Identification of place/transition nets

Maria Paola Cabasino

DIEE, University of Cagliari

PhD Course 27 Mar 07
Outline

- Background of Petri nets
- Motivation
- Identification a *free-labeled* Petri net
- Extensions of the procedure
- Numerical Simulations
- Identification a *labeled* Petri net
- Complexity of the identification procedure
- Coverability graph
A **Place/Transition net** (P/T net) is a structure $N = (P, T, Pre, Post)$ where:

- $P$ is a set of places represented by circles, $|P| = m$;
- $T$ is a set of transitions represented by bars, $|T| = n$;
- $Pre : P \times T \rightarrow \mathbb{N}$ is the pre-incidence function that specifies the arcs directed from places to transitions;
- $Post : P \times T \rightarrow \mathbb{N}$ is the post-incidence function that specifies the arcs directed from transitions to places.
What is a Petri net?

Example of P/T net

\[ P = \{p_1, p_2, p_3, p_4\}, \quad T = \{t_1, t_2, t_3, t_4, t_5\} \]

\[ \text{Pre} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \]

\[ \text{Post} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \quad \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \]
A marking is a vector $M : P \rightarrow \mathbb{N}$ that associates to each place a non-negative number of tokens, represented by black dots. The initial marking is called $M_0$.

$$M_0 = [M_0(p_1) \ M_0(p_2) \ M_0(p_3) \ M_0(p_4)]^T = [1 \ 0 \ 0 \ 0]^T$$

A net system $\langle N, M_0 \rangle$ is a net $N$ with an initial marking $M_0$.
A **marking** is a vector $M : P \rightarrow \mathbb{N}$ that associates to each place a non-negative number of tokens, represented by black dots. The initial marking is called $M_0$.

$$M_0 = \begin{bmatrix} M_0(p_1) & M_0(p_2) & M_0(p_3) & M_0(p_4) \end{bmatrix}^T = [1 \ 0 \ 0 \ 0]^T$$

A **net system** $\langle N, M_0 \rangle$ is a net $N$ with an initial marking $M_0$. 

---

**Maria Paola Cabasino (DIEE, University of Cagliari)**

**Identification of place/transition nets**

---

**PhD Course 27 Mar 07**
What is a Petri net?

Enabling

A transition \( t \) is **enabled** at \( M \) iff \( M \geq Pre(\cdot, t) \)

Transition \( t_2 \) is **enabled** at \( M_0 \) because \( M_0 \geq Pre(\cdot, t_2) \)
What is a Petri net?

Firing

Transition $t_2$ fires yielding a new marking $M_0[\langle t_2 \rangle]M$ with

$$M = M_0 - Pre(\cdot, t_2) + Post(\cdot, t_2) = M_0 + C(\cdot, t_2)$$

$$\begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \end{bmatrix}$$
Language and reachability set

The language of $\langle N, M_0 \rangle$ is the set of all firing sequences that are enabled at the initial marking:

$$L(N, M_0) = \{ \sigma \in T^* \mid M_0[\sigma] \}.$$ 

We also define the set of firing sequences of length less than or equal to $k \in \mathbb{N}$ as:

$$L_k(N, M_0) = \{ \sigma \in L(N, M_0) \mid |\sigma| \leq k \}$$

A marking $M$ is reachable in $\langle N, M_0 \rangle$ if there exists a firing sequence $\sigma$ such that $M_0[\sigma]M$.

The set of all markings reachable in $\langle N, M_0 \rangle$ is called the reachability set of $\langle N, M_0 \rangle$ and is denoted as

$$R(N, M_0) \subseteq \mathbb{N}^m.$$
What is a Petri net?

Language and reachability set

The **language** of \( \langle N, M_0 \rangle \) is the set of all firing sequences that are enabled at the initial marking:

\[
L(N, M_0) = \{ \sigma \in T^* \mid M_0[\sigma] \}. 
\]

We also define the set of firing sequences of length less than or equal to \( k \in \mathbb{N} \) as:

\[
L_k(N, M_0) = \{ \sigma \in L(N, M_0) \mid |\sigma| \leq k \}
\]

A marking \( M \) is **reachable** in \( \langle N, M_0 \rangle \) if there exists a firing sequence \( \sigma \) such that \( M_0[\sigma]M \).

The set of all markings reachable in \( \langle N, M_0 \rangle \) is called the **reachability set** of \( \langle N, M_0 \rangle \) and is denoted as

\[
R(N, M_0) \subseteq \mathbb{N}^m.
\]
Language and reachability set

The language of $\langle N, M_0 \rangle$ is the set of all firing sequences that are enabled at the initial marking:

$$L(N, M_0) = \{ \sigma \in T^* \mid M_0[\sigma] \}.$$  

We also define the set of firing sequences of length less than or equal to $k \in \mathbb{N}$ as:

$$L_k(N, M_0) = \{ \sigma \in L(N, M_0) \mid |\sigma| \leq k \}$$

A marking $M$ is reachable in $\langle N, M_0 \rangle$ if there exists a firing sequence $\sigma$ such that $M_0[\sigma]M$.

The set of all markings reachable in $\langle N, M_0 \rangle$ is called the reachability set of $\langle N, M_0 \rangle$ and is denoted as

$$R(N, M_0) \subseteq \mathbb{N}^m.$$
What is a Petri net?

Language and reachability set

The language of \( \langle N, M_0 \rangle \) is the set of all firing sequences that are enabled at the initial marking:

\[
L(N, M_0) = \{ \sigma \in T^* \mid M_0[\sigma] \}.
\]

We also define the set of firing sequences of length less than or equal to \( k \in \mathbb{N} \) as:

\[
L_k(N, M_0) = \{ \sigma \in L(N, M_0) \mid |\sigma| \leq k \}
\]

A marking \( M \) is reachable in \( \langle N, M_0 \rangle \) if there exists a firing sequence \( \sigma \) such that \( M_0[\sigma]M \).

The set of all markings reachable in \( \langle N, M_0 \rangle \) is called the reachability set of \( \langle N, M_0 \rangle \) and is denoted as

\[
R(N, M_0) \subseteq \mathbb{N}^m.
\]
What is a Petri net?

**Language and reachability set**

The **language** of \( \langle N, M_0 \rangle \) is the set of all firing sequences that are enabled at the initial marking:

\[
L(N, M_0) = \{ \sigma \in T^* \mid M_0[\sigma] \}\.
\]

We also define the set of firing sequences of length less than or equal to \( k \in \mathbb{N} \) as:

\[
L_k(N, M_0) = \{ \sigma \in L(N, M_0) \mid |\sigma| \leq k \}
\]

A marking \( M \) is **reachable** in \( \langle N, M_0 \rangle \) if there exists a firing sequence \( \sigma \) such that \( M_0[\sigma]M \).

The set of all markings reachable in \( \langle N, M_0 \rangle \) is called the **reachability set** of \( \langle N, M_0 \rangle \) and is denoted as

\[
R(N, M_0) \subseteq \mathbb{N}^m.
\]
What is a Petri net?

Structural properties

A P-vector $\vec{x} : P \rightarrow \mathbb{N}$, with $\vec{x} \neq \vec{0}$, is called:

- **P-invariant**: if $\vec{x}^T \cdot C = \vec{0}^T$;
- **P-increasing**: if $\vec{x}^T \cdot C \geq \vec{0}^T$;
- **P-decreasing**: if $\vec{x}^T \cdot C \leq \vec{0}^T$.

It can be shown that if $\vec{x}$ is a P-invariant (resp., P-increasing, P-decreasing) along any evolution the sum of the markings weighted with vector $\vec{x}$ remains constant (resp., does not decrease, does not increase).
What is a Petri net?

### Structural properties

A P-vector $\vec{x} : P \to \mathbb{N}$, with $\vec{x} \neq \vec{0}$, is called:

- **P-invariant**: if $\vec{x}^T \cdot C = \vec{0}^T$;
- **P-increasing**: if $\vec{x}^T \cdot C \geq \vec{0}^T$;
- **P-decreasing**: if $\vec{x}^T \cdot C \leq \vec{0}^T$.

It can be shown that if $\vec{x}$ is a P-invariant (resp., P-increasing, P-decreasing) along any evolution the sum of the markings weighted with vector $\vec{x}$ remains constant (resp., does not decrease, does not increase).
Structural properties

A $T$-vector $\vec{y} : T \rightarrow \mathbb{N}$, with $\vec{y} \neq \vec{0}$, is called:

- **$T$-invariant**: if $C \cdot \vec{y} = \vec{0}$;
- **$T$-increasing**: if $C \cdot \vec{y} \geq \vec{0}$;
- **$T$-decreasing**: if $C \cdot \vec{y} \leq \vec{0}$.

It can be shown that if $\vec{y}$ is a $T$-invariant the firing of a sequence of transitions whose firing vector is $\vec{y}$ does not modify the number of tokens, i.e., it is a stationary sequence.

If $\vec{y}$ is a $T$-increasing the firing of a sequence of transitions whose firing vector is $\vec{y}$ increases the number of tokens, i.e., it is a repetitive non stationary sequence.

Finally if $\vec{y}$ is a $T$-decreasing the firing of a sequence of transitions whose firing vector is $\vec{y}$ decreases the number of tokens.
Structural properties

A $\mathbf{y} : T \rightarrow \mathbb{N}$, with $\mathbf{y} \neq \mathbf{0}$, is called:

- **$T$-invariant**: if $C \cdot \mathbf{y} = \mathbf{0}$;
- **$T$-increasing**: if $C \cdot \mathbf{y} \geq \mathbf{0}$;
- **$T$-decreasing**: if $C \cdot \mathbf{y} \leq \mathbf{0}$.

It can be shown that if $\mathbf{y}$ is a $T$-invariant the firing of a sequence of transitions whose firing vector is $\mathbf{y}$ does not modify the number of tokens, i.e., it is a stationary sequence.

If $\mathbf{y}$ is a $T$-increasing the firing of a sequence of transitions whose firing vector is $\mathbf{y}$ increases the number of tokens, i.e., it is a repetitive non stationary sequence.

Finally if $\mathbf{y}$ is a $T$-decreasing the firing of a sequence of transitions whose firing vector is $\mathbf{y}$ decreases the number of tokens.
What is a Petri net?

Structural properties

A $T$-vector $\vec{y} : T \rightarrow \mathbb{N}$, with $\vec{y} \neq \vec{0}$, is called:

- **$T$-invariant**: if $C \cdot \vec{y} = \vec{0}$;
- **$T$-increasing**: if $C \cdot \vec{y} \geq \vec{0}$;
- **$T$-decreasing**: if $C \cdot \vec{y} \leq \vec{0}$.

