Computational approaches to analyze complex dynamic systems: model-checking and its applications.

Part 3: Model-checking of timed transitions systems: timed extensions of Petri nets

Morgan Magnin

morgan.magnin@irccyn.ec-nantes.fr

www.morganmagnin.net

NII - *Inoue Laboratory*

École Centrale de Nantes - IRCCyN - *MeForBio team*

Lecture Series - Lecture 3 / NII - 2013/04/10

1. Models: from time Petri nets to stopwatch Petri nets
   - From time Petri nets to stopwatch Petri nets
   - Dense-time vs discrete-time
   - State space abstraction dedicated to discrete-time
2. Formalizing specification through timed modal logics
   - TCTL model-checking
   - Introduction to TCTL parametric model-checking
3. Application to biological systems
4. Model-checking tools

Motivations

Objective: formal verification of properties

- Model the system S :
  - Discrete models: finite state automata, Petri nets, ... $\Rightarrow$ Lecture 1
  - Timed models:
    - Timed extensions of finite state automata $\Rightarrow$ Lecture 2
    - **Timed extensions of Petri nets: time/stopwatch Petri nets $\Rightarrow$ Lecture 3**
- Formalize the specification φ :
  - Observers
  - Temporal logics: LTL, CTL, ... $\Rightarrow$ Lecture 1
  - **Timed extensions of temporals logics: TCTL, ... $\Rightarrow$ Lectures 2 & 3**
- Does $S \models \varphi$?

Model-checking algorithms

$\Rightarrow$ State space exploration

Motivations

Objective: formal verification of properties

- Model the system S :
  - Discrete models: finite state automata, Petri nets, ... $\Rightarrow$ Lecture 1
  - Timed models:
    - Timed extensions of finite state automata $\Rightarrow$ Lecture 2
    - **Timed extensions of Petri nets: time/stopwatch Petri nets $\Rightarrow$ Lecture 3**
- Formalize the specification φ :
  - Observers
  - Temporal logics: LTL, CTL, ... $\Rightarrow$ Lecture 1
  - **Timed extensions of temporals logics: TCTL, ... $\Rightarrow$ Lectures 2 & 3**
- Does $S \models \varphi$?

Model-checking algorithms

$\Rightarrow$ State space exploration

Some major issues

Need for modeling tasks with suspending/resuming features

Expressivity/Decidability compromise to discuss ⇒ Lectures 2 & **3**

State space combinatorial explosion

- Need for symbolic approaches ⇒ Lectures 2 & **3**
- Need for new models and abstracted algorithms ⇒ Lecture 4

Some major issues

Need for modeling tasks with suspending/resuming features

Expressivity/**Decidability** compromise to discuss ⇒ Lectures 2 & **3**

State space combinatorial explosion

- Need for symbolic approaches ⇒ Lectures 2 & **3**
- Need for new models and abstracted algorithms ⇒ Lecture 4

Last week's and today's issue

Tricky question

Coming from France, why do I need an average 3-4 days period not to be jet-lagged anymore in Tōkyō?

Observation

- Discrete models do not encompass sufficient information to get a thorough description of the gene regulation network behind the circadian clock **w.r.t. time**
- Some related issues:
  - Is it possible to determine the lower limit of the day/night period cycle during which the circadian clock continues to stabilize?
  - Why does the body better support backward phase delay than advance phase delay?
- $\rightarrow$ On-going modeling project with **biologists** and **computer scientists** (CNRS PEPII funded project CirClock)

Contribution

Scientific challenge

How can we get information about the **production and degradation rates** of a protein in a biological regulatory network?

Objectives of this talk

- From discrete model to timed model → emphasize on the **progressive enrichment** of a model and its drawbacks
- Focus on the introduction of **quantitative** timing information
- Discuss the most appropriate **time semantics** adapted to the model

Joint work with

- G. Bernot, JP. Comet, A. Richard (methodology and application to biology)
- D. Lime, P. Molinaro and O.H. Roux (Petri nets theory)

Overview

1. Models: from time Petri nets to stopwatch Petri nets
   - From time Petri nets to stopwatch Petri nets
   - Dense-time vs discrete-time
   - State space abstraction dedicated to discrete-time
2. Formalizing specification through timed modal logics
   - TCTL model-checking
   - Introduction to TCTL parametric model-checking
3. Application to biological systems
4. Model-checking tools

Time Petri nets - Reminder

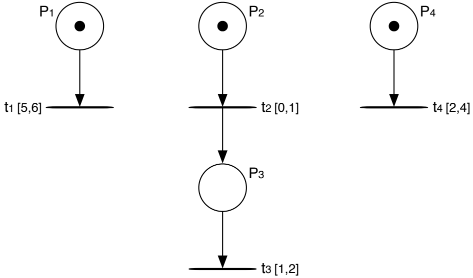


Figure: A dense-time Petri net

$$\begin{array}{lcl} \{P_1, P_2, P_4\} & & \{P_1, P_2, P_4\} \\ \theta(t_1) = 0 & & \theta(t_1) = 0.2 \\ \theta(t_2) = 0 & \xrightarrow{0.2} & \theta(t_2) = 0.2 \\ \theta(t_4) = 0 & & \theta(t_4) = 0.2 \end{array}$$

Time Petri nets - Reminder

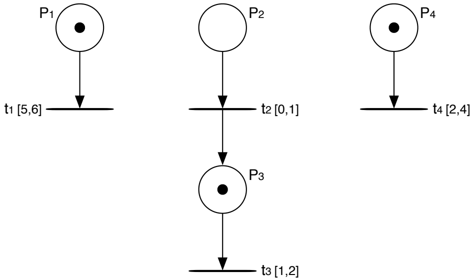


Figure: A dense-time Petri net

$$\begin{array}{l}\{P_1, P_2, P_4\}\\ \theta(t_1) = 0\\ \theta(t_2) = 0\\ \theta(t_4) = 0\end{array} \xrightarrow{0.2} \begin{array}{l}\{P_1, P_2, P_4\}\\ \theta(t_1) = 0.2\\ \theta(t_2) = 0.2\\ \theta(t_4) = 0.2\end{array} \xrightarrow{t_2} \begin{array}{l}\{P_1, P_3, P_4\}\\ \theta(t_1) = 0.2\\ \theta(t_3) = 0\\ \theta(t_4) = 0.2\end{array} \xrightarrow{0.9} \dots$$

Time Petri nets - Reminder

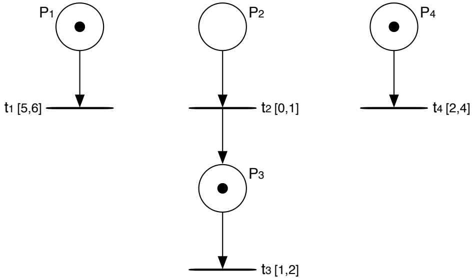


Figure: A discrete-time Petri net

$$\begin{matrix}\{P_1, P_2, P_4\} \\ \theta(t_1) = 0 \\ \theta(t_2) = 0 \\ \theta(t_4) = 0\end{matrix} \xrightarrow{1} \begin{matrix}\{P_1, P_2, P_4\} \\ \theta(t_1) = 1 \\ \theta(t_2) = 1 \\ \theta(t_4) = 1\end{matrix} \xrightarrow{t_2} \begin{matrix}\{P_1, P_3, P_4\} \\ \theta(t_1) = 1 \\ \theta(t_3) = 0 \\ \theta(t_4) = 1\end{matrix} \xrightarrow{1} \begin{matrix}\{P_1, P_3, P_4\} \\ \theta(t_1) = 2 \\ \theta(t_3) = 1 \\ \theta(t_4) = 2\end{matrix} \xrightarrow{1} \ldots$$

Time Petri nets - About read arcs

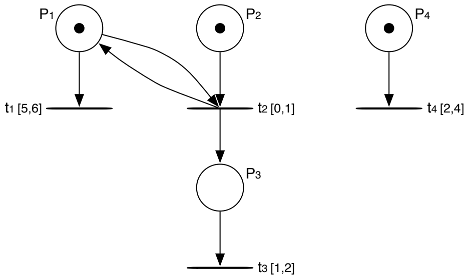


Figure: An other (dense-time) Petri net

$$\begin{array}{lcl} \{P_1, P_2, P_4\} & & \{P_1, P_2, P_4\} \\ \theta(t_1) = 0 & \xrightarrow{0.2} & \theta(t_1) = 0.2 \\ \theta(t_2) = 0 & & \theta(t_2) = 0.2 \\ \theta(t_4) = 0 & & \theta(t_4) = 0.2 \end{array}$$

