Modeling and verification of real-time distributed systems (12h)

Benedikt Bollig
bollig@lsv.ens-cachan.fr

February 2014
Outline

Modeling
1. communication
   a) shared variables
   b) message passing
   c) timing constraints
2. time
   a) synchronous
   b) asynchronous
3. behavior
   a) timed
   b) untimed

Contents
1. distributed timed automata (1abc,2b,3b)
2. timed message sequence charts (1b,2a,3a)
3. timed Petri nets (1a,2a,3a)
Distributed Timed Automata

The Model

Existential Semantics and Region Abstraction
Universal Semantics and Undecidability
Reactive Semantics
Summary
Timed automata

Example:

\[ s_0 \quad x \leq 3 \quad y \leq 3 \]

\[ s_1 \quad x \geq 1, a \quad y \leq 0 \]

\[ s_2 \quad y \geq 1, b \]

- \( s_0 \) \( \xrightarrow{a} \) \( s_0 \) \( x = 0 \)
  - \( y = 0 \)
  - \( a = 0.7 \)

- \( s_0 \) \( \xrightarrow{a} \) \( s_1 \) \( x = 0 \)
  - \( y = 0 \)
  - \( a = 1.5 \)

- \( s_0 \) \( \xrightarrow{a} \) \( s_0 \) \( x = 0 \)
  - \( y = 0 \)
  - \( a = 2.6 \)

- \( s_1 \) \( \xrightarrow{a} \) \( s_1 \) \( x = 1.1 \)
  - \( y = 0 \)
  - \( a = 3.3 \)

- \( s_1 \) \( \xrightarrow{b} \) \( s_2 \) \( x = 1.8 \)
  - \( y = 0 \)
  - \( b = 5.0 \)

- \( s_2 \) \( x = 3.5 \)
  - \( y = 1.7 \)
Timed automata

Definition: timed automaton

A *timed automaton* is a tuple $\mathcal{A} = (S, \Sigma, X, T, \text{Inv}, s_0, F)$ where:

- $S$ is a finite set of *states*
- $\Sigma$ is the alphabet of *actions*
- $X$ is a finite set of *clocks*
- $T \subseteq S \times \Sigma \times \text{Constr}(X) \times 2^X \times S$ is the finite set of *transitions* ($\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$)
- $\text{Inv} : S \to \text{Constr}(X)$ associates with each state an *invariant*
- $s_0 \in S$ is the *initial state*
- $F \subseteq S$ is the set of *final states*

Here, the set $\text{Constr}(X)$ of *clock constraints* over $X$ is given by the grammar

$$\varphi ::= \text{true} \mid \text{false} \mid x \vartriangleleft c \mid \neg \varphi \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2$$

where $x$ ranges over $X$, $\vartriangleleft \in \{<, \leq, >, \geq, =\}$, and $c \in \mathbb{N} = \{0, 1, 2, \ldots\}$. Let $\text{Reset}(\mathcal{A}) = \{x \in X \mid \text{there is } (s, a, \varphi, R, s') \in T \text{ such that } x \in R\}$. We assume that $\text{Inv}(s_0)$ is “satisfied” by the clock valuation over $X$ that maps each clock to 0.
Distributed Timed Automata

**Definition:** distributed timed automaton

\[ D = ((A_p)_{p \in \text{Proc}}, \pi) \] where

- each \( A_p \) is a classical timed automaton
- \( \pi : X \rightarrow \text{Proc} \) assigns processes to clocks. If \( \pi(x) = p \) then
  - clock \( x \) evolves according to local time on process \( p \)
  - only process \( p \) may reset clock \( x \)
  - all processes may read clock \( x \) (i.e., use \( x \) in guards or invariants)

**Example:** DTA with \( \pi(x) = p \) and \( \pi(y) = q \)

\( A_p: \)

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

\( A_q: \)

\[ r_0 \xrightarrow{x \geq 1, b} r_1 \xrightarrow{y \leq 1} r_2 \]
Fix a finite set $\text{Proc}$ of processes.

**Definition: distributed timed automaton**

A *distributed timed automaton (DTA)* over $\text{Proc}$ is a structure $\mathcal{D} = ((\mathcal{A}_p)_{p \in \text{Proc}}, \pi)$:

- $\mathcal{A}_p = (S_p, \Sigma_p, X_p, T_p, \text{Inv}_p, s^p_0, F_p)$ is a timed automaton
- $\pi : X(\mathcal{D}) \to \text{Proc}$ where $X(\mathcal{D}) := \bigcup_{p \in \text{Proc}} X_p$
- for all $p \in \text{Proc}$, $\text{Reset}(\mathcal{A}_p) \subseteq \pi^{-1}(p) \subseteq X_p$
- for all $p, q \in \text{Proc}$, $p \neq q$ implies $\Sigma_p \cap \Sigma_q = \emptyset$

Syntactically, $\pi(x) = p$ means that

- only $p$ may reset $x$, but
- all processes can read $x$ (in guards or invariants).

Semantically, $\pi(x) = p$ means that

- $x$ evolves according to the local time of $p$. 

---

**Distributed timed automata**
Example: DTA with $\pi(x) = p$ and $\pi(y) = q$

$A_p: \quad s_0 \xrightarrow{y \leq 1, a} s_1 \xrightarrow{a, x:=0} s_2$

$A_q: \quad r_0 \xrightarrow{x \geq 1, b} r_1 \xrightarrow{0 < x < 1, b} r_2$
Local Times

- Processes do not have access to the absolute (global) time.
- Each process has its own local time: \( \tau_p : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0} \)
  \( \tau_p(t) \): local time on process \( p \) at absolute time \( t \)
- We require \( \tau_p(0) = 0 \) and that \( \tau_p \) is
  - continuous: \( \lim_{t \to t'} \tau_p(t) = \tau_p(t') \)
  - strictly increasing: \( t < t' \) implies \( \tau_p(t) < \tau_p(t') \)
  - diverging: for all \( t \), there is \( t' \) such that \( \tau_p(t') > t \)

We set \( Rates \) to be the set of tuples \( \tau = (\tau_p)_{p \in Proc} \) where each \( \tau_p \) is a local time function. Note that \( \tau \) can also be seen as a function \( \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}^{Proc} \).
Local Times

Example: Local times

\[ \tau_p \]

\[ \tau_q \]

\[ t \]
Runs of DTA & Untimed behaviours

Example: DTA $\mathcal{D}$ with $\pi(x) = p$ and $\pi(y) = q$

$A_p$: 

\[
\begin{array}{ccc}
 s_0 & \xrightarrow{y \leq 1, a} & s_1 & \xrightarrow{a, x:=0} & s_2 \\
 & 0.2 & & 0.6 & \\
 x = 0 & 0.4 & & 1.2 & \\
y = 0 & 0.2 & & 0.6 & \\
\end{array}
\]

$A_q$: 

\[
\begin{array}{ccc}
 r_0 & \xrightarrow{x \geq 1, b} & r_1 & \xrightarrow{0 < x < 1, b} & r_2 \\
 & 0.4 & & 0.2 & \\
x = 0 & 0.4 & & 0.2 & \\
y = 0 & 0.4 & & 0.2 & \\
\end{array}
\]

If $\tau_p > \tau_q$ then $abab \in L(\mathcal{D}, \tau)$ (e.g. $\tau_p(t) = 2t$ and $\tau_q(t) = t$)

\[
\begin{array}{cccccc}
 s_0 & a & s_1 & b & s_1 & a & s_2 & b & s_2 \\
r_0 & 0.2 & r_0 & 0.6 & r_1 & 0.7 & r_1 & 0.8 & r_2 \\
x = 0 & 0.4 & y = 0 & 0.2 & & 0.7 & & 0.8 & \\
y = 0 & 0.4 & & 0.2 & & 0.7 & & 0.8 & \\
\end{array}
\]

If $\tau_p = \tau_q$ then $abab \notin L(\mathcal{D}, \tau)$ (e.g. $\tau_p(t) = \tau_q(t) = 2t$)

\[
\begin{array}{cccccc}
 s_0 & a & s_1 & b & s_1 & a & s_2 \\
r_0 & 0.2 & r_0 & 0.5 & r_1 & 0.5 & r_1 \\
x = 0 & 0.4 & y = 0 & 0.4 & & 1 & & 1 & \\
y = 0 & 0.4 & & 1 & & 1 & & 1 & \\
\end{array}
\]
TA with independently evolving clocks

**Definition: icTA**

A timed automaton with independently evolving clocks (icTA) is a pair $\mathcal{B} = (A, \pi)$:

- $A = (S, \Sigma, X, T, \text{Inv}, s_0, F)$ is a timed automaton
- $\pi : X \rightarrow \text{Proc}$ assigns “processes” to clocks

If $\pi(x) = p$ then clock $x$ evolves according to local time $\tau_p$.

**Example: icTA $\mathcal{B}$ with $\pi(x) = p$ and $\pi(y) = q$**

![Diagram](image-url)
Semantics of icTA

For a valuation \( \nu : X \rightarrow \mathbb{R}_{\geq 0} \) and \( t = (t_p)_{p \in \text{Proc}} \in \mathbb{R}_{\geq 0}^{|\text{Proc}|} \), let \( \nu + t \) be the valuation defined by \( (\nu + t)(x) = \nu(x) + t_{\pi(x)} \). Intuitively, when time passes, we add to every clock the time elapse that corresponds to the local time of the owner of \( x \).

For \( \tau \in \text{Rates} \), a \( \tau \)-run of \( B \) is a sequence

\[
(s_0, \nu_0) \xrightarrow{a_1, t_1} (s_1, \nu_1) \xrightarrow{a_2, t_2} (s_2, \nu_2) \cdots (s_{n-1}, \nu_{n-1}) \xrightarrow{a_n, t_n} (s_n, \nu_n)
\]

where \( n \geq 0, s_i \in S, \nu_i : X \rightarrow \mathbb{R}_{\geq 0} \) (with \( \nu_0(x) = 0 \) for all \( x \in X \)), \( a_i \in \Sigma_{\varepsilon} \), and \( (t_i)_{1 \leq i \leq n} \) is a non-decreasing sequence of values from \( \mathbb{R}_{\geq 0} \). Further, for all \( i \in \{1, \ldots, n\} \), there are \( \varphi_i \in \text{Constr}(X) \) and \( R_i \subseteq X \) such that the following conditions hold (let \( t_0 = 0 \)):

1. \((s_{i-1}, a_i, \varphi_i, R_i, s_i) \in T\)
2. \( \nu_{i-1} + \tau(t_i) - \tau(t_{i-1}) \models \varphi_i \)
3. \( \nu_{i-1} + \tau(t') - \tau(t_{i-1}) \models \text{Inv}(s_{i-1}) \) for each \( t' \in [t_{i-1}, t_i] \)
4. \( \nu_i = (\nu_{i-1} + \tau(t_i) - \tau(t_{i-1}))[R_i \leftarrow 0] \)
5. \( \nu_i \models \text{Inv}(s_i) \)

In this case, we write \((B, \tau) : s_0 \xrightarrow{a_1 \cdots a_n} s_n\).
Semantics of icTA

Definition: existential and universal semantics of icTA

Let $\mathcal{B} = (S, \Sigma, X, T, \text{Inv}, s_0, F, \pi)$ be an icTA and $\tau \in Rates$. We let

$$L(\mathcal{B}, \tau) := \{w \in \Sigma^* \mid (\mathcal{B}, \tau) : s_0 \xrightarrow{w} s \text{ for some } s \in F\}$$

$$L_\exists(\mathcal{B}) := \bigcup_{\tau \in Rates} L(\mathcal{B}, \tau)$$

$$L_\forall(\mathcal{B}) := \bigcap_{\tau \in Rates} L(\mathcal{B}, \tau)$$

Aim: robustness of an icTA $\mathcal{B}$ against relative local times

Negative Specifications (Safety)

Given a set $\text{Bad}$ of undesired behaviours,

does an icTA $\mathcal{B}$ robustly avoid $\text{Bad}$, i.e., $L_\exists(\mathcal{B}) \cap \text{Bad} = \emptyset$?

Positive Specifications (Liveness)

Given a set $\text{Good}$ of desired behaviours,

does an icTA $\mathcal{B}$ robustly exhibit $\text{Good}$, i.e., $\text{Good} \subseteq L_\forall(\mathcal{B})$?
Semantics of icTA

Example:

Consider the following icTA $B$ over $Proc = \{p, q\}$ with $\pi(x) = p$ and $\pi(y) = q$:

- $L(B, \text{id}) = \{a, ab, b\}$
- $L_\exists(B) = \{a, ab, b, c\}$
- $L_\forall(B) = \{a, ab\}$
Semantics of icTA

One more reason to look at the existential and universal semantics is that the “concrete” semantics can have non-regular behaviors.

Example:

Consider the following icTA $B$, with independent clocks $x$ and $y$:

\[
\begin{align*}
&x = 1 \\
&x := 0 \\
&y = 1 \\
&y := 0 \\
&x, y \leq 1
\end{align*}
\]

Let $\tau = (id, \tau_q)$, where $\tau_q$ is any continuous, strictly increasing function such that $\tau_q(0) = 0$ and $\tau_q(n) = 2^n - 0.5$ for all $n \geq 1$.

Then, $L(B, \tau)$ is the set of finite prefixes of the infinite word $bab^2ab^4ab^8ab^{16}a \ldots$, which is not a regular language.
Semantics of DTA $\mathcal{D}$

Example: Part of the icTA $\mathcal{B}_\mathcal{D}$

$A_p:$

$\begin{align*}
A_p: & \quad s_0 \overset{y \leq 1, a}{\rightarrow} s_1 \overset{a, x:=0}{\rightarrow} s_2 \\
A_q: & \quad r_0 \overset{x \geq 1, b}{\rightarrow} r_1 \overset{0 < x < 1, b}{\rightarrow} r_2
\end{align*}$

$\begin{align*}
\epsilon, y \leq 1 \land x \geq 1 & \rightarrow (s_0, r_0), \emptyset \\
\epsilon, x \geq 1 & \rightarrow (s_1, r_0), \{a\} \\
\epsilon, y \leq 1 & \rightarrow (s_1, r_0), \{b\} \\
\epsilon, y \leq 1 & \rightarrow (s_1, r_1), \{a\} \\
\epsilon, y \leq 1 & \rightarrow (s_1, r_1), \emptyset \\
z:=0, y \leq 1 & \rightarrow (s_1, r_0), \emptyset \\
z:=0, y \leq 1 & \rightarrow (s_1, r_1), \emptyset \\
z:=0 & \rightarrow (s_0, r_0), \emptyset
\end{align*}$

$(T1)$ $(T2)$
Semantics of DTA $\mathcal{D}$ (formally)

**Definition: Semantics of DTA**

Let $\mathcal{D} = ((\mathcal{A}_p)_{p \in \text{Proc}}, \pi)$ be a DTA where $\mathcal{A}_p = (\mathcal{S}_p, \Sigma_p, \mathcal{X}_p, T_p, \text{Inv}_p, s^p_0, F_p)$. We associate with $\mathcal{D}$ the icTA $\mathcal{B}_\mathcal{D} = (S, \Sigma, X, T, \text{Inv}, s^0, F, \pi')$ where

- $\Sigma = \bigcup_{p \in \text{Proc}} \Sigma_p$
- $S = (\prod_{p \in \text{Proc}} S_p) \times 2^\Sigma$
- $X = \{z\} \uplus \bigcup_{p \in \text{Proc}} X_p$
- $s^0 = ((s^p_0)_{p \in \text{Proc}}, \emptyset)$ and $F = (\prod_{p \in \text{Proc}} F_p) \times \{\emptyset\}$
- for $s = (s_p)_{p \in \text{Proc}} \in \prod_{p \in \text{Proc}} S_p$ and $A \subseteq \Sigma$ with $A \neq \emptyset$:
  \[
  \text{Inv}(s, \emptyset) = \bigwedge_{p \in \text{Proc}} \text{Inv}_p(s_p)
  \]
  \[
  \text{Inv}(s, A) = z \leq 0 \land \bigwedge_{p \in \text{Proc}} \text{Inv}_p(s_p)
  \]
- $\pi'(z)$ is any process, and $\pi'$ restricted to $X \setminus \{z\}$ is just $\pi$
Semantics of DTA $\mathcal{D}$ (formally)

Definition: (cntd.)

