Specification models and their analysis – Petri Nets –

Kai Lampka

December 10, 2010

TIK Institut für Technische Informatik und Kommunikationsnetze

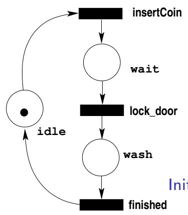
Part I

Petri Nets – Basics

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 $P := \{\text{wait}, \text{wash}, \ldots\}$ (set of places)

Boxes $T := \{\text{insertCoin}, \text{lock\_door}, \ldots\}$ (set of transitions)

Arcs $C := \{\ldots, (\text{lock\_door}, \text{wash}), \ldots\}$ (set of edges)

Weights $\mathcal{W} :=$ here constant 1 for each arc

Initial marking $\mathcal{M}_0 := \{\ m_0(wait) := 0,\ m_0(wash) := 0, m_0(idle) := 1\}$.

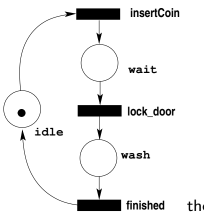
Definition 1.1: (Weigthed) Petri Net

A Petri net (PN) **P** is a 5-tuple $(P, T, C, \mathcal{W}, \mathcal{M}_0)$, where

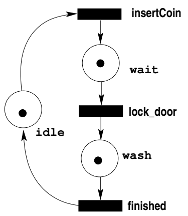
1. $P := \{p_1, \ldots, p_m\}$ is a finite, ordered (indexed) set of places and
2. $T := \{t_1, \ldots, t_n\}$ is a finite, ordered (indexed) set of transitions,
3. $C \subseteq (P \times T) \cup (T \times P)$, is a connection or flow relation,
4. $\mathcal{W} : C \mapsto \mathbb{N}_0$ assigns a weight to each element of C and
5. $m_0 : p \in P \mapsto \mathbb{N}_0$ 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 $\mathcal{M}_0$ (= the initial marking of **P**).

1. An ordering defined on the places yields that the place markings $m_0(p_i)$ can be understood as the component of a vector $\vec{s}_0[i]$ s.t. we can map the initial marking $\mathcal{M}_0$ to the dedicated vector $\vec{s}_0$. **Note, $\vec{s}_i$ gives the number of tokens currently contained in place p_i.**
2. 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.

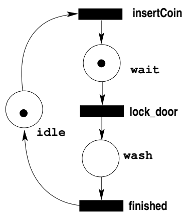
Petri Nets – Introduction



the initial state of the PN
$\vec{s}_0 = (m_0(wait), m_0(wash), m_0(idle))$
$= (0, 0, 1)$



another state of the PN
$\vec{s} = (1, 1, 1)$



and another state
$\vec{s} = (1, 1, 0)$

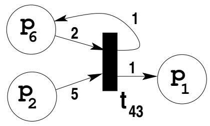
Note:
For differing among the different states of a PN we index them accordingly, i. e., we write $\vec{s}_k$ when referring to the k-th state.

Petri Nets – Operational Semantic (Pre - and Post sets)

Definition 1.2: Pre - and Post sets

1. Pre set of a transition $t \in T$: $\bullet t := \{p | (\mathbf{p}, \mathbf{t}) \in C\}$
2. Post set of a transition $t \in T$: $t\bullet = \{p | (\mathbf{t}, \mathbf{p}) \in C\}$

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



- $\bullet t_{43} := \{$
- $t_{43}\bullet := \{$

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 disceret 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?

Petri Nets – Operational Semantic (Enabling)

- **Enabled transition:**
  a transition $t \in T$ is enabled in a state $\vec{s}$, denoted $\vec{s} \rhd t$, *iff* the places of its pre set hold sufficiently many tokens
  $$\vec{s} \rhd t \Leftrightarrow (\forall p \in \bullet t : \vec{s}[p] \geq \mathcal{W}(p, t))$$
- **Enabling state:**
  a state $\vec{s}$ is denoted enabling for a transition t *iff* $\vec{s} \rhd t$ holds.
- Once a transition is enabled in a state $\vec{s}$ it can be executed (=fired):

When we execute an enabled transition we destroy sufficiently many tokens on the input places and generate the required number of tokens on the output places of t.