Time Petri nets - About read arcs

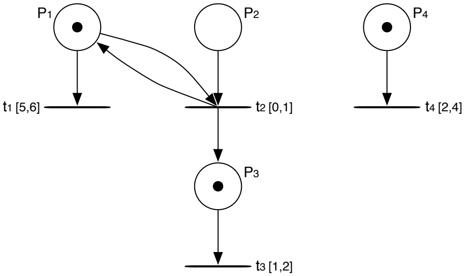


Figure: An other (dense-time) Petri net

$$\begin{array}{l}\{P_1, P_2, P_4\}\\ \theta(t_1) = 0\\ \theta(t_2) = 0\\ \theta(t_4) = 0\end{array} \xrightarrow{0.2} \begin{array}{l}\{P_1, P_2, P_4\}\\ \theta(t_1) = 0.2\\ \theta(t_2) = 0.2\\ \theta(t_4) = 0.2\end{array} \xrightarrow{t_2} \begin{array}{l}\{P_1, P_3, P_4\}\\ \theta(t_1) = 0\\ \theta(t_3) = 0\\ \theta(t_4) = 0.2\end{array} \xrightarrow{0.9} \dots$$

Time Petri nets - About read arcs

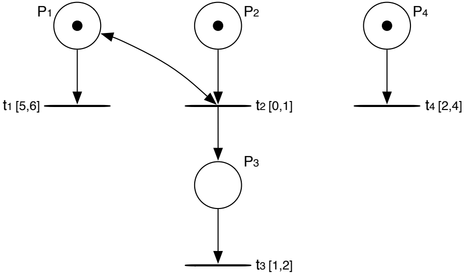


Figure: A Time Petri net with read arcs

$$\begin{array}{l} \{P_1, P_2, P_4\} \\ \theta(t_1) = 0 \\ \theta(t_2) = 0 \\ \theta(t_4) = 0 \end{array} \quad \xrightarrow{0.2} \quad \begin{array}{l} \{P_1, P_2, P_4\} \\ \theta(t_1) = 0.2 \\ \theta(t_2) = 0.2 \\ \theta(t_4) = 0.2 \end{array}$$

Time Petri nets - About read arcs

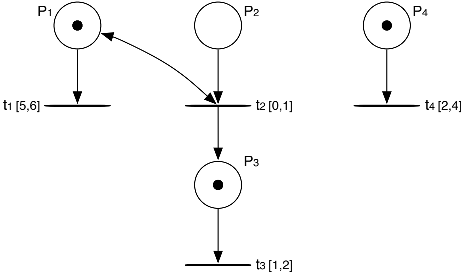


Figure: A Time Petri net with read arcs

$$\begin{array}{l}\{P_1, P_2, P_4\}\\ \theta(t_1)=0\\ \theta(t_2)=0\\ \theta(t_4)=0\end{array} \xrightarrow{0.2} \begin{array}{l}\{P_1, P_2, P_4\}\\ \theta(t_1)=0.2\\ \theta(t_2)=0.2\\ \theta(t_4)=0.2\end{array} \xrightarrow{t_2} \begin{array}{l}\{P_1, P_3, P_4\}\\ \theta(t_1)=\mathbf{0.2}\\ \theta(t_3)=0\\ \theta(t_4)=0.2\end{array} \xrightarrow{0.9} \dots$$

Time Petri nets - About read arcs

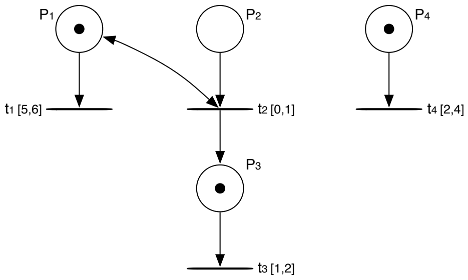


Figure: A Time Petri net with read arcs

Theorem

Time Petri nets with read arcs are more expressive than time Petri nets.

Stopwatch Petri nets - SwPN

Objective

Keep track of the state of a suspended action

Solution

Extend the Time Petri nets with the notion of stopwatches

- Resources and proprieties integrated to the places [RD01] or transitions [BFSV04]
- Activator arcs [BLRV07]
- **Inhibitor hyperarcs** [RL04] :

  If t is enabled by the marking M :
  - t is **inhibited** by $M \Rightarrow \dot{\theta}(t) = 0$
  - t **is not inhibited** by $M \Rightarrow \dot{\theta}(t) = 1$

Stopwatch Petri nets - SwPN

Objective

Keep track of the state of a suspended action

Solution

Extend the Time Petri nets with the notion of stopwatches

- Resources and proprieties integrated to the places [RD01] or transitions [BFSV04]
- Activator arcs [BLRV07]
- **Inhibitor hyperarcs** [RL04] :

  If t is enabled by the marking M :
  - t is **inhibited** by $M \Rightarrow \dot{\theta}(t) = 0$
  - t **is not inhibited** by $M \Rightarrow \dot{\theta}(t) = 1$

Stopwatch Petri nets

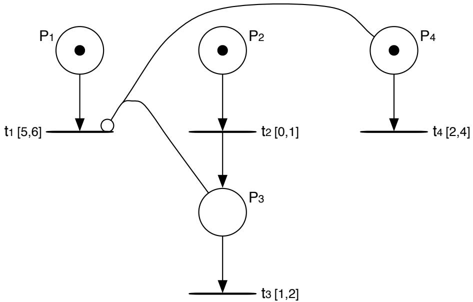


Figure: Stopwatch Petri nets: Petri nets with suspension / resuming features

$$\begin{array}{l} \{P_1, P_2, P_4\} \\ \theta(t_1) = 0 \\ \theta(t_2) = 0 \\ \theta(t_4) = 0 \end{array} \quad \xrightarrow{0.2} \quad \begin{array}{l} \{P_1, P_2, P_4\} \\ \theta(t_1) = 0.2 \\ \theta(t_2) = 0.2 \\ \theta(t_4) = 0.2 \end{array}$$

Stopwatch Petri nets

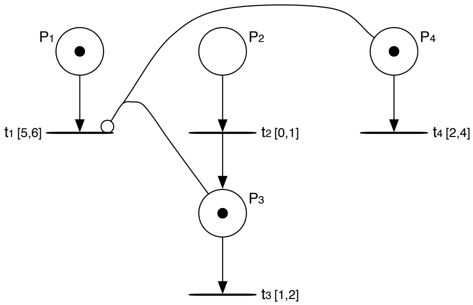


Figure: A SwPN : t_1 inhibited iff $(M(P_3) \geq 1$ **and** $M(P_4) \geq 1)$

$$\begin{array}{l}\{P_1, P_2, P_4\}\\ \theta(t_1)=0\\ \theta(t_2)=0\\ \theta(t_4)=0\end{array} \xrightarrow{0.2} \begin{array}{l}\{P_1, P_2, P_4\}\\ \theta(t_1)=0.2\\ \theta(t_2)=0.2\\ \theta(t_4)=0.2\end{array} \xrightarrow{t_2} \begin{array}{l}\{P_1, P_3, P_4\}\\ \theta(t_1)=0.2\\ \theta(t_3)=0\\ \theta(t_4)=0.2\end{array} \xrightarrow{1} \begin{array}{l}\{P_1, P_3, P_4\}\\ \theta(t_1)=0.2\\ \theta(t_3)=1\\ \theta(t_4)=1.2\end{array}$$

Stopwatch Petri nets

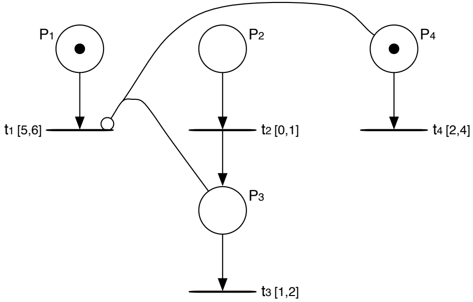


Figure: A SwPN : After the "reactivation" of t_1

$$\begin{array}{l}\{P_1, P_2, P_4\}\\ \theta(t_1) = 0\\ \theta(t_2) = 0\\ \theta(t_4) = 0\end{array} \xrightarrow{0.2} \begin{array}{l}\{P_1, P_2, P_4\}\\ \theta(t_1) = 0.2\\ \theta(t_2) = 0.2\\ \theta(t_4) = 0.2\end{array} \xrightarrow{t_2} \begin{array}{l}\{P_1, P_3, P_4\}\\ \theta(t_1) = 0.2\\ \theta(t_3) = 0\\ \theta(t_4) = 0.2\end{array} \xrightarrow{1} \begin{array}{l}\{P_1, P_3, P_4\}\\ \theta(t_1) = 0.2\\ \theta(t_3) = 1\\ \theta(t_4) = 1.2\end{array} \xrightarrow{t_3} \begin{array}{l}\{P_1, P_4\}\\ \theta(t_1) = 0.2\\ \theta(t_4) = 1.2\end{array}$$