The transitions in $\mathcal{B}_\mathcal{D}$ are of two types:

(T1) $((s, \emptyset), \varepsilon, \varphi, R, (s', A)) \in T$

if there are $\emptyset \neq P \subseteq \text{Proc}$ and $(\tilde{s}_p, a_p, \varphi_p, R_p, \tilde{s}'_p) \in T_p, p \in P$, such that:

- $s_p = \tilde{s}_p$ and $s'_p = \tilde{s}'_p$ for all $p \in P$
- $s_q = s'_q$ for all $q \in \text{Proc} \setminus P$
- $\varphi = \bigwedge_{p \in P} \varphi_p$, $R = \bigcup_{p \in P} R_p \cup \{z\}$, and $A = \{a_p | p \in P\} \setminus \{\varepsilon\}$

(T2) $((s, A), a, \text{true}, \emptyset, (s, A \setminus \{a\})) \in T$

for all $s \in \prod_{p \in \text{Proc}} S_p$, $A \subseteq \Sigma$, and $a \in A$

Definition:

- $L(\mathcal{D}, \tau) := L(\mathcal{B}_\mathcal{D}, \tau)$
- $L_\exists(\mathcal{D}) := L_\exists(\mathcal{B}_\mathcal{D})$
- $L_\forall(\mathcal{D}) := L_\forall(\mathcal{B}_\mathcal{D})$
Semantics of DTA

Example: DTA $\mathcal{D}$ with $\pi(x) = p$ and $\pi(y) = q$

\[ A_p: \quad s_0 \xrightarrow{y \leq 1, a} s_1 \xrightarrow{a, x:=0} s_2 \]

\[ A_q: \quad r_0 \xrightarrow{x \geq 1, b} r_1 \xrightarrow{y \leq 1} 0 < x < 1, b \xrightarrow{} r_2 \]

- if $\tau_p > \tau_q$, then $L(\mathcal{D}, \tau) = \{aa, abab, baab\}$
- if $\tau_p = \tau_q$, then $L(\mathcal{D}, \tau) = \{aa\}$
- $L_\exists(\mathcal{D}) = \{aa, abab, baab\}
- L_\forall(\mathcal{D}) = \{aa\}$

Exercise:

Let $Proc = \{p, q\}$. Show that, for every regular language $L \subseteq \{a, b\}^*$, there is a DTA $\mathcal{D} = ((A_p, A_q), \pi)$ with $A_p = (S_p, \{a\}, X_p, T_p, Inv_p, s_0^p, F_p)$ and $A_q = (S_q, \{b\}, X_q, T_q, Inv_q, s_0^q, F_q)$ such that $L_\exists(\mathcal{D}) = L_\forall(\mathcal{D}) = L$. Note that you may choose the sets of clocks and $\pi$ freely.
Distributed Timed Automata

The Model

- Existential Semantics and Region Abstraction
- Universal Semantics and Undecidability
- Reactive Semantics

Summary
Existential semantics is regular

Goal:
Transform icTA $\mathcal{B}$ into finite automaton recognizing the existential semantics of $\mathcal{B}$.

We proceed in two steps:

1. We define an (infinite, untimed) automaton/transition system $TS(\mathcal{B})$ over the alphabet of $\mathcal{B}$ such that $L(TS(\mathcal{B})) = L_\exists(\mathcal{B})$.

2. We define a bisimulation equivalence relation $\sim$ on the set of states of $TS(\mathcal{B})$ such that:
   - $\sim$ has finite index (finitely many equivalence classes)
   - $L(TS(\mathcal{B})/\sim) = L(TS(\mathcal{B}))$

As $TS(\mathcal{B})/\sim$ is a finite automaton, $L_\exists(\mathcal{B})$ is indeed regular. Moreover, the construction of $TS(\mathcal{B})/\sim$ is effective.
Step 1: The infinite transition system

We define the transition system $TS(B)$ for icTA $\mathcal{B} = (S, \Sigma, X, T, Inv, s_0, F, \pi)$:

- states: pairs $(s, \nu)$ where $s \in S$ and $\nu : X \rightarrow \mathbb{R}_{\geq 0}$
- initial state: $(s_0, \nu_0)$ with $\nu_0(x) = 0$ for all $x \in X$
- final states: states $(s, \nu)$ such that $s \in F$

Then, for $a \in \Sigma_{\varepsilon}$, $(s, \nu) \xrightarrow{a} (s', \nu')$ is a transition in $TS(B)$ if there exist $t \in \mathbb{R}_{\geq 0}$, $\tau \in Rates$, $\varphi \in Constr(X)$, and $R \subseteq X$ such that:

- $(s, a, \varphi, R, s') \in T$
- $\nu + \tau(t) \models \varphi$
- $\nu + \tau(t') \models Inv(s)$ for each $t' \in [0, t]$
- $\nu' = (\nu + \tau(t))[R \leftarrow 0]$
- $\nu' \models Inv(s')$

The language $w \in L(TS(B)) \subseteq \Sigma^*$ of $TS(B)$ is defined as expected.
Step 1: The infinite transition system

**Lemma:**

\[ L(TS(B)) = L_\exists(B) \]

**Exercise:**

Proof of \( \supseteq \).

(\( \tau \)-run \( \rightsquigarrow \) abstract away \( t_i \) \( \rightsquigarrow \) accepting run of \( TS(B) \))

Proof of \( \subseteq \).

Let \( w \in L(TS(B)) \) and let

\[
(s_0, \nu_0) \xrightarrow{a_1} (s_1, \nu_1) \xrightarrow{a_2} (s_2, \nu_2) \cdots (s_{n-1}, \nu_{n-1}) \xrightarrow{a_n} (s_n, \nu_n)
\]

be an accepting run of \( TS(B) \) for \( w = a_1 \cdot \ldots \cdot a_n \). By definition, for each \( 1 \leq i \leq n \), we find \( \hat{t}_i \geq 0 \), \( \tau_i \), \( \varphi_i \), and \( R_i \) such that:

- \((s_{i-1}, a_i, \varphi_i, R_i, s_i) \in T\)
- \(\nu_{i-1} + \tau_i(\hat{t}_i) \models \varphi_i\)
- \(\nu_{i-1} + \tau_i(t') \models \text{Inv}(s_{i-1})\) for each \( t' \in [0, \hat{t}_i]\)
- \(\nu_i = (\nu_{i-1} + \tau_i(\hat{t}_i))[R_i \leftarrow 0]\)
- \(\nu_i \models \text{Inv}(s_i)\)
Step 1: The infinite transition system

Lemma:

\[ L(TS(B)) = L_\exists(B) \]

Proof (cntd.) of \( \subseteq \).

Towards a \( \tau \)-run of \( B \), we define by induction the non-decreasing sequence \( t_0, t_1, \ldots, t_n \) by \( t_0 = 0 \) and \( t_i = t_{i-1} + \hat{t}_i \) for \( 1 \leq i \leq n \).

We also define \( \tau \) in order to obtain a \( \tau \)-run of \( B \):

\[
\tau(t) = \begin{cases} 
\tau(t_{i-1}) + \tau_i(t - t_{i-1}) & \text{if } t \in [t_{i-1}, t_i] \\
\tau(t_n) + \text{id}(t - t_n) & \text{if } t \geq t_n
\end{cases}
\]

Then, we can check:

\begin{itemize}
  \item \( (s_{i-1}, a_i, \varphi_i, R_i, s_i) \in T \)
  \item \( \nu_{i-1} + \tau(t_i) - \tau(t_{i-1}) \models \varphi_i \)
  \item \( \nu_{i-1} + \tau(t') - \tau(t_{i-1}) \models \text{Inv}(s_{i-1}) \) for each \( t' \in [t_{i-1}, t_i] \)
  \item \( \nu_{i-1} + \tau_i(t') \models \text{Inv}(s_{i-1}) \) for each \( t' \in [0, \hat{t}_i] \)
  \item \( \nu_i = (\nu_{i-1} + \tau(t_i) - \tau(t_{i-1}))[R_i \leftarrow 0] \)
  \item \( \nu_i \models \text{Inv}(s_i) \)
\end{itemize}

Therefore, \( (s_0, \nu_0) \xrightarrow{a_1,t_1} (s_1, \nu_1) \xrightarrow{a_2,t_2} (s_2, \nu_2) \cdots (s_{n-1}, \nu_{n-1}) \xrightarrow{a_n,t_n} (s_n, \nu_n) \) is an accepting \( \tau \)-run of \( B \).
Step 2: The bisimulation relation

Next, we define a bisimulation relation on $TS(B)$ that has finite index and preserves final states. We obtain as a quotient a finite automaton accepting $L(TS(B)) = L_\exists(B)$.

Idea:
Bisimulation is based on clock regions; consider two clock valuations to be equivalent if they are $p$-equivalent for every process $p$. In turn, $p$-equivalence is just the usual region equivalence for classical timed automata.

Regions when $\pi(x) = \pi(y)$

Regions when $\pi(x) \neq \pi(y)$
Step 2: The bisimulation relation

For each clock $x \in X$, let $C_x$ be the largest constant clock $x$ is compared with in guards or invariants. Let $p \in \text{Proc}$. Given a clock valuation $\nu$ over $X$, define its $p$-restriction $\nu_p : \pi^{-1}(p) \to \mathbb{R}_{\geq 0}$ by $\nu_p(x) = \nu(x)$ for all $x \in \pi^{-1}(p)$.

As in the classical region construction for timed automata, we obtain a notion of equivalence $\sim_p$ between two such valuations: $\nu_p \sim_p \nu_p'$ if the following hold:

1. for each $x \in \pi^{-1}(p)$, $\nu_p(x) > C_x$ if and only if $\nu_p'(x) > C_x$
2. for each $x \in \pi^{-1}(p)$, $\nu_p(x) \leq C_x$ implies both $\lfloor \nu_p(x) \rfloor = \lfloor \nu_p'(x) \rfloor$ and $\text{fract}(\nu_p(x)) = 0$ iff $\text{fract}(\nu_p'(x)) = 0$
3. for each pair $x, y \in \pi^{-1}(p)$ such that $\nu_p(x) \leq C_x$ and $\nu_p(y) \leq C_y$, we have $\text{fract}(\nu_p(x)) \leq \text{fract}(\nu_p(y))$ iff $\text{fract}(\nu_p'(x)) \leq \text{fract}(\nu_p'(y))$

Remark:

From the result on timed automata, it follows that each $\sim_p$ is an equivalence relation and also a time-abstract bisimulation, i.e, if $\nu_p \sim_p \nu_p'$, then for all $t \in \mathbb{R}_{>0}$, there exists $t' \in \mathbb{R}_{>0}$ such that $\nu_p + t \sim \nu_p' + t'$.

Now, we say that two clock valuations $\nu$ and $\nu'$ over $X$ are equivalent, denoted $\nu \sim \nu'$, if $\nu_p \sim_p \nu_p'$ for all $p \in \text{Proc}$. For valuation $\nu$, let $[\nu]_{\sim}$ (or, simply, $[\nu]$) denote its equivalence class wrt. $\sim$. 
Step 2: The bisimulation relation

**Lemma: Time-abstract bisimulation**

If $\nu \sim \nu'$, then for all $t \in \mathbb{R}^{Proc}_{>0}$, there exists $t' \in \mathbb{R}^{Proc}_{>0}$ such that $\nu + t \sim \nu' + t'$.

**Exercise:**

Prove the lemma.

The equivalence $\sim$ can be naturally extended to states of $TS(\mathcal{B})$ by $(s, \nu) \sim (s', \nu')$ if $s = s'$ and $\nu \sim \nu'$.

To show that this defines a bisimulation relation on $TS(\mathcal{B})$, we first introduce the successor relation on regions.

**Definition:**

Let $\gamma$ and $\gamma'$ be two clock regions. We say that $\gamma'$ is accessible from $\gamma$, written $\gamma \preceq \gamma'$, if either $\gamma = \gamma'$ or there are $\nu \in \gamma$, $\nu' \in \gamma'$, $t \in \mathbb{R}^{Proc}_{>0}$ such that $\nu' = \nu + t$.

Note that $\preceq$ is a partial-order relation. The direct successor relation, written $\gamma \prec \gamma'$, is as usual defined by $\gamma \preceq \gamma'$, $\gamma \neq \gamma'$, and $\gamma'' = \gamma$ or $\gamma'' = \gamma'$ for all clock regions $\gamma''$ with $\gamma \preceq \gamma'' \preceq \gamma'$.
Step 2: The bisimulation relation

Lemma: Bisimulation

If \((s, \nu) \sim (s, \hat{\nu})\) and \((s, \nu) \xrightarrow{a} (s', \nu')\) then \((s, \hat{\nu}) \xrightarrow{a} (s', \hat{\nu}')\) for some \(\hat{\nu}' \sim \nu'\).

Proof:

Assume \((s, \nu) \xrightarrow{a} (s', \nu')\). Let \(t \in \mathbb{R}_{\geq 0}\), \(\tau \in Rates\), \(\varphi \in Constr(X)\) and \(R \subseteq X\) such that ... hold. Consider regions \(\gamma_0 \prec \gamma_1 \prec \cdots \prec \gamma_n\) visited along \(\nu + \tau[0 \ldots t]\):

- there are \(0 = t_0 < t_1 < \cdots < t_n = t\) such that, for \(0 \leq i \leq n\), we have \(\gamma_i = [\nu_i]\) with \(\nu_i = \nu + \tau(t_i) = \nu_{i-1} + \tau(t_i) - \tau(t_{i-1})\)
- for any \(1 \leq i \leq n\) and all \(t_{i-1} < t' < t_i\) we have \(\nu + \tau(t') \in \gamma_{i-1} \cup \gamma_i\)

Assume \((s, \nu) \sim (s, \hat{\nu})\). We construct \(\hat{\tau}\) such that, for each \(0 \leq i \leq n\), we have

\[P(i): \hat{\nu}_i = \hat{\nu} + \hat{\tau}(t_i) \sim \nu_i\]

We start with \(\hat{\tau}(0) = 0\) so that \(P(0)\) holds.

Let now \(1 \leq i \leq n\) and assume we have constructed \(\hat{\tau}\) up to \(t_{i-1}\) with \(P(i-1)\).

Using Lemma [Time-abstract bisimulation], we find \(\hat{\tau} \in \mathbb{R}_{\geq 0}^{Proc}\) such that \(\hat{\nu}_{i-1} + \hat{\tau} \sim \nu_{i-1} + \tau(t_i) - \tau(t_{i-1}) = \nu_i\).

Define \(\hat{\tau}\) on \([t_{i-1}, t_i]\) using a linear interpolation such that \(\hat{\tau}(t_i) = \hat{\tau}(t_{i-1}) + \hat{\tau}\).

We obtain \(\hat{\nu}_i = \hat{\nu} + \hat{\tau}(t_i) = \hat{\nu} + \hat{\tau}(t_{i-1}) + \hat{\tau} = \hat{\nu}_{i-1} + \hat{\tau} \sim \nu_i\ (P(i))\).