It can be shown that if $\vec{y}$ is a **$T$-invariant** the firing of a sequence of transitions whose firing vector is $\vec{y}$ does not modify the number of tokens, i.e., it is a stationary sequence.

If $\vec{y}$ is a **$T$-increasing** the firing of a sequence of transitions whose firing vector is $\vec{y}$ increases the number of tokens, i.e., it is a repetitive non stationary sequence.

Finally if $\vec{y}$ is a **$T$-decreasing** the firing of a sequence of transitions whose firing vector is $\vec{y}$ decreases the number of tokens.
Structural properties

A $T$-vector $\vec{y} : T \to \mathbb{N}$, with $\vec{y} \neq \vec{0}$, is called:

- **$T$-invariant**: if $C \cdot \vec{y} = \vec{0}$;
- **$T$-increasing**: if $C \cdot \vec{y} \geq \vec{0}$;
- **$T$-decreasing**: if $C \cdot \vec{y} \leq \vec{0}$.

It can be shown that if $\vec{y}$ is a $T$-invariant the firing of a sequence of transitions whose firing vector is $\vec{y}$ does not modify the number of tokens, i.e., it is a stationary sequence.

If $\vec{y}$ is a $T$-increasing the firing of a sequence of transitions whose firing vector is $\vec{y}$ increases the number of tokens, i.e., it is a repetitive non stationary sequence.

Finally if $\vec{y}$ is a $T$-decreasing the firing of a sequence of transitions whose firing vector is $\vec{y}$ decreases the number of tokens.
What is a Petri net?

Structural properties

ordinary net \[ \Rightarrow \text{Pre, Post} \in \{0, 1\}^{m \times n}. \]

marked graph \[ \Rightarrow |\bullet p| = |p^\bullet| = 1. \]

state machine \[ \Rightarrow |\bullet t| = |t^\bullet| = 1. \]
**Structural properties**

**ordinary net** \( \implies \) \(\text{Pre, Post} \in \{0,1\}^{m \times n}\).

**marked graph** \( \implies \) \(|\cdot p| = |p^\cdot| = 1\).

**state machine** \( \implies \) \(|\cdot t| = |t^\cdot| = 1\).
**Structural properties**

**ordinary net** \(\implies\) \(\text{Pre, Post} \in \{0, 1\}^{m \times n}\).

**marked graph** \(\implies\) \(|\cdot p| = |p^\cdot| = 1\).

**state machine** \(\implies\) \(|\cdot t| = |t^\cdot| = 1\).
What is a Petri net?

Structural properties

ordinary net $\implies \text{Pre, Post } \in \{0, 1\}^{m \times n}$.

marked graph $\implies |\cdot p| = |p^\bullet| = 1$.

state machine $\implies |\cdot t| = |t^\bullet| = 1$. 
What is a Petri net?

Labeled Petri nets

Given a Petri net $N$ with set of transitions $T$, a labeling function $\varphi : T \rightarrow E \cup \{\varepsilon\}$ assigns to each transition $t \in T$ a symbol, from a given alphabet $E$, or assigns to it the empty string $\varepsilon$.

A labeled Petri net system is a 3-tuple $G = \langle N, M_0, \varphi \rangle$ where $N = (P, T, Pre, Post)$, $M_0$ is the initial marking, and $\varphi : T \rightarrow E \cup \{\varepsilon\}$ is the labeling function.
Labeled Petri nets

Given a Petri net $N$ with set of transitions $T$, a labeling function $\varphi : T \rightarrow E \cup \{\varepsilon\}$ assigns to each transition $t \in T$ a symbol, from a given alphabet $E$, or assigns to it the empty string $\varepsilon$.

A labeled Petri net system is a 3-tuple $G = \langle N, M_0, \varphi \rangle$ where $N = (P, T, Pre, Post)$, $M_0$ is the initial marking, and $\varphi : T \rightarrow E \cup \{\varepsilon\}$ is the labeling function.
What is a Petri net?

Labeled Petri nets

**labeling function** can be classified as:

- **free**: if $\varphi(t)$ is 1 to 1, and no transition is labeled with the empty string;
- **deterministic**: if one may have $\varphi(t) = \varphi(t')$ when $t \neq t'$ but for each reachable marking at most one can be enabled, and no transition is labeled with the empty string;
- **$\lambda$-free**: if one may have $\varphi(t) = \varphi(t')$ when $t \neq t'$, and no transition is labeled with the empty string;
- **arbitrary**: if no restriction is posed on the labeling function $\varphi$. 
A labeled Petri net is a Petri net with a labeling function that assigns a string to each transition. The labeling function can be classified as:

- **free**: if \( \varphi(t) \) is 1 to 1, and no transition is labeled with the empty string;
- **deterministic**: if one may have \( \varphi(t) = \varphi(t') \) when \( t \neq t' \) but for each reachable marking at most one can be enabled, and no transition is labeled with the empty string;
- **\( \lambda \)-free**: if one may have \( \varphi(t) = \varphi(t') \) when \( t \neq t' \), and no transition is labeled with the empty string;
- **arbitrary**: if no restriction is posed on the labeling function \( \varphi \).
labeling function can be classified as:

- **free**: if $\varphi(t)$ is 1 to 1, and no transition is labeled with the empty string;
- **deterministic**: if one may have $\varphi(t) = \varphi(t')$ when $t \neq t'$ but for each reachable marking at most one can be enabled, and no transition is labeled with the empty string;
- **$\lambda$-free**: if one may have $\varphi(t) = \varphi(t')$ when $t \neq t'$, and no transition is labeled with the empty string;
- **arbitrary**: if no restriction is posed on the labeling function $\varphi$. 
What is a Petri net?

Labeled Petri nets

labeling function can be classified as:

- **free**: if $\varphi(t)$ is 1 to 1, and no transition is labeled with the empty string;
- **deterministic**: if one may have $\varphi(t) = \varphi(t')$ when $t \neq t'$ but for each reachable marking at most one can be enabled, and no transition is labeled with the empty string;
- **$\lambda$-free**: if one may have $\varphi(t) = \varphi(t')$ when $t \neq t'$, and no transition is labeled with the empty string;
- **arbitrary**: if no restriction is posed on the labeling function $\varphi$. 

Maria Paola Cabasino (DIEE, University of Cagliari)
Labeled Petri nets

The labeling function can be classified as:

- **free**: if \( \varphi(t) \) is 1 to 1, and no transition is labeled with the empty string;
- **deterministic**: if one may have \( \varphi(t) = \varphi(t') \) when \( t \neq t' \) but for each reachable marking at most one can be enabled, and no transition is labeled with the empty string;
- **\( \lambda \)-free**: if one may have \( \varphi(t) = \varphi(t') \) when \( t \neq t' \), and no transition is labeled with the empty string;
- **arbitrary**: if no restriction is posed on the labeling function \( \varphi \).
Outline

- Background of Petri nets
- **Motivation**
- Identification a *free-labeled* Petri net
- Extensions of the procedure
- Numerical Simulations
- Identification of a *labeled* Petri net
- Complexity of the identification procedure
- Coverability graph
Motivation

Identification: given a pair of observed input-output signals determine a system consistent with the observation.

Two main problems can be solved in this framework:

- *(Time-driven systems)* We want to find the model of an existing system from "external" measurements.

- *(DES)* We have examples of admissible/forbidden behaviors and want to design a system that satisfies these constraints.
Motivation

Identification: given a pair of observed input-output signals determine a system consistent with the observation.

Two main problems can be solved in this framework:

- *(Time-driven systems)* We want to find the model of an existing system from "external" measurements.

- *(DES)* We have examples of admissible/forbidden behaviors and want to design a system that satisfies these constraints.
Identification: given a pair of observed input-output signals determine a system consistent with the observation.

Two main problems can be solved in this framework:

- *(Time-driven systems)* We want to find the model of an existing system from ”external” measurements.

- *(DES)* We have examples of admissible/forbidden behaviors and want to design a system that satisfies these constraints.
Motivation

Identification: given a pair of observed input-output signals determine a system consistent with the observation.

Two main problems can be solved in this framework:

- *(Time-driven systems)* We want to find the model of an existing system from "external" measurements.

- *(DES)* We have examples of admissible(forbidden) behaviors and want to design a system that satisfies these constraints.
Motivation

Identification using Petri nets

Observed behavior

- Usually it is the language of the net, i.e., the set of firable transition sequences (event measurement).
- However, one may assume that some partial information on the state is also known as in the next talk (state measurement).

Should be able to accept both positive examples, i.e., strings in the language of the net, and counterexamples, i.e., strings not in the language.
Motivation

Identification using Petri nets

Observed behavior

- Usually it is the language of the net, i.e., the set of firable transition sequences (event measurement).
- However, one may assume that some partial information on the state is also known as in the next talk (state measurement).

Should be able to accept both positive examples, i.e., strings in the language of the net, and counterexamples, i.e., strings not in the language.
Outline

- Background of Petri nets
- Motivation
- Identification a free-labeled Petri net
- Extensions of the procedure
- Numerical Simulations
- Identification of a labeled Petri net
- Complexity of the identification procedure
- Coverability graph
Introduction

We deal with the problem of identifying a Petri net system, given a finite language that it generates.

We assume $E = T$.

We show that the identification problem can be solved via an integer programming problem where a suitable objective function can be used to determine a minimal net according to a given measure.
Introduction

We deal with the problem of identifying a Petri net system, given a finite language that it generates.

We assume $E = T$.

We show that the identification problem can be solved via an integer programming problem where a suitable objective function can be used to determine a minimal net according to a given measure.
Introduction

We deal with the problem of identifying a Petri net system, given a finite language that it generates.

We assume $E = T$.

We show that the identification problem can be solved via an integer programming problem where a suitable objective function can be used to determine a minimal net according to a given measure.
We deal with the problem of identifying a Petri net system, given a finite language that it generates.

We assume $E = T$.

We show that the identification problem can be solved via an integer programming problem where a suitable objective function can be used to determine a minimal net according to a given measure.
Identification problem 1

Problem 1: Given a set of places \( P = \{p_1, \ldots, p_m\} \) and a set of transitions \( T = \{t_1, \ldots, t_n\} \). Let \( \mathcal{L} \subset T^* \) be a finite prefix-closed language over \( T \), and \( k = \max_{\sigma \in \mathcal{L}} |\sigma| \) be the length of the longest string in \( \mathcal{L} \).

We want to identify the structure of a net \( N = (P, T, \text{Pre}, \text{Post}) \) and an initial marking \( M_0 \) such that \( L_k(N, M_0) = \mathcal{L} \).