Stopwatch Petri nets

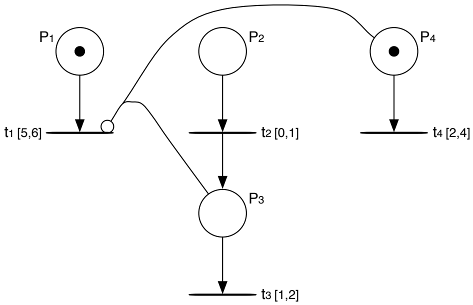


Figure: A discrete-time SwPN

$$\begin{matrix}\{P_1, P_2, P_4\} \\ \theta(t_1) = 0 \\ \theta(t_2) = 0 \\ \theta(t_4) = 0\end{matrix} \xrightarrow{1} \begin{matrix}\{P_1, P_2, P_4\} \\ \theta(t_1) = 1 \\ \theta(t_2) = 1 \\ \theta(t_4) = 1\end{matrix} \xrightarrow{t_2} \begin{matrix}\{P_1, P_3, P_4\} \\ \theta(t_1) = 1 \\ \theta(t_3) = 0 \\ \theta(t_4) = 1\end{matrix} \xrightarrow{1} \begin{matrix}\{P_1, P_3, P_4\} \\ \theta(t_1) = 1 \\ \theta(t_3) = 1 \\ \theta(t_4) = 2\end{matrix} \xrightarrow{t_3} \begin{matrix}\{P_1, P_4\} \\ \theta(t_1) = 1 \\ \theta(t_4) = 2\end{matrix}$$

Stopwatch Petri nets: Definition

Definition

A *Stopwatch Petri Net* (SwPN) is a tuple
$\mathcal{T} = (P, T, {}^\bullet(), ()^\bullet, M_0, a, b, {}^\diamond(.), I)$ where :

- $P = \{p_1, p_2, \ldots, p_m\}$ is a non-empty finite set of *places*;
- $T = \{t_1, t_2, \ldots, t_n\}$ is a non-empty finite set of *transitions* $(T \cap P = \varnothing)$;
- ${}^\bullet() \in (\mathbb{N}^P)^T$ is the *backward incidence function*; $()^\bullet \in (\mathbb{N}^P)^T$ is the *forward incidence function*;
- $M_0 \in \mathbb{N}^P$ is the *initial marking* of the net;
- $a \in (\mathbb{Q}^+)^T$ and $b \in (\mathbb{Q}^+ \cup \{\infty\})^T$ are functions giving for each transition respectively its *earliest* and *latest* firing times $(a \leq b)$;
- ${}^\diamond(.) \in (\mathbb{N}^P)^T$ is the *reset incidence function*;
- $\mathrm{I} = \{(k, t) | k \in \mathbb{N}^P, t \in T\}$ is a finite set of *branch inhibitor hyperarcs*.

Notations

- *enabled*(M) is the set of transitions that are **enabled** by marking M (*i.e.* $t \in enabled(M)$ if $M \geq^{\bullet} t$)
- $\uparrow enabled(M, t')$ is the set of transitions that are **newly enabled** resulting from the fire of t' in marking M
- *inhibited*(M) is the set of transitions that are **inhibited** by marking M (*i.e.* $t \in inhibited(M)$ if $\exists i \leq I(t), {}^{\circ}t^{i} \leq M$)

Semantics

Basic assumptions

- Single-server semantics
- Intermediary semantics
- Strong semantics

Choosing an appropriate time model

- *Dense-time* : continuous evolution of time
- *Discrete-time* : time "jumps" from one integer to the next one at every clock tick

Semantics

Basic assumptions

- Single-server semantics
- Intermediary semantics
- Strong semantics

Choosing an appropriate time model

- *Dense-time* : continuous evolution of time
- *Discrete-time* : time "jumps" from one integer to the next one at every clock tick

Semantics

Basic assumptions

- Single-server semantics
- Intermediary semantics
- Strong semantics

Choosing an appropriate time model

- *Dense-time* : continuous evolution of time
- *Discrete-time* : time "jumps" from one integer to the next one at every clock tick

Semantics

Basic assumptions

- Single-server semantics
- Intermediary semantics
- Strong semantics

Choosing an appropriate time model

- *Dense-time* : continuous evolution of time
- *Discrete-time* : time "jumps" from one integer to the next one at every clock tick

Semantics

Basic assumptions

- Single-server semantics
- Intermediary semantics
- Strong semantics

Choosing an appropriate time model

- *Dense-time* : continuous evolution of time
- *Discrete-time* : time "jumps" from one integer to the next one at every clock tick

Dense-time behavioral semantics [MMR09a]

Timed Transition System $\mathcal{S}_{\mathcal{T}} = (Q, q_0, T, \rightarrow)$ where:

- $Q = \mathbb{N}^P \times (\mathbb{R}^+)^T$: set of all states of the system
- $q_0 = (M_0, \overline{0})$: initial state
- $\rightarrow \in Q \times (T \cup \mathbb{R}) \times Q$ defined by:
  - continuous transition: $(M, \theta) \xrightarrow{\epsilon(d)} (M, \theta')$

$$\text{iff } \forall t_i \in T, \begin{cases} \forall t_i \in enabled(M), \theta'(t_i) = \left\| \begin{array}{l} \theta(t_i) \text{ if } t_i \in enabled(M) \\ \text{ and } t_i \in inhibited(M) \\ \theta(t_i) + d \text{ otherwise,} \end{array} \right. \\ M \geq^\bullet t_i \Rightarrow \theta'(t_i) \leq b(t_i) \end{cases}$$

  - discrete transition: $(M, \theta) \xrightarrow{t_i} (M', \theta')$

$$\text{iff } \begin{cases} t_i \in enabled(M) \text{ and } t_i \notin inhibited(M), \\ M' = M - max({}^\diamond t_i \times M^t, {}^\bullet t_i) + t_i^\bullet, \\ a(t_i) \leq \theta(t_i) \leq b(t_i), \\ \forall t_k, \theta'(t_k) = \left\| \begin{array}{l} 0 \text{ if } t_k \in \uparrow enabled(M, t_i), \\ \theta(t_k) \textit{ otherwise} \end{array} \right. \end{cases}$$

Discrete-time behavioral semantics [MMR09b]

Transition System $\mathcal{S}_{\mathcal{T}} = (Q, q_0, T, \rightarrow)$ where:

- $Q = \mathbb{N}^P \times (\mathbb{R}^+)^T$: set of all states of the system
- $q_0 = (M_0, \overline{0})$: initial state
- $\rightarrow \in Q \times (T \cup \mathbb{R}) \times Q$ defined by:
  - discrete-time transition: $(M, \theta) \xrightarrow{\epsilon(tick)} (M, \theta')$

$$\text{iff } \forall t_i \in T, \begin{cases} \forall t_i \in enabled(M), \theta'(t_i) = \left\| \begin{array}{l} \theta(t_i) \text{ if } t_i \in enabled(M) \\ \text{ and } t_i \in inhibited(M) \\ \theta(t_i) + 1 \text{ otherwise,} \end{array} \right. \\ M \geq^{\bullet} t_i \Rightarrow \theta'(t_i) \leq b(t_i) \end{cases}$$

  - discrete transition: $(M, \theta) \xrightarrow{t_i} (M', \theta')$

$$\text{iff } \begin{cases} t_i \in enabled(M) \text{ and } t_i \notin inhibited(M), \\ M' = M - max(^{\circ}t_i \times M^t, {}^{\bullet}t_i) + t_i^{\bullet}, \\ a(t_i) \leq \theta(t_i) \leq b(t_i), \\ \forall t_k, \theta'(t_k) = \left\| \begin{array}{l} 0 \text{ if } t_k \in\uparrow enabled(M, t_i), \\ \theta(t_k) \text{ otherwise} \end{array} \right. \end{cases}$$

Summary about "logic" arcs

Discrete models

- Read arcs do not add expressivity to Petri nets
- Reset arcs add expressivity to Petri nets
- Logic inhibitor arcs add expressivity to Petri nets

Time models

- Read arcs add expressivity to TPN (but not to SwPN)
- Reset arcs add expressivity to TPN (but not to SwPN)
- Logic inhibitor arcs add expressivity to TPN (but not to SwPN)

Summary about "logic" arcs

Discrete models

- Read arcs do not add expressivity to Petri nets
- Reset arcs add expressivity to Petri nets
- Logic inhibitor arcs add expressivity to Petri nets

Time models

- Read arcs add expressivity to TPN (but not to SwPN)
- Reset arcs add expressivity to TPN (but not to SwPN)
- Logic inhibitor arcs add expressivity to TPN (but not to SwPN)

Expressiveness/Decidability balance

Problem

With a dense-time semantics, the state reachability problem of a SwPN, even bounded, is undecidable [BLRV07].

From the translation of discrete-time TPN into untimed Petri nets, we proved (see next section):

Theorem

With a discrete-time semantics, the state reachability problem of a **bounded** *SwPN is decidable [MMR09b].*

Expressiveness/Decidability balance

Problem

With a dense-time semantics, the state reachability problem of a SwPN, even bounded, is undecidable [BLRV07].

From the translation of discrete-time TPN into untimed Petri nets, we proved (see next section):

Theorem

With a discrete-time semantics, the state reachability problem of a **bounded** *SwPN is decidable [MMR09b].*

Decidability results

	TPN				SwPN			
	Dense-time		Discrete-time		Dense-time		Discrete-time	
	General	Bounded	General	Bounded	General	Bounded	General	Bounded
Boundedness	I	D	I [MMR09b]	D	I	I	I [MMR09b]	D [MMR09b]
k-boundedness	I	D	D	D	I	I	D [MMR09b]	D [MMR09b]
Liveness	I	D	I [MMR09b]	D	I	I	I [MMR09b]	D [MMR09b]
Marking reachability	I	D	I [MMR09b]	D	I	I	I [MMR09b]	D [MMR09b]
State reachability	I	D	I [MMR09b]	D	I	I	I [MMR09b]	D [MMR09b]

Table: Decidability results for TPN and SwPN

Abstractions for discrete-time models: general plan

Problem

The state space of a TPN/SwPN is infinite (in general).