Finally, for \(t' \geq t_n = t\), we let \(\hat{\tau}(t') = \hat{\tau}(t_n) + \text{id}(t' - t_n)\).
Lemma: Bisimulation

If \((s, \nu) \sim (s, \hat{\nu})\) and \((s, \nu) \xrightarrow{a} (s', \nu')\) then \((s, \hat{\nu}) \xrightarrow{a} (s', \hat{\nu}')\) for some \(\hat{\nu}' \sim \nu'\).

Proof (cntd.):
For any \(1 \leq i \leq n\) and all \(t_{i-1} < t' < t_i\) we have

\[
\gamma_{i-1} = [\hat{\nu}_{i-1}] \preceq [\hat{\nu} + \hat{\tau}(t')] \preceq [\hat{\nu}_i] = \gamma_i
\]

and since \(\gamma_{i-1} \not\prec \gamma_i\) we obtain \(\hat{\nu} + \hat{\tau}(t') \in \gamma_{i-1} \cup \gamma_i\).

Therefore, \(\hat{\nu} + \hat{\tau}(t') \models \text{Inv}(s)\) for all \(t' \in [0, t]\) and \(\hat{\nu}_n = \hat{\nu} + \hat{\tau}(t) \models \varphi\).

We let \(\hat{\nu}' = \hat{\nu}_n[R \leftarrow 0] \sim \nu_n[R \leftarrow 0] = \nu'\).

We have \(\hat{\nu}' \models \text{Inv}(s')\) and we deduce that \((s, \hat{\nu}) \xrightarrow{a} (s', \hat{\nu}')\) in \(TS(\mathcal{B})\). \(\square\)
Step 2: The bisimulation relation

It remains to consider the finite quotient $TS(B)/\sim$, which is the finite transition system defined as follows:

- states: equivalence classes $[(s, \nu)]$
- initial state: $[(s_0, \nu_0)]$
- final states: $[(s, \nu)]$ with $s \in F$
- transitions: $[(s, \nu)] \xrightarrow{a} [(s', \nu')]$ for all transitions $(s, \nu) \xrightarrow{a} (s', \nu')$ of $TS(B)$

Since the bisimulation equivalence relation $\sim$ on $TS(B)$ preserves final states, we obtain:

**Corollary:**

$L(TS(B)/\sim) = L(TS(B))$
The region automaton

The finite automaton $TS(B)/\sim$ is not exactly what is usually called the *region automaton* in the classical theory of timed automata.

**Definition: Region automaton**

The region automaton associated with $B$ is actually

$$\mathcal{R}_B = (S', \Sigma, T', s'_0, F')$$

where

1. $S' = S \times \text{Regions}(B)$ with $\text{Regions}(B) = \{[\nu] \mid \nu : X \to \mathbb{R}_{\geq 0}\}$
2. $s'_0 = (s_0, [\nu_0])$
3. $F' = F \times \text{Regions}(B)$
4. for all $a \in \Sigma_\epsilon$ and $s, s' \in S$ and $\gamma, \gamma' \in \text{Regions}(B)$,
   $T'$ contains $(s, \gamma) \xrightarrow{a} (s', \gamma')$ if one of the following holds:
   - $a = \epsilon$, $s = s'$, $\gamma \prec \gamma'$, and $\nu' \models \text{Inv}(s)$ for some $\nu' \in \gamma'$
     (time-elapse transition)
   - there are $\nu \in \gamma$ and $(s, a, \varphi, R, s') \in T$ such that $\nu \models \varphi \land \text{Inv}(s)$, $\nu[R \leftarrow 0] \models \text{Inv}(s')$, and $\nu[R \leftarrow 0] \in \gamma'$
     (discrete transition).
The region automaton

Example:
A part of $\mathcal{R}_B$ for the icTA $B$ from a previous example:
The region automaton

A sequence of time-elapse transitions followed by a discrete transition of $R_B$ is a transition of $TS(B)/\sim$.

Any transition of $TS(B)/\sim$ can be decomposed into a sequence of time-elapse transitions followed by a discrete transition of $R_B$.

**Theorem:**

Let $B = (S, \Sigma, X, T, \text{Inv}, s_0, F, \pi)$ be an icTA and let $C$ be the largest constant a clock is compared with in $B$. Then, the number of states of $TS(B)/\sim$ and of $R_B$ is bounded by $|S| \cdot (2C + 2)^{|X|} \cdot |X|!$ and we have

$$L(R_B) = L(TS(B)/\sim) = L(TS(B)) = L_\exists(B)$$

which is therefore a regular word language.

Assume a regular set $Bad$. Then, since $L_\exists(B)$ is a regular word language, so is $L_\exists(B) \cap Bad$. Thus, we solved the verification problem stated at the beginning:

**Theorem:**

Model checking icTA/DTA wrt. regular safety specifications is decidable.
Distributed Timed Automata

The Model

Existential Semantics and Region Abstraction

- Universal Semantics and Undecidability

Reactive Semantics

Summary
Universal semantics and undecidability

Now, we turn to positive/liveness specifications, which amounts to checking containment in the universal semantics.

Unfortunately, emptiness and universality are undecidable, both for icTA and DTA. We will only show the following result:

**Theorem:**

The following problem is undecidable:

**Input:** Finite set $Proc$ of processes, icTA $B$ over $Proc$

**Question:** $L_{\forall}(B) = \emptyset$?

**Proof:**

The proof is by reduction from Post’s correspondence problem (PCP):

**Post’s correspondence problem (PCP)**

**Input:** Alphabet $A$ and morphisms $f, g : A^* \rightarrow \{0, 1\}^*$.

**Question:** Is there $w \in A^+$ such that $f(w) = g(w)$?
PCP encoding

Let $A = \{a_1, \ldots, a_k\}$, where $k \geq 1$, and $f, g$ be an instance of the PCP.

**Goal:**

An icTA $B$ over $Proc = \{p, q\}$ such that $L_B = \{w \in A^+ \mid f(w) = g(w)\}$.

Idea: Encode sequences over $\{0, 1\}$ in terms of local time functions. Pair $\tau = (\tau_p, \tau_q) \in Rates$ encodes word in $\{0, 1, 2\}^\omega$ using $1 \times 1$-square regions. In the figure, we obtain $dir(\tau) = 101 \ldots$
PCP encoding

With $\tau$, we associate sequences

- $t$-$dir(\tau) = t_1 t_2 \ldots \in (\mathbb{R}_{\geq 0})^\omega$ of time instances
- $dir(\tau) = d_1 d_2 \ldots \in \{0, 1, 2\}^\omega$ of directions

For $i \geq 1$, let $t_i = \min\{t > t_{i-1} \mid \tau_r(t) - \tau_r(t_{i-1}) = 1 \text{ for some } r \in \text{Proc}\}$ (assuming $t_0 = 0$). With this, we set

$$d_i = \begin{cases} 
0 & \text{if } \tau_p(t_i) - \tau_p(t_{i-1}) < 1 \text{ and } \tau_q(t_i) - \tau_q(t_{i-1}) = 1 \\
1 & \text{if } \tau_q(t_i) - \tau_q(t_{i-1}) < 1 \text{ and } \tau_p(t_i) - \tau_p(t_{i-1}) = 1 \\
2 & \text{otherwise}
\end{cases}$$

To "detect" this encoding in automata, we introduce independent clocks $X = \{x, y\}$ ($x$ for $p$, and $y$ for $q$) and use the following guards:

$$guard(0) = (x < 1 \land y = 1)$$
$$guard(1) = (y < 1 \land x = 1)$$
$$guard(2) = (x = 1 \land y = 1)$$

Let $\overline{guard(d)} = \bigvee_{d' \in \{0, 1, 2\} \setminus \{d\}} \overline{guard(d')}$ be the "negation" of $guard$.

Applying a new square then corresponds to resetting both clocks at the same time.
PCP encoding

Goal:
An icTA $\mathcal{B} = (S, A, X, T, \text{Inv}, s_0, F, \pi)$ over $\text{Proc} = \{p, q\}$ and $A = \{a_1, \ldots, a_k\}$ such that $L_{\forall}(\mathcal{B}) = \{w \in A^+ \mid f(w) = g(w)\}$. Recall that $X = \{x, y\}$ with $\pi(x) = p$ and $\pi(y) = q$.

We proceed in two steps:

1. We construct icTA

   $\mathcal{B}_f = (S_f, A, X, T_f, \text{Inv}_f, s^f_0, F_f, \pi)$
   $\mathcal{B}_g = (S_g, A, X, T_g, \text{Inv}_g, s^g_0, F_g, \pi)$

   over $\text{Proc}$ such that, for all $\tau \in \text{Rates}$:

   $L(\mathcal{B}_f, \tau) = \{ w \in A^+ \mid f(w).2 \not\preceq \text{dir}(\tau) \}$
   $L(\mathcal{B}_g, \tau) = \{ w \in A^+ \mid g(w).2 \preceq \text{dir}(\tau) \}$

   Here, $\preceq$ denotes the prefix relation.

2. We build $\mathcal{B} = \mathcal{B}_f \lor \mathcal{B}_g$, which branches non-deterministically into $\mathcal{B}_f$ or $\mathcal{B}_g$. 
PCP encoding

So assume the following:

\[ L(B_f, \tau) = \{ w \in A^+ \mid f(w)_2 \not\leq \text{dir}(\tau) \} \]

\[ L(B_g, \tau) = \{ w \in A^+ \mid g(w)_2 \leq \text{dir}(\tau) \} \]

\[ B = B_f \lor B_g \]

**Lemma:**

\[ L_\forall(B) = \{ w \in A^+ \mid f(w) = g(w) \} \]

**Proof:**

"\( \subseteq \)" : Let \( w \in L_\forall(B) \). Then, \( w \in A^+ \) and, for all \( \tau \in \text{Rates} \), \( w \in L(B_f, \tau) \) or \( w \in L(B_g, \tau) \), i.e., \( f(w)_2 \not\leq \text{dir}(\tau) \) or \( g(w)_2 \leq \text{dir}(\tau) \). Pick one \( \tau \) such that \( f(w)_2 \leq \text{dir}(\tau) \). As then \( g(w)_2 \leq \text{dir}(\tau) \) and \( f(w), g(w) \in \{0, 1\}^* \), we have \( f(w) = g(w) \).

"\( \supseteq \)" : Let \( w \in A^+ \) such that \( f(w) = g(w) \). Let \( \tau \in \text{Rates} \). Trivially, it holds \( f(w)_2 \not\leq \text{dir}(\tau) \) or \( f(w)_2 \leq \text{dir}(\tau) \). As \( f(w) = g(w) \), the latter implies \( g(w)_2 \leq \text{dir}(\tau) \). Therefore, \( w \in L(B, \tau) \). We conclude \( w \in L_\forall(B) \). \( \square \)
PCP encoding

It remains to build $\mathcal{B}_f$ and $\mathcal{B}_g$. Given $a \in A$, $\sigma = d_1 \ldots d_n \in \{0, 1, 2\}^+$ (with $d_j \in \{0, 1, 2\}$ for any $j \in \{1, \ldots, n\}$) and $i \in \{1, 2\}$, we use the transition macro:

$$s \xrightarrow{(a, \sigma)} r_i$$

which actually stands for the following sequence of transitions:

$$s \xrightarrow{a, \text{guard}(d_1)} \{x, y\} \xrightarrow{\varepsilon, \text{guard}(d_2)} \{x, y\} \xrightarrow{\varepsilon, \text{guard}(d_3)} \ldots \xrightarrow{\varepsilon, \text{guard}(d_n)} \{x, y\} \xrightarrow{s_i} r_i$$
PCP encoding

The final automaton $\mathcal{B}$ is given by the following figure. Hereby, its left part depicts $\mathcal{B}_f$, its right part $\mathcal{B}_g$.

This concludes the proof of undecidability.

To show the same for DTA, let $\mathcal{D} = \mathcal{B}[x] \parallel \mathcal{B}[y]$ where we add invariants $x \leq 0 \land y \leq 0$:

$$L_\forall(\mathcal{D}) = L_\forall(\mathcal{B})$$
Undecidability of the universal semantics

Theorem: Undecidability

Let $\mathcal{D}$ be an icTA/DTA.

- **emptiness:** $L_\forall(\mathcal{D}) = \emptyset$ is undecidable.
- **universality:** $L_\forall(\mathcal{D}) = \Sigma^*$ is undecidable.

Even for 2 processes, 1 clock each and bounded drifts: $\exists \alpha > 0$, $\forall t > 0$,

$$1 - \alpha \leq \frac{\tau_q(t)}{\tau_p(t)} < 1 + \alpha \quad \text{or} \quad |\tau_q(t) - \tau_p(t)| \leq \alpha$$

Corollary: Positive specifications

$$\text{Good} \subseteq L_\forall(\mathcal{D})$$

Model checking regular positive specifications for DTA is undecidable.
Distributed Timed Automata

The Model

Existential Semantics and Region Abstraction

Universal Semantics and Undecidability

Reactive Semantics

Summary
### Reactive semantics

Goal of this section: Define a (non-trivial) reactive semantics such that:

#### Theorem: Regularity

For all icTA $\mathcal{B}$:

- $L_{\text{react}}(\mathcal{B})$ is regular
- $L_{\text{react}}(\mathcal{B}) \subseteq L_{\forall}(\mathcal{B})$

#### Corollary: Positive specifications

Model checking regular positive specifications is decidable for the reactive semantics.

$$\text{Good} \subseteq L_{\text{react}}(\mathcal{B})$$
Semantics as a game

Existential semantics: 1-Player game
- Player 1 controls both transitions and time
- $L_{\exists}(D) = \{ w \in \Sigma^* \mid \text{Player 1 has a winning strategy for } w \}$

Universal semantics: 2-Player game with imperfect information
- Player 1 controls transitions
- Player 2 controls time with imperfect information
- $L_{\forall}(D) = \{ w \in \Sigma^* \mid \text{Player 1 has a winning strategy for } w \}$

Reactive semantics: 2-Player game
- Player 1 controls transitions
- Player 2 controls time
- $L_{\text{react}}(D) = \{ w \in \Sigma^* \mid \text{Player 1 has a winning strategy for } w \}$
Reactive Semantics

Idea:
- System observes current region and controls discrete transitions
- Environment controls how local times evolve (time-elapse transitions)
- Not turn-based: system may execute several discrete transitions

$L_{react}(D) = \{w \in \Sigma^* | \text{System has a winning strategy for } w\}$
Reactive Semantics

**Definition: alternating automaton**

An alternating automaton (AA) is a tuple $\mathcal{A} = (S, \Sigma, \delta, s_0, F)$ where

- $S$, $\Sigma$, $s_0$, and $F$ are as usual, and
- $\delta : S \times \Sigma_\epsilon \rightarrow \mathbb{B}^+(S)$ is the *transition function*.

Here, $\mathbb{B}^+(S)$ denotes positive boolean combinations of states from $S$.