The unknowns we want to determine are the elements of the two matrices \( \text{Pre} = \{e_{i,j}\} \in \mathbb{N}^{m \times n} \) and \( \text{Post} = \{o_{i,j}\} \in \mathbb{N}^{m \times n} \) and the elements of the vector \( M_0 = [m_{0,1} \ m_{0,2} \ \cdots \ m_{0,m}]^T \in \mathbb{N}^m \).
Identification problem 1

Problem 1: Given a set of places $P = \{p_1, \ldots, p_m\}$ and a set of transitions $T = \{t_1, \ldots, t_n\}$. Let $\mathcal{L} \subseteq T^*$ be a finite prefix-closed language over $T$, and $k = \max_{\sigma \in \mathcal{L}} |\sigma|$ be the length of the longest string in $\mathcal{L}$.

We want to identify the structure of a net $N = (P, T, \text{Pre}, \text{Post})$ and an initial marking $M_0$ such that $L_k(N, M_0) = \mathcal{L}$.

The unknowns we want to determine are the elements of the two matrices $\text{Pre} = \{e_{i,j}\} \in \mathbb{N}^{m \times n}$ and $\text{Post} = \{o_{i,j}\} \in \mathbb{N}^{m \times n}$ and the elements of the vector $M_0 = \begin{bmatrix} m_{0,1} & m_{0,2} & \cdots & m_{0,m} \end{bmatrix}^T \in \mathbb{N}^m$. 
Identification problem 1

Problem 1: Given a set of places \( P = \{p_1, \ldots, p_m\} \) and a set of transitions \( T = \{t_1, \ldots, t_n\} \). Let \( \mathcal{L} \subset T^* \) be a finite prefix-closed language over \( T \), and \( k = \max_{\sigma \in \mathcal{L}} |\sigma| \) be the length of the longest string in \( \mathcal{L} \).

We want to identify the structure of a net \( N = (P, T, Pre, Post) \) and an initial marking \( M_0 \) such that \( L_k(N, M_0) = \mathcal{L} \).

The unknowns we want to determine are the elements of the two matrices \( Pre = \{e_{i,j}\} \in \mathbb{N}^{m \times n} \) and \( Post = \{o_{i,j}\} \in \mathbb{N}^{m \times n} \) and the elements of the vector \( M_0 = [m_{0,1} \ m_{0,2} \ \cdots \ m_{0,m}]^T \in \mathbb{N}^m \).
Identification problem 1

We prove that a solution to the previous identification problem can be computed thanks to the following theorem.

**Theorem 1:** A solution, to the identification problem presented, satisfies the following set of linear algebraic constraints

\[
\mathcal{G}(m, T, L) \triangleq \\
\begin{align*}
M_0 + \text{Post} \cdot \vec{\sigma} - \text{Pre} \cdot (\vec{\sigma} + \vec{\epsilon}_j) & \geq \vec{0} \\
-KS(\sigma, t_j) + M_0 + \text{Post} \cdot \vec{\sigma} & - \text{Pre} \cdot (\vec{\sigma} + \vec{\epsilon}_j) \leq -\vec{1}_m \\
\vec{1}^T S(\sigma, t_j) & \leq m - 1 \\
M_0 & \in \mathbb{N}^m, \text{Pre}, \text{Post} \in \mathbb{N}^{m \times n} \\
S(\sigma, t_j) & \in \{0, 1\}^m
\end{align*}
\]

\[
\mathcal{E} = \{(\sigma, t_j) \mid \sigma \in L, |\sigma| < k, \sigma t_j \in L\} \\
\mathcal{D} = \{(\sigma, t_j) \mid \sigma \in L, |\sigma| < k, \sigma t_j \not\in L\}
\]
Identification problem 1

We prove that a solution to the previous identification problem can be computed thanks to the following theorem.

**Theorem 1:** A solution, to the identification problem presented, satisfies the following set of linear algebraic constraints

\[
G(m, T, L) \triangleq \begin{cases}
    M_0 + \text{Post} \cdot \bar{\sigma} - \text{Pre} \cdot (\bar{\sigma} + \bar{\epsilon}_j) & \geq \vec{0} \\
    -KS(\sigma, t_j) + M_0 + \text{Post} \cdot \bar{\sigma} - \text{Pre} \cdot (\bar{\sigma} + \bar{\epsilon}_j) & \leq -\vec{1}_m \\
    \vec{1}_T S(\sigma, t_j) & \leq m - 1 \\
    M_0 & \in \mathbb{N}^m, \text{Pre, Post} \in \mathbb{N}^{m \times n} \\
    S(\sigma, t_j) & \in \{0, 1\}^m
\end{cases}
\]

\[\forall (\sigma, t_j) \in \mathcal{E}\]
\[\forall (\sigma, t_j) \in \mathcal{D}\]

\[\mathcal{E} = \{(\sigma, t_j) \mid \sigma \in L, |\sigma| < k, \sigma t_j \in L\}\]
\[\mathcal{D} = \{(\sigma, t_j) \mid \sigma \in L, |\sigma| < k, \sigma t_j \not\in L\}\]
Problem 1: Let us consider the identification problem previously defined and let $f(M_0, Pre, Post)$ be a given performance index. The solution to the identification problem that minimizes $f(M_0, Pre, Post)$ can be computed by solving the following IPP

$$\begin{align*}
\text{min} & \quad f(M_0, Pre, Post) \\
\text{s.t.} & \quad G(m, T, \mathcal{L}).
\end{align*}$$
Let $\mathcal{L} = \{\varepsilon, t_1, t_1 t_1, t_1 t_2, t_1 t_1 t_2, t_1 t_2 t_1\}$ and $m = 2$, thus $k = 3$.  

\[
\begin{cases}
\text{min } \vec{1}^m \cdot M_0 + \vec{1}^m \cdot (\text{Pre} + \text{Post}) \cdot \vec{1}^n \\
\text{s.t. } L_3(N, M_0) = \mathcal{L}
\end{cases}
\]

$\mathcal{E} = \{(\varepsilon, t_1), (t_1, t_1), (t_1, t_2), (t_1 t_2, t_1), (t_1 t_1, t_2)\}$

$\mathcal{D} = \{(\varepsilon, t_2), (t_1 t_2, t_2), (t_1 t_1, t_1)\}$
The procedure identifies a net system with

$$\text{Pre} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{Post} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad M_0 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$
Outline

- Background of Petri nets
- Motivation and literature
- Identification a *free-labeled* Petri net
- **Extensions of the procedure**
- Numerical Simulations
- Identification of a *labeled* Petri net
- Complexity of the identification procedure
- Coverability graph
Extensions of the procedure

Extended identification procedure for free labeled Petri nets

Additional information can easily be incorporated in the ID procedure.

- **Structural constraints** (i.e., P-vectors, T-vectors, Net subclasses, Constraints on $M_0$, ...)
- Synthesis from the reachability graph
- Optimizing the number of places
Additional information can easily be incorporated in the ID procedure.

- **Structural constraints** (i.e., P-vectors, T-vectors, Net subclasses, Constraints on $M_0$, ...)
- Synthesis from the reachability graph
- Optimizing the number of places
Extended identification procedure for free labeled Petri nets

Additional information can easily be incorporated in the ID procedure.

- **Structural constraints** (i.e., P-vectors, T-vectors, Net subclasses, Constraints on $M_0, \ldots$)
- **Synthesis from the reachability graph**
- **Optimizing the number of places**
Extended identification procedure for free labeled Petri nets

Additional information can easily be incorporated in the ID procedure.

- **Structural constraints** (i.e., P-vectors, T-vectors, Net subclasses, Constraints on $M_0, \ldots$)
- **Synthesis from the reachability graph**
- **Optimizing the number of places**
Extensions of the procedure

Structural constraints: **P-vectors**

Assume that some places of the net are known to belong to a **conservative component**, i.e., the weighted sum of their tokens in the component remains constant during any evolution.

This is equivalent to say that some P-invariants for the net are known.

Assume \( \bar{x} \in \mathbb{R}^m \) is a **P-invariant**. We need to add to Problem 1 the following constraint

\[
\bar{x}^T (Post - Pre) = \bar{0}_n^T
\]
Structural constraints: P-vectors

Assume that some places of the net are known to belong to a conservative component, i.e., the weighted sum of their tokens in the component remains constant during any evolution.

This is equivalent to say that some P-invariants for the net are known.

Assume $\vec{x} \in \mathbb{R}^m$ is a P-invariant. We need to add to Problem 1 the following constraint

$$\vec{x}^T (\text{Post} - \text{Pre}) = \vec{0}_n$$
Extensions of the procedure

Structural constraints: \textbf{P-vectors}

Assume that some places of the net are known to belong to a \textit{conservative component}, i.e., the weighted sum of their tokens in the component remains constant during any evolution.

This is equivalent to say that some P-invariants for the net are known.

Assume $\vec{x} \in \mathbb{R}^m$ is a \textit{P-invariant}. We need to add to Problem 1 the following constraint

$$\vec{x}^T (Post - Pre) = \vec{0}_n^T$$
Extensions of the procedure

Structural constraints: \textbf{T-vectors}

Assume that a given \textit{firing sequence} is known to be \textit{stationary}, i.e., the number of the tokens of the net is not modified by the firing of this sequence.

This is equivalent to say that some T-invariants for this net are known.

Assume $\vec{y} \in \mathbb{R}^n$ is a T-invariant. We need to add to Problem 1 the following constraint

$$(Post - Pre)\vec{y} = \vec{0}_m$$
Structural constraints: $T$-vectors

Assume that a given firing sequence is known to be stationary, i.e., the number of the tokens of the net is not modified by the firing of this sequence.

This is equivalent to say that some $T$-invariants for this net are known.

Assume $\vec{y} \in \mathbb{R}^n$ is a $T$-invariant. We need to add to Problem 1 the following constraint

$$(Post - Pre)\vec{y} = \vec{0}_m$$
Extensions of the procedure

Structural constraints: **T-vectors**

Assume that a given firing sequence is known to be stationary, i.e., the number of the tokens of the net is not modified by the firing of this sequence.

This is equivalent to say that some T-invariants for this net are known.

Assume $\vec{y} \in \mathbb{R}^n$ is a T-invariant. We need to add to Problem 1 the following constraint

$$(Post - Pre)\vec{y} = \vec{0}_m$$
Example

Let $\mathcal{L} = \{\varepsilon, t_1, t_1 t_1, t_1 t_2, t_1 t_1 t_2, t_1 t_2 t_1\}$ and $m = 2$, thus $k = 3$.

