Timed Model Checking

François Laroussinie

LSV / ENS Cachan – CNRS UMR 8643

Journée Modèles du temps réel: Réseaux de Petri temporisés et temporels, automates temporisés – 23 mars 2007
Model checking

System

Properties

Formalizing step

? |= ϕ
We want...

- **Expressive models**: communication, variables, timing constraints, probabilities...
- **Expressive specification languages**: natural, powerful, intuitive...
We want...

- **Expressive models**: communication, variables, timing constraints, probabilities.
- **Expressive specification languages**: natural, powerful, intuitive.

**AND EFFICIENT ALGORITHM**!

Main limit of model checking = state explosion problem
State explosion problem

This has motivated

- symbolic methods
- heuristics:
  - on-the-fly algorithms
  - acceleration
  - partial order
  - ...
- abstraction
- ...

Model checkers exist and they work rather nicely!
Quantitative verification

- timing constraints
- probabilities
- dynamic variables
- ...
- extending the specification languages

**NB:** every extension may induce a complexity blow-up.
Quantitative verification

- timing constraints
- probabilities
- dynamic variables
- ...
- extending the specification languages

**NB:** every extension may induce a complexity blow-up.
Outline

1. Timed properties

2. Decision procedures for TA
   - Timed automata
   - Model checking TA
   - Model-checking TCTL
   - Complexity

3. Algorithms in practice

4. Simply-timed systems

5. Conclusion
Outline

1 Timed properties

2 Decision procedures for TA
   • Timed automata
   • Model checking TA
   • Model-checking TCTL
   • Complexity

3 Algorithms in practice

4 Simply-timed systems

5 Conclusion
Classical verification problems

- Reachability of a control state
- $S \sim S'$ ? : (bi)simulation, etc.
- $L(S) \subseteq L(S)$ ? : language inclusion
- $S|A_T$ + reachability : testing automata
- $S \models \Phi$ with $\Phi$ a temporal logic formula : model checking
Classical verification problems

- Reachability of a control state
- $S \sim S'$ ? : (bi)simulation, etc.
- $L(S) \subseteq L(S)$ ? : language inclusion
- $(S|A_T) +$ reachability : testing automata
- $S \models \Phi$ with $\Phi$ a temporal logic formula : model checking

The semantics of the system is a timed transition system.
Timed transition systems

= a transition system + a notion of time

Let $\mathbb{T}$ be a time domain: $\mathbb{N}$, $\mathbb{R}_+$ or $\mathbb{Q}_+$.

**Timed transition system**

$\mathcal{T} = \langle S, s_0, \text{Act}, \rightarrow, I \rangle$

- $S$ is an (infinite) set of states, $s_0 \in S$
- $\rightarrow \subseteq S \times (\mathbb{T} \cup \text{Act}) \times S$
- $I : S \rightarrow 2^{\text{Prop}}$ : assigns atomic propositions to states
Timed transition systems

= a transition system + a notion of time

Let $\mathbb{T}$ be a time domain: $\mathbb{N}$, $\mathbb{R}^+$ or $\mathbb{Q}^+$.

**Timed transition system**

$\mathcal{T} = \langle S, s_0, \text{Act}, \rightarrow, l \rangle$

- $S$ is an (infinite) set of states, $s_0 \in S$
- $\rightarrow \subseteq S \times (\mathbb{T} \cup \text{Act}) \times S$
- $l : S \rightarrow 2^{\text{Prop}}$ : assigns atomic propositions to states

- $s \xrightarrow{a} s'$ : an action transition “$a$” ($a \in \text{Act}$).
- $s \xrightarrow{t} s'$ : a delay transition with duration $t \in \mathbb{R}^+$.

Every finite run $\sigma$ in $\mathcal{T}$ has a finite duration $\text{Time}(\sigma)$. 
Timed properties

Temporal properties:

“Any problem is followed by an alarm”

With \textit{CTL}:

\[ \text{AG} \left( \text{problem} \Rightarrow \text{AF alarm} \right) \]
Timed properties

**Temporal properties:**

“Any problem is followed by an alarm”

With **CTL:** \[ \text{AG} \left( \text{problem} \implies \text{AF alarm} \right) \]

**Timed properties:**

“Any problem is followed by an alarm in at most 10 time units”
Timed CTL – with “subscripts”

We use $TCTL$ whose formulae are built from:

- atomic propositions in Prop (For ex. $\text{Alarm}$, $\text{Problem}$, …)
- boolean combinators ($\land$, $\lor$, $\neg$), and
- temporal operators tagged with timing constraints: $E \ U_{\sim c}$ and $A \ U_{\sim c}$ with $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{N}$.