A run of $\mathcal{A}$ on $w = a_1 \ldots a_n \in \Sigma^*$ is a labeled finite tree $\rho = (V, \sigma, \mu)$ where

- $V \subseteq \mathbb{N}^*$ is the nonempty, finite, prefix-closed set of nodes,
- $\sigma : V \rightarrow S$ and $\mu : V \rightarrow \{0, \ldots, n\}$ are labeling functions

such that, for each node $u \in V$, the following hold:

- if $u$ is the root, then $\sigma(u) = s_0$ and $\mu(u) = 0$
- if $u$ is not a leaf (i.e., $\text{children}(u) \neq \emptyset$), then we have
  - either $\mu(u') = \mu(u)$ for all $u' \in \text{children}(u)$ and
    $\{\sigma(u') \mid u' \in \text{children}(u)\} \models \delta(\sigma(u), \epsilon)$
  - or $\mu(u') = \mu(u) + 1 = i \leq n$ for all $u' \in \text{children}(u)$ and
    $\{\sigma(u') \mid u' \in \text{children}(u)\} \models \delta(\sigma(u), a_i)$

The run is accepting if all leaves are labeled with $F \times \{n\}$. The set of words from $\Sigma^*$ that come with an accepting run is denoted by $L(\mathcal{A})$. 
Reactive Semantics

Lemma: Birget 1993

Given an AA $A$ with $n$ states, one can construct a nondeterministic finite automaton with $2^{O(n^2)}$ states that recognizes $L(A)$.

Let $B = (S, \Sigma, X, T, \text{Inv}, s_0, F, \pi)$ be an icTA over Proc. We associate with $B$ an AA $A_B = (S', \Sigma, \delta', s'_0, F')$ as follows:

- $S' = S \times \text{Regions}(B) \times \{\exists, \forall\}$ and $F' = F \times \text{Regions}(B) \times \{\exists, \forall\}$
- $s'_0 = (s_0, [\nu], \exists)$ where $\nu(x) = 0$ for each $x \in X$
- for $(s, \gamma) \in S \times \text{Regions}(B)$ and $a \in \Sigma_\varepsilon$:

  $$\delta'((s, \gamma, \forall), a) = \begin{cases} 
  \text{False} & \text{if } a \neq \varepsilon \text{ or } \gamma \text{ maximal} \\
  \land \{(s, \gamma', \exists) \mid \gamma \prec \gamma'\} & \text{otherwise}
  \end{cases}$$

  $$\delta'((s, \gamma, \exists), a) = \begin{cases} 
  \lor \{(s', \gamma', \exists) \mid (s, \gamma) \xrightarrow{a}_d (s', \gamma')\} & \text{if } a \neq \varepsilon \text{ or } \gamma \text{ maximal} \\
  (s, \gamma, \forall) \lor \lor \{(s', \gamma', \exists) \mid (s, \gamma) \xrightarrow{\varepsilon}_d (s', \gamma')\} & \text{otherwise}
  \end{cases}$$

where $\xrightarrow{a|\varepsilon}_d$ denotes a discrete transition of the region automaton $R_B$

Definition:

For an icTA $B$, let $L_{\text{react}}(B) = L(A_B)$ be the reactive semantics of $B$. Moreover, for a DTA $D$, $L_{\text{react}}(D) = L_{\text{react}}(B_D)$ is the reactive semantics of $D$. 
The following theorem follows from the previous lemma:

**Theorem:**

Let $B = (S, \Sigma, X, T, \text{Inv}, s_0, F, \pi)$ be an icTA and let $n$ be the number of states of $R_B$ (which is bounded by $|S| \cdot (2C + 2)^{|X|} \cdot |X|!$ where $C$ is the largest constant a clock is compared with in $B$). Then, $L_{\text{react}}(B)$ is regular and one can compute a non-deterministic finite automaton with $2^{O(n^2)}$ states that recognizes $L_{\text{react}}(B)$. 
Reactive Semantics

Lemma:

For any icTA \( B \), \( L_{\text{react}}(B) \subseteq L_\forall(B) \).

Proof: Assume \( B = (S, \Sigma, X, T, \text{Inv}, s_0, F, \pi) \) and \( A_B = (S', \Sigma, \delta', s'_0, F') \). Let \( w \in L_{\text{react}}(B) = L(A_B) \) and \( \rho = (V, \sigma, \mu) \) be an accepting run of \( A_B \) on \( w \).

We pick \( \tau \in \text{Rates} \). We construct inductively a maximal branch \( u_0u_1\ldots u_n \in V^* \) in \( \rho \) and two sequences \( t_0, t_1, \ldots, t_n \) and \( \nu_0, \nu_1, \ldots, \nu_n \):

1) Let \( u_0 = \varepsilon \), \( t_0 = 0 \) and \( \nu_0(x) = 0 \) for all \( x \in X \). Note that \( \sigma(u_0) = (s_0, [\nu_0], \exists) \).

2) Assume that the sequences have been constructed up to \( k \) and that \( \sigma(u_k) = (s_k, [\nu_k], pl_k) \).

If \( u_k \) is a leaf, the construction is over and \( k = n \).

Otherwise:

- Assume that \( pl_k = \forall \). Let \( t_{k+1} > t_k \) be such that \( [\nu_k] \prec [\nu_{k+1}] \) with \( \nu_{k+1} = \nu_k + \tau(t_{k+1}) - \tau(t_k) \). By definition of \( \delta' \), there exists a child \( u_{k+1} \) of \( u_k \) such that \( \sigma(u_{k+1}) = (s_k, [\nu_{k+1}], \exists) \).

- Assume now that \( pl_k = \exists \). Choose \( u_{k+1} \) in \( \text{children}(u_k) \).
  - Either \( \sigma(u_{k+1}) = (s_k, [\nu_k], \forall) \) and we let \( t_{k+1} = t_k \) and \( \nu_{k+1} = \nu_k \).
  - Otherwise, the move from \( u_k \) to \( u_{k+1} \) corresponds to some discrete transition of \( R_B \) with label \( a_{k+1} \in \Sigma_\varepsilon \) and some reset set \( R \subseteq X \). We let \( t_{k+1} = t_k \) and \( \nu_{k+1} = \nu_k[R \leftarrow 0] \) so that we have \( \sigma(u_{k+1}) = (s_{k+1}, [\nu_{k+1}], \exists) \).
Lemma:
For any icTA $B$, $L_{react}(B) \subseteq L_{\forall}(B)$.

Proof (cntd):
The discrete moves along the constructed branch correspond to the sequence $0 < i_1 < \cdots < i_\ell \leq n$ of indices $k$ such that $pl_{k-1} = pl_k = \exists$. As $\rho$ is an accepting run for $w$, we have $w = a_{i_1} \cdot a_{i_2} \cdot \ldots \cdot a_{i_\ell}$ and $s_{i_\ell} = s_n \in F$. One can verify that the sequence

$$(s_0, \nu_0) \xrightarrow{a_{i_1}, t_{i_1}} (s_{i_1}, \nu_{i_1}) \xrightarrow{a_{i_2}, t_{i_2}} \ldots \xrightarrow{a_{i_\ell}, t_{i_\ell}} (s_{i_\ell}, \nu_{i_\ell})$$

is a $\tau$-run of $B$ so that $w \in L(B, \tau)$. \qed
Reactive Semantics

Lemma:
Suppose that $|Proc| \geq 2$. There are some DTA $D$ over $Proc$ and some $\tau \in Rates$ such that $L_{react}(D) \subseteq L(D, \tau) \subseteq L(D, \tau) \subseteq L(D)$.

Example: (Proof)
Consider the following icTA $B$ (which can also be viewed as a DTA):

- $L_\exists(B) = \{a, ab, b, c\}$
- $L(B, id) = \{a, ab, b\}$
- $L_\forall(B) = \{a, ab\}$
- $L_{react}(B) = \{a\}$
Exercise:
Consider the icTA $\mathcal{B} = (\{s_0, \ldots, s_5\}, \{a, b\}, \{x, y\}, T, \text{Inv}, s_0, \{s_3, s_4, s_5\}, \pi)$ over $\{p, q\}$ with $\pi(x) = p$ and $\pi(y) = q$, which is depicted below. Determine $L_\exists(\mathcal{B})$, $L_\forall(\mathcal{B})$, and $L_{\text{react}}(\mathcal{B})$ for invariants (a) $\varphi = \text{true}$, and (b) $\varphi = (x = 0) \land (y = 0)$. Justify your solution.

<table>
<thead>
<tr>
<th>$L_\exists(\mathcal{B})$</th>
<th>$\varphi = \text{true}$</th>
<th>$\varphi = (x = 0) \land (y = 0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${a}$</td>
<td>${a}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>$L_\forall(\mathcal{B})$</td>
<td>${a}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>$L_{\text{react}}(\mathcal{B})$</td>
<td>${a}$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
Distributed Timed Automata
The Model
Existential Semantics and Region Abstraction
Universal Semantics and Undecidability
Reactive Semantics

Summary
Summary

- Distributed system using clocks with local times to synchronize
- Regular existential semantics suited for negative specifications
- Regular reactive semantics suited for positive specifications
- Undecidable universal semantics
Message Sequence Charts with timing constraints

Message Sequence Charts (MSCs):

- visual specification formalism
- ITU standardized
- describes the interaction between processes by means of message exchange

Example: MSC (with timing constraints)
Introducing timing (T-MSC)

- Attach time stamps (which are non-negative real numbers) to events.
- This is the natural formalism for timing which extends from timed words.
- However, it is not natural for specification by engineers! So we introduce another model...
Introducing timing (TC-MSC)

- We attach time intervals to “selected pairs” of events.
- We can restrict the pairs and thus control timing.
- This is natural for specification.
Plan

2 Message Sequence Charts with Timing Constraints (TC-MSCs)
   - Message Sequence Charts (MSCs)
     - Message Sequence Charts with Timing Constraints (TC-MSCs)
     - Realizability of Single TC-MSCs
     - Message Sequence Graphs with Timing Constraints
     - Timed Channel Systems
Let us formalize MSCs, first without timing, and then with timing constraints.

**We fix:**
- \(\text{Proc}\) (finite set of at least two *processes*)
- \(\text{Msg}\) (nonempty finite set of *message types*)

These parameters determine:
- \(\text{Ch} := \{(p, q) \in \text{Proc} \times \text{Proc} \mid p \neq q\}\) (set of *channels*)
- \(\text{Act}_p := \{p!q(m), p?q(m) \mid p \neq q \in \text{Proc}, m \in \text{Msg}\}\) (actions executed by process \(p \in \text{Proc}\))
- \(\text{Act} := \bigcup_{p \in \text{Proc}} \text{Act}_p\) (all the actions)
MSCs

We will model MSCs as \textit{Act-labeled posets}:

\begin{definition}
An \textit{Act-labeled poset} is a structure \( M = (E, \leq, \lambda) \) where:
\begin{itemize}
\item \((E, \leq)\) is a partially ordered set
\item \(\lambda : E \rightarrow \text{Act}\)
\end{itemize}
\end{definition}

Hereby, \(E\) will be the set of \textit{events}, which are ordered by \(\leq\).

\begin{notation}
\begin{itemize}
\item \(\downarrow e := \{ e' \in E \mid e' \leq e \}\) (past of event \(e \in E\))
\item \(E_p := \{ e \in E \mid \lambda(e) \in \text{Act}_p \}\) (events of \(p \in \text{Proc}\))
\item \(E_a := \{ e \in E \mid \lambda(e) = a \}\) (events executing \(a \in \text{Act}\))
\item \(<_p := < \cap (E_p \times E_p)\) (order on process \(p \in \text{Proc}\))
\item \(<_{\text{msg}} \subseteq E \times E\) defined by \(e <_{\text{msg}} e'\) if there are \((p, q) \in \text{Ch}\) and \(m \in \text{Msg}\) such that \(\lambda(e) = p!q(m), \lambda(e') = q?p(m)\), and
\[
| \downarrow e \cap \bigcup_{m' \in \text{Msg}} E_{p!q(m')} | = | \downarrow e' \cap \bigcup_{m' \in \text{Msg}} E_{q?p(m')} |
\]
\end{itemize}
\end{notation}
MSCs

**Definition: MSC**

An MSC is an Act-labeled poset \( M = (E, \leq, \lambda) \) such that:

- for all \( p \in \text{Proc} \), \( <_p \) is a (strict) linear order
- for all \( e \in E \), there is \( e' \in E \) such that \( e <_{\text{msg}} e' \) or \( e' <_{\text{msg}} e \)
- \( \leq = (<_{\text{msg}} \cup \bigcup_{p \in \text{Proc}} <_p)^* \)

**Example: MSC**

Here, \( \text{Proc} = \{p, q, r\} \), \( \text{Msg} = \{m_1, m_2\} \), and \( E = \{e_1, e'_1, e_2, e'_2, e_3, e'_3\} \).

We have \( \lambda(e_2) = q?p(m_1) \), \( e_1 <_{\text{msg}} e_2 \leq e_3 <_r e'_3 \), \( e'_1 \not\leq e'_2 \), and \( e'_2 \not\leq e'_1 \).
Plan

2 Message Sequence Charts with Timing Constraints (TC-MSCs)
   Message Sequence Charts (MSCs)
   Message Sequence Charts with Timing Constraints (TC-MSCs)
   Realizability of Single TC-MSCs
   Message Sequence Graphs with Timing Constraints
   Timed Channel Systems
MSCs with timing constraints

Idea:
Annotate MSCs with timing constraints, which are taken from the set of intervals \( Int \) containing \([a, b], (a, b], [a, b), (a, b), [a, \infty), (a, \infty)\) where \( a, b \in \mathbb{N} \).

Definition: TC-MSC
An **MSC with timing constraints** (TC-MSC) is a pair \( TC = (M, C) \) where:

- \( M = (E, \leq, \lambda) \) is an MSC
- \( C \subseteq (E \times E) \times Int \) is a set of timing constraints such that:
  for all \( ((e, f), I) \in C \) and \( ((e', f'), I') \in C \), both \( e < f \) and \( (e, f) = (e', f') \) implies \( I = I' \)

Remark:
Thus, timing constraints are only allowed between ordered pairs of events, and each such pair has at most one constraint.
Realization of TC-MSC

\[
\begin{align*}
[1,4] & \quad [1,4] \\
[2,3] & \quad [1,2] \\
(0,2) \\
\end{align*}
\]
Realization of TC-MSC
While TC-MSCs serve as specifications, concrete executions map, to each event, a time stamp from $\mathbb{R}_{\geq 0}$. We call such a structure a timed MSC:

**Definition: timed MSC**

A *timed MSC* (T-MSC) is a pair $T = (M, \tau)$ where $M = (E, \leq, \lambda)$ is an MSC and $\tau : E \rightarrow \mathbb{R}_{\geq 0}$ such that, for all $(e_1, e_2) \in \leq$, we have $\tau(e_1) \leq \tau(e_2)$.

**Definition: realization**

Let $TC = (M, C)$ with $M = (E, \leq, \lambda)$ be a TC-MSC. A *realization* of $TC$ is a T-MSC $(M, \tau)$ such that, for all $((e_1, e_2), I) \in C$, we have $\tau(e_2) - \tau(e_1) \in I$. We say that $TC$ is *realizable* if there is a realization of $TC$.

**Example: TC-MSC and realization**

```
<table>
<thead>
<tr>
<th></th>
<th>User</th>
<th>ATM</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3.9</td>
<td></td>
<td>1</td>
<td>2.3</td>
</tr>
<tr>
<td>[0, 4]</td>
<td></td>
<td>[0, 2]</td>
<td></td>
</tr>
</tbody>
</table>
```

User actions:
- **pin-request** at time 3.9
- **card** at time 0

ATM actions:
- **card-data** at time 1
- **card-OK** at time 3.3
Plan

2 Message Sequence Charts with Timing Constraints (TC-MSCs)
   Message Sequence Charts (MSCs)
   Message Sequence Charts with Timing Constraints (TC-MSCs)
   Realizability of Single TC-MSCs
   Message Sequence Graphs with Timing Constraints
   Timed Channel Systems
Realizability problem for TC-MSCs

We consider the realizability problem for TC-MSCs:

Theorem: [Alur et al. 1996]

For a given TC-MSC $TC = (M, C)$ with $M = (E, \leq, \lambda)$, one can decide in time $O(|E|^3)$ if $TC$ is realizable (and, if so, determine a realization of $TC$).