Hp: $m_1 + m_2$ constant $\implies \vec{x} = [1 \ 1]^T$

$$
\text{Pre} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{Post} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad M_0 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}
$$

\begin{align*}
\text{(a)} & \quad \text{p}_1 \quad \text{t}_1 \quad \text{p}_2 \\
\text{(b)} & \quad \text{p}_1 \quad \text{p}_2
\end{align*}
Extensions of the procedure

Net subclasses

- Ordinary:
  \[ Pre, Post \in \{0, 1\}^{m \times n}. \]

- Marked graph:
  \[
  \begin{cases}
    Pre \cdot \vec{1}_n = 1 \\
    Post \cdot \vec{1}_n = 1.
  \end{cases}
  \]

- State machine:
  \[
  \begin{cases}
    \vec{1}_m^T \cdot Pre = 1 \\
    \vec{1}_m^T \cdot Post = 1.
  \end{cases}
  \]
Net subclasses

■ Ordinary:

$$Pre, Post \in \{0, 1\}^{m \times n}.$$ 

■ Marked graph:

$$\begin{cases}
Pre \cdot \vec{1}_n = 1 \\
Post \cdot \vec{1}_n = 1.
\end{cases}$$

■ State machine:

$$\begin{cases}
\vec{1}_m^T \cdot Pre = 1 \\
\vec{1}_m^T \cdot Post = 1.
\end{cases}$$
Net subclasses

- **Ordinary:**

  \( Pre, Post \in \{0, 1\}^{m \times n} \).

- **Marked graph:**

  \[
  \begin{cases}
  Pre \cdot \vec{1}_n = 1 \\
  Post \cdot \vec{1}_n = 1.
  \end{cases}
  \]

- **State machine:**

  \[
  \begin{cases}
  \vec{1}_m^T \cdot Pre = 1 \\
  \vec{1}_m^T \cdot Post = 1.
  \end{cases}
  \]
Net subclasses

- **Ordinary:**

  \[ \text{Pre, Post } \in \{0, 1\}^{m \times n}. \]

- **Marked graph:**

  \[
  \begin{align*}
  \text{Pre} \cdot \vec{1}_n &= 1 \\
  \text{Post} \cdot \vec{1}_n &= 1.
  \end{align*}
  \]

- **State machine:**

  \[
  \begin{align*}
  \vec{1}^T_m \cdot \text{Pre} &= 1 \\
  \vec{1}^T_m \cdot \text{Post} &= 1.
  \end{align*}
  \]
Extensions of the procedure

Constraints on the initial marking

\textbf{GMEC} (Generalized Mutual Exclusion Constraint): \((\vec{w}, k)\), where \(\vec{w} \in \mathbb{Z}^m, k \in \mathbb{Z}\).

This constraint defines a set of legal markings:

\[ M(\vec{w}, k) = \{ M \in \mathbb{N}^m \mid \vec{w}^T M \leq k \}. \]

If \( M_0 \in M(\vec{w}, k) \implies \vec{w}^T M_0 \leq k \)

Example: consider a Petri net with an initial marking that can not contain a number of tokens greater than 1 in places \( p_1 \) and \( p_2 \).

\[ M(p_1) + M(p_2) \leq 1. \]
Constraints on the initial marking

**GMEC** (Generalized Mutual Exclusion Constraint): \((\vec{w}, k)\), where \(\vec{w} \in \mathbb{Z}^m, k \in \mathbb{Z}\).

This constraint defines a set of legal markings:

\[
\mathcal{M}(\vec{w}, k) = \{ M \in \mathbb{N}^m | \vec{w}^T M \leq k \}.
\]

If \( M_0 \in \mathcal{M}(\vec{w}, k) \implies \vec{w}^T M_0 \leq k \)

**Example**: consider a Petri net with an initial marking that cannot contain a number of tokens greater than 1 in places \(p_1\) and \(p_2\).

\[
M(p_1) + M(p_2) \leq 1.
\]
Constraints on the initial marking

**GMEC** (Generalized Mutual Exclusion Constraint): $(\vec{w}, k)$, where $\vec{w} \in \mathbb{Z}^m$, $k \in \mathbb{Z}$.

This constraint defines a set of legal markings:

$$\mathcal{M}(\vec{w}, k) = \{ M \in \mathbb{N}^m \mid \vec{w}^T M \leq k \}.$$ 

If $M_0 \in \mathcal{M}(\vec{w}, k) \implies \vec{w}^T M_0 \leq k$

Example: consider a Petri net with an initial marking that cannot contain a number of tokens greater than 1 in places $p_1$ and $p_2$.

$$M(p_1) + M(p_2) \leq 1.$$
Extensions of the procedure

Constraints on the initial marking

GMEC (Generalized Mutual Exclusion Constraint): \((\vec{w}, k)\), where \(\vec{w} \in \mathbb{Z}^m, k \in \mathbb{Z}\).

This constraint defines a set of legal markings:

\[
\mathcal{M}(\vec{w}, k) = \{ M \in \mathbb{N}^m \mid \vec{w}^T M \leq k \}.
\]

If \(M_0 \in \mathcal{M}(\vec{w}, k) \implies \vec{w}^T M_0 \leq k\)

Example: consider a Petri net with an initial marking that cannot contain a number of tokens greater than 1 in places \(p_1\) and \(p_2\).

\[
M(p_1) + M(p_2) \leq 1.
\]
Assume that the net system we want to synthesize is bounded, and thus its language is regular.

The language is given in terms of a finite state automaton

\[ G = (Q, T, \delta, q_0) \]

where \( Q \) is the set of states, the alphabet \( T \) is the set of transitions of the net, \( \delta : Q \times T \rightarrow Q \) is the transition function, and \( q_0 \) is the initial state.
Assume that the net system we want to synthesize is bounded, and thus its language is regular.

The language is given in terms of a finite state automaton

\[ G = (Q, T, \delta, q_0) \]

where \( Q \) is the set of states, the alphabet \( T \) is the set of transitions of the net, \( \delta : Q \times T \rightarrow Q \) is the transition function, and \( q_0 \) is the initial state.
Assume that the net system we want to synthesize is bounded, and thus its language is regular.

The language is given in terms of a finite state automaton

\[ G = (Q, T, \delta, q_0) \]

where \( Q \) is the set of states, the alphabet \( T \) is the set of transitions of the net, \( \delta : Q \times T \rightarrow Q \) is the transition function, and \( q_0 \) is the initial state.
Problem 2: Let $G = (Q, T, \delta, q_0)$ be a given finite state automaton. Chosen a set of places $P$ of cardinality $m$, we want to identify the structure of a net $N = (P, T, Pre, Post)$ and an initial marking $M_0$ such that $L(N, M_0) = L(G)$.

The unknowns we want to determine are the elements of the two matrices $Pre = \{e_{i,j}\} \in \mathbb{N}^{m \times n}$ and $Post = \{o_{i,j}\} \in \mathbb{N}^{m \times n}$ and the elements of the vector $M_0 = [m_{0,1} \ m_{0,2} \ \cdots \ m_{0,m}]^T \in \mathbb{N}^m$. 
Problem 2: Let \( G = (Q, T, \delta, q_0) \) be a given finite state automaton. Chosen a set of places \( P \) of cardinality \( m \), we want to identify the structure of a net \( N = (P, T, \text{Pre}, \text{Post}) \) and an initial marking \( M_0 \) such that \( L(N, M_0) = L(G) \).

The unknowns we want to determine are the elements of the two matrices \( \text{Pre} = \{ e_{i,j} \} \in \mathbb{N}^{m \times n} \) and \( \text{Post} = \{ o_{i,j} \} \in \mathbb{N}^{m \times n} \) and the elements of the vector \( M_0 = \begin{bmatrix} m_{0,1} & m_{0,2} & \cdots & m_{0,m} \end{bmatrix}^T \in \mathbb{N}^m \).
Extensions of the procedure

Synthesis of bounded Petri nets from regular languages

The identification procedure previously defined considers sequences of bounded length.

An automaton is able to generate sequences of unbounded length every time that there is a cycle. Thus we have to distinguish between sequences that pass through cycles (that can be extended indefinitely) and sequences that do not pass through cycles (whose length is finite).

To this aim, it is sufficient to define $\mathcal{L}$ as the set of sequences that are generated by the automaton without passing through a cycle. Then, we need to impose as additional constraints the set of minimal $T$-invariants.
The identification procedure previously defined considers sequences of bounded length.

An automaton is able to generate sequences of unbounded length every time that there is a cycle. Thus we have to distinguish between sequences that pass through cycles (that can be extended indefinitely) and sequences that do not pass through cycles (whose length is finite).

To this aim, it is sufficient to define $\mathcal{L}$ as the set of sequences that are generated by the automaton without passing through a cycle. Then, we need to impose as additional constraints the set of minimal $T$-invariants.
The identification procedure previously defined considers sequences of bounded length.

An automaton is able to generate sequences of unbounded length every time that there is a cycle. Thus we have to distinguish between sequences that pass through cycles (that can be extended indefinitely) and sequences that do not pass through cycles (whose length is finite).

To this aim, it is sufficient to define $\mathcal{L}$ as the set of sequences that are generated by the automaton without passing through a cycle. Then, we need to impose as additional constraints the set of minimal $T$-invariants.
Consider the finite state automaton $G$.

Minimal $T$-invariant: $[1 \ 1]^T$ and $\mathcal{L}_T(G) = \{\epsilon, t_1, t_1t_1\}$ thus $\mathcal{E} = \{(\epsilon, t_1), (t_1, t_1), (t_1, t_2), (t_1t_1, t_2)\}$ and $\mathcal{D} = \{(\epsilon, t_2), (t_1t_1, t_1)\}$.

Now, assume that we want to determine the Petri net system that minimizes the sum of initial tokens and all arcs.
Example

For $m = 1$ we get no feasible solution, while for $m = 2$ we find the net system in Figure (b), whose reachability graph is shown in Figure (a).

Note that in this particular case the reachability graph of the net is isomorphic to the given automaton $G$, but this is not necessarily guaranteed by our procedure.
We have also solved the problem where the number $m$ of the places is not given, but it is only known to be less or equal to a given value $\bar{m}$.

For this problem we have formulated another IPP similar to the previous one.

\[
\begin{array}{l}
\min_{m \leq \bar{m}} f(m, M_0, Pre, Post) \\
\text{s.t. } G(m, T, L).
\end{array}
\] (1)
Optimizing the number of places

We have also solved the problem where the number \( m \) of the places is not given, but it is only known to be less or equal to a given value \( \bar{m} \).

