Petri Nets – Introduction

A Petri Net (PN) is a weighted(?), bipartite(?) digraph(?) invented by Carl Adam Petri in his PhD-thesis “Kommunikation mit Automaten” (1962). Many flavors of Petri nets are in use; we start with a simple kind. Example:

- Circles $P := \{\text{wait, wash, } \ldots \}$ (set of places)
- Boxes $T := \{\text{insertCoin, lock_door, } \ldots \}$ (set of transitions)
- Arrows $C := \{\ldots (\text{lock_door, wash}), \ldots \}$ (set of edges)
- Weights $W := \text{here constant } 1 \text{ for each arc}$
- Initial marking $M_0 := \{m_0(\text{wait}) := 0, m_0(\text{wash}) := 0, m_0(\text{idle}) := 1\}$

For differing among the different states of a PN we index them accordingly, i.e., we write $\vec{s}_k$ when referring to the k-th state.

Note:

For differing among the different states of a PN we index them accordingly, i.e., we write $\vec{s}_k$ when referring to the k-th state.

Petri Nets – Operational Semantic (Enabling)

The operational (or execution) semantic of PN stem from the movement of tokens in the net:

- Transitions consume tokens from input places (pre set).
- Transitions add tokens to their output places (post set).
- One execute one transition at a time according to a discreet event system semantic.
- Execution of transition is atomic and instantenous (= zero-time).
- Thus concurrency of transition’s execution is resolved by their interleavings (= interleaving semantic).

But when can we actually execute transitions?

Definition 1.2: Pre - and Post sets

- Pre set of a transition $t \in T$: $\text{pre}(t) := \{p | (p, t) \in C\}$
- Post set of a transition $t \in T$: $\text{post}(t) := \{p | (t, p) \in C\}$

analytically we define pre $(\bullet p)$ and post sets $(p \bullet)$ for each place $p \in P$.

Definition 1.1: (Weighted) Petri Net

A Petri net (PN) $P$ is a 5-tuple $(P, T, C, W, M_0)$, where

- $P := \{p_1, \ldots, p_n\}$ is a finite, ordered (indexed) set of places and
- $T := \{t_1, \ldots, t_l\}$ is a finite, ordered (indexed) set of transitions,
- $C \subseteq (P \times T) \cup (T \times P)$ is a connection or flow relation,
- $W : C \mapsto \mathbb{N}_0$ assigns a weight to each element of $C$ and
- $m_0 : p \in P \mapsto \mathbb{N}_0$ gives the initial marking for place $p$, i.e., it assigns a number of token to place $p$. The set of all such initial markings is denoted $M_0 (= \text{the initial marking of } P)$.

An ordering defined on the places yields that the place markings $m_0(p)$ can be understood as the component of a vector $\vec{s}_0 := (s_0(p))$ s.t. we can map the initial marking $M_0$ to the dedicated vector $\vec{s}_0$. Note: $\vec{s}_0$ gives the number of tokens currently contained in place $p$.

A vector of this kind is denoted in the following as state vector, as it uniquely defines the state of the PN we also simply say state.
Petri Nets – Operational Semantic (Firing)

Given a state $s$ of a PN we want to compute its successor state $s'$ w. r. t. an enabled transition. To do so we define a transfer or transition function of a transition $t$ as follows:

$$
\delta(\delta(\delta(\delta(s, t), t), t), t), t) :=
$$

- if $s \cap t$ then $\delta(s, t) := s'$ else undefined.
- With $\delta$ we can construct sets of triples $(s, t, s')$, where we use the notation $s \rightarrow t s'$.

Petri Nets – Reachability set

Moving the tokens around the net by executing enabled transitions is denoted token game; essentially executing $t$.
- Executing transition function $\delta$ in a fixed point iteration, starting with state $s_0$, yields a set of states, denoted as a PN’s set of reachable states or reachability set.