+ all the standard abbreviations: $AG_{\sim c}$, $AF_{\sim c}$ etc.
We use $TCTL$ whose formulae are built from:

- atomic propositions in $Prop$ (For ex. $\text{Alarm}$, $\text{Problem}$, $\ldots$ )
- boolean combinators ($\land$, $\lor$, $\neg$), and
- temporal operators tagged with timing constraints: $E_\sim U_c$ and $A_\sim U_c$ with $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{N}$.

+ all the standard abbreviations: $AG_\sim$, $AF_\sim$ etc.

$$s \models E\varphi U_\sim c \psi \iff \exists \rho = \rho' \cdot \rho'' \in \text{Exec}(s) \text{ with } s \xrightarrow{\rho'} s' \text{ and }$$
$$\text{Time}(\rho') \sim c \text{ and } s' \models \psi, \text{ and } s'' \models \varphi \text{ for all } s \prec \rho' s'' \prec \rho s'$$
We use \( TCTL \) whose formulae are built from:

- atomic propositions in Prop (For ex. Alarm, Problem, \ldots )
- boolean combinators (\( \land, \lor, \neg \)), and
- temporal operators tagged with timing constraints: \( E \_ U_{\sim c} \) and \( A \_ U_{\sim c} \) with \( \sim \in \{<, \leq, =, \geq, >\} \) and \( c \in \mathbb{N} \).

\[ s \models A\varphi U_{\sim c} \psi \iff \forall \rho \in \text{Exec}(s), \rho = \rho' \cdot \rho'' \text{ s.t } s \xrightarrow{\rho'} s' \text{ and } \text{Time}(\rho') \sim c \text{ and } s' \models \psi, \text{ and } s'' \models \varphi \text{ for all } s <_\rho s'' <_\rho s' \]
Timed CTL – with “subscripts”

We use $TCTL$ whose formulae are built from:

- atomic propositions in Prop (For ex. Alarm, Problem, …)
- boolean combinators ($\land$, $\lor$, $\neg$), and
- temporal operators tagged with timing constraints: $E \_ U_{\sim c}$- and $A \_ U_{\sim c}$- with $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{N}$.

+ all the standard abbreviations: $AG_{\sim c}$, $AF_{\sim c}$ etc.

“Any problem is followed by an alarm in at most 10 time units”

$$AG \left( \text{problem} \Rightarrow AF_{\leq 10} \text{ alarm} \right)$$
$TCTL_c$ formulae are built from:

- atomic propositions (For ex. $\text{Alarm}$, $\text{Problem}$, \ldots)
- boolean combinators ($\land$, $\lor$, $\lnot$)
- classical temporal operators $E\_U\_ \text{ and } A\_U\_ \text{, and}$
- formula clocks ($x, y \in K$), constraints ($x \sim c$ or $x - y \sim c$), and reset operator ($x \text{ in } \varphi$).

$TCTL_c$ formulae are interpreted over pairs $(s, u)$:

- $s$ is a state of the TTS
- $u : K \rightarrow \mathbb{T}$ is a valuation for the formulae clocks
Timed CTL – with clocks

\( TCTL_c \) formulae are built from:

- atomic propositions (For ex. Alarm, Problem, . . .)
- boolean combinators (\( \land, \lor, \neg \))
- classical temporal operators \( E.U. \) and \( A.U. \), and
- formula clocks \( (x, y \in K) \), constraints \( (x \sim c \text{ or } x - y \sim c) \), and reset operator \( (x \text{ in } \varphi) \).

“Any problem is followed by an alarm in at most 10 time units”

\[
AG \left( \text{problem } \Rightarrow x \text{ in } \mathbf{AF}(x \leq 10 \land \text{alarm}) \right)
\]
Outline

1. Timed properties

2. Decision procedures for TA
   - Timed automata
   - Model checking TA
   - Model-checking TCTL
   - Complexity

3. Algorithms in practice

4. Simply-timed systems

5. Conclusion
Outline

1. Timed properties

2. Decision procedures for TA
   - Timed automata
   - Model checking TA
   - Model-checking TCTL
   - Complexity

3. Algorithms in practice

4. Simply-timed systems

5. Conclusion
$\mathcal{A} = \text{an automaton (locations and transitions)} + \text{clocks}$

**OFF**

- **true, b, x := 0**
- **x = 10, i, −**

**ON**

- **x ≤ 10**
- **x < 10, b, x := 0**
\( A = \text{an automaton (locations and transitions) + clocks} \)

