg u\text{Pred}(\neg X)) \]

- The states from which one can ensure \( \smiley \) in no more than 1 step is:

\[ \text{Attr}_1(\smiley) = \pi(\smiley) \]

- The states from which one can ensure \( \smiley \) in no more than 2 steps is:

\[ \text{Attr}_2(\smiley) = \pi(\text{Attr}_1(\smiley)) \]

- \( \ldots \)
Timed games with a reachability objective

We write:

\[ \pi(X) = X \cup \text{Pred}_\delta(\text{cPred}(X), \neg \text{uPred}(\neg X)) \]

- The states from which one can ensure 😊 in no more than 1 step is:
  \[ \text{Attr}_1(😊) = \pi(😊) \]

- The states from which one can ensure 😊 in no more than 2 steps is:
  \[ \text{Attr}_2(😊) = \pi(\text{Attr}_1(😊)) \]

- ... 

- The states from which one can ensure 😊 in no more than \( n \) steps is:
  \[ \text{Attr}_n(😊) = \pi(\text{Attr}_{n-1}(😊)) \]
Timed games with a reachability objective

We write:

\[ \pi(X) = X \cup \text{Pred}_\delta(c\text{Pred}(X), \neg \text{uPred}(\neg X)) \]

- The states from which one can ensure \( \smiley \) in no more than 1 step is:

  \[ \text{Attr}_1(\smiley) = \pi(\smiley) \]

- The states from which one can ensure \( \smiley \) in no more than 2 steps is:

  \[ \text{Attr}_2(\smiley) = \pi(\text{Attr}_1(\smiley)) \]

- \( \ldots \)

- The states from which one can ensure \( \smiley \) in no more than \( n \) steps is:

  \[ \text{Attr}_n(\smiley) = \pi(\text{Attr}_{n-1}(\smiley)) = \pi^n(\smiley) \]
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $c\text{Pred}(X)$ and $u\text{Pred}(X)$.
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.
- Does $\pi$ also preserve unions of regions?
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.

- Does $\pi$ also preserve unions of regions?
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.
- Does $\pi$ also preserve unions of regions?
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.
- Does $\pi$ also preserve unions of regions?

$\text{cPred}(X)$
$\text{uPred}(\neg X)$
$\text{Pred}_\delta(\text{cPred}(X), \neg \text{uPred}(\neg X))$
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.

- Does $\pi$ also preserve unions of regions? Yes!

... and is correct

\[
\text{cPred}(X) \quad \text{uPred}(¬X) \\
\text{Pred}_δ(\text{cPred}(X), ¬\text{uPred}(¬X))
\]
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.
- Does $\pi$ also preserve unions of regions? Yes!

(but it does not preserve zones...)

cPred($X$)
uPred($\neg X$)
$\text{Pred}_\delta(\text{cPred}(X), \neg \text{uPred}(\neg X))$
Stability w.r.t. regions

- if $X$ is a union of regions, then:
  - $\text{Pred}_a(X)$ is a union of regions,
  - and so are $\text{cPred}(X)$ and $\text{uPred}(X)$.
- Does $\pi$ also preserve unions of regions? Yes!

\[ \text{cPred}(X), \text{uPred}(\neg X), \text{Pred}_\delta(\text{cPred}(X), \neg \text{uPred}(\neg X)) \]

(but it does not preserve zones...)

$\leadsto$ the computation of $\pi^*(\text{😊})$ terminates!
Stability w.r.t. regions

- if \( X \) is a union of regions, then:
  - \( \text{Pred}_a(X) \) is a union of regions,
  - and so are \( \text{cPred}(X) \) and \( \text{uPred}(X) \).
- Does \( \pi \) also preserve unions of regions? Yes!

\[ \leadsto \text{the computation of } \pi^* \text{ (😊) terminates!} \]

... and is correct
Timed games with a safety objective

- We can use operator $\tilde{\pi}$ defined by

\[
\tilde{\pi}(X) = \operatorname{Pred}_\delta(X \cap c\operatorname{Pred}(X), \neg u\operatorname{Pred}(\neg X))
\]

instead of $\pi$, and compute $\tilde{\pi}^*(\neg \frownie)$
Timed games with a safety objective

- We can use operator $\tilde{\pi}$ defined by

$$
\tilde{\pi}(X) = \text{Pred}_\delta(X \cap c\text{Pred}(X), \neg u\text{Pred}(\neg X))
$$

instead of $\pi$, and compute $\tilde{\pi}^*(\neg \frownie)$

- It is also stable w.r.t. regions.
Some remarks

Control games

Our games are control games,
Some remarks

Control games

Our games are control games, and in particular they:
- are asymmetric
  - the environment can preempt any decision of the controller
  - we take the point-of-view of the controller
Some remarks

Control games

Our games are control games, and in particular they:
- are asymmetric
  - the environment can preempt any decision of the controller
  - we take the point-of-view of the controller
- are neither concurrent nor turn-based
Some remarks

Control games

Our games are control games, and in particular they:

- are asymmetric
  - the environment can preempt any decision of the controller
  - we take the point-of-view of the controller
- are neither concurrent nor turn-based
- do not take into account Zenoness considerations
Some remarks

Control games

Our games are control games, and in particular they:

- are asymmetric
  - the environment can preempt any decision of the controller
  - we take the point-of-view of the controller
- are neither concurrent nor turn-based
- do not take into account Zenoness considerations
  \[ \leadsto \text{can be done adding a Büchi winning condition} \]
Some remarks

Control games

Our games are control games, and in particular they:

- are asymmetric
  - the environment can preempt any decision of the controller
  - we take the point-of-view of the controller
- are neither concurrent nor turn-based
- do not take into account Zenoness considerations
  - can be done adding a Büchi winning condition

Alternative models [AFH+03,BLMO07]

- concurrent and symmetric games
- some incorporate non-Zenoness in the winning condition

[AFH+03] de Alfaro, Faella, Henzinger, Majumdar, Stoelinga.
[BLMO07] Brihaye, Laroussinie, Markey, Oreiby. Timed Concurrent Game Structures (CONCUR’07).
Some remarks

Control games

Our games are control games, and in particular they:

- are asymmetric
  - the environment can preempt any decision of the controller
  - we take the point-of-view of the controller
- are neither concurrent nor turn-based
- do not take into account Zenoness considerations
  \[\leadsto\] can be done adding a Büchi winning condition

Alternative models [AFH+03,BLMO07]

- concurrent and symmetric games
- some incorporate non-Zenoness in the winning condition
  \[\leadsto\] those games are not determined 😞
Some remarks

Control games

Our games are control games, and in particular they:

- are asymmetric
  - the environment can preempt any decision of the controller
  - we take the point-of-view of the controller
- are neither concurrent nor turn-based
- do not take into account Zenoness considerations
  \[ \leadsto \text{can be done adding a Büchi winning condition} \]

Alternative models [AFH+03,BLMO07]

- concurrent and symmetric games
- some incorporate non-Zenoness in the winning condition
  \[ \leadsto \text{those games are not determined} \]
  ... and they may not represent a proper interaction with an environment

[AFH+03] de Alfaro, Faella, Henzinger, Majumdar, Stoelinga.
[BLMO07] Brihaye, Laroussinie, Markey, Oreiby. Timed Concurrent Game Structures (CONCUR’07).
Application of timed games to strong timed bisimulation

This is a relation between ◦ and ◦ such that:

∀ d > 0 ∃ /u1D6FF (d) ...