For this problem we have formulated another IPP similar to the previous one.

\[
\begin{align*}
\min_{m \leq \bar{m}} \quad & f(m, M_0, \text{Pre}, \text{Post}) \\
\text{s.t.} \quad & G(m, T, \mathcal{L}).
\end{align*}
\]
Optimizing the number of places

We have also solved the problem where the number $m$ of the places is not given, but it is only known to be less or equal to a given value $\bar{m}$.

For this problem we have formulated another IPP similar to the previous one.

\[
\begin{align*}
\min_{m \leq \bar{m}} & \quad f(m, M_0, Pre, Post) \\
\text{s.t.} & \quad G(m, T, \mathcal{L}).
\end{align*}
\]
Example

Let us assume we are given the language

$$\mathcal{L} = \{ \varepsilon, t_1, t_1t_2, t_1t_3, t_1t_2t_1, t_1t_2t_3, t_1t_3t_1, t_1t_3t_2 \}$$

thus $k = 3$.

We assume that the total number of places is bounded by $\bar{m} = 5$. The sets of enabling/disabling constraints are

$$\mathcal{E} = \{ (\varepsilon, t_1), (t_1, t_2), (t_1, t_3), (t_1t_2, t_1), (t_1t_2, t_3), (t_1t_3, t_1), (t_1t_3, t_2) \}$$

$$\mathcal{D} = \{ (\varepsilon, t_2), (\varepsilon, t_3), (t_1, t_1), (t_1t_2, t_2), (t_1t_3, t_3) \}.$$
The procedure identifies a net system with $m = 3$ and

\[
\begin{align*}
\text{Pre} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},
\quad \text{Post} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix},
\quad M_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},
\end{align*}
\]
Outline

- Background of Petri nets
- Motivation
- Identification a *free-labeled* Petri net
- Extensions of the procedure
- Numerical Simulations
- Identification a *labeled* Petri net
- Complexity of the identification procedure
- Coverability graph
In this section we investigate the computational complexity of solving Petri nets identification problems starting from the language generated.

In particular, we investigate how the computational time is depended on the cardinality of the set of finite length strings that describe the language, and the chosen performance index.
In this section we investigate the computational complexity of solving Petri nets identification problems starting from the language generated.

In particular, we investigate how the computational time is depended on the cardinality of the set of finite length strings that describe the language, and the chosen performance index.
Numerical simulations: Why?

In this section we investigate the computational complexity of solving Petri nets identification problems starting from the language generated.

In particular, we investigate how the computational time is depended on the cardinality of the set of finite length strings that describe the language, and the chosen performance index.
The sender-receiver process

\[ \mathcal{L} = L_k(N, M_0) \text{ with } k = 2q \]

Minimal T-invariant \( \vec{y} = \vec{1}_n \)
The sender-receiver process with $q=2$

$\mathcal{L} = \varepsilon, t_1, t_1t_3, t_1t_3t_4, t_1t_3t_4t_2$

Minimal $T$-invariant $\vec{y} = [1 \ 1 \ 1 \ 1]^T$
Numerical simulations

The sender-receiver process with $q=2$

\[ \mathcal{L} = \varepsilon, t_1, t_1 t_3, t_1 t_3 t_4, t_1 t_3 t_4 t_2 \]

Minimal T-invariant $\vec{y} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$
The sender-receiver process with $q=2$

$$\mathcal{L} = \varepsilon, t_1, t_1 t_3, t_1 t_3 t_4, t_1 t_3 t_4 t_2$$

Minimal $T$-invariant $\vec{y} = [1 \ 1 \ 1 \ 1]^T$
The sender-receiver process with $q=2$

\[ \mathcal{L} = \varepsilon, t_1, t_1 t_3, t_1 t_3 t_4, t_1 t_3 t_4 t_2 \]

Minimal T-invariant $\vec{y} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$
Numerical simulations

The sender-receiver process with $q=2$

\[
\mathcal{L} = \varepsilon, t_1, t_1 t_3, t_1 t_3 t_4, t_1 t_3 t_4 t_2
\]

Minimal $T$-invariant $\vec{y} = [1 \ 1 \ 1 \ 1 \ 1]^T$
Now I present the results of various identification problems carried out considering different values of $q$, namely $q = 2, \ldots, 6$.

In particular, our goal here is to synthesize a net system that generates the same language of the net system in Figure.
Numerical simulations

The sender-receiver process

For any $q$ we considered two different cases. We assume:

(C1) $J = \bar{K} \cdot \vec{1}^T_m \bar{z} + \vec{1}^T_m \cdot M_0 + \vec{1}^T_m \cdot (\text{Pre} + \text{Post}) \cdot \vec{1}_n$,

(C2) $J = \vec{1}^T_m \cdot M_0$. 
Numerical simulations

The sender-receiver process

We carried out all numerical simulations using an appropriate tool we developed in MATLAB. Given:

- the language \( \mathcal{L} = L_k(N, M_0) \),
- the structural constraints,
- an upper bound on the number of places \( \bar{m} \)

it generates the set of constraints of the IPP in the syntax of CPLEX.

Then, given the desired performance index, the resulting IPP can be directly solved using ILOG CPLEX.

For both cases C1 and C2, and for any value of \( q \), we limit the computational time to one hour. This constraint did not allow us to obtain the optimal solution in all cases examined.

Simulations have been run on a PC Athlon 64, 4000+ processor.
Numerical results are summarized in this table:

<table>
<thead>
<tr>
<th></th>
<th>First admissible solution</th>
<th>Optimal solution (max 1 hour)</th>
<th>% after 1 hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>q=2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>0,03 sec (141)</td>
<td>0,03 sec (141)</td>
<td>0%</td>
</tr>
<tr>
<td>C2</td>
<td>&lt;0,01 sec (113)</td>
<td>&lt;0,01 sec (113)</td>
<td>0%</td>
</tr>
<tr>
<td>q=3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>&lt;4 sec (598)</td>
<td></td>
<td>16,71%</td>
</tr>
<tr>
<td>C2</td>
<td>&lt;0,6 sec (607)</td>
<td>0,78 sec (2744)</td>
<td>0%</td>
</tr>
<tr>
<td>q=4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>&lt;29 sec (1706)</td>
<td></td>
<td>59,91%</td>
</tr>
<tr>
<td>C2</td>
<td>&lt;8 sec (2479)</td>
<td></td>
<td>50,00%</td>
</tr>
<tr>
<td>q=5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>&lt;200 sec (3855)</td>
<td></td>
<td>75,05%</td>
</tr>
<tr>
<td>C2</td>
<td>&lt;20 sec (3625)</td>
<td></td>
<td>75,00%</td>
</tr>
<tr>
<td>q=6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>&lt;65 sec (6949)</td>
<td></td>
<td>97,61%</td>
</tr>
<tr>
<td>C2</td>
<td>&lt;20 sec (6067)</td>
<td></td>
<td>50,00%</td>
</tr>
</tbody>
</table>
Considerations

- The optimal solution can be computed within an hour only for $q = 2$ (both in case C1 and C2), and for $q = 3$ (in case C2).

- In all the other cases the number of constraints was too high, and regardless of the considered performance index, one hour was not enough to determine the optimal solution.
Considerations

- The optimal solution can be computed within an hour only for $q = 2$ (both in case C1 and C2), and for $q = 3$ (in case C2).

- In all the other cases the number of constraints was too high, and regardless of the considered performance index, one hour was not enough to determine the optimal solution.
Numerical simulations

Considerations

<table>
<thead>
<tr>
<th>q</th>
<th>First admissible solution</th>
<th>Optimal solution (max 1 hour)</th>
<th>% after 1 hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>C1 0,03 sec (141)</td>
<td>0,03 sec (141)</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>C2 &lt;0,01 sec (113)</td>
<td>&lt;0,01 sec (113)</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>C1 &lt; 4 sec (598)</td>
<td></td>
<td>16,71%</td>
</tr>
<tr>
<td></td>
<td>C2 &lt;0,6 sec (607)</td>
<td>0,78 sec (2744)</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>C1 &lt; 29 sec (1706)</td>
<td></td>
<td>59,91%</td>
</tr>
<tr>
<td></td>
<td>C2 &lt;8 sec (2479)</td>
<td></td>
<td>50,00%</td>
</tr>
<tr>
<td>5</td>
<td>C1 &lt;200 sec (3855)</td>
<td></td>
<td>75,05%</td>
</tr>
<tr>
<td></td>
<td>C2 &lt;20 sec (3625)</td>
<td></td>
<td>75,00%</td>
</tr>
<tr>
<td>6</td>
<td>C1 &lt; 65 sec (6949)</td>
<td></td>
<td>97,61%</td>
</tr>
<tr>
<td></td>
<td>C2 &lt;20 sec (6067)</td>
<td></td>
<td>50,00%</td>
</tr>
</tbody>
</table>

- the distance from the optimal solution in quite all cases examined also depend on the considered performance index (case C1 or C2).
Considerations

- However this fact is not a serious limitation of our procedure because in general, when computing an identification problem, we are mainly interested in determining an admissible solution, rather than an optimal one.
As it can be seen by looking at the first column of the table, the computational times are very very short also for large values of $q$ if we consider an arbitrary solution, e.g., the first admissible one computed by CPLEX when solving an optimization problem.
Outline

- Background of Petri nets
- Motivation
- Identification a free-labeled Petri net
- Extensions of the procedure
- Numerical Simulations
- Identification a labeled Petri net
- Complexity of the identification procedure
- Coverability graph
Labeled Petri net

No transition is labeled with $\varepsilon$

The same label $e \in E$ may be associated to more than one transition.

We denote

$$T_e = \{ t \in T \mid \varphi(t) = e \} = \{ t^e_1, \ldots, t^e_{n_e} \}, \quad e \in E$$

where $n_e = |T_e|$, the set of transitions with label $e$.

We assume that the total number of transitions $T_e$ sharing the same label $e \in E$ is known.
Labeled Petri net

No transition is labeled with $\varepsilon$

The same label $e \in E$ may be associated to more than one transition.

We denote

$$T_e = \{ t \in T \mid \varphi(t) = e \} = \{ t_1^e, \ldots, t_{n_e}^e \}, \quad e \in E$$

where $n_e = |T_e|$, the set of transitions with label $e$.

We assume that the total number of transitions $T_e$ sharing the same label $e \in E$ is known.
Labeled Petri net

No transition is labeled with $\varepsilon$

The same label $e \in E$ may be associated to more than one transition.

We denote

$$T_e = \{ t \in T \mid \varphi(t) = e \} = \{ t^e_1, \ldots, t^e_{n_e} \}, \quad e \in E$$

where $n_e = \| T_e \|$, the set of transitions with label $e$.