Clocks progress synchronously with time (Time domain = \( \mathbb{R}_+ \))

**Transitions**: \( l \xrightarrow{g,a,r} l' \in T \) with:

- \( g \) is the guard,
- \( a \) is the label,
- \( r \) is the set of clocks to be reset to 0
Timed automata - definition  [Alur & Dill]

\[ \mathcal{A} = \text{an automaton (locations and transitions) } + \text{ clocks} \]

Clocks progress synchronously with time (Time domain = \( \mathbb{R}_+ \))

Transitions: \( \ell \xrightarrow{g,a,r} \ell' \in T \) with:

- \( g \) is the guard,
- \( a \) is the label,
- \( r \) is the set of clocks to be reset to 0

Semantics: . . . a timed transition system.

\( \blacktriangle \text{ States: } (\ell, v) \) where \( v \) is a valuation for the clocks
Timed automata - definition  [Alur & Dill]

\[\mathcal{A} = \text{an automaton (locations and transitions)} + \text{clocks}\]

Clocks progress synchronously with time (Time domain = \(\mathbb{R}_+\))

**Transitions**: \(\ell \xrightarrow{g,a,r} \ell' \in T\) with:
- \(g\) is the guard,
- \(a\) is the label,
- \(r\) is the set of clocks to be reset to 0

**Semantics**: \(\ldots\) a timed transition system.

▲ **States**: \((\ell, v)\) where \(v\) is a valuation for the clocks

▲ **Action transition**:

\((\ell, v) \xrightarrow{a} (\ell', v')\) iff  
\[
\begin{cases} 
\exists \ell \xrightarrow{g,a,r} \ell' \in \mathcal{A}, \\
    v \models g, \quad v' = v[r \leftarrow 0]
\end{cases}
\]
Timed automata - definition  [Alur & Dill]

\[ \mathcal{A} = \text{an automaton (locations and transitions) + clocks} \]

Clocks progress synchronously with time (Time domain = \( \mathbb{R}_+ \))

**Transitions:** \( \ell \xrightarrow{g, a, r} \ell' \in T \) with:
- \( g \) is the guard,
- \( a \) is the label,
- \( r \) is the set of clocks to be reset to 0

**Semantics:** . . . a timed transition system.

▲ **States:** \((\ell, v)\) where \( v \) is a valuation for the clocks

▲ **Action transition:**
\[
(\ell, v) \xrightarrow{a} (\ell', v') \iff \exists \ell \xrightarrow{g, a, r} \ell' \in \mathcal{A}, v \models g, v' = v[r \leftarrow 0]
\]

▲ **Delay transition:**
\[
\forall t \in \mathbb{R}_+, (\ell, v) \xrightarrow{t} (\ell, v + t) \iff \forall 0 \leq t' \leq t, v + t' \models \text{Inv}(\ell)
\]
Delay transitions satisfy the following properties:

- **Time-determinism:** $q \xrightarrow{t} q' \land q \xrightarrow{t} q'' \Rightarrow q' = q''$

- **0-delay:** $q \xrightarrow{0} q$

- **Time-Additivity:**
  
  $q \xrightarrow{t} q' \land q' \xrightarrow{t'} q'' \Rightarrow q \xrightarrow{t+t'} q''$

- **Time-Continuity:**

  $q \xrightarrow{t+t'} q' \Rightarrow \exists q'' \text{ s.t. } q \xrightarrow{t} q'' \xrightarrow{t'} q'$
Outline

1 Timed properties

2 Decision procedures for TA
   - Timed automata
   - Model checking TA
   - Model-checking TCTL
   - Complexity

3 Algorithms in practice

4 Simply-timed systems

5 Conclusion
$A$ defines an \textit{infinite} timed transition system.

Decision procedures are based on the \textit{region graph} technique (Alur, Courcoubetis, Dill)
Model checking TA

\(\mathcal{A}\) defines an infinite timed transition system.

Decision procedures are based on the region graph technique (Alur, Courcoubetis, Dill)

From \(\mathcal{A}\) and \(\Phi\), one defines an equivalence \(\equiv_{\mathcal{A},\Phi}\) s.t.:

- \(v \equiv_{\mathcal{A},\Phi} v' \Rightarrow (\ell, v) \models \Phi \iff (\ell, v') \models \Phi\)

- \(\mathbb{R}_+^X/\equiv_{\mathcal{A},\Phi}\) is finite

A region = An equivalence class of \(\equiv_{\mathcal{A},\Phi}\)

Reduce \(\mathcal{A} \models \Phi\) to \((\mathcal{A} \times \mathbb{R}_+^X/\equiv_{\mathcal{A},\Phi}) \models \Phi\)

...and use a standard model checking algorithm.
Model checking TA

\( \mathcal{A} \) defines an infinite timed transition system.