... and vice-versa (swap ◦ and ◦) for the bisimulation relation.

Theorem

Strong timed (bi)simulation between timed automata is decidable and EXPTIME-complete.
Application of timed games to strong timed bisimulation

This is a relation between ⊗ and ⊘ such that:

\[ \forall d > 0 \exists \text{...} \]

... and vice-versa (swap ⊗ and ⊘) for the bisimulation relation.

Theorem

Strong timed (bi)simulation between timed automata is decidable and
EXPTIME-complete.
Application of timed games to strong timed bisimulation

This is a relation between $\bullet$ and $\bullet$ such that:

\[ \forall a \exists \quad \exists a \forall \]

... and vice-versa (swap $\bullet$ and $\bullet$) for the bisimulation relation.

Theorem
Strong timed (bi)simulation between timed automata is decidable and
EXPTIME-complete.
Application of timed games to strong timed bisimulation

This is a relation between $\bullet$ and $\bullet$ such that:

\[ a \quad \forall \quad \exists \quad \delta(d) \quad \forall d > 0 \]

Theorem
Strong timed (bi)simulation between timed automata is decidable and EXPTIME-complete.
Application of timed games to strong timed bisimulation

This is a relation between $\bullet$ and $\bullet$ such that:

\[ \forall \exists a / \varnothing (d) \]
\[ \forall d > 0 \exists \delta(d) / \varnothing (d) \]

Theorem

Strong timed (bi)simulation between timed automata is decidable and EXPTIME-complete.
Application of timed games to strong timed bisimulation

This is a relation between $\bullet$ and $\bullet$ such that:

$\forall \quad a \quad \exists$  

$\forall d > 0 \quad \exists \quad \delta(d)$

$\exists \quad a$  

$\exists \quad \delta(d)$

... and vice-versa (swap $\bullet$ and $\bullet$) for the bisimulation relation.
Application of timed games to strong timed bisimulation

This is a relation between $\bullet$ and $\bullet$ such that:

$\forall a \exists (d)$

$\forall d > 0 \exists \delta(d)$

... and vice-versa (swap $\bullet$ and $\bullet$) for the bisimulation relation.

Theorem

Strong timed (bi)simulation between timed automata is decidable and EXPTIME-complete.
timed automaton $A$

g_1,a,Y_1 := 0

g_2,a,Y_2 := 0

$A$

$B$

g,a,Y := 0
timed automaton $\mathcal{A}$

$\vdash p, q$

tester

$g_1, a, Y_1 := 0$
$g_2, a, Y_2 := 0$

$\vdash p, p'_1$
$\vdash p', p'_2$

... $\vdash q, q'$

timed automaton $\mathcal{B}$

$g, a, Y := 0$

$\vdash p', q'$
timed automaton $\mathcal{A}$

$\mathcal{A}$

$\mathcal{B}$

$\mathcal{A}$

$\mathcal{B}$
Let $g_1, a, Y_1 := 0$, $g_2, a, Y_2 := 0$.

**Timed Automaton $A$**

- **State $p$**
  - Transition: $g_1, a, Y_1 := 0$ to $p_1$
  - Transition: $g_2, a, Y_2 := 0$ to $p_2$

**Timed Automaton $B$**

- **State $q$**
  - Transition: $g, a, Y := 0$ to $q'$

**Tester**

- States: $p, q$
  - Transition: $g_1, a, Y_1 := 0, z := 0$
  - Transition: $g, a, Y := 0, z := 0$
  - Transition: $g_2, a, Y_2 := 0, z := 0$

**Prover**

- States: $p_1, q'$, $p_2, q'$
  - Transition: $g_1 \land (z=0), a, Y$
  - Transition: $g_2 \land (z=0), a, Y$

**Logical Expressions**

- $g_1 \land (z=0), a, Y$
- $g_2 \land (z=0), a, Y$
\[ g_1, a, Y_1 := 0 \]
\[ g_2, a, Y_2 := 0 \]

\[ p \]
\[ p' \]
\[ p'' \]

\[ g_1, a, Y_1 := 0, z := 0 \]
\[ g_2, a, Y_2 := 0, z := 0 \]

\[ p, q \]
\[ g, a, Y := 0, z := 0 \]

\[ p', q' \]

\[ g \land (z = 0), a, Y \]

\[ g_1 \land (z = 0), a, Y_1 \]
\[ g_2 \land (z = 0), a, Y_2 \]

\[ g_1, a, Y_1 := 0 \]
\[ g_2, a, Y_2 := 0 \]

\[ p, q \]
\[ g_1, a, Y := 0 \]
\[ g_2, a, Y := 0 \]

\[ q \]
\[ q' \]

\[ g, a, Y := 0 \]

\[ \neg g, a \]

\[ (z = 0) \land \neg g, a \]
\( \mathcal{A} \) and \( \mathcal{B} \) are strongly timed bisimilar iff the prover \( \bigcirc \) has a winning strategy to avoid \( \frownie \).
What else?

- **Implementation**: Uppaal-Tiga implements a forward algorithm to compute winning states and winning strategies \([\text{CDF+05, BCD+07}]\)

\[\text{CDF+05}\] Cassez, David, Fleury, Larsen, Lime. Efficient on-the-fly algorithms for the analysis of timed games \((\text{CONCUR’05})\).

\[\text{BCD+07}\] Berhmann, Cougnard, David, Fleury, Larsen, Lime. Uppaal-Tiga: Time for playing games! \((\text{CAV’07})\).
What else?

- **Implementation**: Uppaal-Tiga implements a forward algorithm to compute winning states and winning strategies [CDF+05, BCD+07]
- A climate controller in a pig stable (Skov A/S) [JRLD07]

What else?

- **Implementation:** Uppaal-Tiga implements a forward algorithm to compute winning states and winning strategies [CDF+05, BCD+07]
  - A climate controller in a pig stable (Skov A/S) [JRLD07]
  - A pump controller (Hydac Gmbh) [CJL+09]

What else?