Proof:
We show the theorem for the case where intervals are of the form $[a, b]$ or $[a, \infty)$.

Idea:
Reduce realizability of TC-MSC to finding negative-weight cycles in a graph, which can be solved in cubic time.
Realizability problem for TC-MSCs

Example: realizable TC-MSC \((M, C)\)

\[
\begin{align*}
\tau(e_1) - \tau(e_2) &\leq 0 \\
\tau(e_1) - \tau(e_3) &\leq 0 \\
\tau(e_1) - \tau(e_4) &\leq 0 \\
\tau(e_2) - \tau(e_3) &\leq 0 \\
\tau(e_2) - \tau(e_4) &\leq 0 \\
\tau(e_3) - \tau(e_4) &\leq 0 \\
\tau(e_4) - \tau(e_1) &\leq 4 \\
\tau(e_2) - \tau(e_3) &\leq -4
\end{align*}
\]

Basic constraints

\[
\tau(f) - \tau(e) \leq b \quad \Rightarrow \quad e \xrightarrow{b} f
\]

\[
\begin{align*}
e &\sim f \\
\Rightarrow \quad \tau(f) - \tau(e) &\leq b \\
\Rightarrow \quad \text{no negative-weight cycle}
\end{align*}
\]
Realizability problem for TC-MSCs

Definition: graph of TC-MSC

Let $TC = (M, C)$ be a TC-MSC with $M = (E, \leq, \lambda)$. We define the weighted graph $G_{TC} = (V, Arcs, weight)$ where:

- $V = E \cup \{e_0\}$ (the fresh node $e_0$ is used to compute a realization)
- $Arcs = \text{arcs}^{-1}$
  - $\cup \{(e, f) | \exists a, b \in \mathbb{N} : ((e, f), [a, b]) \in C\}$
  - $\cup \{(e_0, e) | e \in E\}$
- for $(e, f) \in Arcs$, we let $weight(e, f) =$
  - $0$ if $e = e_0$
  - $b$ if $e < f$ and $((e, f), [a, b]) \in C$
  - $\min(\{0\} \cup \{-a | ((f, e), I) \in C \text{ with } I = [a, b] \text{ or } I = [a, \infty]\})$ if $f < e$

![Diagram](image_url)
Realizability problem for TC-MSCs

Lemma:

1. If $G_{TC}$ contains some negative-weight cycle, then $TC$ is not realizable.
2. If $G_{TC}$ contains no negative-weight cycle, then $TC$ is realizable and we can compute a realization of $TC$.

Note that negative-weight cycles can indeed be detected in cubic time.

Proof: 1.

Suppose $\rho = (e_1, \ldots, e_n, e_1)$ with $n \geq 2$ is a negative-weight cycle (note that $e_0$ is not part of the cycle). Suppose $(M, \tau)$ is a realization of $TC$. We have

\[
\tau(e_2) - \tau(e_1) \leq \text{weight}(e_1, e_2) \\
\tau(e_3) - \tau(e_2) \leq \text{weight}(e_2, e_3) \\
\vdots \\
\tau(e_n) - \tau(e_{n-1}) \leq \text{weight}(e_{n-1}, e_n) \\
\tau(e_1) - \tau(e_n) \leq \text{weight}(e_n, e_1)
\]

When we build the sum on both sides, we obtain

\[
\tau(e_1) - \tau(e_1) \leq \text{weight}(\rho) < 0
\]

But this is a contradiction.
Realizability problem for TC-MSCs

Lemma:
1. If $G_{TC}$ contains some negative-weight cycle, then $TC$ is not realizable.
2. If $G_{TC}$ contains no negative-weight cycle, then $TC$ is realizable and we can compute a realization of $TC$.

Proof: 2.

- Suppose $G_{TC}$ contains no negative-weight cycle.
- For $e \in E$, let $\tau'(e)$ be the minimal weight of a path from $e_0$ to $e$ (it exists!).
- Pick any $(e, f) \in Arcs$ such that $e \neq e_0$.
- We have $\tau'(f) \leq \tau'(e) + weight(e, f)$, which implies $\tau'(f) - \tau'(e) \leq weight(e, f)$.
- Thus, $\tau'$ satisfies the constraints imposed by $TC$.

However, there might be negative values in the image of $\tau'$. To obtain a realization, we need a mapping $\tau : E \to \mathbb{R}_{\geq 0}$, which we obtain from $\tau'$ as follows: For $e \in E$, we set $\tau(e) = \tau'(e) - \min_{f \in E} \tau'(f)$. Then, $(M, \tau)$ is a realization of $TC$. □
Realizability problem for TC-MSCs

Example:

\[ \begin{align*}
\text{Realization:} \\
\tau(e_1) &= 0 \\
\tau(e_2) &= 0 \\
\tau(e_3) &= 4 \\
\tau(e_4) &= 4
\end{align*} \]
Floyd-Warshall Algorithm

Computes distances and detects negative-weight cycles.

Floyd-Warshall\((G = (\{1, \ldots, n\}, Arcs, weight))\)

for \(i = 1\) to \(n\)
\[
    d^{(0)}[i, i] \leftarrow 0
\]
for \(j = 1\) to \(n\)
    if \(i \neq j\) and \((i, j) \in Arcs\) then \(d^{(0)}[i, j] \leftarrow \text{weight}(i, j)\)
    if \(i \neq j\) and \((i, j) \notin Arcs\) then \(d^{(0)}[i, j] \leftarrow \infty\)

for \(k = 1\) to \(n\)
    for \(i = 1\) to \(n\)
        for \(j = 1\) to \(n\)
            \[
            d^{(k)}[i, j] \leftarrow \min(d^{(k-1)}[i, j], d^{(k-1)}[i, k] + d^{(k-1)}[k, j])
            \]

for \(i = 1\) to \(n\)
    if \(d^{(n)}[i, i] < 0\) then “there is a negative-weight cycle”

If there is no negative-weight cycle, then \(d^{(n)}[i, j]\) is the weight of the shortest path from \(i\) to \(j\). Running time \(O(n^3)\).

More efficient alternative (single source): Bellman-Ford algorithm; \(O(n \cdot |Arcs|)\).
**Realizability problem for TC-MSCs**

**Exercise:**

Let $Proc = \{p, q, r\}$ and $Msg = \{m\}$. Let the TC-MSC $TC$ (over $Proc$ and $Msg$) be given as follows (the message type is omitted):

![Graph](image_url)

Apply the realizability algorithm to $TC$. In particular, determine the weighted graph $G_{TC}$. If $TC$ is realizable, then determine the realization that the algorithm outputs.
Realizability problem for TC-MSCs

Exercise:
Reduce the realizability problem for TC-MSCs (in its full generality, i.e., considering all intervals from $Int$) to the emptiness problem for timed automata. More precisely: give an effective transformation of a TC-MSC $TC$ into a timed automaton $A$ such that $TC$ is realizable iff $L(A) \neq \emptyset$.

Notice: In line with this lecture, a timed automaton is an icTA $A$ over one single process. Its language $L(A)$ is defined as $L(A, \text{id})$. 
Plan

Message Sequence Charts with Timing Constraints (TC-MSCs)
Message Sequence Charts (MSCs)
Message Sequence Charts with Timing Constraints (TC-MSCs)
Realizability of Single TC-MSCs

- Message Sequence Graphs with Timing Constraints

Timed Channel Systems
Message Sequence Graphs with timing constraints

Idea:
Define more complex behaviors by means of automata constructs.

Example:
Concatenation of MSCs and TC-MSCs

Definition: (asynchronous) concatenation of MSCs

Let $M^1 = (E^1, \leq^1, \lambda^1)$ and $M^2 = (E^2, \leq^2, \lambda^2)$ be MSCs (assume $E^1 \cap E^2 = \emptyset$). We set $M^1 \circ M^2 := (E, \leq, \lambda)$ where:

- $E = E^1 \cup E^2$
- $\lambda(e) = \lambda^i(e)$ if $e \in E^i$
- $\leq = (\leq^1 \cup \leq^2 \cup \bigcup_{p \in \text{Proc}} (E^1_p \times E^2_p))^*$

Note that $M^1 \circ M^2$ is again an MSC, and that concatenation is associative.

Concatenation of TC-MSCs is parametrized by a partial mapping $\gamma : \text{Proc} \to \text{Int}$:

Definition: concatenation of TC-MSCs

Let $TC^1 = (M^1, C^1)$ and $TC^2 = (M^2, C^2)$ be TC-MSCs where, for $i \in \{1, 2\}$, $M^i = (E^i, \leq^i, \lambda^i)$ (assume $E^1 \cap E^2 = \emptyset$). Let $\gamma : \text{Proc} \to \text{Int}$ be a partial mapping. Then, $TC^1 \circ_\gamma TC^2$ is defined if, for all $p \in \text{dom}(\gamma)$, both $E^1_p \neq \emptyset$ and $E^2_p \neq \emptyset$.

If defined, we set

$$TC^1 \circ_\gamma TC^2 := (M^1 \circ M^2, C)$$

where $C = C^1 \cup C^2 \cup \{((\max(E^1_p), \min(E^2_p)), \gamma(p)) \mid p \in \text{dom}(\gamma)\}$. 
Message Sequence Graphs with timing constraints

Definition: TC-MSG

A TC-MSG is a structure $G = (S, \Delta, S_{in}, S_{F}, \Phi, \gamma)$ where

- $S$ is a non-empty finite set of states
- $S_{in} \subseteq S$ and $S_{F} \subseteq S$ are the sets of initial and final states, resp.
- $\Delta \subseteq S \times S$ is the transition relation
- $\Phi$ is a mapping from $S$ into the set of TC-MSCs
- $\gamma : \Delta \rightarrow \{ \gamma | \gamma : \text{Proc} \rightarrow \text{Int} \}$ specifies edge constraints such that, for all $(s, s') \in \Delta$, the concatenation $\Phi(s) \circ \gamma(s, s') \Phi(s')$ is defined
**TC-MSGs**

**Definition: language of TC-MSG**

For a path $\rho = (s_0, s_1, \ldots, s_n)$ through $G$ (i.e., $n \geq 0$ and $(s_i, s_{i+1}) \in \Delta$ for all $i \in \{0, \ldots, n-1\}$), let

$$\Phi(\rho) := \Phi(s_0) \circ \gamma(s_0, s_1) \circ \gamma(s_1, s_2) \cdots \circ \gamma(s_{n-1}, s_n) \circ \gamma(s_n)$$

which is indeed defined. Then, the *language* of $G$ is the set $L(G) := \{\Phi(\rho) \mid \rho = (s_0, s_1, \ldots, s_n) \text{ is a path through } G \text{ such that } s_0 \in S_{\text{in}} \text{ and } s_n \in S_{\text{f}}\}$. 

---

\[ r \xrightarrow{m_1} [0, 3] \xrightarrow{m_2} [0, 2] \xrightarrow{m_3} (2, 3) \xrightarrow{m_2} [0, 3] \xrightarrow{m_1} [1, 1] \xrightarrow{m_2} [0, 2] \xrightarrow{m_3} [1, 1] \xrightarrow{r} s \]

\[ ([0, 2], [1, 1]) \xrightarrow{m_2} ([2, 3], [1, 1]) \xrightarrow{m_1} ([0, 3], [1, 1]) \xrightarrow{m_2} ([0, 3], [1, 1]) \xrightarrow{m_3} ([1, 1], [1, 1]) \xrightarrow{r} s \]
Realizability of TC-MSGs

Definition: realizability of TC-MSGs

A TC-MSG $G$ is called realizable if $L(G)$ contains some realizable TC-MSC.

E.g., the previous example TC-MSG is realizable. But the problem is difficult (in the following, let $n$ stand for a constraint $[n, n]$):

Example:

To reach the last node from the first one, we need to iterate:

- $k$-times the first loop
- $\ell$-times the second loop

such that $ka - \ell b = 1$. 
Realizability of TC-MSGs

Theorem:

Realizability is decidable for TC-MSGs \((S, \Delta, S_{in}, S_F, \Phi, \overline{\gamma})\) such that, for all \((s, s') \in \Delta\), we have \(\text{dom}(\overline{\gamma}(s, s')) = \emptyset\).

Proof:

It is sufficient to check some TC-MSCs that are on a path from an initial to a final state.

Unfortunately, realizability is, in general, undecidable:

Theorem: [Gastin et al. 2008]

Realizability of TC-MSGs is undecidable (even when we restrict to TC-MSGs with timing constraints on processes).
Proof of undecidability

The proof is by reduction from emptiness of 2-counter machines (2-CM), which are equipped with two counters, $c_1$ and $c_2$, whose value is initially 0.

A 2-CM is a sequence of labeled instructions:

$$
1 : instr_1 \\
2 : instr_2 \\
\vdots \\
n : instr_n
$$

Each instruction is one of the following (where $c \in \{c_1, c_2\}$):

- $\ell : accept$
- $\ell : c++$
- $\ell : if \ c == 0 \ goto \ \ell' \ else \ c --$

The semantics of an instruction and of a program are as expected:

- $c++$ increments counter $c$ by 1
- $c--$ decrements counter $c$ by 1

We construct a TC-MSG $G$ that contains a realizable TC-MSC iff the 2-CM accepts, i.e., it executes an instruction of the form $\ell : accept$. 
Simulation of 2-CM

Idea:
We employ two processes, $p$ and $q$, to simulate one counter $c$. The idea is that the difference $t_q - t_p$ between the execution times of the current events of $p$ and $q$ tracks the counter value. In particular, we maintain $t_p \leq t_q$ as an invariant.
Simulation of 2-CM

Proof:

- The edge constraints 1, 1 make sure that $t_q - t_p$ is preserved.
- The initial state 0 executes two copies of $\text{Init}$ to make sure that we initially have $t_q - t_p = 0$ for both counters.
- For all $\ell : c++$, create a state $\ell$ in $G$. It executes both $c++$ and, for the other counter, $\text{Freeze}$.
- For all $\ell : \text{if } c == 0 \text{ goto } \ell' \text{ else } c--$, create two states:
  - $\ell\_\_\_\_$ executes the test $c == 0$ for counter $c$ and $\text{Freeze}$ for the other counter
  - $\ell\_\_\-$ executes the decrement $c --$ for counter $c$ and $\text{Freeze}$ for the other counter
  When the next instruction to be executed is of that form, $G$ branches non-deterministically (and with edge constraints 1, 1) into $\ell\_\_\_\_$ or $\ell\_\_\-$.
- For $\ell : \text{accept}$, create an accepting state $\ell$, executing two copies of $\text{Freeze}$.

With this, the 2-CM accepts iff $L(G)$ contains some realizable TC-MSC.
Plan

2 Message Sequence Charts with Timing Constraints (TC-MSCs)
Message Sequence Charts (MSCs)
Message Sequence Charts with Timing Constraints (TC-MSCs)
Realizability of Single TC-MSCs
Message Sequence Graphs with Timing Constraints

Timed Channel Systems
Timed Channel System

User

- $s_1$
  - $(\text{pswd})$
  - $x \rightarrow 0$

- $s_2$
  - $(\text{wrong})$
  - $x \in [5, 7]$

- $s_3$

Server

- $t_1$
  - $(\text{correct})$
  - $(\text{pswd} \in [0, 4])$

- $t_2$
  - $(\text{wrong})$

User Server

- User
  - 1
    - pswd
  - 7
    - wrong
  - 8
    - pswd
  - 10
    - correct

- Server
  - 3
  - 5
  - 9
  - 10
An timed channel system (TCS) is a tuple \( T = (S, \text{Msg}, C, X, \Delta, s_0) \) where

- \( S \) is a finite set of states
- \( \text{Msg} \) is a finite set of messages
- \( C \) is a finite set of channels
- \( X \) is a finite set of clocks
- \( \Delta \) is a finite set of transitions
- \( s_0 \in S \) is the initial state

A transition is a triple \( (s, op, s') \) where \( s, s' \in S \) and \( op \) is one of the following:

- \( \text{nop} \) (no operation)
- \( c!(m) \) (appends \( m \in \text{Msg} \) to \( c \in C \) according to FIFO)
- \( c?(m \in I) \) (removes \( m \in \text{Msg} \) from \( c \in C \) if the age of \( m \) is in \( I \in \text{Int} \))
- \( x \in I \) (checks whether value of \( x \in X \) is in \( I \in \text{Int} \))
- \( x \leftarrow 0 \) (resets \( x \in X \))
Timed Channel System
Semantics of $\mathcal{T} = (S, \text{Msg}, C, X, \Delta, s_0)$ is a transition system $(\text{Conf}_\mathcal{T}, \rightarrow_\mathcal{T})$.