Decision procedures are based on the region graph technique (Alur, Courcoubetis, Dill)

From \( \mathcal{A} \) and \( \Phi \), one defines an equivalence \( \equiv_{\mathcal{A}, \Phi} \) s.t.:

- \( \nu \equiv_{\mathcal{A}, \Phi} \nu' \Rightarrow ( (\ell, \nu) \models \Phi \iff (\ell, \nu') \models \Phi ) \)

- \( \mathbb{R}_{+}^{X} / \equiv_{\mathcal{A}, \Phi} \) is finite

A region = An equivalence class of \( \equiv_{\mathcal{A}, \Phi} \)

Reduce \( \mathcal{A} \models \Phi \) to \( (\mathcal{A} \times \mathbb{R}_{+}^{X} / \equiv_{\mathcal{A}, \Phi} ) \models \Phi \)

...and use a standard model checking algorithm.

! The size of \( \mathbb{R}_{+}^{X} / \equiv_{\mathcal{A}, \Phi} \) is in \( O(|X|! \cdot M^{|X|}) \)!
Property $\Phi$: reachability of the control state $q_F$.

Two configurations $(\ell, \nu)$ and $(\ell, \nu')$ have the same behavior (w.r.t. $\Phi$) when

1. Any action transition enabled from $\nu$ is also enabled from $\nu'$; and the target configurations have the same behavior... (and vice versa)

2. For any delay transition $t$ from $\nu$, there is a delay transition $t'$ s.t. $(q, \nu + t)$ and $(q, \nu' + t')$ have the same behavior... (and vice versa)
Property Φ: reachability of the control state $q_F$.

Two configurations $(ℓ, v)$ and $(ℓ, v')$ have the same behavior (w.r.t. Φ) when

1. Any action transition enabled from $v$ is also enabled from $v'$; and the target configurations have the same behavior... (and vice versa)

2. For any delay transition $t$ from $v$, there is a delay transition $t'$ s.t. $(q, v + t)$ an $(q, v' + t')$ have the same behavior... (and vice versa)

This is a (time abstract) bisimulation!
Equivalence on clocks valuations

Time abstract bisimulation: \((\ell, v) \approx (\ell, v')\) when

1. For any \(\ell \xrightarrow{g,a,r} \ell'\), we have \((\ell, v) \xrightarrow{a} (\ell', v[r \leftarrow 0])\) \(\Rightarrow\) there exists \((\ell, v') \xrightarrow{a} (\ell', v'[r \leftarrow 0])\) and \((\ell', v[r \leftarrow 0]) \approx (\ell', v'[r \leftarrow 0])\) (and vice versa)

2. For any \((\ell, v) \xrightarrow{t} (\ell, v + t)\); there exists \((\ell, v') \xrightarrow{t'} (\ell, v' + t')\) and \((\ell, v + T) \approx (\ell, v' + t')\) (and vice versa)
Equivalence on clocks valuations

Time abstract bisimulation: \((\ell, v) \approx (\ell, v')\) when

1. For any \(\ell \xrightarrow{g,a,r} \ell'\), we have \((\ell, v) \xrightarrow{a} (\ell', v[r \leftarrow 0])\) \(\Rightarrow\) there exists \((\ell, v') \xrightarrow{a} (\ell', v'[r \leftarrow 0])\) and \((\ell', v[r \leftarrow 0]) \approx (\ell', v'[r \leftarrow 0])\)
   (and vice versa)

2. For any \((\ell, v) \xrightarrow{t} (\ell, v + t)\); there exists \((\ell, v') \xrightarrow{t'} (\ell, v' + t')\)
   and \((\ell, v + T) \approx (\ell, v' + t')\)
   (and vice versa)

For (1), it is sufficient to satisfy the same constraints \(x \sim k\) with \(k \in \{0, \ldots, M_x\}\).
The region abstraction

\[ X = \{x, y\} \]

\[ M_x = 3 \text{ and } M_y = 2 \]
The region abstraction

\[ X = \{x, y\} \]
\[ M_x = 3 \text{ and } M_y = 2 \]

- “compatibility” between regions and constraints \( x \sim k \) and \( y \sim k \)
The region abstraction

$X = \{x, y\}$

$M_x = 3$ and $M_y = 2$

- “compatibility” between regions and constraints $x \sim k$ and $y \sim k$
- “compatibility” between regions and time elapsing
The region abstraction

\[ X = \{x, y\} \]

\[ M_x = 3 \text{ and } M_y = 2 \]