- **Implementation:** Uppaal-Tiga implements a forward algorithm to compute winning states and winning strategies [CDF+05, BCD+07]
  - A climate controller in a pig stable (Skov A/S) [JRLD07]
  - A pump controller (Hydac Gmbh) [CJL+09]

- **Partial observation/Incomplete information:**
  - action-based observation: **undecidable** [BDMP03]
  - finite-observation of states: **decidable** [CDL+07]

---

[CDF+05] Cassez, David, Fleury, Larsen, Lime. Efficient on-the-fly algorithms for the analysis of timed games *(CONCUR’05).*


[CJL+09] Cassez, Jessen, Larsen, Raskin, Reynier. Automatic Synthesis of Robust and Optimal Controllers – An Industrial Case Study *(HSCC’09).*

[BDMP03] Bouyer, D’Souza, Madhusudan, Petit. Timed control with partial observability *(CAV’03).*

[CDL+07] Cassez, David, Larsen, Lime, Raskin. Timed control with observation based and stuttering invariant strategies *(ATVA’07).*
What else?

- **Implementation:** Uppaal-Tiga implements a forward algorithm to compute winning states and winning strategies [CDF+05, BCD+07]
  - A climate controller in a pig stable (Skov A/S) [JRLD07]
  - A pump controller (Hydac Gmbh) [CJL+09]

- **Partial observation/Incomplete information:**
  - action-based observation: undecidable [BDMP03]
  - finite-observation of states: decidable [CDL+07]

- **Quantitative constraints**, see the next lecture!

[CDF+05] Cassez, David, Fleury, Larsen, Lime. Efficient on-the-fly algorithms for the analysis of timed games (*CONCUR’05*).
[CJL+09] Cassez, Jessen, Larsen, Raskin, Reynier. Automatic Synthesis of Robust and Optimal Controllers – An Industrial Case Study (*HSCC’09*).
[BDMP03] Bouyer, D’Souza, Madhusudan, Petit. Timed control with partial observability (*CAV’03*).
Real-time Model Checking
— Priced timed automata —

Nicolas MARKEY

Lav. Spécification & Vérification
CNRS & ENS Cachan – France

March 3, 2010
Time is not always sufficient

Timed automata are (rather) well understood – Can we go further?
Time is not always sufficient

Timed automata are (rather) well understood – Can we go further?

Compute $D \times C \times A \quad B \quad A \quad B \quad C \times D$ using two processors:

$P_1$ (fast):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 picosec.</td>
</tr>
<tr>
<td>$\times$</td>
<td>3 picosec.</td>
</tr>
</tbody>
</table>

$P_2$ (slow):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5 picosec.</td>
</tr>
<tr>
<td>$\times$</td>
<td>7 picosec.</td>
</tr>
</tbody>
</table>
Time is not always sufficient

Timed automata are (rather) well understood – Can we go further?

Compute $D \times C \times A \times B + A + B + C \times D$ using two processors:

$P_1$ (fast):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>idle</td>
<td>10 W</td>
</tr>
<tr>
<td>in use</td>
<td>90 W</td>
</tr>
</tbody>
</table>

2 picosec.

$P_2$ (slow):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>idle</td>
<td>20 W</td>
</tr>
<tr>
<td>in use</td>
<td>30 W</td>
</tr>
</tbody>
</table>

5 picosec.

$P_1$ (fast):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>idle</td>
<td>10 W</td>
</tr>
<tr>
<td>in use</td>
<td>90 W</td>
</tr>
</tbody>
</table>

3 picosec.

$P_2$ (slow):

<table>
<thead>
<tr>
<th></th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>idle</td>
<td>20 W</td>
</tr>
<tr>
<td>in use</td>
<td>30 W</td>
</tr>
</tbody>
</table>

7 picosec.
Time is not always sufficient

Timed automata are (rather) well understood – Can we go further?

Compute $D \times C \times A \times B + A + B + C \times D$ using two processors:

**$P_1$ (fast):**

<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2 picosec.</td>
<td></td>
</tr>
<tr>
<td>$\times$</td>
<td>3 picosec.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>energy</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>idle</td>
<td>10 W</td>
</tr>
<tr>
<td></td>
<td>in use</td>
<td>90 W</td>
</tr>
</tbody>
</table>

**$P_2$ (slow):**

<table>
<thead>
<tr>
<th></th>
<th>time</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5 picosec.</td>
<td></td>
</tr>
<tr>
<td>$\times$</td>
<td>7 picosec.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>energy</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>idle</td>
<td>20 W</td>
</tr>
<tr>
<td></td>
<td>in use</td>
<td>30 W</td>
</tr>
</tbody>
</table>

**Diagram:**

- $P_1$ (fast):
  - $T_2$
  - $T_3$
  - $T_5$
  - $T_6$

- $P_2$ (slow):
  - $T_1$
  - $T_4$

- Timings:
  - $T_1$ to $T_6$: 13 picoseconds
  - $T_1$ to $T_6$: 1.37 nanojoules
Time is not always sufficient

Timed automata are (rather) well understood – Can we go further?

Compute $D \times C \times A \times B \times A \times B \times C \times D$ using two processors:

$P_1$ (fast):

<table>
<thead>
<tr>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 picosec.</td>
</tr>
<tr>
<td>× 3 picosec.</td>
</tr>
</tbody>
</table>

energy

<table>
<thead>
<tr>
<th>idle</th>
<th>in use</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 W</td>
<td>90 W</td>
</tr>
</tbody>
</table>

$P_2$ (slow):

<table>
<thead>
<tr>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 picosec.</td>
</tr>
<tr>
<td>× 7 picosec.</td>
</tr>
</tbody>
</table>

energy

<table>
<thead>
<tr>
<th>idle</th>
<th>in use</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 W</td>
<td>30 W</td>
</tr>
</tbody>
</table>

12 picoseconds $\times 1.39$ nanojoules
Time is not always sufficient

Timed automata are (rather) well understood – Can we go further?

Compute $D \times C \times A \times B$ $A \times B$ $C \times D$ using two processors:

$P_1$ (fast):

<table>
<thead>
<tr>
<th>time</th>
<th>2 picosec.</th>
<th>× 3 picosec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>energy</td>
<td>idle 10 W</td>
<td>in use 90 W</td>
</tr>
</tbody>
</table>

$P_2$ (slow):

<table>
<thead>
<tr>
<th>time</th>
<th>5 picosec.</th>
<th>× 7 picosec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>energy</td>
<td>idle 20 W</td>
<td>in use 30 W</td>
</tr>
</tbody>
</table>

0 5 10 15 20 25

$P_1$ $P_2$

$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$

19 picoseconds
1.32 nanojoules
Time is not always sufficient

- **hybrid automata**: timed automata augmented with variables whose derivative is not constant.