Computation of the state space of dense-time SwPN

Abstractions techniques (semi-algorithms) ⇒ Group states in equivalence classes

Computation of the state space of discrete-time SwPN

- **Enumerate** the set of states
- Adapt the dense-time **symbolic** methods to discrete-time

Abstractions for discrete-time models: general plan

Problem

The state space of a TPN/SwPN is infinite (in general).

Computation of the state space of dense-time SwPN

Abstractions techniques (semi-algorithms) ⇒ Group states in equivalence classes

Computation of the state space of discrete-time SwPN

- **Enumerate** the set of states
- Adapt the dense-time **symbolic** methods to discrete-time

Abstractions for discrete-time models: general plan

Problem

The state space of a TPN/SwPN is infinite (in general).

Computation of the state space of dense-time SwPN

Abstractions techniques (semi-algorithms) $\Rightarrow$ Group states in equivalence classes

Computation of the state space of discrete-time SwPN

- **Enumerate** the set of states
- Adapt the dense-time **symbolic** methods to discrete-time

From discrete-time to untimed model: the **fictitious clock** model

Principle

- Simulate time elapsing with a dedicated *tick* transition.
- *Clock* classically associated to transitions: viewed as a place whose marking is incremented by one with every *tick* transition.
- Discrete-time SwPNs can then be described as *parallel composition* of PNs with reset arcs and inhibitor hyperarcs (synchronized product of PNs)
- Then follows:
  - Expressivity and decidability results;
  - Practical way to enumerate the state space of discrete-time *bounded* SwPNs

From discrete-time to untimed model: the **fictitious clock** model

Principle

- Simulate time elapsing with a dedicated *tick* transition.
- *Clock* classically associated to transitions: viewed as a place whose marking is incremented by one with every *tick* transition.
- Discrete-time SwPNs can then be described as *parallel composition* of PNs with reset arcs and inhibitor hyperarcs (synchronized product of PNs)
- Then follows:
  - Expressivity and decidability results;
  - Practical way to enumerate the state space of discrete-time *bounded* SwPNs

From discrete-time to untimed model: the **fictitious clock** model

Principle

- Simulate time elapsing with a dedicated *tick* transition.
- *Clock* classically associated to transitions: viewed as a place whose marking is incremented by one with every *tick* transition.
- Discrete-time SwPNs can then be described as *parallel composition* of PNs with reset arcs and inhibitor hyperarcs (synchronized product of PNs)
- Then follows:
  - Expressivity and decidability results;
  - Practical way to enumerate the state space of discrete-time *bounded* SwPNs

From discrete-time to untimed model: the **fictitious clock** model

Principle

- Simulate time elapsing with a dedicated *tick* transition.
- *Clock* classically associated to transitions: viewed as a place whose marking is incremented by one with every *tick* transition.
- Discrete-time SwPNs can then be described as *parallel composition* of PNs with reset arcs and inhibitor hyperarcs (synchronized product of PNs)
- Then follows:
  - Expressivity and decidability results;
  - Practical way to enumerate the state space of discrete-time *bounded* SwPNs

From discrete-time TPNs to untimed PNs [MMR06]

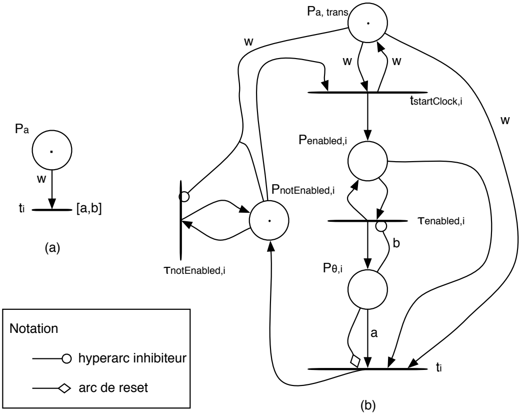


Figure: Translating a discrete-time TPN into an untimed PN with reset arcs and inhibitor hyperarcs

From discrete-time SwPN to untimed PNs [MMR06]

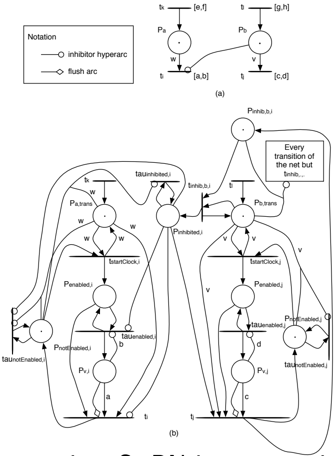


Figure: Translating a discrete-time SwPN into an untimed PN with reset arcs and inhibitor hyperarcs

State space of discrete-time models: enumerative approach

Computation of the state space of discrete-time models

Translation of discrete-time Petri nets towards models that can be analyzed thanks to **efficient data-structures**:

- **BDD**-inspired datastructures for untimed Petri nets : Markg tool [MDR02] (Lecture 1)
- **Symbolic representation** thanks to counter automata : FAST [BFLP03] and LEVER [VV06] tools

State space of discrete-time models: enumerative approach

Computation of the state space of discrete-time models

Translation of discrete-time Petri nets towards models that can be analyzed thanks to **efficient data-structures**:

- **BDD**-inspired datastructures for untimed Petri nets : Markg tool [MDR02] (Lecture 1)
- **Symbolic representation** thanks to counter automata : FAST [BFLP03] and LEVER [VV06] tools

State space of discrete-time models: enumerative approach

Computation of the state space of discrete-time models

Translation of discrete-time Petri nets towards models that can be analyzed thanks to **efficient data-structures**:

- **BDD**-inspired datastructures for untimed Petri nets : Markg tool [MDR02] (Lecture 1)
- **Symbolic representation** thanks to counter automata : FAST [BFLP03] and LEVER [VV06] tools

Experimental results [MMR06]

Net	Discrete-time - Markg		Discrete-time - FAST		Discrete-time - LEVER	
	Time	Memory	Time	Memory	Time	Memory
Ex 12	336.15 s	55 040 KB	0.22 s	3 000 KB	0.07 s	1 320 KB
Ex 13	NA	NA	62.04 s	20 196 KB	0.26 s	3 596 KB
Ex 14	612.85 s	55 116 KB	0.65 s	3 404 KB	0.05 s	1 320 KB
Ex 15	NA	NA	NA	NA	NA	NA
Ex 16 (avec ∞)	0.10 s	1 320 KB	0.33 s	2 524KB	NA	NA
Ex 17 (avec ∞)	1 148.18 s	139 800 KB	0.55 s	3 352KB	NA	NA

Table: PENTIUM ; 2 GHz; 2Go RAM

Experimental results [MMR06]

Net	Dense-time - Roméo		Discrete-time - Markg		Discrete-time - FAST		Discrete-time - Lever	
	Time	Memory	Temps	Memory	Time	Memory	Time	Memory
Ex 11	NA	NA	1.07s	95 796 KB	0.72 s	3 502 KB	0.43 s	1 320 KB
Ex 12	0.04 s	1 320 KB	336.15 s	55 040 KB	0.22 s	3 000 KB	0.07 s	1 320 KB
Ex 13	0.11 s	1 320 KB	NA	NA	62.04 s	20 196 KB	0.26 s	3 596 KB
Ex 14	0.10 s	1 320 KB	612.85 s	55 116 KB	0.65 s	3 404 KB	0.05 s	1 320 KB
Ex 15	0.11 s	1 320 KB	NA	NA	NA	NA	NA	NA
Ex 16 (avec ∞)	0.10 s	1 320 KB	0.10 s	1 320 KB	0.33 s	2 524 KB	NA	NA
Ex 17 (avec ∞)	0.10 s	1 320 KB	1 148.18 s	139 800 KB	0.55 s	3 352 KB	NA	NA

Table: Pentium ; 2 GHz; 2Go RAM

About the experimental results [MMR06]

Discussion

- Computationally speaking:
  - Large firing intervals $[1, 1000] \Rightarrow$ **Combinatorial explosion** of the number of states!
  - Dense-time may be more efficient than discrete-time: then, what is the advantage of discrete-time?
- "Theoretically and practically" speaking:
  - On some models, computation terminates **only for discrete-time semantics**.
  - Which may result from the difference **in terms of decidability** between dense-time and discrete-time!

About the experimental results [MMR06]

Discussion

- Computationally speaking:
  - Large firing intervals $[1, 1000] \Rightarrow$ **Combinatorial explosion** of the number of states!
  - Dense-time may be more efficient than discrete-time: then, what is the advantage of discrete-time?
- "Theoretically and practically" speaking:
  - On some models, computation terminates **only for discrete-time semantics**.
  - Which may result from the difference **in terms of decidability** between dense-time and discrete-time!