- “compatibility” between regions and constraints \( x \sim k \) and \( y \sim k \)
- “compatibility” between regions and time elapsing
The region abstraction

- “compatibility” between regions and constraints $x \sim k$ and $y \sim k$
- “compatibility” between regions and time elapsing

$X = \{x, y\}$

$M_x = 3$ and $M_y = 2$
The region abstraction

\[ X = \{x, y\} \]
\[ M_x = 3 \text{ and } M_y = 2 \]

region defined by
\[ I_x = ]1; 2[, \ I_y = ]0; 1[ \]
\[ \{x\} < \{y\} \]

- “compatibility” between regions and constraints \( x \sim k \) and \( y \sim k \)
- “compatibility” between regions and time elapsing
The region abstraction

- “compatibility” between regions and constraints $x \sim k$ and $y \sim k$
- “compatibility” between regions and time elapsing

\[
X = \{x, y\}
\]
\[
M_x = 3 \text{ and } M_y = 2
\]

region defined by
\[
I_x = ]1; 2[, \ I_y = ]0; 1[
\]
\[
\{x\} < \{y\}
\]

successor by delay transition:
\[
I_x = ]1; 2[, \ I_y = ]2; 2[
\]
Region graph
Region graph
Region graph

\[ \mathcal{A} \times \mathbb{R}_+^X_{\equiv A, \Phi} \text{ is a standard cartesian product.} \]

Every node contains information on the value of any clock.
An example [AD 90's]
Outline

1 Timed properties

2 Decision procedures for TA
   • Timed automata
   • Model checking TA
   • Model-checking TCTL
   • Complexity

3 Algorithms in practice

4 Simply-timed systems

5 Conclusion
Consider the time-switch...

\[
\begin{align*}
true, & \quad b, \quad x := 0 \quad \text{for } x < 10, \quad b, \quad x := 0 \\
\text{OFF} & \quad \text{ON} \\
\text{ON}, & \quad x \leq 10 \quad \text{for } x = 10, \quad i, \quad \neg \\
& \\
\text{How to decide: } (\text{ON}, x = 6.8) \models AF_{<5} \text{OFF} ~? \end{align*}
\]
Consider the time-switch... How to decide: \((\text{ON}, x = 6.8) \models \text{AF}_{\leq 5} \text{OFF}\) ?

- Add a new clock \(z\)
Consider the time-switch...

How to decide: $(\text{ON}, x = 6.8) \models \text{AF}_{<5} \text{OFF}$?

- Add a new clock $z$

- $z$ is not used in $\mathcal{A}$:
  
  \[(q, \nu(z = t)) \models \varphi \iff (q, \nu(z = t')) \models \varphi \iff (q, \nu) \models \varphi\]
Consider the time-switch...
How to decide: \((\text{ON}, x = 6.8) \models \text{AF}_{<5}\text{OFF}\) ?

- Add a new clock \(z\)

- \(z\) is not used in \(\mathcal{A}\):
  \[(q, \nu(z = t)) \models \varphi \iff (q, \nu(z = t')) \models \varphi \iff (q, \nu) \models \varphi\]

- \(z\) is used to measure time elapsing and evaluate the timed constraint in \(\varphi\):
  \[(q, \nu) \models \text{AF}_{<5}\text{OFF} \iff (q, \nu(z = 0)) \models \text{AF (OFF} \wedge z < 5)\]
Consider the time-switch...
How to decide: \((ON, x = 6.8) \models AF_{\leq 5} OFF\)?

- Add a new clock \(z\)
- \(z\) is not used in \(A\):
  \[(q, \nu(z = t)) \models \varphi \iff (q, \nu(z = t')) \models \varphi \iff (q, \nu) \models \varphi\]
- \(z\) is used to measure time elapsing and evaluate the timed constraint in \(\varphi\):
  \[(q, \nu) \models AF_{\leq 5} OFF \iff (q, \nu(z = 0)) \models AF (OFF \land z < 5)\]
- \(\iff (q, [\nu(z = 0)]) \models AF (OFF \land P_{z < 5})\)
  \(\Rightarrow\) CTL model checking over a finite graph!
Idea: We reduce the verification of $TCTL$ formula over TAs to a model checking problem over the region graph for a $CTL$ formula.

- We add a new clocks $z_\Phi$ to handle timing constraints "$\sim c$"

$X' = X \cup \{z_\Phi\}$
Idea: We reduce the verification of TCTL formula over TAs to a model checking problem over the region graph for a CTL formula.