⇒ examples: leaking gas burner, water-level monitor, ...

\[
\begin{align*}
    x &\leq 1 \\
x &\geq 30, x &\geq 0 \\
y &\leq 1 \\
z &\geq 1
\end{align*}
\]

\[
\begin{align*}
    x &\leq 1, x &\leq 0 \\
x &\geq 30, x &\geq 0
\end{align*}
\]

Theorem

*Reachability is undecidable (even for timed automata with one stopwatch).*

Time is not always sufficient

- **hybrid automata**: timed automata augmented with variables whose derivative is not constant.

→ examples: leaking gas burner, water-level monitor, ...

\[
\begin{align*}
x &\leq 1 \\
x &\leq 1, x &\leq 0 \\
x &\geq 30, x &\leq 0 \\
true \\
x &1 \\
y &1 \\
z &0
\end{align*}
\]

- **timed automata with observers**: similar to hybrid automata, but the behavior only depends on clock variables.

Outline of the talk

1. Introduction

2. Timed automata with observers

3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies

4. Resource-management problems

5. Conclusions and perspectives
Outline of the talk

1. Introduction
2. Timed automata with observers
3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies
4. Resource-management problems
5. Conclusions and perspectives
Timed automata with (linear) observers

Example

\[ x = 1 \]
\[ x := 0 \]

Timed automata with (linear) observers

Example

\[ \begin{align*}
-3 + 6 & \quad \rightarrow \\
-6 - 1 & \quad \rightarrow \\
-6 + 2 & \quad \rightarrow \\
\end{align*} \]

\[ \begin{align*}
x & = 1 \\
\end{align*} \]

\[ \begin{align*}
x & := 0 \\
\end{align*} \]

\[ \begin{align*}
-3 - 3 + 6 & + 6 - 6 - 6 - 6 - 6 \quad \cdots \quad \text{in Weighted Timed Automata (2001).}
\end{align*} \]


Timed automata with (linear) observers

Example

\[-3 + 6 \quad -6 + 2 \quad -1\]

\[x = 1 \quad x := 0\]

\[\text{Ref: [1] Alur, La Torre, Pappas. } \textit{Optimal Paths in Weighted Timed Automata} \text{ (2001).}\]

\[\text{[2] Behrmann et al. } \textit{Minimum-cost reachability for priced timed automata} \text{ (2001).}\]
Timed automata with (linear) observers

Example

\[ -3 + 6 - 6 + 2 - 1 \]

\[ x = 1 \]

\[ x := 0 \]

\[ -3 - 3 + 6 + 6 - 6 - 6 + 2 \]

Timed automata with (linear) observers

Example

\[ x = 1 \]


Timed automata with (linear) observers

Example

\[ x = 0 \]

\[ x = 1 \]


Timed automata with (linear) observers

Example

\[-3 + 6 - 6 + 2 - 1 \]

\[x = 1\]

\[x := 0\]

Timed automata with (linear) observers

Example

\[ x = 1 \]

Timed automata with (linear) observers

Example

\[ x = 1 \]


Timed automata with (linear) observers

Example

\[ x = 1 \]


Outline of the talk

1. Introduction

2. Timed automata with observers

3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies

4. Resource-management problems

5. Conclusions and perspectives
Outline of the talk

1. Introduction
2. Timed automata with observers
3. Resource-optimization problems
   - Optimal reachabililty
   - Weighted temporal logics
   - Optimal strategies
4. Resource-management problems
5. Conclusions and perspectives
Optimal reachability

Example

\[ \dot{p} = 5 \quad y = 0 \]
\[ \dot{p} = 7 \]
\[ \dot{p} = 5 \]

\( x \leq 2 \)
\( y : = 0 \)
\( x \geq 3 \)

Minimal cost for reaching \( \smiley \):

\[
\inf_{0 \leq t \leq 2} \min \left( 5t + 7(3 - t) + 1, 5t + 5(3 - t) + 4 \right) = 18
\]

The optimal schedule consists in waiting 2 time units in ; going through .
Optimal reachability

Example

Minimal cost for reaching 😊:

\[
\inf_{0 \leq t \leq 2} \min (5t + 7(3-t) + 1 + 5t + 5(3-t) + 4) = 18
\]

The optimal schedule consists in waiting 2 time units in ; going through.
Optimal reachability

Example

\[ \dot{p} = 5 \quad y = 0 \]
\[ \dot{p} = 7 \]
\[ \dot{p} = 5 \]

\( x \leq 2 \)
\( y = 0 \)
\( x \geq 3 \)

Minimal cost for reaching 😊:

\[
inf_0 \leq t \leq 2 \min (5t + 7(3 - t) + 1, 5t + 5(3 - t) + 4) = 18
\]

The optimal schedule consists in waiting 2 time units and going through.
Optimal reachability

Example

\[ \dot{p} = 5 \quad y = 0 \]
\[ \dot{p} = 7 \]
\[ \dot{p} = 5 \]

\[ x \leq 2 \]
\[ y := 0 \]
\[ x \geq 3 \]

Minimal cost for reaching 😊:

\[
\begin{align*}
5t & 7 & 3 - t & 1 \\
5t & 5 & 3 - t & 4
\end{align*}
\]
Optimal reachability

Example

\[
\begin{align*}
p &= 5 \\
y &= 0 \\
p &= 7 \\
p &= 5
\end{align*}
\]

\[
x \leq 2 \\
y := 0 \\
x \geq 3
\]

Minimal cost for reaching 😊:

\[
\begin{align*}
\min \begin{pmatrix} 5t & 7 & 3 - t & 1 \\ 5t & 5 & 3 - t & 4 \end{pmatrix}
\end{align*}
\]
Optimal reachability

Example

\[ \dot{p} = 5 \quad y = 0 \]
\[ \dot{p} = 7 \]
\[ \dot{p} = 5 \]

\[ x \leq 2 \]
\[ y := 0 \]
\[ x \geq 3 \]
\[ p + = 1 \]
\[ p + = 4 \]
\[ x \geq 3 \]

Minimal cost for reaching 😊:

\[ \inf_{0 \leq t \leq 2} \min \begin{pmatrix} 5t & 7 & 3 - t & 1 \\ 5t & 5 & 3 - t & 4 \end{pmatrix} = 18 \]

The optimal schedule consists in waiting 2 time units in; going through.
Optimal reachability

Example

\[ \dot{p} = 5 \quad y = 0 \]
\[ \dot{p} = 7 \]
\[ \dot{p} = 5 \]

\[ x \leq 2 \]
\[ y := 0 \]
\[ x \geq 3 \]

Minimal cost for reaching \( \smiley \):

\[
\inf_{0 \leq t \leq 2} \min \begin{pmatrix}
5t & 7 & 3 - t & 1 \\
5t & 5 & 3 - t & 4
\end{pmatrix}
\]

\( = 18 \)

The optimal schedule consists in waiting 2 time units in; going through.
Optimal reachability

Example

\[ \dot{p} = 5 \]
\[ y = 0 \]
\[ \dot{p} = 7 \]
\[ \dot{p} = 5 \]

\[ x \leq 2 \]
\[ x \geq 3 \]
\[ y = 0 \]
\[ p+1 \]
\[ p+4 \]

Minimal cost for reaching \( \smiley \):

\[
\inf_{0 \leq t \leq 2} \min \begin{pmatrix}
5t & 7 & 3 - t & 1 \\
5t & 5 & 3 - t & 4
\end{pmatrix} = 18
\]

The optimal schedule consists in:

- waiting 2 time units in \( \bigcirc \);
- going through \( \bigcirc \).
Optimal reachability

**Theorem**

*Optimal reachability in priced timed automata is PSPACE-complete.*
Optimal reachability

Theorem

Optimal reachability in priced timed automata is PSPACE-complete.