**Configuration from $\text{Conf}_\mathcal{T}$**

Triple $\langle s, \nu, \gamma \rangle$ where $s \in S$, $\nu : X \rightarrow \mathbb{R}_{\geq 0}$, and $\gamma : C \rightarrow (\text{Msg} \times \mathbb{R}_{\geq 0})^*$.

**Transitions** $\rightarrow_\mathcal{T} = \overrightarrow{d}_\mathcal{T} \cup \overrightarrow{t}_\mathcal{T}$

$\langle s, \nu, \gamma \rangle \xrightarrow{\overrightarrow{d}_\mathcal{T}} \langle s', \nu', \gamma' \rangle$ if there is an operation $\text{op}$ such that $(s, \text{op}, s') \in \Delta$ and one of the following holds:

- $\text{op} = \text{nop}$ and $\nu = \nu'$ and $\gamma = \gamma'$
- $\text{op} = c!(m)$ and $\nu = \nu'$ and $\gamma' = \gamma[c \mapsto (m, 0) \cdot \gamma(c)]$
- $\text{op} = c?(m \in I)$ and $\nu = \nu'$ and $\exists t \in I$ such that $\gamma = \gamma'[c \mapsto \gamma'(c) \cdot (m, t)]$
- $\text{op} = x \in I$ and $\nu = \nu'$ and $\gamma = \gamma'$ and $\nu(x) \in I$
- $\text{op} = x \leftarrow 0$ and $\nu' = \nu[x \mapsto 0]$ and $\gamma = \gamma'$

$\langle s, \nu, \gamma \rangle \xrightarrow{\overrightarrow{t}_\mathcal{T}} \langle s', \nu', \gamma' \rangle$ if $s = s'$ and there is $t \in \mathbb{R}_{\geq 0}$ such that

- $\nu' = \nu + t$
- $\gamma'(c) = \gamma(c) + t$

Here, $[(m_1, t_1) \ldots (m_n, t_n)] + t := (m_1, t_1 + t) \ldots (m_n, t_n + t)$. 
Reachability Problem for TCS

Definition: Control-state reachability

Let \( T = (S, Msg, C, X, \Delta, s_0) \) be a TCS. A state \( s \in S \) is reachable in \( T \) if
\[
\langle s_0, \nu_0, \gamma_0 \rangle \xrightarrow{\ast} T \langle s, \nu, \gamma \rangle
\]
for some \( \nu_0, \gamma_0, \nu, \gamma \) such that
- \( \nu_0(x) = 0 \) for all \( x \in X \)
- \( \gamma_0(c) = \varepsilon \) for all \( c \in C \)

Definition: Control-state reachability problem

The control-state reachability problem for TCS is defined as follows:

**Input:** TCS \( T = (S, Msg, C, X, \Delta, s_0) \) and \( s \in S \).

**Question:** Is \( s \) reachable in \( T \)?

Theorem:

The control-state reachability problem for TCS is undecidable.

\( \Rightarrow \) Alternative semantics: under- and over-approximation
Under- and over-approximation

Let $\mathcal{T} = (S, Msg, C, X, \Delta, s_0)$ be a TCS.

**B-bounded semantics for $B \in \mathbb{N}$:**

$\langle s, \nu, \gamma \rangle \xrightarrow{\mathcal{T}, B} \langle s', \nu', \gamma' \rangle$ if

- $\langle s, \nu, \gamma \rangle \xrightarrow{\mathcal{T}} \langle s', \nu', \gamma' \rangle$
- $|\gamma'(c)| \leq B$ for all $c \in Ch$

**Lossy semantics:**

$\xrightarrow{\mathcal{T}} = \xrightarrow{\mathcal{T}, d} \cup \xrightarrow{\mathcal{T}, t} \cup \xrightarrow{\mathcal{T}, l}$ where $\langle s, \nu, \gamma \rangle \xrightarrow{\mathcal{T}, l} \langle s', \nu', \gamma' \rangle$ if

- $s = s$
- $\nu = \nu'$
- $\gamma'(c) \sqsubseteq \gamma(c)$ for all $c \in C$

Here, $a_1 \ldots a_m \sqsubseteq b_1 \ldots b_n$ is the subword ordering: there is a strictly increasing injective mapping $g : \{1, \ldots, m\} \rightarrow \{1, \ldots, n\}$ such that $a_i = b_{g(i)}$. 
Reachability Problem for TCS

**Definition:** $B$-bounded and lossy control-state reachability

Let $\mathcal{T} = (S, \text{Msg}, C, X, \Delta, s_0)$ be a TCS and $s \in S$. We say that $s$ is

- $B$-reachable in $\mathcal{T}$ if $\langle s_0, \nu_0, \gamma_0 \rangle \xrightarrow{T,B}^* \langle s, \nu, \gamma \rangle$
- lossy-reachable in $\mathcal{T}$ if $\langle s_0, \nu_0, \gamma_0 \rangle \xrightarrow{T,\text{lossy}}^* \langle s, \nu, \gamma \rangle$

for some $\nu_0, \gamma_0, \nu, \gamma$ such that $\nu_0(x) = 0$ for all $x \in X$ and $\gamma_0(c) = \varepsilon$ for all $c \in C$.

**Definition:** Bounded control-state reachability problem

**Input:** TCS $\mathcal{T}$, state $s \in S$, and $B \in \mathbb{N}$.

**Question:** Is $s$ $B$-reachable in $\mathcal{T}$?

**Definition:** Lossy control-state reachability problem

**Input:** TCS $\mathcal{T}$ and state $s$.

**Question:** Is $s$ lossy-reachable in $\mathcal{T}$?

**Theorem:**

The following problems are decidable:

- the bounded control-state reachability problem for TCS
- the lossy control-state reachability problem for TCS [Abdulla et al. 2012]
Reachability Problem for TCS

Theorem:
The bounded control-state reachability problem for TCS is decidable.

Idea (suppose $B = 3$):

\[
\begin{align*}
&c!(m) & c!(m) & c?(m \in I) & c!(m) & c!(m) & c?(m \in I) \\
x_c^1 \leftarrow 0 & x_c^2 \leftarrow 0 & x_c^1 \in I & x_c^1 \leftarrow 0 & x_c^3 \leftarrow 0 & x_c^2 \in I
\end{align*}
\]

Channel $c \implies (m, 3) (m, 1) (m, 2) (m, 1)$

Proof:
Let $\mathcal{T} = (S, \text{Msg}, C, X, \Delta, s_0)$ be a TCS and $B \in \mathbb{N}$. Translate $\mathcal{T}$ into a “timed automaton” $A_{\mathcal{T}} = (S', X', \Delta', s'_0)$ (every operation is of the form $\text{nop}, x \in I$, or $x \leftarrow 0$ where $x \in X'$ and $I \in \text{Int}$):

1. $X' = X \cup \{x_c^i | i \in \{1, \ldots, B\} \text{ and } c \in C\}$
2. $S' = S \times \{\gamma | \gamma : C \rightarrow (\text{Msg} \times \{1, \ldots, B\})^{\leq B}\}$
   - message $(m, i)$ in channel $c$ should be checked with clock $x_c^i$
3. $s'_0 = \langle s_0, \gamma_0 \rangle$ where $\gamma_0(c) = \varepsilon$ for all $c \in C$
Reachability Problem for TCS

Proof: (cntd.)

We have a transition
\[(\langle s, \gamma \rangle, \alpha, \langle s', \gamma' \rangle) \in \Delta'\]
if there is \(op\) such that \((s, op, s') \in \Delta\) and one of the following holds:

- simulation of “non-channel” transition in \(T\):
  - \(\alpha = op \in \{\text{nop}\} \cup \{x \in I \mid x \in X\} \cup \{x \leftarrow 0 \mid x \in X\}\)
  - \(\gamma = \gamma'\)

- simulation of send transition in \(T\):
  - there are \(c, m,\) and \(w = (m_1, i_1) \ldots (m_n, i_n)\) such that
    - \(op = c!(m)\)
    - \(\gamma(c) = w\)
    - \(\alpha = x^i_c \leftarrow 0\)
    - \(\gamma' = \gamma[c \mapsto (m, i) \cdot w]\)
  - where \(i = \min(\{1, \ldots, B\} \setminus \{i_1, \ldots, i_n\})\)

- simulation of receive transition in \(T\):
  - there are \(c, m,\) and \(i \in \{1, \ldots, B\}\) such that
    - \(op = c?(m \in I)\) for \(c \in C, m \in \text{Msg},\) and \(I \in \text{Int}\)
    - \(\gamma = \gamma'[c \mapsto \gamma'(c) \cdot (m, i)]\)
    - \(\alpha = x^i_c \in I\)
Reachability Problem for TCS

Proof: (cntd.)

We reduced $\mathcal{T} = (S, \text{Msg}, C, X, \Delta, s_0)$ to a particular TCS $\mathcal{A}_T = (S', X', \Delta', s'_0)$ such that, for all $s \in S$, we have

- $s$ is reachable in $\mathcal{T}$ iff
- $\langle s, \gamma \rangle$ is reachable in $\mathcal{A}_T$ for some $\gamma$

Correctness proof is via mimicking a \textit{timed} run of $\mathcal{T}$ in $\mathcal{A}_T$, and vice versa (Exercise).

Theorem: [Abdulla, Atig, Cederberg 2012]

The lossy control-state reachability problem for TCS is decidable.

Proof: (sketch)

- Reduce problem to reachability problem in untimed model (infinite-state).
- Show that untimed model generates transition system that is well quasi ordered.
Petri Nets

\[
\begin{array}{c}
p_0 \\
p_1 \\
t_1 \\
p_2 \\
t_2 \\
t_3 \\
t_4
\end{array}
\]

\[
\begin{array}{c}
|1| \\
|2| \\
|1| \\
|1| \\
|1| \\
|1| \\
|1|
\end{array}
\]
# Introducing timing

## Two possibilities

1. transitions have ages $\implies$ Time Petri Nets ['70s]
2. tokens have ages $\implies$ Timed Petri Nets ['90s]

## Time Petri Nets

Transitions carry constraints in terms of intervals.
- upper bounds are invariants (urgency)
- transition $t$ is reset when $t$ is newly activated after firing
Plan

3 Time(d) Petri Nets
- Time Petri Nets (TPN)
  Decision problems for TPN
- Timed Petri Nets (TdPN)
  Decision problems for TdPN
- Decidability of Coverability for TdPN
- Expressiveness (credits to Serge Haddad)
Time Petri Net (TPN)
Time Petri Net (TPN)

Notation:

- The set $\text{Int}$ of intervals contains $[a, b], (a, b], [a, b), (a, b), [a, \infty), (a, \infty)$ where $a, b \in \mathbb{N}$.
- Given a set $P$, let $\text{Bag}(P)$ be the set of mappings $m : P \rightarrow \mathbb{N}$ with finite support, i.e., such that $\sum_{p \in P} m(p) \in \mathbb{N}$.

Definition: Time Petri Net

A time Petri net (TPN) is a tuple $\mathcal{N} = (P, T, \text{Pre}, \text{Post}, \varphi, m_0)$ where:

- $P$ is a finite set of places
- $T$ is a finite set of transitions such that $P \cap T = \emptyset$
- $\text{Pre} : T \rightarrow \text{Bag}(P)$
- $\text{Post} : T \rightarrow \text{Bag}(P)$
- $\varphi : T \rightarrow \text{Int}$
- $m_0 \in \text{Bag}(P)$ is the initial marking
**Example: \( \mathcal{N} = (P, T, \text{Pre}, \text{Post}, \varphi, m_0) \)**

- \( P = \{p_0, p_1, p_2\} \quad T = \{t_1, \ldots, t_4\} \)
- \( \text{Pre}(t_1)(p_0) = 1 \quad \text{Post}(t_1)(p_1) = 2 \quad \text{Post}(t_3)(p_i) = 0 \text{ for } i \in \{0,1,2\} \)
- \( \varphi(t_1) = [1, 1] \)
- \( m_0(p_0) = 1 \quad m_0(p_1) = m_0(p_2) = 0 \)
Time Petri Net (TPN)

Let $\mathcal{N} = (P, T, \text{Pre}, \text{Post}, \varphi, m_0)$ be a TPN.

Notation:
For $m, m' \in \text{Bag}(P)$, we write $m \leq m'$ if $m(p) \leq m'(p)$ for all $p \in P$.

Definition: active transitions

Let $m \in \text{Bag}(P)$. A transition $t \in T$ is active in $m$ if $\text{Pre}(t) \leq m$.
The set of transitions that are active in $m$ is denoted by $\text{Active}(m)$.

Definition: configuration

A configuration of $\mathcal{N}$ is a pair $(m, \nu)$ where:

- $m \in \text{Bag}(P)$
- $\nu : \text{Active}(m) \rightarrow \mathbb{R}_{\geq 0}$
- $\nu(t) \in \varphi(t)\downarrow$ for all $t \in \text{Active}(m)$
  (where $\varphi(t)\downarrow$ is the downward closure of interval $\varphi(t)$)

The set of configurations of $\mathcal{N}$ is denoted by $\text{Conf}_\mathcal{N}$.

Intuitively, every $t \in \text{Active}(m)$ is a clock with valuation $\nu(t)$ satisfying invariant $\varphi(t)\downarrow$, and $t$ can be fired if $\nu(t) \in \varphi(t)$. 
**Time Petri Net (TPN)**

Example: configuration \((m, \nu)\)

- \(m(p_0) = m(p_1) = 1\) \(\quad m(p_2) = 0\)
- \(Active(m) = \{t_1, t_2, t_3\}\)
- \(\nu(t_1) = \nu(t_2) = 1\) \(\quad \nu(t_3) = 0\)

Transitions \(t_1, t_2\) are fireable, while \(t_3, t_4\) are not fireable.
Let $\mathcal{N} = (P, T, Pre, Post, \varphi, m_0)$ be a TPN.

**Notation: (resets)**

For $m \in Bag(P)$ and $t \in Active(m)$, let

$$Reset(m, t) := \{ t' \in Active(m - Pre(t) + Post(t)) \mid t = t' \text{ or } t' \notin Active(m - Pre(t)) \}$$

be the set of transitions that are reset when firing $t$ in $m$. 