- We add a new clocks $z_\Phi$ to handle timing constraints “$\sim c$”
  \[ X' = X \cup \{z_\Phi\} \]
- Build the region graph \( \mathbb{R}^+_M X' \) with \( M = \max(M_\Phi, M_\Lambda) \).
Idea: We reduce the verification of TCTL formula over TAs to a model checking problem over the region graph for a CTL formula.

- We add a new clocks $z_\Phi$ to handle timing constraints $\sim c$
  $X' = X \cup \{z_\Phi\}$
- Build the region graph $\mathbb{R}_+^X_M$ with $M = \max(M_\Phi, M_A)$.
- For any constraint $\sim c$, label states $(\ell, [v])$ with $P_{\sim c}$ when $v \models z_\Phi \sim c$. 
Idea: We reduce the verification of TCTL formula over TAs to a model checking problem over the region graph for a CTL formula.

- We add a new clocks $z_\Phi$ to handle timing constraints $\sim c$.
  \[ X' = X \cup \{ z_\Phi \} \]
- Build the region graph $\mathbb{R}_+^{X'_M}$ with \( M = \max(M_\Phi, M_A) \).
- For any constraint $\sim c$, label states $(\ell, [v])$ with $P_{\sim c}$ when $v \models z_\Phi \sim c$.
- $(\ell, v) \models E\varphi U_{\sim c} \psi$ iff(*) $(\ell, \gamma_v[x_\Phi \leftarrow 0]) \models E\varphi U(\psi \land P_{z_\Phi \sim c})$
Outline

1 Timed properties

2 Decision procedures for TA
   - Timed automata
   - Model checking TA
   - Model-checking TCTL
   - Complexity

3 Algorithms in practice

4 Simply-timed systems

5 Conclusion
Complexity of timed verification

**Theorem**

- Reachability in TA is **PSPACE-C** (Alur & Dill)

- Reachability in TA with three clocks is **PSPACE-C** (Courcoubetis & Yannakakis)

- Reachability in TA with constants in \{0, 1\} is **PSPACE-C** (Courcoubetis & Yannakakis)

- Model checking **Timed CTL** over TA is **PSPACE-C** (Alur, Courcoubetis & Dill)

(But tools (Kronos, UppAal) exist and have been applied successfully for verifying industrial case studies.)
PSPACE-hardness of reachability

Let $\mathcal{M} = \langle \Sigma, Q, q_0, q_f, T \rangle$ be a linear bounded Turing machine. Let $w_0$ an input word over $\{a, b\}^*$, with $|w_0| = n$.

The behavior of $\mathcal{M}$ over $w_0$ can be encoded by a TA $A_{\mathcal{M},w_0}$:

- Locations of $A_{\mathcal{M},w_0}$ are
  $$(q, i) \begin{cases} q \text{ is a state of } \mathcal{M} \\ i \text{ is the position of the tape head} \end{cases}$$
- Clocks are $\{x_1, \ldots, x_n, y_1, \ldots, y_n, t\}$
  The cell $C_i$ contains
  $$\begin{cases} a \text{ iff } x_i = y_i \\ b \text{ iff } x_i < y_i \end{cases}$$
- Transitions of $A_{\mathcal{M},w_0}$ are built from those of $\mathcal{M}$:
  - guards $x_i = y_i$ (resp. $x_i < y_i$) checks the current cell
  - and resetting $\{x_i\}$ (resp. $\{x_i, y_i\}$) writes $b$ (resp. $a$)
- An initial transition resets clocks according to $w_0$. 

Complexity of timed model checking

One clock = one component
⇒ same complexity as for “non-flat” systems.
One clock = one component
⇒ same complexity as for “non-flat” systems.

<table>
<thead>
<tr>
<th>Complexity of model checking non-flat systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model checking</td>
</tr>
<tr>
<td>Reachability</td>
</tr>
<tr>
<td>$CTL$ model checking</td>
</tr>
<tr>
<td>$AF \mu$-calculus</td>
</tr>
</tbody>
</table>

(Papadimitriou, Vardi, Kupferman, Wolper, Rabinovich, . . .)

BUT . . .
Undecidability is close

Timed automata require strong constraints on:

- clocks (synchronous, same speed)
- constraints \((x \sim k, x - y \sim k)\)
- updates \((x := 0, x := k)\)

Almost every extension leads to undecidability.
Undecidability is close

Timed automata require strong constraints on:

- clocks (synchronous, same speed)
- constraints ($x \sim k$, $x - y \sim k$)
- updates ($x := 0$, $x := k$)

Almost every extension leads to undecidability.

And... 