Petri Nets – Operational Semantic (Firing)

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

$$\delta(\vec{s},t) := \vec{s}\,' \text{ with } \vec{s}\,'[i] := \begin{cases} \vec{s}\,[i] & \Leftrightarrow p_i \notin t\bullet \cup \bullet t \\ \vec{s}\,[i] - \mathcal{W}(p_i,t) & \Leftrightarrow p_i \in \bullet t \cap \overline{t\bullet} \\ \vec{s}\,[i] + \mathcal{W}(t,p_i) & \Leftrightarrow p_i \in t\bullet \cap \overline{\bullet t} \\ \vec{s}\,[i] - \mathcal{W}(p_i,t) + \mathcal{W}(t,p_i) & \Leftrightarrow p_i \in t\bullet \cap \bullet t \end{cases}$$

- if $\vec{s} \rhd t$ then $\delta(\vec{s},t) := \vec{s}\,'$ else undefined.
- With δ we can construct sets of triples $(\vec{s},t,\vec{s}\,')$, where we use the notation $\vec{s} \xrightarrow{t} \vec{s}\,'$.

Petri Nets – Reachability set

- Moving the tokens around the net by executing enabled transitions is denoted **token game**; essentially executing δ.
- Executing transition function δ in a fixed point iteration, starting with state $\vec{s}_0$, 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

1. $\mathbb{S}^0 := \{\vec{s}_0\}$
2. $\mathbb{S}^i := \mathbb{S}^{i-1} \cup \{\delta(\vec{s},t) \mid \forall \vec{s} \in \mathbb{S}^{i-1}, \forall t \in T \text{ where } \vec{s} \rhd t\}$
3. we are intrested in the largest of such sets $\mathbb{S}^0 \subseteq \mathbb{S}^1 \subseteq \ldots \subseteq \mathbb{S}^k$ which we denote as set of reachable states $\mathbb{S}$ of a PN $\mathbf{P}$ and w. r. t. $\vec{s}_0$.

Note:

This allows one to construct a (not necessarily finite) LTS for each PN and an initial state $\vec{s}_0$. Such an LTS constitutes the semantic model of a PN.

Petri Nets – Reachability graph

Definition 1.4: Reachability graph

A reachability graph $RG(\mathbf{P},\vec{s}_0)$ of a PN $\mathbf{P}$ and its initial state $\vec{s}_0$ is a LTS $\mathcal{L}(\mathbb{S},\mathbb{S}_0,\mathcal{A}ct,\mathbb{E})$ with

- $\mathbb{S}$ which is the set of reachable states of the PN.
- $\mathbb{S}_0 := \{\vec{s}_0\}$ with $\vec{s}_0$ as the initial state of the PN
- $\mathcal{A}ct$ which is the set of the transition labels of the PN
- $\mathbb{E} \subseteq \mathbb{S} \times \mathcal{A}ct \times \mathbb{S}$ induced by the PN as follows:

$$(\vec{s} \in \mathbb{S} \wedge \vec{s} \rhd t) \Rightarrow \left(\vec{s} \xrightarrow{t} \delta(\vec{s},t) \in \mathbb{E}\right)$$

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 $\vec{s}_0$ by executing the transitions of a PN.

⟶ Example 1.1: Washing maschine

Petri Nets – Reachability question

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 $\mathbf{P}$ and an initial (system) state $\vec{s}_0$ the reachability question answers the question, if it is possible to reach a dedicated state $\vec{s}_b$ by executing a sequence of transition $(t_i,\ldots,t_j)$.

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 $\vec{s}_b$. With $\sigma := t_i, t_j \ldots t_k$ one also writes $\vec{s}_0 \xrightarrow{\sigma} \vec{s}_b$ for indicating that $\vec{s}_b$ can be reached by executing transition sequence σ.