Proof.

- The region abstraction is not fine enough:

Ref:
Optimal reachability

Theorem

*Optimal reachability in priced timed automata is PSPACE-complete.*

Proof.

- The idea is: “take transitions *close to integer dates*”;

Optimal reachability

Theorem

Optimal reachability in priced timed automata is PSPACE-complete.

Proof.

- The idea is: “take transitions close to integer dates”;
- Corner-point abstraction: only consider corners of regions:

![Diagram]

Optimal reachability

Theorem

**Optimal reachability in priced timed automata is PSPACE-complete.**

Proof.

- The idea is: “take transitions close to integer dates”;
- Corner-point abstraction: only consider corners of regions:

Optimal reachability

Theorem

*Optimal reachability in priced timed automata is PSPACE-complete.*

Proof.

- The idea is: “take transitions *close to integer dates*”;
- Corner-point abstraction: only consider *corners* of regions:

Outline of the talk

1. Introduction
2. Timed automata with observers
3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies
4. Resource-management problems
5. Conclusions and perspectives
Weighted temporal logic

Example

Decorate temporal modalities with constraints on cost:

\[ G(failure ⇒ F ≤ 250 \text{ repaired}) \]

\[ A G(failure ⇒ E F \text{ time} ≤ 5 (repair ∧ A F \text{ cost} ≤ 150 \text{ running})) \]
Weighted temporal logic

Example

Decorate temporal modalities with constraints on cost:

\[ 1.4 \ 3.4 \ 0.2 \ 1.3 \ 1.2 \]
\[ \models U = 5 \]

Example

\[ G (\text{failure} \Rightarrow F \leq 250 \ \text{repaired}) \]
\[ A G (\text{failure} \Rightarrow E F \text{time} \leq 5 (\text{repair} \land A F \text{cost} \leq 150 \ \text{running})) \]
Weighted temporal logic

Example

Decorate temporal modalities with constraints on cost:

\[ G(failure \Rightarrow F \leq 250 \text{ repaired}) \]

\[ A G(failure \Rightarrow E F \text{ time} \leq 5 (repair \land A F \text{ cost} \leq 150 \text{ running})) \]
Weighted temporal logic

Example
Decorate temporal modalities with constraints on cost:

\[ \begin{align*}
\text{Example} & \quad G (\text{failure} \Rightarrow F \leq 250 \text{ repaired}) \\
\text{A G (failure} & \Rightarrow \text{E F}_{\leq 5} (\text{repair} \land \text{A F}_{\leq 150} \text{ running}))
\end{align*} \]
Weighted temporal logic

Example

Decorate temporal modalities with constraints on cost:

\[
1.4 \quad 3.4 \quad 0.2 \quad 1.3 \quad 1.2
\]
\[
\models U = 5
\]

Example

\[
G (\text{failure} \Rightarrow F_{\leq 250} \text{repaired})
\]
\[
A G (\text{failure} \Rightarrow EF_{\text{time} \leq 5} \text{repair} \land AF_{\text{cost} \leq 150} \text{running})
\]
Undecidability results

**Theorem**

*WMTL model-checking is undecidable.*
Undecidability results

Theorem
WMTL model-checking is undecidable.

Proof.
- encoding of a two-counter machine;

Undecidability results

**Theorem**

*WMTL model-checking is undecidable.*

**Proof.**

- encoding of a two-counter machine;
- Holds even for one clock and one cost variable.

Undecidability results

**Theorem**

*WMTL model-checking is undecidable.*

**Proof.**
- encoding of a **two-counter machine**;
- Holds even for one clock and one cost variable.

**Theorem**

*WCTL model-checking is undecidable.*
**Undecidability results**

**Theorem**

\[\text{WMTL model-checking is undecidable.}\]

**Proof.**
- encoding of a two-counter machine;
- Holds even for one clock and one cost variable.

**Theorem**

\[\text{WCTL model-checking is undecidable.}\]

**Proof.**
- encoding of a two-counter machine;


Undecidability results

**Theorem**

*WMTL* model-checking is undecidable.

**Proof.**
- encoding of a two-counter machine;
- Holds even for one clock and one cost variable.

**Theorem**

*WCTL* model-checking is undecidable.

**Proof.**
- encoding of a two-counter machine;
- requires three clocks.

Refs:  
Decidable subcases

Theorem

WCTL model-checking is *PSPACE-complete* on 1-clock weighted timed automata.

Decidable subcases

Theorem

WCTL model-checking is \textbf{PSPACE-complete} on 1-clock weighted timed automata.

Proof.

- region-based algorithm;

Decidable subcases

Theorem

WCTL model-checking is **PSPACE-complete** on 1-clock weighted timed automata.

Proof.

- region-based algorithm;
- but region are not fine enough:

```
\[ \dot{p} = 2, \dot{p} = 1, x = 1 \]
```

\[ \mathcal{E} \mathcal{F}_{\leq 1} \]

Decidable subcases

Theorem

\textit{WCTL model-checking is PSPACE-complete on 1-clock weighted timed automata.}

\textit{Proof.}

- region-based algorithm;
- but region are not fine enough:

\[
\begin{align*}
\dot{p} &= 2 \\
\dot{x} &= 1 \\
\dot{p} &= 1 \\
\dot{x} &= 1
\end{align*}
\]

Refine regions: granularity $1/M|\phi|$ is sufficient.

Decidable subcases

**Theorem**

WCTL model-checking is \textbf{PSPACE-complete} on 1-clock weighted timed automata.

\textit{Proof.}

- region-based algorithm;
- but region are not fine enough:

\[ \dot{p} = \frac{2}{\dot{p}} = \frac{1}{x} = 1 \]

\[ 0 \leq 1 \]

**Refine regions:** granularity $1/M|\phi|$ is sufficient.

Decidable subcases

**Theorem**

WCTL model-checking is *PSPACE-complete* on 1-clock weighted timed automata.