About the experimental results [MMR06]

Discussion

- Computationally speaking:
  - Large firing intervals $[1, 1000] \Rightarrow$ **Combinatorial explosion** of the number of states!
  - Dense-time may be more efficient than discrete-time: then, what is the advantage of discrete-time?
- "Theoretically and practically" speaking:
  - On some models, computation terminates **only for discrete-time semantics**.
  - Which may result from the difference **in terms of decidability** between dense-time and discrete-time!

How discrete time and dense-time discretization are connected?

Question

Can we consider the state space of discrete-time nets as the **discretization** of the state space of the corresponding dense-time model?

Problems

- Identify the cases when this assumption is valid
- Design an algorithm to symbolically compute the state space
- Model-check TCTL properties of SwPN using this algorithm

How discrete time and dense-time discretization are connected?

Question

Can we consider the state space of discrete-time nets as the **discretization** of the state space of the corresponding dense-time model?

Problems

- Identify the cases when this assumption is valid
- Design an algorithm to symbolically compute the state space
- Model-check TCTL properties of SwPN using this algorithm

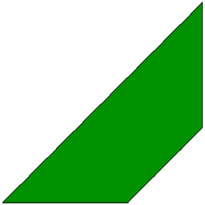
Abstraction of the state space of dense-time SwPN

Problem

Group states in d'equivalence classes (abstraction)

$\Rightarrow$ Use state class graph [BM83]

$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \right.$$

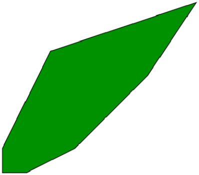


$$\left. \right\}$$

TPN and some sub-classes of SwPN : the firing domain can be encoded by a Difference Bound Matrix (DBM) $[d_{ij}]_{i,j \in [0..n]}$:

$$\begin{cases} -d_{0i} \leq \theta_i - 0 \leq d_{i0}, \\ \theta_i - \theta_j \leq d_{ij} \end{cases}$$

$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \right.$$

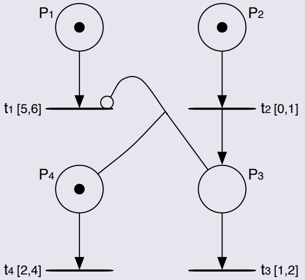


$$\left. \right\}$$

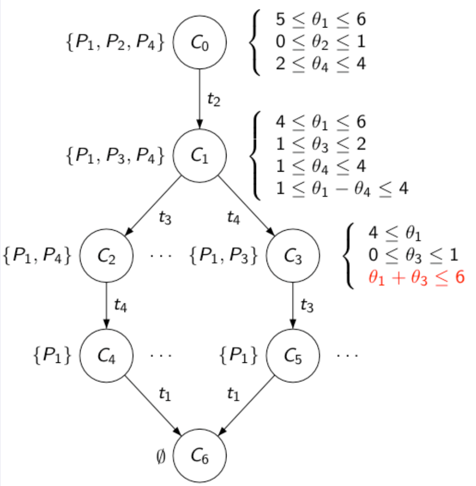
SwPN : polyhedra $A\bar{\theta} \leq B$

State class graph

SwPN

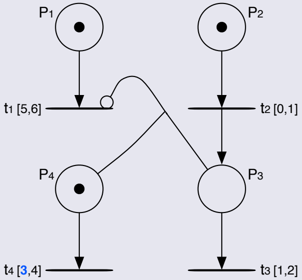


State class graph

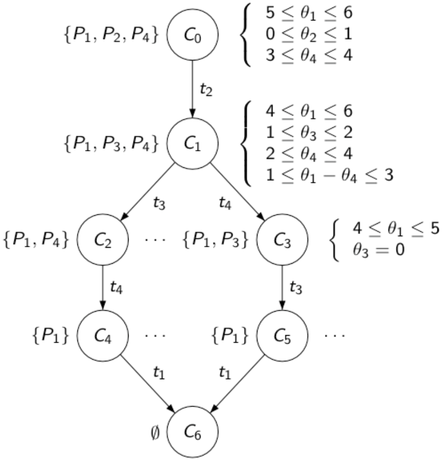


Non-DBM vs DBM polyhedral constraints

SwPN



State class graph



State classes for discrete-time SwPN

Objective

Extend the principle of state classes to discrete-time

State classes for discrete-time SwPN

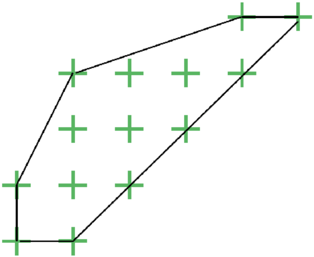
$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \right.$$

$$\left. \right\}$$

$\Rightarrow$ Define symbolic state classes

Symbolic state classes for discrete-time SwPN

$$C = \left\{ \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \right.$$



$$\left. \right\}$$

Problems caused by the discretization of symbolic classes

Question

Does the successor of a symbolic discrete-time state class $(M, Poly)$ is equal to the discretization of the dense-time successor of this class?

Problems caused by the discretization of symbolic classes

Question

Does the successor of a symbolic discrete-time state class $(M, Poly)$ is equal to the discretization of the dense-time successor of this class?

Answer

- Yes in the case of TPN [Pop91, PZ96].
- And for SwPN ?

Problems caused by the discretization of symbolic classes

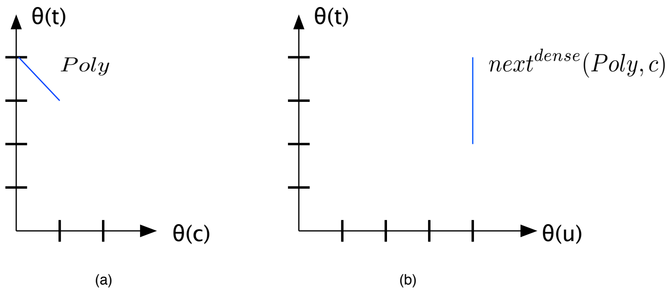
Question

Does the successor of a symbolic discrete-time state class $(M, Poly)$ is equal to the discretization of the dense-time successor of this class?

Answer

- Yes in the case of TPN [Pop91, PZ96].
- And for SwPN ?

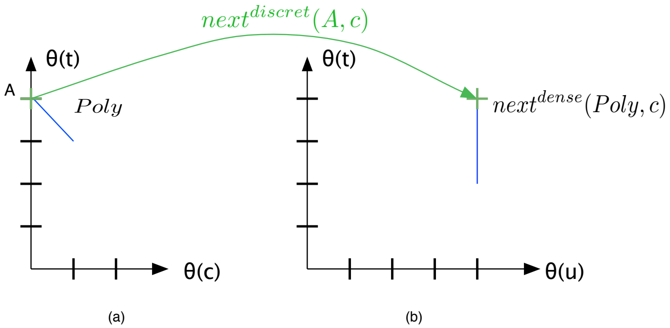
Problems caused by the discretization of symbolic classes



Question

Do all points resulting from the discretization of $next^{dense}(Poly, c)$ have a predecessor whose all coordinates are integers?
$next^{discret}(\mathcal{D}isc(Poly)) = \mathcal{D}isc(next^{dense}(Poly))$?

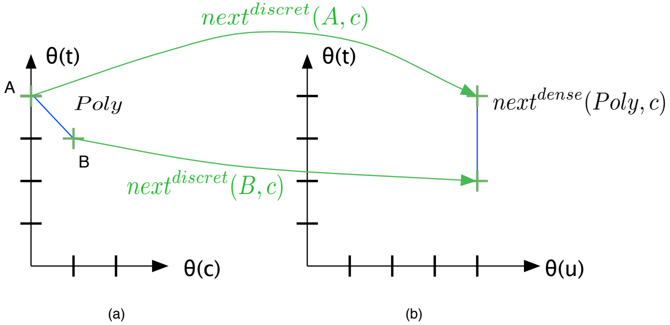
Problems caused by the discretization of symbolic classes



Question

Do all points resulting from the discretization of $next^{dense}(Poly, c)$ have a predecessor whose all coordinates are integers?
$next^{discret}(\mathcal{D}isc(Poly)) = \mathcal{D}isc(next^{dense}(Poly))$?

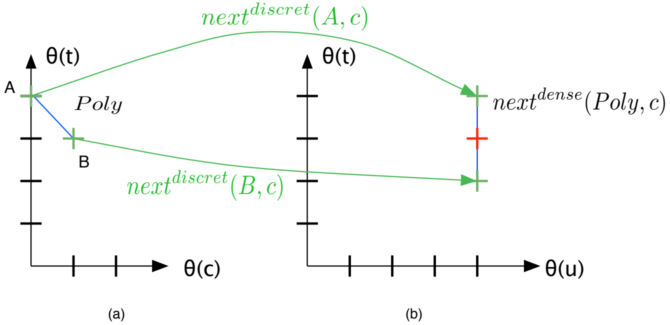
Problems caused by the discretization of symbolic classes



Question

Do all points resulting from the discretization of $next^{dense}(Poly, c)$ have a predecessor whose all coordinates are integers?
$next^{discret}(\mathcal{D}isc(Poly)) = \mathcal{D}isc(next^{dense}(Poly))$?