We assume that the total number of transitions $T_e$ sharing the same label $e \in E$ is known.
No transition is labeled with $\varepsilon$

The same label $e \in E$ may be associated to more than one transition.

We denote

$$T_e = \{ t \in T \mid \varphi(t) = e \} = \{ t_1^e, \ldots, t_{n_e}^e \}, \quad e \in E$$

where $n_e = |T_e|$, the set of transitions with label $e$.

We assume that the total number of transitions $T_e$ sharing the same label $e \in E$ is known.
**Problem 3.** Given a set of places $P = \{p_1, \ldots, p_m\}$ and a set of transitions $T = \{t_1, \ldots, t_n\}$. Let

$$T = \bigcup_{e \in E} T_e$$

and $\varphi : T \to E$ be a labeling function over $E$.

Let $\mathcal{L} \subset E^*$ be a given finite prefix-closed language over $E^*$, and

$$k = \max_{w \in \mathcal{L}} |w|$$

be the length of the longest word in $\mathcal{L}$. 
Identification problem 3

We want to identify the structure of a deterministic\(^1\) net \(N = (P, T, Pre, Post)\) labeled by \(\varphi\) and an initial marking \(M_0\) such that

\[
L^E_k(N, M_0) = \mathcal{L}.
\]

The unknowns we want to determine are the elements of the two matrices \(\text{Pre} = \{e_{i,j}\} \in \mathbb{N}^{m \times n}\) and \(\text{Post} = \{o_{i,j}\} \in \mathbb{N}^{m \times n}\) and the elements of the vector \(M_0 = \left[ m_{0,1} \ m_{0,2} \ \cdots \ m_{0,m} \right]^T \in \mathbb{N}^m\).

\(^1\)Determinism is a desirable property and we assume that net enjoys it. However, it may also possible to solve this problem without assuming that the net be deterministic.
We want to identify the structure of a deterministic\textsuperscript{1} net $N = (P, T, Pre, Post)$ labeled by $\varphi$ and an initial marking $M_0$ such that

$$L^E_k(N, M_0) = \mathcal{L}.$$ 

The unknowns we want to determine are the elements of the two matrices 

$Pre = \{ e_{i,j} \} \in \mathbb{N}^{m \times n}$ and $Post = \{ o_{i,j} \} \in \mathbb{N}^{m \times n}$ and the elements of the vector $M_0 = \left[ \begin{array}{c} m_{0,1} \ m_{0,2} \ \cdots \ m_{0,m} \end{array} \right]^T \in \mathbb{N}^m$.

\textsuperscript{1}Determinism is a desirable property and we assume that net enjoys it. However, it may also possible to solve this problem without assuming that the net be deterministic.
Identification problem 3

Let us define the set of enabling conditions

$$\mathcal{E} = \{(w, e) \mid w \in \mathcal{L}, |w| < k, we \in \mathcal{L}\},$$

and the set of disabling conditions

$$\mathcal{D} = \{(w, e) \mid w \in \mathcal{L}, |w| < k, we \notin \mathcal{L}\}.$$
Identification problem 3

Let us define the set of **enabling conditions**

$$\mathcal{E} = \{(w, e) \mid w \in \mathcal{L}, |w| < k, we \in \mathcal{L}\},$$

and the set of **disabling conditions**

$$\mathcal{D} = \{(w, e) \mid w \in \mathcal{L}, |w| < k, we \notin \mathcal{L}\}.$$ 

The marking reached after observing the sequence $w$ is denoted $M_w$ and in particular $M_\varepsilon = M_0$, being $\varepsilon$ the empty string.
Identification of deterministic Petri nets

Identification problem 3 (cont’d)

**Theorem 3.** Solving Problem 3 is equivalent to find an admissible solution of the following IPP denoted $G(m, T, L, \varphi)$.

\[
\forall (w, e) \in \mathcal{E} \begin{cases} 
M_w - \text{Pre}(\cdot, t_1^e) \geq -z_1^{e,w} \cdot \vec{K} \\
\quad \vdots \\
M_w - \text{Pre}(\cdot, t_{n_e}^e) \geq -z_{n_e}^{e,w} \cdot \vec{K} \\
M_{we} - M_w - \text{Post}(\cdot, t_1^e) + \text{Pre}(\cdot, t_1^e) \leq z_1^{e,w} \cdot \vec{K} \\
\quad \vdots \\
M_{we} - M_w - \text{Post}(\cdot, t_{n_e}^e) + \text{Pre}(\cdot, t_{n_e}^e) \geq -z_{n_e}^{e,w} \cdot \vec{K} \\
\sum_{e=1}^{n_e} z_e^{e,w} + \ldots + z_{n_e}^{e,w} = n_e - 1 \\
\sum_{e=1}^{n_e} z_e^{e,w} \in \{0, 1\} \\
\quad \vdots 
\end{cases}
\]

(At least one transition with label $e$ is enabled after $w$) and fires
Theorem 3. Solving Problem 3 is equivalent to find an admissible solution of the following IPP denoted $\mathcal{G}(m, T, \mathcal{L}, \varphi)$.

\[
\forall (w, e) \in \mathcal{E} : |T_e| > 1, \forall t^e_j \in T_e \left\{ \begin{array}{l}
\ldots \\
-K\bar{S}(w, t^e_j) + M_w - \text{Pre}(\cdot, t^e_j) \leq -\mathbf{1} \\
\bar{1}^T\bar{S}(w, t^e_j) \leq m - z_{j}^{e,w} \\
\bar{S}(w, t^e_j) \in \{0, 1\}^m \\
\ldots \end{array} \right.
\]

(The only transition with label $e$ enabled after $w$ is the one that fires)
Identification problem 3 (cont’d)

**Theorem 3.** Solving Problem 3 is equivalent to find an admissible solution of the following IPP denoted $G(m, T, L, \varphi)$.

\[
\forall (w, e) \in D, \ \forall t_j^e \in T_e \quad \begin{cases} 
\ldots \\
- KS(w, t_j^e) + M_w - Pre(\cdot, t_j^e) \leq -\mathbf{1} \\
\mathbf{1}^T S(w, t_j^e) \leq m - 1 \\
S(w, t_j^e) \in \{0, 1\}^m
\end{cases}
\]

(All transitions with label $e$ are disabled after $w$)
Example

Let $m = n = 3$, $L(t_1) = a$, $L(t_2) = L(t_3) = b$ and the net language is $\mathcal{L}' = \{a^r b^q, \ r \geq q \geq 0\}$. Assume we want to minimize the sum of initial tokens and the sum of all arcs.

For $k = 3$, the language we consider is $\mathcal{L} = \{\varepsilon, a, aa, ab, aaa, aab\}$. This implies that

$$\mathcal{E} = \{ (\varepsilon, a), (a, a), (a, b), (aa, a), (aa, b) \}, \quad \mathcal{D} = \{ (\varepsilon, b), (ab, a), (ab, b) \}. $$

The resulting net system is

![Net Diagram](attachment:image.png)
Example (cont’d)

For \( k = 4 \), the language is \( \mathcal{L} = \{ \varepsilon, a, aa, ab, aaa, aab, aaaa, aaab, aabb \} \). This implies that

\[
\mathcal{E} = \{ (\varepsilon, a), (a, a), (a, b), (aa, a), (aa, b), (aaa, a), (aaa, b), (aab, b) \},
\]

\[
\mathcal{D} = \{ (\varepsilon, b), (ab, a), (ab, b), (aab, a) \}.
\]

The resulting net system is

![Net System Diagram](image)

The same net system is also obtained if \( k = 5 \).
For $k \geq 5$ we obtain the net in figure whose language is exactly $\mathcal{L}' = \{a^r b^q, \ r \geq q \geq 0\}$.
Outline

- Background of Petri nets
- Motivation
- Identification of a *free-labeled* Petri net
- Extensions of the procedure
- Numerical Simulations
- Identification of a *labeled* Petri net
- **Complexity of the identification procedure**
- Coverability graph
Computational complexity free-labeled PN

The number of unknown depends on the length $k$ of the longest string in $\mathcal{L}$, and on the number of strings in $\mathcal{L}$ of length $r$ (for $r = 0, \ldots, k$) that we denote $\nu_r$.

The number of unknowns is

$$u_{\text{free}} = m + 2(m \times n) + m \left( \sum_{r=0}^{k-1} (n\nu_r - \nu_{r+1}) \right).$$

where each term corresponds, respectively, to: $M_0$; $Pre$ and $Post$; the binary vectors $S_{\sigma,j}$.

In the worst case it is exponential:

$$\mathcal{O}(mn^k).$$
The number of unknowns is

\[ u_{\text{free}} = m + 2(m \times n) + m \left( \sum_{r=0}^{k-1} (n\nu_r - \nu_{r+1}) \right) . \]

where each term corresponds, respectively, to: \( M_0 \); \( Pre \) and \( Post \); the binary vectors \( S_{\sigma,j} \).

In the worst case it is exponential:

\[ \mathcal{O}(mn^k) . \]
The number of unknowns depends on the length $k$ of the longest string in $\mathcal{L}$, and on the number of strings in $\mathcal{L}$ of length $r$ (for $r = 0, \ldots, k$) that we denote $\nu_r$.

The number of unknowns is

$$u_{\text{free}} = m + 2(m \times n) + m \left( \sum_{r=0}^{k-1} (n\nu_r - \nu_{r+1}) \right).$$

where each term corresponds, respectively, to: $M_0$; $Pre$ and $Post$; the binary vectors $S_{\sigma,j}$.

In the worst case it is exponential:

$$O(mn^k).$$
The number of unknowns depends on the length $k$ of the longest string in $\mathcal{L}$, and on the number of strings in $\mathcal{L}$ of length $r$ (for $r = 0, \ldots, k$) that we denote $\nu_r$.

The number of unknowns is

$$u_{\text{free}} = m + 2(m \times n) + m \left( \sum_{r=0}^{k-1} (n\nu_r - \nu_{r+1}) \right).$$

where each term corresponds, respectively, to: $M_0$; $Pre$ and $Post$; the binary vectors $S_{\sigma,j}$.

In the worst case it is exponential:

$$\mathcal{O}(mn^k).$$
The number of unknowns now depends on $\tau = \max_{e \in E} |T_e|$, on the length $k$ of the longest string in $\mathcal{L}$, and on the number of strings in $\mathcal{L}$ of length $r$ (for $r = 0, \ldots, k$) that we denote $\nu_r$.