**Proof.**

- region-based algorithm;
- but region are not fine enough:

```
0  1
```

```
E \not\rightarrow EF_{\leq 1} \quad U_{\geq 1}
```

Decidable subcases

Theorem

WCTL model-checking is PSPACE-complete on 1-clock weighted timed automata.

Proof.

- region-based algorithm;
- but region are not fine enough:
- Refine regions: granularity $1/M^{\varphi}$ is sufficient.
Outline of the talk

1. Introduction

2. Timed automata with observers

3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies

4. Resource-management problems

5. Conclusions and perspectives
Weighted timed games

Example

Timed games can also be extended with weights:

\[
\begin{align*}
\dot{p} &= 2 \\
\dot{p} &= 5 \\
\dot{p} &= 0 \\
\dot{p} &= 3 \\
x &\leq 1 \\
x &+ = 4 \\
x &< 1 \\
x &= 1
\end{align*}
\]

A strategy for a player indicates which (action or delay) transition to play; a strategy is winning if all its outcomes are.
Weighted timed games

Example

Timed games can also be extended with weights:

\[\begin{align*}
&\dot{p} = 2 \\
&\dot{p} = 5 \\
&\dot{p} = 0 \\
&\dot{p} = 3
\end{align*}\]

A strategy for a player indicates which (action or delay) transition to play; a strategy is winning if all its outcomes are.
Weighted timed games

Example

Timed games can also be extended with weights:

\[ \dot{p} = 2 \quad \dot{p} = 5 \quad \dot{p} = 0 \quad \dot{p} = 3 \]

\[ x \leq 1 \]

\[ p + = 4 \]

A strategy for a player indicates which (action or delay) transition to play;

A strategy is winning if all its outcomes are.

\[ x \leq 1 \]

\[ x < 1 \]
Optimal winning strategy

Example

\[ \dot{p} = 5 \quad y = 0 \]
\[ \dot{p} = 6 \]
\[ \dot{p} = 3 \]
\[ x \leq 2 \]
\[ y := 0 \]
\[ x \geq 3 \]
\[ p + = 1 \]
\[ p + = 9 \]
\[ x \geq 3 \]

Minimal cost for reaching the goal:

\[ \inf_{0 \leq t \leq 2} \max \left( 5t + 6(3-t) + 1, 5t + 3(3-t) + 9 \right) = \frac{56}{3} \]

which is achieved with \( t = \frac{1}{3} \)

Corollary

Regions are not sufficient for solving priced timed games.
Optimal winning strategy

Example

\[ \dot{p} = 5 \quad y = 0 \]
\[ \dot{p} = 6 \]
\[ \dot{p} = 3 \]

\( x \leq 2 \)

\( y = 0 \)

\( x \geq 3 \)

\( p + = 1 \)

\( p + = 9 \)

Minimal cost for reaching 😊:

\[
\inf_{0 \leq t \leq 2} \max (5t + 6(3-t) + 1, 5t + 3(3-t) + 9) = \frac{56}{3}
\]

which is achieved with \( t = \frac{1}{3} \)

Corollary

Regions are not sufficient for solving priced timed games.
Optimal winning strategy

Example

Minimal cost for reaching 😊:

\[ 5t \quad 6 \quad 3 - t \quad 1 \]
Optimal winning strategy

Example

\[
\begin{align*}
\dot{p} &= 5 \\
y &= 0 \\
\dot{p} &= 6 \\
\dot{p} &= 3 \\
x &\leq 2 \\
y &= 0 \\
x &\geq 3 \\
p &= 1 \\
p &= 9 \\
\end{align*}
\]

Minimal cost for reaching 😊:

\[
\begin{align*}
5t &\quad 6 & 3 - t &\quad 1 \\
5t &\quad 3 & 3 - t &\quad 9 \\
\end{align*}
\]

Corollary

Regions are not sufficient for solving priced timed games.
Optimal winning strategy

Example

Minimal cost for reaching 😊:

\[
\max \begin{pmatrix}
5t & 6 & 3 - t & 1 \\
5t & 3 & 3 - t & 9
\end{pmatrix}
\]

which is achieved with \( t = \frac{1}{3} \)

Corollary

Regions are not sufficient for solving priced timed games.
Optimal winning strategy

Example

\[
\begin{align*}
\dot{p} &= 5 \\
y &= 0 \\
\dot{p} &= 6 \\
\dot{p} &= 3
\end{align*}
\]

\[
x \leq 2 \\
y := 0 \\
x \geq 3
\]

\[
p + = 1 \\
p + = 9 \\
x \geq 3
\]

Minimal cost for reaching 😊:

\[
\inf_{0 \leq t \leq 2} \max \begin{pmatrix}
5t & 6 & 3 - t & 1 \\
5t & 3 & 3 - t & 9
\end{pmatrix} = \frac{56}{3}
\]

which is achieved with \( t = \frac{1}{3} \)

Corollary

Regions are not sufficient for solving priced timed games.
Optimal winning strategy

Example

Minimal cost for reaching 😊:

\[ \inf_{0 \leq t \leq 2} \max \begin{pmatrix} 5t & 6 & 3 - t & 1 \\ 5t & 3 & 3 - t & 9 \end{pmatrix} = \frac{56}{3} \]

which is achieved with \( t = \frac{1}{3} \).
Optimal winning strategy

Example

Minimal cost for reaching ☹:

\[
\inf_{0 \leq t \leq 2} \max \begin{pmatrix} 5t & 6 & 3 - t & 1 \\ 5t & 3 & 3 - t & 9 \end{pmatrix} = \frac{56}{3}
\]

which is achieved with \( t = \frac{1}{3} \)

Corollary

Regions are not sufficient for solving priced timed games.
Optimal winning strategy

Example

Minimal cost for reaching 😊:

\[
\inf_{0 \leq t \leq 2} \max \begin{pmatrix}
5t & 6 & 3 - t & 1 \\
5t & 3 & 3 - t & 9
\end{pmatrix}
\]

which is achieved with \( t = 1/3 \)

Corollary

Regions are not sufficient for solving priced timed games.
Computing optimal winning strategies is undecidable

Theorem

*Computing optimal strategies in priced timed games is undecidable.*

Computing optimal winning strategies is undecidable

**Theorem**

*Computing optimal strategies in priced timed games is undecidable.*

**Proof.**

The proof relies on simple *modules* that will allow encoding a *two-counter machine*:

Computing optimal winning strategies is undecidable

Theorem

*Computing optimal strategies in priced timed games is undecidable.*

Proof.