Problems caused by the discretization of symbolic classes



Question

Do all points resulting from the discretization of $next^{dense}(Poly, c)$ have a predecessor whose all coordinates are integers?
$next^{discret}(\mathcal{D}isc(Poly)) = \mathcal{D}isc(next^{dense}(Poly))$?

Problems caused by the discretization of symbolic classes

Question

Do all points resulting from the discretization of $next^{dense}(Poly, c)$ have a predecessor whose all coordinates are integers?
$next^{discret}(\mathcal{D}isc(Poly)) = \mathcal{D}isc(next^{dense}(Poly))$?

Our answer

- Yes in the case of DBM
- Identification of the shape of the first non-DBM polyhedra appearing during the computation: $x + y \quad (-z) \sim c$
- No as soon as such a non-DBM polyhedra appears!

Problems caused by the discretization of symbolic classes

Question

Do all points resulting from the discretization of $next^{dense}(Poly, c)$ have a predecessor whose all coordinates are integers?
$next^{discret}(\mathcal{D}isc(Poly)) = \mathcal{D}isc(next^{dense}(Poly))$?

Our answer

- Yes in the case of DBM
- Identification of the shape of the first non-DBM polyhedra appearing during the computation: $x + y \quad (-z) \sim c$
- No as soon as such a non-DBM polyhedra appears!

Problems caused by the discretization of symbolic classes

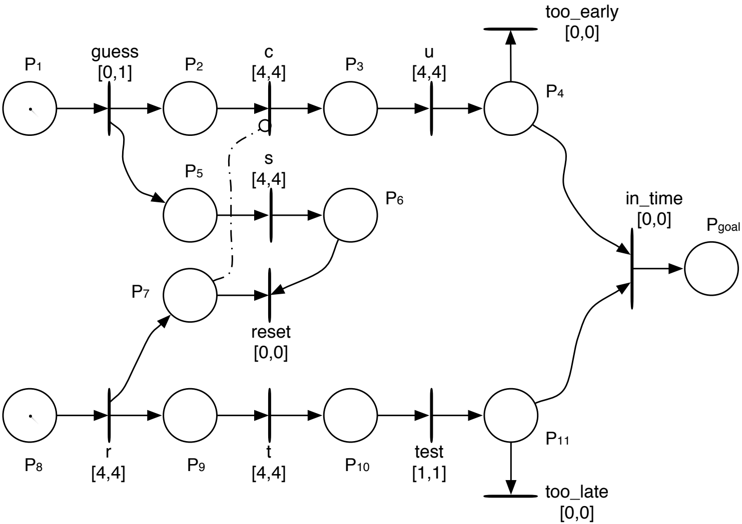
Question

Do all points resulting from the discretization of $next^{dense}(Poly, c)$ have a predecessor whose all coordinates are integers?
$next^{discret}(\mathcal{D}isc(Poly)) = \mathcal{D}isc(next^{dense}(Poly))$?

Our answer

- Yes in the case of DBM
- Identification of the shape of the first non-DBM polyhedra appearing during the computation: $x + y \ \ (-z) \sim c$
- No as soon as such a non-DBM polyhedra appears!

SwPN counter-example: discrete-time ≠ discretization of dense-time



Theorem

The state space of a discrete-time TPN and the discretization of the corresponding dense-time net are equal.

⇒ What for SwPN?

Theorem

The state space of a discrete-time SwPN and the discretization of the corresponding dense-time net are not equal [MMR09b].

Remark about Timed Automata

A similar result exists:

Theorem

For every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$) [BS90].

⇒ Finding a relevant granularity is hard.

Theorem

The state space of a discrete-time TPN and the discretization of the corresponding dense-time net are equal.

⇒ What for SwPN?

Theorem

The state space of a discrete-time SwPN and the discretization of the corresponding dense-time net are not equal [MMR09b].

Remark about Timed Automata

A similar result exists:

Theorem

For every $k \geq 1$, there exists a digital circuit such that the reachability set of states in dense-time is strictly larger than the one in discrete time (with granularity $\frac{1}{k}$) [BS90].

⇒ Finding a relevant granularity is hard.

Issue

How can we **symbolically** compute the state space of discrete-time SwPN?

Theorem

As long as *the state class graph of SwPN does not involve non-DBM polyhedra :*

- *The discretization of the state space of dense-time net leads to* **state all belonging** *to the state space of the discrete-time net;*
- *Set of the untimed traces of the dense-time net* $\subseteq$ *Set of the untimed traces of the discrete-time net*

Issue

How can we **symbolically** compute the state space of discrete-time SwPN?

Theorem

As long as *the state class graph of SwPN does not involve non-DBM polyhedra :*

- *The discretization of the state space of dense-time net leads to* **state all belonging** *to the state space of the discrete-time net;*
- *Set of the untimed traces of the dense-time net* $\subseteq$ *Set of the untimed traces of the discrete-time net*

Symbolic computation of the state space of discrete time nets

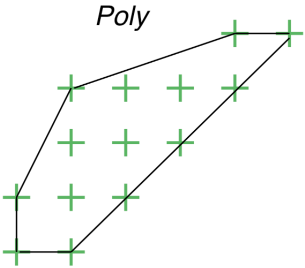
Principles

- Compute the state space of the corresponding dense-time network **as long as a non-DBM polyhedron does not appear** ;
- All non-DBM polyhedron *Poly* is **decomposed into a union of DBM**, $DBM\_split(Poly)$, s.t. $\mathcal{D}isc(Poly) = \mathcal{D}isc(DBM\_split(Poly))$

Symbolic computation of the state space of discrete time nets

Principles

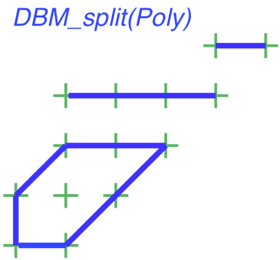
- Compute the state space of the corresponding dense-time network **as long as a non-DBM polyhedron does not appear** ;
- All non-DBM polyhedron *Poly* is **decomposed into a union of DBM**, $DBM\_split(Poly)$, s.t. $\mathcal{D}isc(Poly) = \mathcal{D}isc(DBM\_split(Poly))$



Symbolic computation of the state space of discrete time nets

Principles

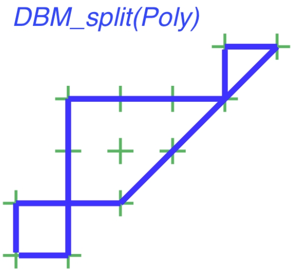
- Compute the state space of the corresponding dense-time network **as long as a non-DBM polyhedron does not appear** ;
- All non-DBM polyhedron *Poly* is **decomposed into a union of DBM**, $DBM\_split(Poly)$, s.t. $\mathcal{D}isc(Poly) = \mathcal{D}isc(DBM\_split(Poly))$



Symbolic computation of the state space of discrete time nets

Principles

- Compute the state space of the corresponding dense-time network **as long as a non-DBM polyhedron does not appear** ;
- All non-DBM polyhedron *Poly* is **decomposed into a union of DBM**, $DBM\_split(Poly)$, s.t. $\mathcal{D}isc(Poly) = \mathcal{D}isc(DBM\_split(Poly))$



Theorem

For discrete-time SwPN, the algorithm to symbolically compute their state space is correct w.r.t. marking and state reachability and language.

Theorem

The termination of the algorithm is ensured for bounded discrete-time nets.

Benchmark: symbolic vs enumerative

Net	Symbolic algo. - ROMÉO		Enumerative algo. - Markg	
	Time	Memory	Time	Memory
Ex 1	0.12 s	1 320 KB	1.03 s	96 032 KB
Ex 3	34.42 s	1 320 KB	2.95 s	111 700 KB
Ex 4	45.12 s	20 012 KB	NA	NA
Ex 11	0.52 s	2 940 KB	1.07 s	95 796 KB
Ex 15	0.15 s	1 320 KB	1 148.18 s	139 800 KB

Table: PENTIUM ; 2 GHz; 2Gb RAM

⇒ Model-checking of timed **quantitative** properties

Benchmark: symbolic vs enumerative

Net	Symbolic algo. - ROMÉO		Enumerative algo. - Markg	
	Time	Memory	Time	Memory
Ex 1	0.12 s	1 320 KB	1.03 s	96 032 KB
Ex 3	34.42 s	1 320 KB	2.95 s	111 700 KB
Ex 4	45.12 s	20 012 KB	NA	NA
Ex 11	0.52 s	2 940 KB	1.07 s	95 796 KB
Ex 15	0.15 s	1 320 KB	1 148.18 s	139 800 KB

Table: PENTIUM ; 2 GHz; 2Gb RAM

$\Rightarrow$ Model-checking of timed **quantitative** properties

Overview

Model-checking formal properties - Reminder

Qualitative properties

- LTL
- CTL
- CTL*

Quantitative timing properties

A TCTL formula:

$$\varphi := ap \mid \neg ap \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid A\varphi U_I \varphi \mid E\varphi U_I \varphi$$

with:

- *ap* an atomic assertion
- I an interval from $\mathbb{R}^+$ with integer bounds s.t. $[n, m]$, $[n, m[$, $]n, m]$, $]n, m[$, or $[m, \infty[$, $n, m \in \mathbb{N}$

Model-checking of TCTL properties - Reminder

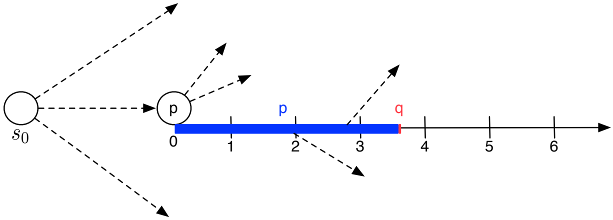


Figure: $s_0 \models E(\mathrm{p}U_{[2;4]}\mathrm{q})$

Some additional TCTL example - Reminder

Bounded liveness/response [DT98]

- "Whenever a property p becomes true, q must be true within n seconds" ($n \in \mathbb{N}$)
- $AG(p \Rightarrow AF_{[0,n]}q)$
- Denoted $p \rightarrow_{[0,n]} q$ in most model-checkers

Model-checking formal properties on TPN and SwPN: TPN-TCTL

Quantitative timing properties

A TPN-TCTL formula:

$$\varphi := \mathrm{GMEC} \mid \neg\varphi \mid \varphi \Rightarrow \psi \mid A\varphi U_I \varphi \mid E\varphi U_I \varphi$$

with:

- GMEC (*Generalized Mutual Exclusion Constraints*) is a **conjunction/disjunction of linear constraints** that limit the weighted sum of tokens in a subset of places (e.g. $(M(P_1) + 4M(P_2) \geq 3) \wedge (M(P_2) + 3M(P_3) \leq 10)$;
- I an interval from $\mathbb{R}^+$ with **integer bounds** s.t. $[n, m]$, $[n, m[$, $]n, m]$, $]n, m[$, or $[m, \infty[$, $n, m \in \mathbb{N}$

A taste of TCTL model-checking thanks to observers

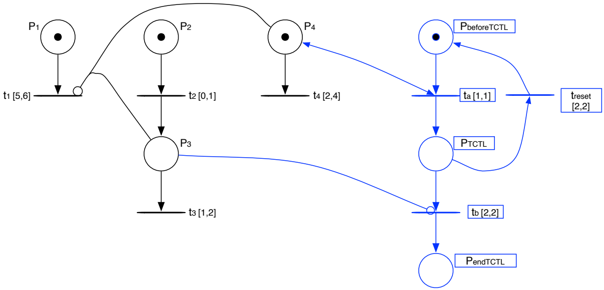


Figure: Addition of an observer to the initial net to model-check
$E((M(P_4) = 1)U_{[1;2]}(M_(P_3) = 0))$

A taste of TCTL model-checking thanks to observers

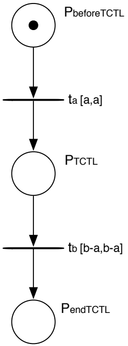


Figure: Generic observer to model-check $\mathrm{E}\varphi U_{[a;b]}\psi$

TPN-TCTL model-checking on time and stopwatch Petri nets

How to verify quantitative time properties

- **Translate TPN into TA** and use available TA model-checking tools (e.g. UPPAAL)
- Build an **observer** (*i.e.* a supervisory net) corresponding to the expected property and model-check the global network
- **On-the-fly** computation of the state space, then checking of the property

Parametric TPN-TCTL model-checking on time and stopwatch Petri nets [TLR09]

Parametric SwPN

A pSwPN is a n-tuple $\mathcal{N} = (P, T, Par, ^{\bullet}(.), (.)^{\bullet}, ^{\circ}(.), a, b, M_0, D_p)$, where:

- $Par = \{\lambda_1, \lambda_2, \ldots, \lambda_l\}$ is a finite set of **parameters** (we denote $\Gamma(Par)$ the set of linear expressions over Par);
- $a : T \to \Gamma(Par)$ is the function that gives the *earliest firing time* of a transition, expressed as a **linear expression over the set of parameters**;
- $b : T \to \Gamma(Par) \cup \{\infty\}$ is the function that gives the *latest firing time* of a transition, that is either a **linear expression over the set of parameters** or equal to ∞;
- $D_p \subseteq \mathbb{N}^{Par}$ is the *domain of the parameters*;

Parametric TCTL for SwPN [TLR09]

Quantitative timing properties

A PTCTL formula:

$$E\varphi\ U_I\psi \mid A\varphi\ U_I\psi \mid EF_I\varphi \mid AF_I\varphi \mid EG_I\varphi \mid AG_I\varphi \mid \varphi \leadsto_{I_r} \psi$$

with:

- φ et ψ are **GMEC** (*Generalized Mutual Exclusion Constraints*);
- I et I_r are **parametric intervals with integer bounds** s.t. $[n, m]$, $[n, m[$, $]n, m]$, $]n, m[$, or $[m, \infty[$, $n, m \in \mathbb{N}$, with the restriction that $I_r = [0, m]$ or $I_r = [0, \infty[$

Parametric TCTL for SwPN: Application [TLR09]

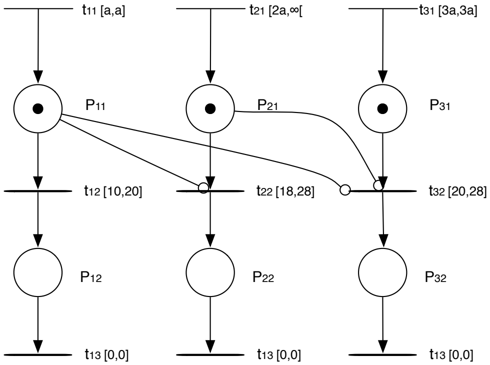


Figure: Example of a parametric SwPN modeling a system of three tasks, two being periodic, one sporadic [BFSV04]

Parametric TCTL for SwPN: Application [TLR09]

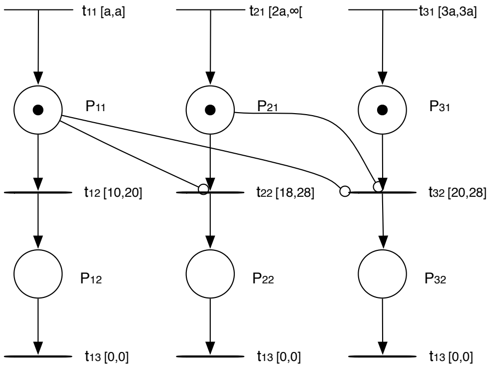


Figure: Verification of the *schedulability* of the tasks (do a task always terminate before an other instance of the same task is created?): $\forall P_i, AG_{[0,\infty[}(M(P_i) \leq 1)$

Parametric TCTL for SwPN: Application [TLR09]

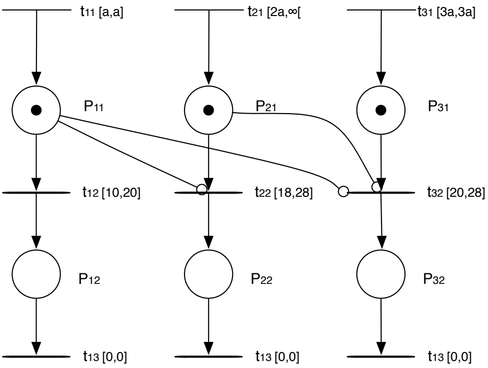


Figure: Verification of the *schedulability* of the tasks (do a task always terminate before an other instance of the same task is created?):
$\forall P_i, AG_{[0,\infty[}(M(P_i) \leq 1), whichleadsto\{a > 48\}$

Parametric TCTL for SwPN: Application [TLR09]

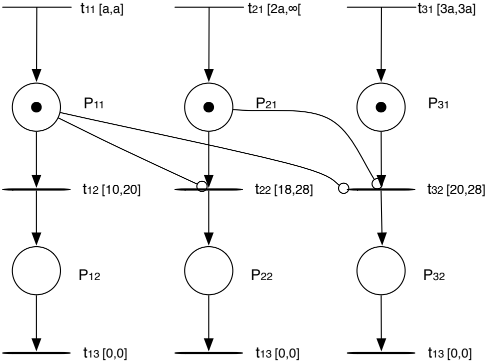


Figure: With the new constraint $\{a > 48\}$, what is the worst case response time of task 3? $M(P_{31}) > 0 \rightsquigarrow_{[0,b]} M(P_{32}) > 0$

Parametric TCTL for SwPN: Application [TLR09]