The number of unknowns is

$$u_{\lambda\text{-free}} = m + 2mn + m \left( \sum_{r=1}^{k} \nu_r \right) + \tau \left( \sum_{r=1}^{k} \nu_r \right) + m \tau \left( \sum_{r=1}^{k} \nu_r \right) + m \tau \left( \sum_{r=0}^{k-1} (n \nu_r - \nu_{r+1}) \right)$$

where each term corresponds, respectively, to: $M_0$; $Pre$ and $Post$; $M_w$; the binary variables $z_{j,w}^e$; the binary vectors $\bar{S}(w, t_j^e)$; the binary vectors $S(w, t_j^e)$.

In the worst case it is exponential:

$$\mathcal{O}(m n^{k+1}).$$
Computational complexity labeled PN

The number of unknown now depends on $\tau = \max_{e \in E} |T_e|$, on the length $k$ of the longest string in $L$, and on the number of strings in $L$ of length $r$ (for $r = 0, \ldots, k$) that we denote $\nu_r$.

The number of unknowns is

$$u_{\lambda\text{-free}} = m + 2mn + m \left( \sum_{r=1}^{k} \nu_r \right) + \tau \left( \sum_{r=1}^{k} \nu_r \right) + m\tau \left( \sum_{r=1}^{k} \nu_r \right) + m\tau \left( \sum_{r=0}^{k-1} (n\nu_r - \nu_{r+1}) \right)$$

where each term corresponds, respectively, to: $M_0$; $Pre$ and $Post$; $M_w$; the binary variables $z_{j}^{e,w}$; the binary vectors $\bar{S}(w, t_j^e)$; the binary vectors $S(w, t_j^e)$.

In the worst case it is exponential:

$$\mathcal{O}(mn^{k+1}).$$
The number of unknown now depends on $\tau = \max_{e \in E} |T_e|$, on the length $k$ of the longest string in $L$, and on the number of strings in $L$ of length $r$ (for $r = 0, \ldots, k$) that we denote $\nu_r$.

The number of unknowns is

$$u_{\lambda\text{-free}} = m + 2mn + m \left( \sum_{r=1}^{k} \nu_r \right) + \tau \left( \sum_{r=1}^{k} \nu_r \right) + m\tau \left( \sum_{r=1}^{k} \nu_r \right) + m\tau \left( \sum_{r=0}^{k-1} (n\nu_r - \nu_{r+1}) \right)$$

where each term corresponds, respectively, to: $M_0$; $Pre$ and $Post$; $M_w$; the binary variables $z_{j}^{e,w}$; the binary vectors $\bar{S}(w, t_{j}^{e})$; the binary vectors $S(w, t_{j}^{e})$.

In the worst case it is exponential:

$$O(mn^{k+1}).$$
The number of unknown now depends on $\tau = \max_{e \in E} |T_e|$, on the length $k$ of the longest string in $L$, and on the number of strings in $L$ of length $r$ (for $r = 0, \ldots, k$) that we denote $\nu_r$.

The number of unknowns is

$$u_{\lambda\text{-free}} = m + 2mn + m \left( \sum_{r=1}^{k} \nu_r \right) + \tau \left( \sum_{r=1}^{k} \nu_r \right) + m\tau \left( \sum_{r=1}^{k} \nu_r \right) + m\tau \left( \sum_{r=0}^{k-1} (n\nu_r - \nu_{r+1}) \right)$$

where each term corresponds, respectively, to: $M_0$; $Pre$ and $Post$; $M_w$; the binary variables $z_{j}^{e,w}$; the binary vectors $\bar{S}(w, t_{j}^{e})$; the binary vectors $S(w, t_{j}^{e})$.

In the worst case it is exponential:

$$O(mn^{k+1}).$$
Outline

- Background of Petri nets
- Motivation
- Identification a free-labeled Petri net
- Extensions of the procedure
- Numerical Simulations
- Identification a labeled Petri net
- Complexity of the identification procedure
- Coverability graph
A Motivating example

(a): unbounded queue

(b): (infinite) reachability graph

(c): coverability graph

$t_1$: arrival of the customers

$t_2$: departure of the customers after service
Motivations

Generally, in an identification problem the information about the language of the net to identify is given by an automaton, that generates the language (theory of regions).

If the net is bounded the input is the reachability graph.

If the net is unbounded we suppose that the input of the identification problem is the coverability graph (CG).

CG is an automaton that generates a language that is a superset of the net language, and that contains structural information on the net that goes beyond the language, i.e., $\omega$-increasing and $\omega$-stationary productions.
Motivations

Generally, in a identification problem the information about the language of the net to identify is given by an automaton, that generates the language (theory of regions).

If the net is bounded the input is the reachability graph.

If the net is unbounded we suppose that the input of the identification problem is the coverability graph (CG).

CG is an automaton that generates a language that is a superset of the net language, and that contains structural information on the net that goes beyond the language, i.e., $\omega$-increasing and $\omega$-stationary productions.
Motivations

Generally, in a identification problem the information about the language of the net to identify is given by an automaton, that generates the language (theory of regions).

If the net is **bounded** the input is the reachability graph.

If the net is **unbounded** we suppose that the input of the identification problem is the coverability graph (CG).

CG is an automaton that generates a language that is a superset of the net language, and that contains structural information on the net that goes beyond the language, i.e., $\omega$-increasing and $\omega$-stationary productions.
Motivations

Generally, in an identification problem the information about the language of the net to identify is given by an automaton, that generates the language (theory of regions).

If the net is **bounded** the input is the **reachability graph**.

If the net is **unbounded** we suppose that the input of the identification problem is the **coverability graph** (CG).

CG is an automaton that generates a language that is a superset of the net language, and that contains structural information on the net that goes beyond the language, i.e., $\omega$-increasing and $\omega$-stationary productions.
Motivations

Generally, in a identification problem the information about the language of the net to identify is given by an automaton, that generates the language (theory of regions).

If the net is bounded the input is the reachability graph.

If the net is unbounded we suppose that the input of the identification problem is the coverability graph (CG).

CG is an automaton that generates a language that is a superset of the net language, and that contains structural information on the net that goes beyond the language, i.e., $\omega$-increasing and $\omega$-stationary productions.
Repetitive sequences

A sequence $\sigma \in T^*$ is called repetitive if there exists a marking $M \in R(N, M_0)$ such that

$$M_1[\sigma] \prec M_2[\sigma] \prec M_3[\sigma] \ldots$$

(2)

i.e., if it can fire infinitely often starting from $M_1$. It is possible to distinguish two different types of repetitive sequences:

- **stationary** sequence: if in (2) it holds $M_i = M_{i+1}$ for all $i = 1, 2, \ldots$
- **increasing** sequence: if in (2) it holds $M_i \preceq M_{i+1}$ for all $i = 1, 2, \ldots$
The sequence $\sigma = t_1 t_2$ is repetitive, in particular it is an increasing sequence.

$$M_0 = [1 \ 0 \ 0]^T$$

$$M_0 [t_1 t_2] M_2$$
The sequence $\sigma = t_1 t_2$ is repetitive, in particular it is an increasing sequence.

\[ M_1 = [0 \ 1 \ 0]^T \]

\[ M_0[t_1] M_1[t_2] M_2 \]
The sequence $\sigma = t_1 t_2$ is repetitive, in particular it is an increasing sequence.

$$M_2 = [1 \ 0 \ 1]^T$$

$$M_0[t_1 t_2] M_2$$
In a CG $G$ of a Petri net a node $q'$ is \( \omega \)-increasing and $\pi$ is the corresponding \( \omega \)-increasing production if and only if there exists in the graph a node $\bar{q} \neq q'$ such that:

\[
 s(\pi) = \bar{q}, \quad e(\pi) = q'
\]

and there exists another production $\pi'$ such that:

\[
 s(\pi') = e(\pi') = q', \quad \ell(\pi) = \ell(\pi') = \sigma, \quad \nu(\pi) \cap \nu(\pi') = q'.
\]

- $\nu(\pi)$: the set of all nodes of the production
- $s(\pi) = \bar{q}$: the start node of the production
- $\ell(\pi)$: the sequence of the production
- $e(\pi) = \bar{q}$: the end node
In a CG $G$ of a Petri net, a production $\pi$ with $s(\pi) = q$ corresponds to a sequence $\omega$-stationary iff

$$e(\pi) = q.$$ 

$s(\pi) = q$: the start node of the production
$e(\pi) = q$: the end node
Example: Petri net and coverability graph

Consider the net in figure (a) and its CG in (b).

\[ \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \]

\[ \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \]

\[ \begin{bmatrix} 1 & 0 & \omega \end{bmatrix} \]

\[ \begin{bmatrix} 0 & 1 & \omega \end{bmatrix} \]

\[ \begin{bmatrix} \omega & \omega & \omega \end{bmatrix} \]

\[ \begin{bmatrix} \omega & \omega & \omega \end{bmatrix} \]
Consider the net in figure (a) and its CG in (b).

Example: Petri net and coverability graph
Example: $\omega$-increasing productions

For example the node $q_2$ is $\omega$-increasing, and $\pi = t_1 t_2$ is the correspondent $\omega$-production in fact:

$s(\pi) = q_0$, $e(\pi) = q_2$, $\nu(\pi) = \{q_0, q_1, q_2\}$

and $\exists s(\pi') = e(\pi') = q_2$, $\ell(\pi) = \ell(\pi') = t_1 t_2$,

$\nu(\pi') = \{q_2, q_3\} \cap \nu(\pi) = q_2$. 

Maria Paola Cabasino (DIEE, University of Cagliari)
The \( \omega \)-stationary productions are associated to the node \( q_6 \) and the firing vectors are \( S = \{ \bar{\sigma}', \bar{\sigma}'', \bar{\sigma}''' \} \) where

\[
\sigma' = t_1, \sigma'' = t_3 \text{ and } \sigma''' = t_4.
\]
Identification problem 4

Problem 4: Let $G = (Q, T, \delta, q_0)$ be a given finite state automaton. Chosen a set of places $P$ of cardinality $m$, we want to identify the structure of a free-labeled Petri net $N = (P, T, Pre, Post)$ and an initial marking $M_0$ such that the CG of $\langle N, M_0 \rangle$ is isomorphic to $G$.

The unknowns we want to determine are the elements of the two matrices $Pre = \{e_{i,j}\} \in \mathbb{N}^{m \times n}$ and $Post = \{o_{i,j}\} \in \mathbb{N}^{m \times n}$ and the elements of the vector $M_0 = \begin{bmatrix} m_{0,1} & m_{0,2} & \cdots & m_{0,m} \end{bmatrix}^T \in \mathbb{N}^m$. 
Identification problem 4

Problem 4: Let $\mathcal{G} = (Q, T, \delta, q_0)$ be a given finite state automaton. Chosen a set of places $P$ of cardinality $m$, we want to identify the structure of a free-labeled Petri net $N = (P, T, Pre, Post)$ and an initial marking $M_0$ such that the CG of $\langle N, M_0 \rangle$ is isomorphic to $\mathcal{G}$.