The proof relies on simple modules that will allow encoding a two-counter machine:

- Adding the value of clock $x$ to the cost:

  Add $x$

  $p_0$ $x$ $p_1$

  $y_0$ $y_1$

  $z_0$ $z_1$

Computing optimal winning strategies is undecidable

**Theorem**

*Computing optimal strategies in priced timed games is undecidable.*

**Proof.**

The proof relies on simple modules that will allow encoding a two-counter machine:

- Adding the value of clock $x$ to the cost:
- Adding $1 - x$ to the cost:

---

Computing optimal winning strategies is undecidable

Theorem

*Computing optimal strategies in priced timed games is undecidable.*

**Proof.**

The proof relies on simple *modules* that will allow encoding a two-counter machine:

- Checking that \( y = 2x \):

Computing optimal winning strategies is undecidable

**Theorem**

*Computing optimal strategies in priced timed games is undecidable.*

**Proof.**

The proof relies on simple modules that will allow encoding a two-counter machine:

- **Checking that** $y = 2x$:  

  ![Diagram](image)

Computing optimal winning strategies is undecidable

**Theorem**

*Computing optimal strategies in priced timed games is undecidable.*

**Proof.**

The proof relies on simple modules that will allow encoding a two-counter machine:

- Checking that $y = 2x$:
- Dividing clock $x$ by 2:

![Diagram]

Computing optimal winning strategies is undecidable

Theorem

Computing optimal strategies in priced timed games is undecidable.

Proof.

The proof relies on simple modules that will allow encoding a two-counter machine:

- encode counter $c_1$ as $x_1 = 2^{1-c_1}$ and counter $c_2$ as $x_2 = 3^{1-c_1}$;
- by cleverly juggling with clocks, we can achieve this encoding with three clocks.

Turn-based 1-clock priced timed games are decidable

Example

- Optimal strategies do not always exist:
  - $\dot{p} = 2$, $\dot{p} = 1$

```latex
x = 1
x = 0
```

- Optimal strategies may require memory:
  - $\dot{p} = 2$, $\dot{p} = 1$

```latex
x < 1, x := 0
x > 0
```
Turn-based 1-clock priced timed games are decidable

Example

- Optimal strategies do not always exist:

\[ \dot{p} = 2 \quad \dot{p} = 1 \]

\[ x = 1 \]
\[ x = 0 \]

- Optimal strategies may require memory:

\[ x < 1, \quad x = 0 \]
\[ x > 0 \]
Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games* always admit \( \varepsilon \)-optimal winning strategies, and such strategies can be computed.

Turn-based 1-clock priced timed games are decidable

Theorem

*Turn-based 1-clock priced timed games always admit ε-optimal winning strategies, and such strategies can be computed.*

Proof.

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games* always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.

**Proof.**


Theorem

*Turn-based 1-clock priced timed games* always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.

Proof.
Turn-based 1-clock priced timed games are decidable

Theorem

*Turn-based 1-clock priced timed games always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.*

Proof.


Turn-based 1-clock priced timed games are decidable

Theorem

*Turn-based 1-clock priced timed games always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.*

Proof.

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.*

**Proof.**

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.*

**Proof.**


Turn-based 1-clock priced timed games are decidable

Theorem

*Turn-based 1-clock priced timed games always admit $\epsilon$-optimal winning strategies, and such strategies can be computed.*

Proof.

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.*

**Proof.**


Turn-based 1-clock priced timed games are decidable

Theorem

**Turn-based 1-clock priced timed games** always admit \( \varepsilon \)-optimal winning strategies, and such strategies can be computed.

Proof.

Theorem

**Turn-based 1-clock priced timed games** always admit \( \varepsilon \)-optimal winning strategies, and such strategies can be computed.

Proof.

Turn-based 1-clock priced timed games are decidable

Theorem

Turn-based 1-clock priced timed games always admit \( \varepsilon \)-optimal winning strategies, and such strategies can be computed.

Proof.

Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games* always admit $\varepsilon$-optimal winning strategies, and such strategies can be computed.

**Proof.**


Turn-based 1-clock priced timed games are decidable

**Theorem**

*Turn-based 1-clock priced timed games always admit \( \varepsilon \)-optimal winning strategies, and such strategies can be computed.*

**Proof.**

Turn-based 1-clock priced timed games are decidable

Theorem

*Turn-based 1-clock priced timed games* always admit *ε*-optimal winning strategies, and such strategies can be computed.

Proof.

Turn-based 1-clock priced timed games are decidable

Theorem

*Turn-based 1-clock priced timed games* always admit *ε-optimal winning strategies*, and such strategies can be computed.

Proof.

- The procedure terminates;
- There is a **positive granularity** for with the **region abstraction is correct**;
- The **optimal cost functions** are piecewise affine, continuous, decreasing functions. Their slopes are rates of the automaton.

Refs:


Outline of the talk

1. Introduction

2. Timed automata with observers

3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies

4. Resource-management problems

5. Conclusions and perspectives
Managing resources

Example

In some cases, resources can both be consumed and regained.

The aim is then to keep the level of resources within given bounds.
Managing resources

Example

\[ \ell_0 - 3 \xrightarrow{x \cdot 0} \ell_1 6 \xrightarrow{x \cdot 1} \ell_2 -6 \]

Three variants of the problem:
1. **Lower bound**: The aim is to maintain the level of resources above a given bound.
2. **Interval**: The aim is to keep the level of resources within an interval.
3. **Lower bound with finite capacity**: The aim is to keep the level of resources above a given lower bound, but with a finite capacity.
Managing resources

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
Managing resources

Example

Three variants of the problem:

1. **Lower bound**: the aim is to maintain the level of resources above a given bound.

$$\ell_0 - 3 \rightarrow \ell_1 6 \rightarrow \ell_2 -6$$

$$x: 0 \rightarrow x: 1$$
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.

2. **interval**: the aim is to keep the level of resources within an interval.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.

2. **interval**: the aim is to keep the level of resources within an interval.
Managing resources

Example

$$-3 \ell_0 + 6 \ell_1 - 6 \ell_2 = 1$$

$$x = 0$$

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
2. **interval**: the aim is to keep the level of resources within an interval.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.

2. **interval**: the aim is to keep the level of resources within an interval.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.

2. **interval**: the aim is to keep the level of resources within an interval.
Managing resources

Example

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.

2. **interval**: the aim is to keep the level of resources within an interval.

3. **lower bound with finite capacity**: the aim is to keep the level of resources above a given lower bound, but with a finite capacity.
Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.
2. **interval**: the aim is to keep the level of resources within an interval.
3. **lower bound with finite capacity**: the aim is to keep the level of resources above a given lower bound, but with a finite capacity.
Managing resources

Example

\[ -3 \ell_0 + 6 \ell_1 - 6 \ell_2 = 1 \]

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.

2. **interval**: the aim is to keep the level of resources within an interval.