- Model-checking TLTL is undecidable
- Satisfiability of TCTL is undecidable
Outline

1 Timed properties

2 Decision procedures for TA
   - Timed automata
   - Model checking TA
   - Model-checking TCTL
   - Complexity

3 Algorithms in practice

4 Simply-timed systems

5 Conclusion
Complexity and algorithms

Tools use special data-structures and heuristics:

- DBM (Difference Bounded Matrix)
- CDD (Clock Difference Diagram)
- NDD (Numerical Decision Diagram)
- ... on-the-fly algorithms
- compositionnal algorithm
- ...
Complexity and algorithms

Tools use special data-structures and heuristics:

- **DBM** (Difference Bounded Matrix)
- **CDD** (Clock Difference Diagram)
- **NDD** (Numerical Decision Diagram)
- ... on-the-fly algorithms
- compositionnal algorithm
The DBM data structure

DBM (Difference Bounded Matrix) data structure [BM83, Dil90]

\[(x_1 \geq 3) \land (x_2 \leq 5) \land (x_1 - x_2 \leq 4)\]

\[
\begin{pmatrix}
  x_0 & x_1 & x_2 \\
  +\infty & -3 & +\infty \\
  +\infty & +\infty & 4 \\
  5 & +\infty & +\infty \\
\end{pmatrix}
\]
The DBM data structure

**DBM (Difference Bounded Matrice) data structure** [BM83, Dil90]

\[(x_1 \geq 3) \land (x_2 \leq 5) \land (x_1 - x_2 \leq 4)\]

\[
\begin{pmatrix}
    x_0 & x_1 & x_2 \\
    +\infty & -3 & +\infty \\
    +\infty & +\infty & 4 \\
    5 & +\infty & +\infty
\end{pmatrix}
\]

- Existence of a normal form

\[
\begin{pmatrix}
    0 & -3 & 0 \\
    9 & 0 & 4 \\
    5 & 2 & 0
\end{pmatrix}
\]
The DBM data structure

DBM (Difference Bounded Matrice) data structure [BM83, Dil90]

\[(x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)\]

\[
\begin{pmatrix}
  x_0 & x_1 & x_2 \\
  +\infty & -3 & +\infty \\
  +\infty & +\infty & 4 \\
  5 & +\infty & +\infty \\
\end{pmatrix}
\]

- Existence of a normal form

- All operations on valuations can be computed using DBMs
Reachability algorithm – UppAal

1. Algorithm : (\(\mathcal{A}\): timed automaton; \(L_f\): set of final locations) {
2. \(k := \) the maximal constant in \(\mathcal{A}\);
3. \(\text{Visited} := \emptyset\);
4. \(\text{Waiting} := \{(l_0, \text{Extra}_k(Z_0))\}\);
5. Repeat
6. Get and Remove \((l, Z)\) from \(\text{Waiting}\);
7. If \(l \in L_f\)
8. then \{Return “Yes” and a witness trace;\}
9. else \{If there is no \((l, Z')\) \(\in \text{Visited}\) s.t. \(Z \subseteq Z'\)
10. then \{\(\text{Visited} := \text{Visited} \cup \{(l, Z)\}\);\}
11. \(\text{Succ} := \{(l', \text{Extra}_k(\text{Post}(Z, e))) \mid e = (l, -, -, l')\}\);\}
12. \(\text{Waiting} := \text{Waiting} \cup \text{Succ};\}\}
13. Until (\(\text{Waiting} = \emptyset\));
14. Return “No”; }
As for standard systems, the reachability algorithm can be done in forward or in backward.

There is no special problems in backward analysis.

In forward analysis, termination requires a special operator (the \textit{k-extrapolation}).
(Otherwise constants used in DBM may grow for ever.)

This abstraction entails an over-approximation and the procedure is correct only for TA without diagonal constraints (see Bouyer’s works).

NB: forward analysis is more convenient for verifying extended TA (for ex. with integer variables).
Outline

1. Timed properties

2. Decision procedures for TA
   - Timed automata
   - Model checking TA
   - Model-checking TCTL
   - Complexity

3. Algorithms in practice

4. Simply-timed systems

5. Conclusion
Complexity of timed verification

- Reachability in TA with three clocks is PSPACE-C (Courcoubetis & Yannakakis)
- Model checking Timed CTL over TA is PSPACE-C (Alur, Courcoubetis & Dill)
- ...
Reachability in TA with three clocks is PSPACE-C (Courcoubetis & Yannakakis)

Model checking Timed CTL over TA is PSPACE-C (Alur, Courcoubetis & Dill)

...  