The unknowns we want to determine are the elements of the two matrices $Pre = \{e_{i,j}\} \in \mathbb{N}^{m \times n}$ and $Post = \{o_{i,j}\} \in \mathbb{N}^{m \times n}$ and the elements of the vector $M_0 = \begin{bmatrix} m_{0,1} & m_{0,2} & \cdots & m_{0,m} \end{bmatrix}^T \in \mathbb{N}^m$. 
We proved that a solution to the previous identification problem can be computed thanks to the following theorem.

**Theorem 4:** A solution, to the identification problem presented, satisfies the following linear algebraic constraints:

- Enabling constraints (a)
- Constraints related to $\omega$-increasing sequences (b)
- Constraints related to $\omega$-stationary sequences (c)
- Blocking constraints (d)
- Equivalence constraints (e)
- Discriminating constraints (f)
- Integrity constraints (g)
We proved that a solution to the previous identification problem can be computed thanks to the following theorem.

Theorem 4: A solution, to the identification problem presented, satisfies the following linear algebraic constraints:

- Enabling constraints (a)
- Constraints related to $\omega$-increasing sequences (b)
- Constraints related to $\omega$-stationary sequences (c)
- Blocking constraints (d)
- Equivalence constraints (e)
- Discriminating constraints (f)
- Integrity constraints (g)
We proved that a solution to the previous identification problem can be computed thanks to the following theorem.

**Theorem 4:** A solution, to the identification problem presented, satisfies the following linear algebraic constraints:

- Enabling constraints
- Constraints related to $\omega$-increasing sequences
- Constraints related to $\omega$-stationary sequences
- Blocking constraints
- Equivalence constraints
- Discriminating constraints
- Integrity constraints
Identification problem 4

We proved that a solution to the previous identification problem can be computed thanks to the following theorem.

**Theorem 4:** A solution, to the identification problem presented, satisfies the following linear algebraic constraints:

- **Enabling constraints**
- Constraints related to $\omega$-increasing sequences
- Constraints related to $\omega$-stationary sequences
- Blocking constraints
- Equivalence constraints
- Discriminating constraints
- Integrity constraints
We proved that a solution to the previous identification problem can be computed thanks to the following theorem.

**Theorem 4:** A solution, to the identification problem presented, satisfies the following linear algebraic constraints:

- Enabling constraints \((a)\)
- Constraints related to \(\omega\)-increasing sequences \((b)\)
- Constraints related to \(\omega\)-stationary sequences \((c)\)
- Blocking constraints \((d)\)
- Equivalence constraints \((e)\)
- Discriminating constraints \((f)\)
- Integrity constraints \((g)\)
Identification problem 4

We proved that a solution to the previous identification problem can be computed thanks to the following theorem.

**Theorem 4:** A solution, to the identification problem presented, satisfies the following linear algebraic constraints:

- Enabling constraints (a)
- Constraints related to $\omega$-increasing sequences (b)
- Constraints related to $\omega$-stationary sequences (c)
- Blocking constraints (d)
- Equivalence constraints (e)
- Discriminating constraints (f)
- Integrity constraints (g)
We proved that a solution to the previous identification problem can be computed thanks to the following theorem.

**Theorem 4**: A solution, to the identification problem presented, satisfies the following linear algebraic constraints:

- Enabling constraints (a)
- Constraints related to $\omega$-increasing sequences (b)
- Constraints related to $\omega$-stationary sequences (c)
- Blocking constraints (d)
- Equivalence constraints (e)
- Discriminating constraints (f)
- Integrity constraints (g)
Identification problem 4

We proved that a solution to the previous identification problem can be computed thanks to the following theorem.

**Theorem 4:** A solution, to the identification problem presented, satisfies the following linear algebraic constraints:

- Enabling constraints
- Constraints related to $\omega$-increasing sequences
- Constraints related to $\omega$-stationary sequences
- Blocking constraints
- Equivalence constraints
- Discriminating constraints
- Integrity constraints
We proved that a solution to the previous identification problem can be computed thanks to the following theorem.

**Theorem 4:** A solution, to the identification problem presented, satisfies the following linear algebraic constraints:

- **Enabling constraints** (a)
- Constraints related to $\omega$-increasing sequences (b)
- Constraints related to $\omega$-stationary sequences (c)
- Blocking constraints (d)
- Equivalence constraints (e)
- Discriminating constraints (f)
- Integrity constraints (g)
Identification problem 4

We proved that a solution to the previous identification problem can be computed thanks to the following theorem.

**Theorem 4:** A solution, to the identification problem presented, satisfies the following linear algebraic constraints:

- Enabling constraints (a)
- Constraints related to $\omega$-increasing sequences (b)
- Constraints related to $\omega$-stationary sequences (c)
- Blocking constraints (d)
- Equivalence constraints (e)
- Discriminating constraints (f)
- Integrity constraints (g)
In this talk we have provided a solution to the problem of identifying a Petri net system that generates a given language, that is based on the solution of IPPs.

Both the case of free labeled Petri net systems and the case of $\lambda$-free labeled Petri nets are considered.

Furthermore we have also considered the problem of synthesizing a net when additional information about the model (structural constraints, conservative components, stationary sequences) or about its initial marking is given.

We also treated the problem of synthesizing a bounded net starting from an automaton that generates its language.
Conclusion and future work

In this talk we have provided a solution to the problem of identifying a Petri net system that generates a given language, that is based on the solution of IPPs.

Both the case of free labeled Petri net systems and the case of $\lambda$-free labeled Petri nets are considered.

Furthermore we have also considered the problem of synthesizing a net when additional information about the model (structural constraints, conservative components, stationary sequences) or about its initial marking is given.

We also treated the problem of synthesizing a bounded net starting from an automaton that generates its language.
Conclusion and future work

In this talk we have provided a solution to the problem of identifying a Petri net system that generates a given language, that is based on the solution of IPPs.

Both the case of free labeled Petri net systems and the case of $\lambda$-free labeled Petri nets are considered.

Furthermore we have also considered the problem of synthesizing a net when additional information about the model (structural constraints, conservative components, stationary sequences) or about its initial marking is given.

We also treated the problem of synthesizing a bounded net starting from an automaton that generates its language.
Conclusion and future work

In this talk we have provided a solution to the problem of identifying a Petri net system that generates a given language, that is based on the solution of IPPs.

Both the case of free labeled Petri net systems and the case of λ-free labeled Petri nets are considered.

Furthermore we have also considered the problem of synthesizing a net when additional information about the model (structural constraints, conservative components, stationary sequences) or about its initial marking is given.

We also treated the problem of synthesizing a bounded net starting from an automaton that generates its language.
Conclusion and future work

In this talk we have provided a solution to the problem of identifying a Petri net system that generates a given language, that is based on the solution of IPPs.

Both the case of free labeled Petri net systems and the case of $\lambda$-free labeled Petri nets are considered.

Furthermore we have also considered the problem of synthesizing a net when additional information about the model (structural constraints, conservative components, stationary sequences) or about its initial marking is given.

We also treated the problem of synthesizing a bounded net starting from an automaton that generates its language.
Our approach is based on integer programming that is a well accepted methodology for optimization of discrete system. However we have shown that the computational complexity of the IPPs that describe the problem highly increases with the number of places $m$, with the number of transition sharing the same label, and with the length $k$.

This problem may be partially overcome using appropriate heuristics, that compute solutions recursively, with increasing values of $k$, but that may only provide suboptimal solutions with respect to the chosen performance index.

This problem will be the object of our future work in this topic.
Our approach is based on integer programming that is a well accepted methodology for optimization of discrete system. However we have shown that the computational complexity of the IPPs that describe the problem highly increases with the number of places $m$, with the number of transition sharing the same label, and with the length $k$.

This problem may be partially overcome using appropriate heuristics, that compute solutions recursively, with increasing values of $k$, but that may only provide suboptimal solutions with respect to the chosen performance index.

This problem will be the object of our future work in this topic.
Conclusion and future work

Our approach is based on integer programming that is a well accepted methodology for optimization of discrete system. However we have shown that the computational complexity of the IPPs that describe the problem highly increases with the number of places $m$, with the number of transition sharing the same label, and with the length $k$.

This problem may be partially overcome using appropriate heuristics, that compute solutions recursively, with increasing values of $k$, but that may only provide suboptimal solutions with respect to the chosen performance index.

This problem will be the object of our future work in this topic.
Conclusion and future work

Our approach is based on integer programming that is a well accepted methodology for optimization of discrete system. However we have shown that the computational complexity of the IPPs that describe the problem highly increases with the number of places $m$, with the number of transition sharing the same label, and with the length $k$.

This problem may be partially overcome using appropriate heuristics, that compute solutions recursively, with increasing values of $k$, but that may only provide suboptimal solutions with respect to the chosen performance index.

This problem will be the object of our future work in this topic.
Conclusion and future work

In this talk we considered also the problem of identifying an unbounded Petri net system given its unlabeled coverability graph.

The solution we propose ensures that the coverability graph of the resulting net system is isomorphic to $G$, and is based on a linear algebraic characterization of the net systems whose coverability graph is isomorphic to $G$.

Therefore, also in this case the identification problem is written in terms of an IPP.
Conclusion and future work

In this talk we considered also the problem of identifying an unbounded Petri net system given its unlabeled coverability graph.

The solution we propose ensures that the coverability graph of the resulting net system is isomorphic to $G$, and is based on a linear algebraic characterization of the net systems whose coverability graph is isomorphic to $G$.

Therefore, also in this case the identification problem is written in terms of an IPP.
Conclusion and future work

In this talk we considered also the problem of identifying an unbounded Petri net system given its unlabeled coverability graph.

The solution we propose ensures that the coverability graph of the resulting net system is isomorphic to $G$, and is based on a linear algebraic characterization of the net systems whose coverability graph is isomorphic to $G$.

Therefore, also in this case the identification problem is written in terms of an IPP.
Conclusion and future work

In this talk we considered also the problem of identifying an unbounded Petri net system given its unlabeled coverability graph.

The solution we propose ensures that the coverability graph of the resulting net system is isomorphic to $G$, and is based on a linear algebraic characterization of the net systems whose coverability graph is isomorphic to $G$.

Therefore, also in this case the identification problem is written in terms of an IPP.


A. Giua and C. Seatzu. **Identification of unbounded Petri nets from their coverability graph.** In *Proc. 45th IEEE Conf. on Decision and Control*, San Diego, CA, USA, December 2006.