Definition 1.3: Reachability set of a PN

$$
\delta(S_0) := \{s_0\}
$$

$S^i := S^{i-1} \cup \{\delta(s, t) | \forall s \in S^{i-1}, \forall t \in T \text{ where } s \cap t\}$

we are interested in the largest of such sets $S_0 \subseteq S_1 \subseteq \ldots \subseteq S^i$ which we denote as set of reachable states $S$ of a PN $P$ and w. r. t. $s_0$.

Note:

This allows one to construct a (not necessarily finite) LTS for each PN and an initial state $s_0$. Such an LTS constitutes the semantic model of a PN.

Petri Nets – Token game (Reachability of states)

One the basis of the token game we can now pose interesting questions about the properties of a PN and ultimately about the modeled system itself. E.g.:

- Can we reach a state s.t. each place holds at least $N$ but at most $K$ tokens (under- or overflow of buffers in a chip-design)?
- Can we reach a state where everything is blocked?

Such questions are denoted reachability problems, since they can be solved in principle by checking if a respective state can be derived from the initial state $s_0$ by executing the transitions of a PN.

Example 1.1: Washing machine

We formalize this as follows:

Definition 1.5: Reachability problem/question

Given a high-level model which can be mapped to a LTS, e. g. a PN $P$ and an initial (system) state $s_0$, the reachability question answers the question, if it is possible to reach a dedicated state $s_f$ by executing a sequence of transition $(t_1, \ldots, t_k)$.

Formally we are looking for a sequence of state-to-state transitions in the LTS of the high-level model s.t. the reach state $s_f$. With $\sigma := t_1 \cap t_2 \cap \ldots \cap t_k$ one also writes $s_0 \xrightarrow{\sigma} s_f$ for indicating that $s_f$ can be reached by executing transition sequence $\sigma$.

Petri Nets – Reachability question

Recall: A yes/no-question is decidable if and only if there is a computation which after finitely many steps returns with either yes or no.
- A yes/no-question is semi-deciable if and only if the computation may return after finitely many steps with either a yes or no answer .

Is reachability for PN decidable? How would you proceed?

Petri Nets – Algebraic approach (state equations)

- An incidence matrix shows the relationship between two classes of objects, it possesses one row for each element of class $x$ and one column for each element of class $y$.
- For a PN the incidence matrix $A \in \mathbb{N}^{|P| \times |T|}$ describes the token-flow w. r. t. place $i$ (row index) and transition $j$ (column index).

$$
a_{ij} = W(t_j, t_i) - W(t_i, t_j) \text{ (gain of place } i \text{ when transition } j \text{ fires)}
$$

Petri Nets – How-to solve the reachability problem

There are several ways to answer this question, we discuss two semi-decision procedures, e. g. the methods are based on necessary (not sufficient!) conditions.

Algebraic method

Solution of a system of linear equations (works for standard PN only).
- Absence of solution implies non-reachability of the resp. state. In case of a solution we do not know anything.

Algorithmic method (brute-force method)

Generation of the reachability graph and check the states on-the-fly.
- Termination implies reachability of the searched state. If the reachability graph is not finite the method may not terminate.

Example 1.2: Washing machine

$$
A := ? \quad \vec{s}_0 := ?
$$
Let the firing vector $\vec{u}_i \in \mathbb{N}^{T_1}$ be the indicator for the firing of transition $t_i$, i.e., $\vec{u}_i[i] := 1$ and $\vec{u}_i[j] := 0$ for $i \neq j$.

What is the PN semantic of $\vec{x}_i + A \cdot \vec{u}_i$?  
→ Example 1.3: Washing machine

- For a transition sequence $\sigma$ of length $|\sigma|$ one can construct the linear combination of the firing vectors:

$$\vec{T} := \sum_{k=1}^{\sigma} \vec{u}_k$$

where $\sigma[k]$ refers to the $k$-th transition symbol in the sequence.

→ Question 1.1: What is the PN semantic of $\vec{x}_i + A \cdot \vec{T}$?

**Petri Nets – Algebraic approach (state equations)**

This allows one to test for the reachability of state $\vec{s}$ as follows:

- Solve $(A \cdot \vec{f} = \vec{s} - \vec{x}_0)$
- if there is no such solution $\vec{T} \in \mathbb{N}^{T_1}$ then there is no sequence of transition firings leading from $\vec{x}_0$ to $\vec{s}$.