Decidability

- 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 – 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 sloution we do not know anything.

Algorithmic method (brute-force method)

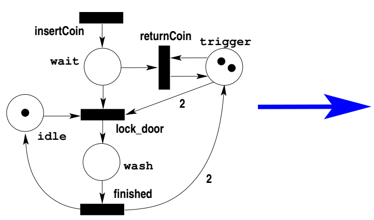
Generation of the reachability graph and check the states on-the-fly.

Termination implies reachability of the searched state. If the reachability graph is not finite the method may not terminate.

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 $A \in \mathbb{N}_0^{|P|\times|T|}$ describes the token-flow w. r. t. place i (row index) and transition j (column index).

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



A := ?

$\vec{s}_0$:= ?

⟶ Example 1.2: Washing maschine

Petri Nets – Algebraic approach (state equations)

- Let the firing vector $\vec{u}_i \in \mathbb{N}_0^{|T|}$ 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 $\vec{s}_c + A \cdot \vec{u}_i$?

  $\longrightarrow$ Example 1.3: Washing machine

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

$$\vec{f} := \sum_{k:=1}^{|\sigma|} \vec{u}_{\sigma[k]}$$

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

  $\longrightarrow$ Question 1.1: What is the PN semantic of $\vec{s}_c + A \cdot \vec{f}$?

Petri Nets – Algebraic approach (state equations)

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

1. Solve$(A \cdot \vec{f} = \vec{s} - \vec{s}_0)$
2. if there is no such solution $\vec{f} \in \mathbb{N}_0^{|T|}$ than there is no sequence of transition firings leading from $\vec{s}_0$ to $\vec{s}$.
3. Formally:

$$(\nexists \vec{f} \in \mathbb{N}_0^{|T|} : A \cdot \vec{f} = \vec{s} - \vec{s}_0) \Rightarrow \nexists \sigma : \vec{s}_0 \xrightarrow{\sigma} \vec{s}$$

$\longrightarrow$ Example 1.4: Washing machine

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

Petri Nets – Algorithmic approach (enumerative)

1. Algebraic solution based on state equations: Solution of a system of linear equation is a necessary condition for reachability for standard PN only.

   Absence of solution implies non-reachability of the resp. state.

2. **Algorithmic solution** (brute-force method): Generation of the reachability graph and check the states on-the-fly:

   Termination implies reachability of the searched state.

Petri Nets – Explicit Reachability Set Generation

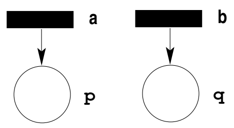
Reachability algorithm

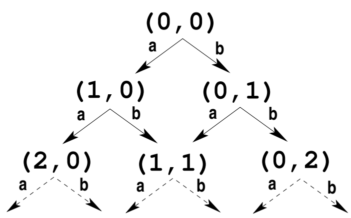
```
S_tmp := ∅, S := ∅
put s_0 into S_tmp
put s_0 into S
Call function DFS()

Function DFS()
  While (S_tmp ≠ ∅)
     take s from S_tmp
     forall t ∈ T do
        s' := δ(s, t)
        If (s' = s_test) terminate
        Else if (s' ∉ S)
           put s' into S_tmp
           put s' into S
        Else do_nothing
        endif
     od
  endwhile
```

Once this algorithm terminates we know that $\vec{s}_{test}$ is reachable or not reachable.

$\longrightarrow$ Example 1.5: Washing machine: DFS/BFS

Petri Nets – Interleaving semantic





- The execution order of transitions is partly uncoordinated.
- This yields an exponential size of a PN's underlying LTS , i. e., $|\mathbb{S}|$ is exponential w. r. t. the number of concurrently enabled transitions

(= state space explosion problem)

$\longrightarrow$ Example 1.6: Washing machine with 2 tokens in place idle

Petri Nets – Properties to be verified

- After we have learned how to answer the reachability problem, (where the proposed techniques may fail), we will now deal with other properties to be verified.
- These properties can be posed as reachability problems. Hence they can be answered by either directly applying the discussed techniques or slightly adapted versions.