3. **lower bound with finite capacity**: the aim is to keep the level of resources above a given lower bound, but with a finite capacity.
Managing resources

Example

\[ x = 0 \]

Three variants of the problem:

1. **lower bound**: the aim is to maintain the level of resources above a given bound.

2. **interval**: the aim is to keep the level of resources within an interval.

3. **lower bound with finite capacity**: the aim is to keep the level of resources above a given lower bound, but with a finite capacity.
Results in the untimed case

Theorem

In the untimed case, the following results hold:

<table>
<thead>
<tr>
<th></th>
<th>existential problem</th>
<th>universal problem</th>
<th>games</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower bound</td>
<td>$\in$ PTIME</td>
<td>$\in$ PTIME</td>
<td>$\in$ UP $\cap$ coUP</td>
</tr>
<tr>
<td>Lower bound, finite capacity</td>
<td>$\in$ PTIME</td>
<td>$\in$ PTIME</td>
<td>$\in$ NP PTIME-hard</td>
</tr>
<tr>
<td>Interval</td>
<td>$\in$ PSPACE NP-hard</td>
<td>$\in$ PTIME</td>
<td>EXPTIME-c.</td>
</tr>
</tbody>
</table>

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.

Results in the 1-clock case

Theorem

*In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.*

Proof.

- **Corner-point abstraction:**

\[ -3 \xrightarrow{x>0} 6 \xrightarrow{x} -6 \]

\[ x \xrightarrow{0} 0 \xrightarrow{x} 1 \]

Results in the 1-clock case

**Theorem**

*In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.*

**Proof.**

- **Corner-point abstraction:**

  ![Diagram](image)

# Results in the 1-clock case

**Theorem**

*In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.*

**Proof.**

- **Corner-point abstraction:** Only correct if no discrete costs!

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.

Proof.

- Corner-point abstraction: Only correct if no discrete costs!

Results in the 1-clock case

**Theorem**

*In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.*

**Proof.**

- **Corner-point abstraction:** Only correct if no discrete costs!
- In the presence of discrete costs:

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.

Proof.

- **Corner-point abstraction:** Only correct if no discrete costs!
- In the presence of discrete costs:
  - compute **optimal final resource-level** along a non-resetting path;

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of an infinite run with resource level above a given lower bound is decidable in EXPTIME.

Proof.

- **Corner-point abstraction**: Only correct if no discrete costs!
- In the presence of discrete costs:
  - compute optimal final resource-level along a non-resetting path;
  - compose the resulting functions for general paths.

Results in the 1-clock case

Theorem

*In the 1-clock case, the existence of a strategy for maintaining the resource level within a given interval is undecidable.*

Results in the 1-clock case

Theorem

In the 1-clock case, the existence of a strategy for maintaining the resource level within a given interval is undecidable.

Proof.

- Encoding of a two-counter machine: both counters are stored in one cost, as \( \ell = 5 - 2^{-c_1} \cdot 3^{-c_2} \).

Results in the 1-clock case

Theorem

*In the 1-clock case, the existence of a strategy for maintaining the resource level within a given interval is undecidable.*

Proof.

- Encoding of a two-counter machine: both counters are stored in one cost, as \( \ell = 5 - 2^{-c_1} \cdot 3^{-c_2} \).
- The following module is used to increment and decrement:

![Diagram showing the encoding of a two-counter machine and the modules used for increment and decrement.](image)

Results in the 1-clock case

**Theorem**

In the 1-clock case, the existence of a strategy for maintaining the resource level within a given interval is undecidable.

**Proof.**

- **Encoding of a two-counter machine**: both counters are stored in one cost, as $\ell = 5 - 2^{-c_1} \cdot 3^{-c_2}$.
- The following module is used to increment and decrement:

```
Initial level  

5 - e

module ok

Final level  

5 - ne/6
```

Outline of the talk

1. Introduction
2. Timed automata with observers
3. Resource-optimization problems
   - Optimal reachability
   - Weighted temporal logics
   - Optimal strategies
4. Resource-management problems
5. Conclusions and perspectives
Conclusions and perspectives

- **Weighted timed automata** are a powerful formalism for modeling **resources**:
  - expressive enough for many applications;
  - several problems remain **decidable**;
  - some algorithms can be made **symbolic** and are **implemented** in Uppaal CORA.
Conclusions and perspectives

- **Weighted timed automata** are a powerful formalism for modeling **resources**:
  - expressive enough for many applications;
  - several problems remain **decidable**;
  - some algorithms can be made **symbolic** and are implemented in Uppaal CORA.

- Many open problems:
  - energy constraints for automata with several clocks;
  - timed automata with observers having richer dynamics.

\[
\begin{align*}
-3 & \rightarrow 6 \\
-6 & \rightarrow 2 \\
0 & \rightarrow -1 \\
1 & \rightarrow 0 \\
\end{align*}
\]

\[
\begin{align*}
x & = 1 \\
x & = 0 \\
\frac{dp}{dt} & = 2 \times p
\end{align*}
\]
Conclusions and perspectives

- Weighted timed automata are a powerful formalism for modeling resources:
  - expressive enough for many applications;
  - several problems remain decidable;
  - some algorithms can be made symbolic and are implemented in Uppaal CORA.

- Many open problems:
  - energy constraints for automata with several clocks;
  - timed automata with observers having richer dynamics.
Real-time Model Checking
— Open problems —

Patricia BOUYER-DECITRE, Kim G. LARSEN, Nicolas MARKEY

March 3, 2010
Algorithms and datastructures

- **Fully symbolic exploration** of timed automata:
  - BDD-like representation of transitions;
  - Clock Difference Diagrams, but no canonical form...

- **Zone-based implementations** for the verification of priced-timed automata:
  - optimal infinite runs;
  - multi-priced timed automata;
  - energy timed automata and games...

- **Algorithms and implementations** for model-checking linear-time properties.

- **Partial-order reductions** for timed automata.

- **Bounded model-checking** for timed automata using SAT solvers.
Timed games

- Non-zero-sum games:
  - multi-player timed games;
  - instead of computing winning strategies, we look for equilibria.

- Partial observability:
  - minimal set of observations needed to ensure controllability
  - CEGAR approach to incomplete information.

- Probabilistic timed games:

Weighted timed automata and games

- Priced timed games with two clocks
- Optimal infinite runs with safety
- Timed automata with exponential observers and even richer dynamics
- Energy timed games with several clocks.
Implementability issues

- **Timed automata are not implementable!**
  - computers are digital;
  - communications are not instantaneous;
  - different clocks have (slightly) different rates.

- **Modified semantics**
  - tube semantics
  - probabilistic semantics
  - guard enlargement (robustness)

- **Zone-based algorithms** for robustness checking

- **Robust model-checking** of weighted timed automata.

- **Robust control:**
  - synthesis of implementable controllers.