Formally:

$$\{\vec{T} \in \mathbb{N}^{T_1} : A \cdot \vec{f} = \vec{s} - \vec{x}_0\} \Rightarrow \exists \vec{s} : \vec{x}_0 \Rightarrow \vec{s}$$

→ Example 1.4: Washing machine

→ Question 1.2: If there is a solution, what do we know, what can be the problem?

**Deadlock-freeness**

One of the most basic properties related to the reachability is the question about the reachability of deadlocks.

**Definition 1.6: Deadlock**

A state $\vec{s} \in S$ of a PN $P$ is denoted deadlock iff

$$\exists t \in T : \vec{s} \not\rightarrow t$$

- A PN $P$ where no deadlock exists is denoted deadlock-free.
- A PN $P$ whose reachability graph is non-terminal is deadlock-free (or vice-versa)

→ Example 1.7: Dining philosophers

**K-boundedness**

**Definition 1.7: K-Boundedness**

- A place $p_j$ of a PN $P$ with its initial state $\vec{x}_0$ is denoted $K$-bounded iff it never holds more than $K_j$ tokens:

$$p_j \text{ is } K_j\text{-bounded} \iff \exists K_j \in \mathbb{N}_0 : \forall \vec{s} : \vec{s}[p_j] \leq K_j$$

otherwise $p_j$ is denoted as unbounded.

- A PN is denoted $K$-bounded if each of its places is $K_j$-bounded, i.e., $P$ is denoted $K$-bounded $\iff \forall p_j \in P : p_j$ is $K_j$-bounded

- Example: Check a system design for buffer-overflows.

- In the literature 1-bounded PN are denoted as safe.

→ Question 1.3: Is the dining philosopher PN bounded?
Besides reachability problems, there are properties which ask about the structure of the reachability graph.

**Definition 1.8: Home state**

- A state \( s' \) is denoted as home state iff it is reachable from every other state, i.e.,
  \[
  \forall s \in S : \exists \sigma : s \xrightarrow{\sigma} s'.
  \]
- If \( s_0 \) is a home state than the PN \( P \) is denoted reversible.

In fact this is much more complex as a simple reachability query.

- Algebraic approach: can we exploit this one for home state detection?
- Algorithmic approach: is much more complicated (cycle detection).

---

**Timed PN (TPN)**

Enabled transitions are executed within some time interval \([a, b] \)  
(→ Timed Automata)

**Generalized Stochastic PN (GSPN)**

In a GSPN we have 2 types of transitions

- Weighted transition \( w \) is executed with some probability:  
  \[
  \text{Prob}_w(s) := \frac{W(w)}{\sum_{k \in E(s \rightarrow t)} W(t_k)}
  \]
- Markovian transition \( m \) is executed after an exponentially distributed delay time \( t \)  
  \[
  F_{\text{delay},m}(t) := 1 - e^{-\lambda_m t}
  \]
  (→ Continuous-time Markov chains)

---

**Colored PN**

- Tokens carry colors
- Transition are colored, i.e., they only consume and generate tokens of a specific color
- Colored PN posses Turing-power:
  - cannot be transformed into regular PN
  - reachability is not decidable.

---

**PN with inhibitor arcs**

- A place \( p \) connected to a transition \( t \) via an inhibitor arc \( (\Rightarrow p \in \Delta t) \)
  suppresses the transition’s execution, i.e., \( t \) can only fire iff \( s' \mid \pi[t] \geq W(p, t) \)
  holds.
- one solely needs to extend the rule for enabledness of \( t \):
  \[
  s' \models t \iff (\forall p \in \pi[t] : s'[p] \geq W(p, t))
  \land (\forall p \in \Delta t : s'[p] < W(p, t))
  \]

**Note:**

PN with more than one inhibitor arc possesses Turing-power, i.e.,

- they cannot be transformed into regular weighted PN
- reachability of a state is not decidable; the methods we looked at are therefore the best one can do.