**Example:**

$Reset(m, t_1) = \{t_1, t_2\}$

$Reset(m, t_1) = \{t_1\}$
Definition: semantics of $\mathcal{N} = (P, T, Pre, Post, \varphi, m_0)$

Let the infinite transition system $TS_\mathcal{N} = (Conf_\mathcal{N}, q_0, \rightarrow)$ be given as follows:

- $q_0 = (m_0, \nu_0)$ where $\nu_0(t) = 0$ for all $t \in Active(m_0)$
  (note that $q_0 \in Conf_\mathcal{N}$)
- (delay transition) for all $d \in \mathbb{R}_{>0}$:
  $$(m, \nu) \xrightarrow{d} (m, \nu + d)$$
- (discrete transition) for all $t \in Active(m)$ such that $\nu(t) \in \varphi(t)$:
  $$(m, \nu) \xrightarrow{t} (m - Pre(t) + Post(t), \nu')$$

where, for all $t' \in Active(m - Pre(t) + Post(t))$:

$$\nu'(t') = \begin{cases} 
0 & \text{if } t' \in Reset(m, t) \\
\nu(t') & \text{otherwise (in that case: } t' \in Active(m)\text{)}
\end{cases}$$
Plan

3 Time(d) Petri Nets

Time Petri Nets (TPN)

- Decision problems for TPN

Timed Petri Nets (TdPN)

Decision problems for TdPN

Decidability of Coverability for TdPN

Expressiveness (credits to Serge Haddad)
### Definition: Reachability for TPN

**Input:** TPN \( \mathcal{N} = (P, T, \text{Pre}, \text{Post}, \varphi, m_0) \) and \( m \in \text{Bag}(P) \)

**Question:** Is there \( \nu : \text{Active}(m) \to \mathbb{R}_{\geq 0} \) such that \((m, \nu)\) is reachable in \( TS_\mathcal{N} \) from \( q_0 \)?

### Definition: Coverability for TPN

**Input:** TPN \( \mathcal{N} = (P, T, \text{Pre}, \text{Post}, \varphi, m_0) \) and \( m \in \text{Bag}(P) \)

**Question:** Are there \( m' \geq m \) and \( \nu' : \text{Active}(m') \to \mathbb{R}_{\geq 0} \) such that \((m', \nu')\) is reachable in \( TS_\mathcal{N} \) from \( q_0 \)?

### Theorem:

For Petri nets (i.e., TPN with trivial timing constraints),

- reachability is decidable [Mayr 1981].
- coverability is in EXPSPACE [Rackoff 1978].

Both problems are EXPSPACE-hard [Lipton et al. 1976].
Theorem: [Jones et al. 1977]
Reachability for TPN is undecidable.

Proof: (Idea)
The proof is by reduction from emptiness of 2-counter machines (2-CM).
We construct a TPN $\mathcal{N}$ such that the 2-CM executes an instruction of the form $\ell : \text{accept iff a configuration with a token in place } \ell \text{ is reachable in } TS_\mathcal{N}$. 
Proof of undecidability

A 2-CM is a sequence of labeled instructions:

\[\begin{align*}
1 : instr_1 \\
2 : instr_2 \\
\vdots \\
n : instr_n
\end{align*}\]

Each instruction is one of the following (where \(c \in \{c_1, c_2\}\)):

- \(\ell : \text{accept}\)
- \(\ell : c++\)
- \(\ell : \text{if } c == 0 \text{ goto } \ell' \text{ else } c --\)

The semantics of an instruction and of a program are as expected:

- \(c++\) increments counter \(c\) by 1
- \(c--\) decrements counter \(c\) by 1
Proof of undecidability

Idea:
We introduce places $c_1$ and $c_2$ to simulate counters:
The number of tokens in $c_i$ is the current counter value of $c_i$.

\[
\ell : c_1 \quad \text{++} \\
\]

[Diagram]

\[
\ell \quad \quad \quad t^{\ell} \quad \quad \quad \ell + 1 \\
\quad \quad \quad [0, 0] \\
\quad \quad \quad \quad c_1
\]
Proof of undecidability

\[ \ell : \text{if } c_1 == 0 \text{ goto } \ell' \text{ else } c_1 -- \]

- When going into \( \ell \), both transitions are reset (or non-active).
- Due to urgency, \( t^\ell_{\_\_} \) must fire if there is token in \( c_1 \).
- Only if there is no token in \( c_1 \), \( t^{\ell\_\_0} \) can fire.
Corollary:

Coverability for TPN is undecidable.

Proof:

Follows directly from the previous proof:
Place $\ell$ cannot contain more than one token.
3 **Time(d) Petri Nets**

Time Petri Nets (TPN)
Decision problems for TPN

- **Timed Petri Nets (TdPN)**
Decision problems for TdPN
Decidability of Coverability for TdPN
Expressiveness (credits to Serge Haddad)
Timed Petri Net (TdPN)

\[
(p_0, 0) \xrightarrow{1} (p_0, 1) \xrightarrow{t_1} (p_0, 0) + (p_1, 0) \xrightarrow{t_1} (p_0, 0) + (p_1, 0) + (p_1, 0)
\]

\[
\xrightarrow{2} (p_0, 2) + (p_1, 2) + (p_1, 2) \xrightarrow{t_2} (p_0, 0) + (p_1, 2) + (p_1, 2) + (p_2, 0)
\]

\[
\xrightarrow{t_3} (p_0, 0) + (p_1, 2) \xrightarrow{1} (p_0, 1) + (p_1, 3)
\]
Timed Petri Net (TdPN)

Goal:
- Reachability for TdPN is undecidable.
- Coverability for TdPN is decidable.

Definition: Timed Petri Net

A *timed Petri net* (TdPN) is a tuple $\mathcal{N} = (P, T, \text{Pre}, \text{Post}, m_0)$ where:
- $P$ is a finite set of *places*
- $T$ is a finite set of *transitions* such that $P \cap T = \emptyset$
- $\text{Pre} : T \rightarrow \text{Bag}(P \times \text{Int})$
- $\text{Post} : T \rightarrow \text{Bag}(P \times \text{Int})$
- $m_0 \in \text{Bag}(P)$ is the initial marking
Timed Petri Net (TdPN)

Example: $\mathcal{N} = (P, T, Pre, Post, m_0)$

- $P = \{p_0, p_1, p_2\}$  $T = \{t_1, t_2, t_3\}$
- $Pre(t_1) = (p_0, [0, 1])$  $Post(t_1) = (p_1, [0, 0])$
- $m_0(p_0) = 1$  $m_0(p_1) = m_0(p_2) = 0$

(we start with one token in $p_0$ with age 0)
**Timed Petri Net (TdPN)**

**Definition:** configurations of $\mathcal{N} = (P, T, Pre, Post, m_0)$

The set of configurations of $\mathcal{N}$ is $Conf_{\mathcal{N}} := Bag(P \times \mathbb{R}_{\geq 0})$.

**Example:**

$$(p_0, 2.4) + (p_1, 4) + (p_1, 4) + (p_1, 3) + (p_2, 1.2) + (p_2, 5)$$

**Example:** (satisfaction of precondition)

$$(p_0, 2.4) + (p_1, 4) + (p_1, 3) \models (p_0, [2, 3]) + (p_1, [4, 4]) + (p_1, [0, \infty])$$

**Notation:**

For $\gamma \in Conf_{\mathcal{N}}$ and $\alpha \in Bag(P \times \text{Int})$, we write

$$\gamma \models \alpha$$

if there is $\beta \in Bag(P \times \mathbb{R}_{\geq 0} \times \text{Int})$ such that

- $\Pi_{1,2}(\beta) = \gamma$ and $\Pi_{1,3}(\beta) = \alpha$ where $[\Pi_{1,2}(\beta)](p, x) = \sum_{I \in \text{Int}} \beta(p, x, I)$
- for all $(p, x, I) \in \text{dom}(\beta): x \in I$
Timed Petri Net (TdPN)

Definition: semantics of $\mathcal{N} = (P, T, Pre, Post, m_0)$

Let the infinite transition system $TS_\mathcal{N} = (Conf_\mathcal{N}, q_0, \rightarrow)$ be given as follows:

- $q_0 = \sum_{p \in P} m_0(p) \cdot (p, 0)$
- (delay transition) for all $d \in \mathbb{R}_{>0}$:
  $$\gamma \xrightarrow{d} \gamma + d$$
- (discrete transition) for all $t \in T$
  $$\gamma \xrightarrow{t} \gamma'$$

if there are $\gamma^-, \gamma^+ \in Bag(P \times \mathbb{R}_{\geq 0})$ with $\gamma^- \leq \gamma$ such that

- $\gamma' = \gamma - \gamma^- + \gamma^+$
- $\gamma^- \models Pre(t)$
- $\gamma^+ \models Post(t)$
Timed Petri Net (TdPN)

Example:

\[
\begin{align*}
(p_0, 0) & \overset{1}{\rightarrow} (p_0, 1) \overset{t_1}{\rightarrow} (p_0, 0) + (p_1, 0) \overset{t_1}{\rightarrow} (p_0, 0) + (p_1, 0) + (p_1, 0) \\
& \overset{2}{\rightarrow} (p_0, 2) + (p_1, 2) + (p_1, 2) \overset{t_2}{\rightarrow} (p_0, 0) + (p_1, 2) + (p_1, 2) + (p_2, 0) \\
& \overset{t_2}{\rightarrow} (p_0, 0) + (p_1, 2) \overset{1}{\rightarrow} (p_0, 1) + (p_1, 3)
\end{align*}
\]
Plan

3 Time(d) Petri Nets
   Time Petri Nets (TPN)
   Decision problems for TPN
   Timed Petri Nets (TdPN)
   Decision problems for TdPN
   Decidability of Coverability for TdPN
   Expressiveness (credits to Serge Haddad)
Decision problems for TPN

**Definition: Reachability for TdPN**

**INPUT:** TdPN $\mathcal{N} = (P, T, Pre, Post, m_0)$ and $\gamma \in Bag(P \times Q_{\geq 0})$

**QUESTION:** Is $\gamma$ reachable in $TS_\mathcal{N}$ from $q_0$?

**Notation:**

For $\Gamma \subseteq Conf_\mathcal{N}$, let:

$$\Gamma^\uparrow := \{ \gamma' \in Conf_\mathcal{N} \mid \text{there is } \gamma \in \Gamma \text{ such that } \gamma \leq \gamma' \}$$

Here, $\gamma \leq \gamma'$ if $\gamma(p, x) \leq \gamma'(p, x)$ for all $(p, x) \in P \times R_{\geq 0}$.

**Definition: Coverability for TdPN**

**INPUT:** TdPN $\mathcal{N} = (P, T, Pre, Post, m_0)$ and $\gamma \in Bag(P \times Q_{\geq 0})$

**QUESTION:** Is there a configuration in $\{\gamma\}^\uparrow$ that is reachable in $TS_\mathcal{N}$ from $q_0$?
Reachability for TdPN is undecidable.

Proof: (Idea)

The proof is by reduction from emptiness of 2-counter machines (2-CM).

We construct a TdPN $\mathcal{N}$ such that the 2-CM executes an instruction of the form

\[ \ell : \text{accept when both counters are 0} \]

iff configuration $(\ell, 0)$ (no tokens in $c_1$ and $c_2$) is reachable in $TS_{\mathcal{N}}$. 
Proof of undecidability

A 2-CM is a sequence of labeled instructions:

1 : instr_1
2 : instr_2
  ...
n : instr_n

Each instruction is one of the following (where $c \in \{c_1, c_2\}$):

- $\ell$ : accept
- $c$ : $c++$
- $\ell$ : if $c == 0$ goto $\ell'$ else $c--$

The semantics of an instruction and of a program are as expected:

- $c++$ increments counter $c$ by 1
- $c--$ decrements counter $c$ by 1
Proof of undecidability

Idea:

We introduce places $c_1$ and $c_2$ to simulate counters:
The number of tokens in $c_i$ is the current counter value of $c_i$.

$\ell : c_1 \rightarrow c_1$
Proof of undecidability

$\ell : \text{if } c_1 == 0 \text{ goto } \ell' \text{ else } c_1 --$

- Invariant: All tokens are 0 after each transition.
- If there is token in $c_1$, then fire $t_{\ell} - -$.
- If there is token in $c_1$ and $t_{\ell} - -$ is missed, all tokens in $c_1$ will be “dead”.
Plan

3 Time(d) Petri Nets
Time Petri Nets (TPN)
Decision problems for TPN
Timed Petri Nets (TdPN)
Decision problems for TdPN

Decidability of Coverability for TdPN
Expressiveness (credits to Serge Haddad)
Decidability of coverability for TdPN

**Definition: Coverability for TdPN**

**Input:** TdPN $\mathcal{N} = (P, T, Pre, Post, m_0)$ and $\hat{\gamma} \in Bag(P \times \mathbb{Q}_{\geq 0})$

**Question:** Is there a configuration in $\{\hat{\gamma}\}^\uparrow$ that is reachable in $TS_{\mathcal{N}}$ from $q_0$?

**Theorem:**

Coverability for TdPN is decidable.

**Proof: (Outline)**

- Wlog., we assume that all ages in $\hat{\gamma}$ are integers (otherwise, change granularity).
- Classify configurations into infinitely many regions (like in timed automata, but infinite).
- Order regions wrt. well-quasi-ordering (wqo) $\leq$.
- Iterate computation of $pre(\{\hat{\gamma}\}^\uparrow)$ until there are no new configurations (process terminates because $\leq$ is wqo).
- Compare result with $q_0$. 
Decidability of coverability for TdPN

We fix a TdPN $\mathcal{N} = (P, T, Pre, Post, m_0)$ and $\hat{\gamma} \in Bag(P \times \mathbb{Q}_{\geq 0})$. Let $\text{max}$ be the maximal integer part occurring in $\mathcal{N}$ or $\hat{\gamma}$.

Idea:

- Instead of clocks, consider tokens $\leadsto$ infinitely many.
- Keep only integer part and order them according to fractional part (as in timed automata – but now for infinitely many clocks).

Example: ($\text{max} = 4$)

$$(p, 1) + (p, 2.8) + (q, 3) + (q, 0.8) + (q, 5.1) + (r, 1.5)$$

$$\Downarrow$$

$$\underbrace{(p, 1) + (q, 3)}_{a_0} \quad \underbrace{(r, 1)}_{a_1} \quad \underbrace{(p, 2) + (q, 0)}_{a_2} \quad (q, \infty)_{a_{\infty}}$$

Definition: region

A region is a sequence $\mathcal{R} = a_0a_1 \ldots a_na_{\infty}$ with $n \geq 0$ such that:

- $a_i \in Bag(P \times \{0, \ldots, \text{max}\})$ for all $i \in \{0, \ldots, n\}$
- $a_{\infty} \in Bag(P \times \{\infty\})$
- $|a_i| > 0$ for all $i \in \{1, \ldots, n\}$
Decidability of coverability for TdPN

Example: \( \max = 4 \)

\[
\begin{aligned}
(p, 1) + (q, 3) + (r, 1.5) + (p, 2.8) + (q, 0.8) + (q, 5.1) \\
\in (p, 1) + (q, 3) + (r, 1) + (p, 2) + (q, 0) + (q, \infty)
\end{aligned}
\]

Notation for region \( \mathcal{R} = a_0 a_1 \ldots a_n a_\infty \):

Let \( [\mathcal{R}] \) be the set of all \( \gamma \in \text{Conf}_N \) such that \( \exists \gamma_1, \ldots, \gamma_n, \gamma_\infty \in \text{Bag}(P \times \mathbb{R}_{\geq 0}) \):

- \( \gamma = a_0 + \gamma_1 + \ldots + \gamma_n + \gamma_\infty \)
- for all \( i \in \{1, \ldots, n\} \) and \( (p, x) \leq \gamma_i \), we have \( 0 < x - \lfloor x \rfloor \)
- for all \( 1 \leq i \leq n \) (let \( \lfloor x \rfloor = \infty \) if \( x > \max \)):
  - \( \lfloor \gamma_i \rfloor = a_i \) and \( \lfloor \gamma_\infty \rfloor = a_\infty \)
- for all \( 1 \leq i \leq n \) and \( (p, x) + (q, y) \leq \gamma_i \):
  - (in each group, identical fractional parts)
  \[
  0 < x - \lfloor x \rfloor = y - \lfloor y \rfloor
  \]
- for all \( 1 \leq i < j \leq n \), \( (p, x) \leq \gamma_i \), and \( (q, y) \leq \gamma_j \):
  - (fractional parts are ordered)
  \[
  x - \lfloor x \rfloor < y - \lfloor y \rfloor
  \]
wqo on regions

Definition: wqo

A well-quasi-ordering (wqo) over a set $X$ is a reflexive and transitive binary relation $\preceq \subseteq X \times X$ such that, for every infinite sequence $x_1, x_2, x_3, \ldots$, there are $i < j$ with $x_i \preceq x_j$.