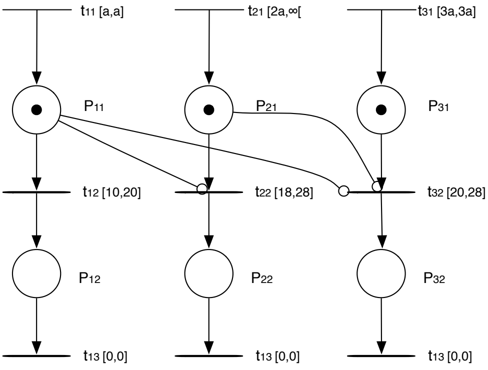


Figure: With the new constraint $\{a > 48\}$, what is the worst case response time of task 3? $M(P_{31}) > 0 \rightsquigarrow_{[0,b]} M(P_{32}) > 0, whichleadsto\{b = 96\}$

Overview

Short insight on the meaning of Petri net components - Reminder

Intuitive meaning

- Marking of a place: **presence/absence** or **quantity** of a component
- Arc : **precedence** or **succession**
- Transition : **event** andor **transformation**
- Weight : necessary **quantity**, consumed and/or produced

Petri nets for modeling biochemical networks - Reminder

Principle of qualitative modeling

- Places : reactants, products, enzymes
- Transitions : reactions, catalysis
- Weight on the arcs: stoichiometry

Application to the modeling of biochemical reactions - Reminder

$$2NAD^+ + 2H_2O \rightarrow 2NADH + 2H^+ + O_2$$

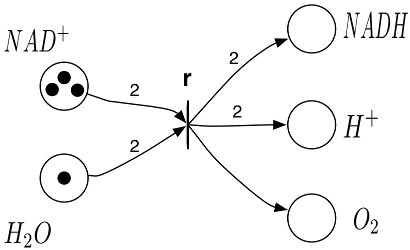


Figure: An example of a very simple translation from a biochemical reaction

Additional modeling features

Critical issues

- How to **test** the number of tokens in a place (e.g. concentration level of a protein) without decrementing it? $\Rightarrow$ Read arcs
- How to **reset the number of tokens** in a place (e.g. reset of a system) whatever was its previous concentration? $\Rightarrow$ Reset arcs
- How to **integrate quantitative information about speed of reactions** that can start, stop or resume, depending on thresholds (e.g. a concentration of a protein)? $\Rightarrow$ Inhibitor (hyper)arcs that allow to model stopwatches

Additional modeling features

TCTL model-checking

- Allows to validate a model
- Can help biologists to **infer discrete parameters** (e.g. tendency of evolution)
- Can help biologists to **infer timing quantitative parameters** (e.g. speed of reactions)

Overview

1. Models: from time Petri nets to stopwatch Petri nets
   - From time Petri nets to stopwatch Petri nets
   - Dense-time vs discrete-time
   - State space abstraction dedicated to discrete-time
2. Formalizing specification through timed modal logics
   - TCTL model-checking
   - Introduction to TCTL parametric model-checking
3. Application to biological systems
4. Model-checking tools

Model-checking tools

Verification of time extensions of Petri nets: Roméo [GLMR05]

- Provides a GUI and command-line tools for editing and verifying not only time Petri nets, but also **stopwatch Petri nets**
- Tool box for *validation* (simulation) and **verification** (model-checking)
- Developed at IRCCyN (Nantes, France)
- Tackles reachability and safety verification, and parametric analysis
- `http://romeo.rts-software.org/`

Model-checking tools

Verification of timed automata: UPPAAL [LPY97]

- Tool box for *validation* (simulation) and **verification** (model-checking)
- Developed by Uppsala University (Sweden) and Aalborg University (Denmark)
- Provides a GUI and command-line tools for editing and verifying TA
- Continuously **kept up-to-date** with new extensions (e.g. games) and improvements
- `http://www.uppaal.org`

Model-checking tools

Verification of linear hybrid automata: HyTech [HHWT95]

- Addresses **linear** hybrid automata
- Tackles reachability and safety verification, and parametric analysis
- http://embedded.eecs.berkeley.edu/research/hytech/
- Not maintained anymore ⇒ the tool has been overtaken by **PHAVer**: http://www-verimag.imag.fr/~frehse/phaver_web/ [Fre05]

Petri nets in the modeling of biological systems

Summary

- Intuitive modeling, easy to adapt to biological systems
- Progressive introduction of time delays and **parametric inference**
- But **combinatorial explosion** of state space, despite symbolic approaches

Further work

- Analysis of **multi-scale networks**
- Application to the circadian clock (on-going project with University of Nice I3S BioInfo and Circadian System Biology teams)
- Take advantage of the potential of **chronometric** information modeling
- New approaches to study **large networks** $\rightarrow$ **Lecture 4: Approaches inspired by pi-calculus and static analysis**

- Sébastien Bardin, Alain Finkel, Jérôme Leroux, and Laure Petrucci.
  FAST: Fast Acceleration of Symbolic Transition systems.
  In Warren A. Hunt, Jr and Fabio Somenzi, editors, Proceedings of the 15th International Conference on Computer Aided Verification (CAV'03), volume 2725 of Lecture Notes in Computer Science, pages 118–121, Boulder, Colorado, USA, July 2003. Springer.

- G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario.
  Time state space analysis of real-time preemptive systems.
  IEEE transactions on software engineering, 30(2):97–111, February 2004.

- Bernard Berthomieu, Didier Lime, Olivier (H.) Roux, and Francois Vernadat.
  Reachability problems and abstract state spaces for time Petri nets with stopwatches.
  Journal of Discrete Event Dynamic Systems (DEDS), 17(2), 2007. To appear.

- B. Berthomieu and M. Menasche.

An enumerative approach for analyzing time Petri nets.
IFIP Congress Series, 9:41–46, 1983.

Patricia Bouyer, Nicolas Markey, and Ocan Sankur.
Robust reachability in timed automata: A game-based approach.
In ICALP (2), pages 128–140, 2012.

J. A. Brzozowski and C-J. Seger.
Advances in Asynchronous Circuit Theory – Part I: Gate and Unbounded Inertial Delay Models.
Bulletin of the European Association of Theoretical Computer Science, October 1990.

Conrado Daws and Stavros Tripakis.
Model checking of real-time reachability properties using abstractions.
In Proceedings of the 4th International Conference on Tools and Algorithms for Construction and Analysis of Systems, TACAS '98, pages 313–329, London, UK, UK, 1998. Springer-Verlag.

Goran Frehse.

Phaver: Algorithmic verification of hybrid systems past hytech.
In HSCC, pages 258–273, 2005.

Guillaume Gardey, Didier Lime, Morgan Magnin, and Olivier (H.) Roux.
Romeo: a tool for analyzing time petri nets.
In Proc. 17th International Conference on Computer Aided Verfication (CAV'05), volume 3576 of Lecture Notes in Computer Science, pages 418–423, Edinburgh, Scotland, UK, July 2005. Springer–Verlag.

Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi.
Hytech: The next generation.
In IEEE Real-Time Systems Symposium, pages 56–65, 1995.

Kim G. Larsen, Paul Pettersson, and Wang Yi.
Uppaal in a nutshell.
Int. Journal on Software Tools for Technology Transfer, 1:134–152, 1997.

Pierre Molinaro, David Delfieu, and Olivier (H.) Roux.

Improving the calculus of the marking graph of Petri net with bdd like structure.
In 2002 IEEE international conference on systems, man and cybernetics (SMC 02), Hammamet, Tunisia, October 2002.

M. Magnin, P. Molinaro, and O.H. Roux.
Decidability, expressivity and state-space computation of stopwatch petri nets with discrete-time semantics.
In Discrete Event Systems, 2006 8th International Workshop on, pages 33–38, 2006.

Morgan Magnin, Pierre Molinaro, and Olivier H. Roux.
Expressiveness of Petri nets with stopwatches. Dense-time part.
Fundamenta Informaticae, 97(1-2):111–138, 2009.

Morgan Magnin, Pierre Molinaro, and Olivier H. Roux.
Expressiveness of Petri nets with stopwatches. Discrete-time part.
Fundamenta Informaticae, 97(1-2):139–176, 2009.

Louchka Popova.

On time petri nets.
Journal Inform. Process. Cybern., EIK (formerly: Elektron. Inform. verarb. Kybern.), 27(4):227–244, 1991.

L. Popova-Zeugmann.
Essential states in time petri nets, 1996.

Olivier H. Roux and Anne-Marie Déplanche.
Extension des réseaux de Petri t-temporels pour la modélisation de l'ordonnancement de tâches temps-réel.
In 3e congrès Modélisation des Systèmes Réactifs (MSR'2001), pages 327–342, Toulouse, France, 2001. Hermes Science.

Olivier (H.) Roux and Didier Lime.
Time Petri nets with inhibitor hyperarcs. Formal semantics and state space computation.
In The 25th International Conference on Application and Theory of Petri Nets, (ICATPN'04), volume 3099 of Lecture Notes in Computer Science, pages 371–390, Bologna, Italy, June 2004. Springer.

- Louis-Marie Traonouez, Didier Lime, and Olivier (H.) Roux.
Parametric model-checking of stopwatch petri nets.
Journal of Universal Computer Science, 15(17):3273–3304, December 2009.

- Abhay Vardhan and Mahesh Viswanathan.
Lever: A tool for learning based verification.
In CAV, pages 471–474, 2006.