Excursion:

Properties that can be answered as reachability queries, are denoted safety properties. But remember, the here presented techniques may fail, but sometimes we can not do better, depending on the employed high-level formalism, e. g. PN with inhibitor arcs.

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

$$\nexists t \in T : \vec{s} \rhd t$$

- 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)

$\longrightarrow$ Example 1.7: Dining philosophers

K-boundedness

Definition 1.7: K-Boundedness

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

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

   otherwise p_j is denoted as *unbounded*.

2. A PN is denoted K-bounded if each of its places is K_j-bounded, i. e.,

$$\mathbf{P} \text{ is denoted } K\text{-bounded} \Leftrightarrow \forall p_j \in P : p_j \text{ is } K_j\text{-bounded}$$

- Example: Check a system design for buffer-overflows.
- In the literature 1-bounded PN are denoted as *safe*.

$\longrightarrow$ Question 1.3: Is the dining philosopher PN bounded?

Reversibility and home states

Besides reachability problems, there are properties which ask about the structure of the reachability graph.

Definition 1.8: Home state

1. A state $\vec{s}\,'$ is denoted as home state *iff* it is reachable from every other state, i. e.,

$$\vec{s}\,' \text{ is a home state } \Leftrightarrow \forall \vec{s} \in \mathbb{S} : \exists \sigma : \vec{s} \xrightarrow{\sigma} \vec{s}\,'.$$

2. If $\vec{s}_0$ is a home state than the PN **P** is denoted reversible.

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

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

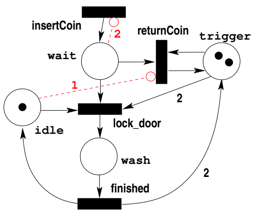
Petri Nets – Extensions

Many flavors of Petri nets are in use, e.g.

1. PN with inhibitor arcs
2. Colored PN
3. PN with timed behaviour

Petri Nets – Extensions

PN with inhibitor arcs



- A place p connected to a transition t via an inhibitor arc ($\Rightarrow p \in \triangle t$) suppresses the transition's execution, i. e., t can only fire *iff* $\vec{s}\,[p] < \mathcal{W}(p,t)$ holds.
- one solely needs to extend the rule for enabledness of t:

$$\vec{s} \rhd t \Leftrightarrow \{(\forall p \in \bullet t : \vec{s}\,[p] \geq \mathcal{W}(p,t)) \wedge : (\forall p \in \triangle t : \vec{s}\,[p] < \mathcal{W}(p,t))\}$$

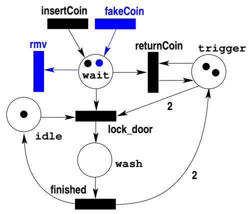
Note:

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

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

Petri Nets – Extensions

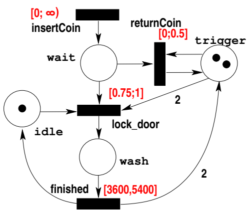
Colored PN



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

Petri Nets – Extensions

Timed PN (TPN)

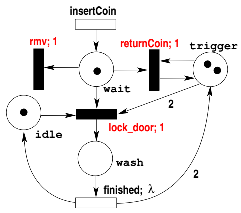


Enabled transitions are executed within some time interval $[a,b]$

(→ Timed Automata)

Petri Nets – Extensions

Generalized Stochastic PN (GSPN)



In a GSPN we have 2 types of transitions

- Weighted transition w is executed with some probability:

$$Prob_w(\vec{s}) := \frac{\mathcal{W}(w)}{\sum_{t_k \in \{t \mid \vec{s} \rhd t\}} \mathcal{W}(t_k)}$$

- Markovian transition m is executed after an exponentially distributed delay time t

$$F_{delay\_of\_m}(t) := 1 - e^{-\lambda_m t}$$

(→ Continous-time Markov chains)