Definition: wqo on regions

For two regions $\mathcal{R}$ and $\mathcal{R}'$, we let $\mathcal{R} \preceq \mathcal{R}'$ if $[\mathcal{R}']^\uparrow \subseteq [\mathcal{R}]^\uparrow$.

Lemma:

Let $\mathcal{R} = a_0a_1 \ldots a_na_\infty$ and $\mathcal{R}' = b_0b_1 \ldots b_mb_\infty$. We have $\mathcal{R} \preceq \mathcal{R}'$ iff there is $f : \{1, \ldots, n\} \to \{1, \ldots, m\}$ strictly increasing (which implies $n \leq m$) such that:

- $a_0 \leq b_0$
- $a_\infty \leq b_\infty$
- $a_i \leq b_{f(i)}$ for all $i \in \{1, \ldots, n\}$

Proof: (Exercise)

If $\mathcal{R}'$ contains more tokens, its upward closure is smaller.
wqo on regions

Example: \( f(1) = 1 \)

\[
\emptyset \quad (q, 3) \quad (r, \infty) \quad \preceq \quad (p, 1) \quad (r, 0) + (q, 3) \quad (p, 2) \quad (r, \infty)
\]

\[
a_0 \quad a_1 \quad a_\infty \quad b_0 \quad b_1 \quad b_2 \quad b_\infty
\]

Lemma:

The relation \( \preceq \) is a wqo.

Proof:

Follows from previous lemma and Higman’s Lemma:
The subsequence relation over strings over a finite alphabet is a wqo.
Decidability of coverability for TdPN

**Definition:**

Let $\gamma \in \text{Conf}_N$ and $t \in T$.

- $t\text{-}pre(\gamma) := \{ \gamma' \mid \gamma' \xrightarrow{t} \gamma \}$
- $delay\text{-}pre(\gamma) := \{ \gamma' \mid \gamma' \xrightarrow{d} \gamma \text{ for some } d > 0 \text{ such that } R(\gamma') \neq R(\gamma) \text{ and } R(\gamma' + d') \in \{ R(\gamma'), R(\gamma) \} \text{ for all } d' \in [0, d) \}$
- $pre(\gamma) = delay\text{-}pre(\gamma) \cup \bigcup_{t \in T} t\text{-}pre(\gamma)$

Here, $R(\gamma)$ is the unique region $R'$ with $\gamma \in R'$.

**Lemma:**

Let $R$ be a region and $t \in T$.

- $delay\text{-}pre([R]^\dagger) = [R_1]^\dagger \cup \ldots \cup [R_k]^\dagger$
  for some effectively computable $k$ and $R_i$
  set $delay\text{-}pre(R) := \{ R_1, \ldots, R_k \}$
- $t\text{-}pre([R]^\dagger) = [R_1]^\dagger \cup \ldots \cup [R_k]^\dagger$
  for some effectively computable $k$ and $R_i$
  set $t\text{-}pre(R) := \{ R_1, \ldots, R_k \}$
The algorithm

Algorithm

\[ S := \{ R(\hat{\gamma}) \} \]

repeat

\[ S' := S \]

\[ S := S \cup ((\text{delay-pre}(S) \cup \bigcup_{t \in T} t-\text{pre}(S))) \backslash \{ R' \mid \text{there is } R \in S \text{ such that } R \preceq R' \} \]

until \( S = S' \)

check if \( R \preceq R(q_0) \) for some \( R \in S \)

Proof: termination

By the wqo property.

Proof: correctness

By the previous lemma, \([S]^{\uparrow} = \text{pre}^*([R(\hat{\gamma})]^{\uparrow}).\)
Computation of time-delay predecessors

Computation of \( \text{delay-pre}(\lceil R \rceil^\uparrow) \) for \( R = a_0a_1 \ldots a_na_\infty \)

We distinguish three cases:

1. \( a_0 \cap (P \times \{0\}) \neq \emptyset \)

   \[ \implies \text{delay-pre}(\lceil R \rceil^\uparrow) = \emptyset \]

   (cannot let elapse \( d > 0 \) and reach 0)

2. \( a_0 \cap (P \times \{0\}) = \emptyset \) and \( a_0 \neq \emptyset \)

   \[ \implies \text{delay-pre}(\lceil R \rceil^\uparrow) = [\emptyset a_1 \ldots a_na_{n+1}a_\infty]^\uparrow \]

   where \( a_{n+1} = "a_0 - 1" \)

   (very small reverse time elapse; no token in \( a_1 \ldots a_n \) reaches border)

3. \( a_0 = \emptyset \)

   Three cases:
   - 3.1 tokens of \( a_1 \) will first reach integral value
   - 3.2 some tokens of \( a_\infty \) will first reach \( \max \) (for some \( b_\infty \leq a_\infty \))
   - 3.3 both at the same time

3.3 \( \implies \text{delay-pre}(\lceil R \rceil^\uparrow) = [a'_0a_2 \ldots a_na'_\infty]^\uparrow \)

   where \( a'_\infty = a_\infty - b_\infty \) and \( a'_0 = a_1 + b_\infty[\infty \rightarrow \max] \)
Computation of transition predecessors

**Notation:**

Let \( \mathcal{R} = a_0a_1 \ldots a_n a_\infty \) be a region. For \((p, x) \in a_i\), we write \((i, x) \models I\) if the “real value” of \(x\) belongs to \(I\).

**Example:**

For \(\mathcal{R} = \emptyset(p, 2)\emptyset\), we have \((1, 2) \models (2, 3]\).

**Computation of \(t\)-\(pre([\mathcal{R}]^\uparrow)\) for \(\mathcal{R} = a_0a_1 \ldots a_n a_\infty\)**

Transition \(t\) produces a bag of tokens \(Post(t)\). This bag might appear in the \(a_i\)'s or only in the upward closure. \(\implies\) Choose:

- \(post_0, \ldots, post_n \in Bag(P \times \{0, \ldots, \text{max}\} \times \text{Int})\)
- \(post_\infty \in Bag(P \times \{\infty\} \times \text{Int})\)

such that

- \((i, x) \models I\) for all \(i \in \{0, \ldots, n, \infty\}\) and \((p, x, I) \leq post_i\)
- \(\Pi_{1,2}(post_i) \leq a_i\) for all \(i \in \{0, \ldots, n, \infty\}\)
- \((\sum_i \Pi_{1,3}(post_i)) \leq Post(t)\)
Computation of transition predecessors

We obtain \( R' = a'_0a'_1...a'_na'_\infty \leq R \) by deleting all \( post_i \) from \( a_i \). Choose:

- \( n'' \geq n' \)
- \( pre_0, ..., pre_{n''} \in Bag(P \times \{0, \ldots, max\} \times Int) \)
- \( pre_\infty \in Bag(P \times \{\infty\} \times Int) \)
- \( f : \{1, \ldots, n'\} \rightarrow \{1, \ldots, n''\} \) strictly increasing such that
  - \( (i, x) \models I \) for all \( i \in \{0, \ldots, n'', \infty\} \) and \( (p, x, I) \leq pre_i \)
  - \( (\sum_i \Pi_{1,3}(pre_i)) = Pre(t) \)
  - \( a''_0 = a'_0 + \Pi_{1,2}(pre_0) \) and \( a''_\infty = a'_\infty + \Pi_{1,2}(pre_\infty) \)
  - for all \( i \in \{1, \ldots, n''\} \):
    - \( a''_i = \begin{cases} a'_j + \Pi_{1,2}(pre_i) & \text{if } f(j) = i \text{ for some } j \in \{1, \ldots, n'\} \\ \Pi_{1,2}(pre_i) & \text{otherwise} \end{cases} \)

In this way, we obtain one \( t \)-predecessor \( R'' \) of \( [R]^\uparrow \) with \( R' \preceq R'' \). It depends on the choices of \( post_i, pre_i, n', n'', f \) etc. We obtain finitely many regions \( R_1, \ldots, R_k \) and one can show: \( t\cdot pre([R]^\uparrow) = [R_1]^\uparrow \cup \ldots \cup [R_k]^\uparrow \)
Exercises

Exercise:
Try to formalize the following specification as a TPN and as a TdPN:

- There are concurrent events $e$ and $e'$ which may (but do not have to) occur.
- Event $e$ may only occur at instants in $[0, 1]$.
- Event $e'$ may occur at every instant.

Exercise:
Try to formalize the following specification as a TPN and as a TdPN:

- There is a single event which must occur at instant in $[0, 1]$. 

Plan

3 Time(d) Petri Nets
   Time Petri Nets (TPN)
   Decision problems for TPN
   Timed Petri Nets (TdPN)
   Decision problems for TdPN
   Decidability of Coverability for TdPN

Expressiveness (credits to Serge Haddad)
Definition:
A T(d)PN with acceptance condition is a T(d)PN that, in addition, has:

- a labeling function \( \lambda : T \rightarrow \Sigma \cup \{\varepsilon\} \) for some alphabet \( \Sigma \)
- a finite set \( F \subseteq Bag(P) \)

Definition: language

- The timed language of a T(d)PN \( \mathcal{N} \) is denoted by \( L_t(\mathcal{N}) \).
- The untimed language of a T(d)PN \( \mathcal{N} \) is denoted by \( L(\mathcal{N}) \).

Both are defined in the obvious manner on the basis of \( TS_\mathcal{N} \).
Time(d) Petri nets with acceptance condition

\[
\begin{align*}
(p_0, 0) &\xrightarrow{1} (p_0, 1) \xrightarrow{t_1} (p_0, 0) + (p_1, 0) \xrightarrow{t_1} (p_0, 0) + (p_1, 0) + (p_1, 0) \\
&\xrightarrow{2} (p_0, 2) + (p_1, 2) + (p_1, 2) \xrightarrow{t_2} (p_0, 0) + (p_1, 2) + (p_1, 2) + (p_2, 0) \\
&\xrightarrow{t_3} (p_0, 0) + (p_1, 2) \xrightarrow{1} (p_0, 1) + (p_1, 3)
\end{align*}
\]

Example:

Suppose \((1, 1, 0)\) is a final configuration. Then:

\[(a, 1)(a, 1)(b, 3)(c, 3) \in L_t(\mathcal{N})\]
A hierarchy of languages

Theorem: (for untimed Petri nets)

\[ \{a^n b^n c^n \mid n \in \mathbb{N}\} \subset \{a^n b^n \mid n \in \mathbb{N}\} \subset \{wcw^{-1} \mid w \in \{a, b\}^*\} \subset \{wcw \mid w \in \{a, b\}^*\} \]
A hierarchy of languages

Example: An (untimed) Petri net for \( \{ a^n b^n c^n \mid n \in \mathbb{N} \} \)

\[
F = \{ (1, 0, 0, 0, 0, 0), (0, 0, 0, 0, 0, 1) \}
\]
Observation: There are non regular untimed languages of (unbounded) nets while all TA untimed languages are regular.

For timed languages:

**Theorem: From TA to nets**
- Timed Automata can be “simulated” by bounded TPNs.
- Timed Automata can be “simulated” by bounded TdPNs.

But these simulations are not valid wrt. bisimulation.

**Theorem: From bounded nets to TA**
- Bounded TPNs can be simulated by Timed Automata.
- Bounded TdPNs can be simulated by Timed Automata.

And these simulations are valid wrt. bisimulation.
From TA to TPN

The structural part

- There is a place per state.
- There is an \textit{untimed} net transition per automata transition, i.e. with interval $[0, \infty)$, which takes as input the source state.

Time guards

- There is a place $T_{x \sim c}$ per conjunct of a guard $x \sim c$ which is an input for the transitions where it occurs.
- The marking of such a place is ruled by an “independent” timed subnet.

Clock Resets

- Clock resets consists (in zero time) to reinitialize all the timed subnets related to the clocks to be reset \textbf{whatever the state of these subnets}.
- The clock resets take place after the automata transition and before producing the token in the destination state.
From TA to TPN: an Example

\[ x \geq 1 \land y < 1 ; c ; x := 0 \]

\[ x < 1 ; a ; \emptyset \]
\[ x < 1 ; b ; y := 0 \]
From TA to TdPN

The structural part

- There is a place per state.
- There is a net transition per automata transition which takes as *untimed* input, i.e. with interval \([0, \infty)\), the source state.

Time guards

- There is a place \(T_{x \sim c}\) per conjunct of a guard \(x \sim c\) which is an untimed input for the transitions where it occurs.
- The *control* that a token of such a place has been checked at appropriate time is performed by an “independent” timed subnet *with an acceptance condition*.

Clock Resets

- Clock resets consists (in zero time) in reinitializing all the timed subnets related to the clocks to be reset *whatever the state of these subnets*.
- The clock resets take place after the automata transition and before producing the token in the destination state.
From TA to TdPN: an Example

Acceptance condition
\[ T_{x \leq 1} = 0 \]
From Bounded TPN to TA

There is one clock $x_t$ per transition $t$.

Build the reachability graph.
- The locations of the TA are the reachable markings.
- The transitions of the TA are the transitions of the reachability graph.

Define the invariants. Given $T_m$ the set of transitions enabled at $m$
The invariant of $m$ is: $\bigwedge_{t \in T_m} x_t \leq l(t)$.

Define the guards and updates. Given a transition $m \xrightarrow{t} m'$,
- The guard is $x(t) \geq e(t)$
- The clocks to be reset are those associated with the newly enabled transitions.

Warning: There exists an alternative structural translation but it uses both networks of TA and finite counters.
From Bounded TPN to TA: an Example

$$t_3, c, [4, 6]$$

$$t_2, b, [2, 3]$$

$$t_1, a, [1, \infty[$$

$$x_2 \leq 3$$

$$x_3 \geq 4; c; \emptyset$$

$$x_2 \leq 3 \land x_3 \leq 6$$

$$x_3 \geq 4; c; x_1 := x_2 := x_3 := 0$$

$$x_1 \geq 1; a; x_1 := 0$$

$$x_1 \geq 1; a; x_1 := x_2 := 0$$

$$x_2 \geq 2; b; x_1 := x_3 := 0$$

$$x_2 \geq 2; b; \emptyset$$

$$x_3 \leq 6$$
From Bounded TdPN to TA

Transform the net such that all intervals of output arcs are $[0, 0]$.

Build the reachability graph with token identities $p_i$ and instances of transitions

▷ The locations of the TA are the reachable markings.
▷ The transitions of the TA are the transitions of the reachability graph.

There is one clock $x_{p_i}$ per token $p_i$ in place $p$.

Define the guards and updates. Given a transition $m \xrightarrow{t} m'$,

▷ For every input arc $(p, t)$ labelled by interval $[a, b]$ and consumed token $p_i$ the guard is $a \leq x_{p_i} \leq b$
▷ The clocks to be reset are those associated with the tokens that are produced.
From Bounded TdPN to TA: an Example
Main References


On relation between TA and TPN