Are there simpler models for which timed model checking can be efficient?
Simple models for timed verification

It is possible to use classical Kripke structures as timed models. There is no inherent concept of time: time elapsing is encoded by events.

For example:

- each transition = one time unit (Emerson et al.)
- or: a “tick” proposition labels states where one t.u. elapses.

Timed model checking can be efficient (polynomial-time)!
Simple models for timed verification

It is possible to use classical Kripke structures as timed models. There is no inherent concept of time: time elapsing is encoded by events.

For example:

- each transition = one time unit (Emerson et al.)
- or: a “tick” proposition labels states where one t.u. elapses.

Timed model checking can be efficient (polynomial-time)!

Can we extend these models without loosing efficiency?
Model checking one clock TA

Theorem

1. Model checking $TCTL$ on 1C-TA is PSPACE-complete.
2. Reachability in 1C-TA is NLOGSPACE-complete.
3. Model checking $TCTL_{\leq,\geq}$ on 1C-TA is P-complete.

1 - Reduction from QBF.
Model checking one clock TA

Theorem

1. Model checking $TCTL$ on 1C-TA is PSPACE-complete.
2. Reachability in 1C-TA is NLOGSPACE-complete.
3. Model checking $TCTL_{\leq,\geq}$ on 1C-TA is P-complete.

2 - Idea: A configuration $(\ell, x)$ can be encoded as $(\ell, N_i)$ where $N_i$ is the number of the interval (whose bounds are constants in $\mathcal{A}$) containing $x$. 
Model checking one clock TA

Theorem

1. Model checking $TCTL$ on 1C-TA is PSPACE-complete.
2. Reachability in 1C-TA is NLOGSPACE-complete.
3. Model checking $TCTL_{\leq,\geq}$ on 1C-TA is P-complete.

3 - Sets of valuations are represented as lists of intervals.
Theorem

1. Reachability in 2C-TA is NP-hard.  
   (reduction from SUBSET-SUM)

2. Model checking $CTL$ on 2C-TA is PSPACE-complete.  
   (reduction from QBF)
Overview of results for 1-2 clocks TAs

[ACD93, CY92, LMS04]

<table>
<thead>
<tr>
<th></th>
<th>1C-TAs</th>
<th>2C-TAs</th>
<th>TAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reachability</td>
<td>NLOGSPACE-C</td>
<td>NP-hard</td>
<td>PSPACE-C</td>
</tr>
<tr>
<td>$TCTL_{\leq,\geq}$</td>
<td>P-complete</td>
<td>PSPACE-C</td>
<td>PSPACE-C</td>
</tr>
<tr>
<td>$TCTL$</td>
<td>PSPACE-complete</td>
<td>PSPACE-C</td>
<td>PSPACE-C</td>
</tr>
</tbody>
</table>

1C-TA are an interesting class!

- Language inclusion is decidable for 1C-TA (for finite timed words) – Abdulla et al.
- Checking emptiness of 1C-Alternating TA is decidable – Lasota & Walukiewicz.

And allows interesting extensions...
Problem: looking for efficient model-checking algorithms for probabilistic, non-deterministic and timed systems.

\[
\text{PTCTL} \ni \text{request} \Rightarrow \left( \mathbb{P}_{\geq 0.99} \mathbb{F}_{\leq 5} \text{response} \right)
\]

- Model-checking PTCTL formulae over PTA is \text{EXPTIME}-complete.
- Model checking PTCTL\(^0/1\) over one clock Probabilistic Timed Automata is \text{EXPTIME}-complete.
- The almost-sure reachability in two clocks PTA is \text{EXPTIME}-complete.
- Model checking PTCTL\(^{0/1}_{\leq, \geq}\) over one clock Probabilistic Timed Automata is \text{P}-complete.
Model-checking *Priced* Timed Automata

**Problem:** quantitative verification of timed automata extended with (discrete and continuous) prices (against WCTL properties).

- model-checking 3-clock priced timed automata is undecidable (can encode 2-counter machines) [BMM06].
- when there is only one clock, we can refine regions and get a (PSPACE) model-checking algorithm [BLM07].

And there are also some extensions to Timed *Games* [BLMR06,BCFL04].
Outline

1. Timed properties
2. Decision procedures for TA
   - Timed automata
   - Model checking TA
   - Model-checking TCTL
   - Complexity
3. Algorithms in practice
4. Simply-timed systems
5. Conclusion
Perspectives

- Timed model checkers:
  - Tackle complexity blow-up.
  - Symbolic data-structures
  - More expressive models (for specific applications).

- Adding probabilities, costs

- Timed control, timed games, synthesis.