**Technische Universität München (I7)**
**Prof. J. Esparza / P. Meyer**

**Summer term 2015**
**Thursday 16th April, 2015**

# Petri nets – Homework 1

Discussed on Wednesday 22nd and Thursday 23rd April, 2015.

*For questions regarding the exercises, please send an email to meyerphi@in.tum.de or just drop by at room 03.11.042.*

## Exercise 1.1 Milner's scheduler

We want to specify a simple scheduler for a set of $n$ agents $P_1, \ldots, P_n$. Each agent $P_i$ performs a task repeatedly, and the scheduler is required to ensure that they begin the task in cyclic order starting with $P_1$. The different task-performances need not exclude each other in time—for example $P_2$ can begin before $P_1$ finishes—but the scheduler is required to ensure that each agent finishes one performance before it begins another.

We assume that $P_i$ requests task initiation by an action $a_i$ and signals completion by an action $b_i$. The scheduler can then be specified by requiring that:

(1) It must perform $a_1, \ldots, a_n$ cyclically, starting with $a_1$.

(2) It must perform $a_i$ and $b_i$ alternately, for each $i$.

However, a scheduler which imposes a fixed sequence, say $a_1 b_1 a_2 b_2 \ldots$, is not good enough, the scheduler must allow *any* sequence of actions compatible with the conditions (1) and (2) above. For example, for $n = 2$, the sequences $a_1 a_2 b_1 b_2 a_1$ and $a_1 b_1 a_2 a_1 b_2 b_1$ are compatible with the specification, but the sequences $a_1 b_1 a_1$ and $a_1 b_1 a_2 a_1 a_2$ are not.

(a) For $n = 2$ agents, give a Petri net $(N, M_0)$ which models a scheduler satisfying the given requirements.

The Petri net must have transitions $\{a_1, a_2, b_1, b_2\} \subseteq T$. Firing one of these transitions is equivalent to the execution of the corresponding action. Further, for any firing sequence $\sigma$ enabled at $M_0$, the requirements above must hold when interpreting the firing sequence as a sequence of actions. The Petri net may also have additional transitions, which do not correspond to an action and may occur at any time.

The Petri net should also be live and bounded.

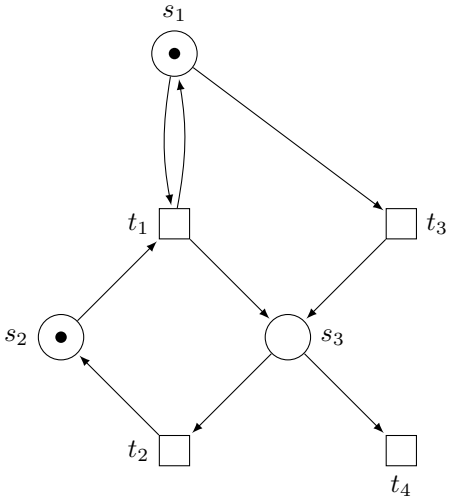
(b) How many reachable markings does the Petri net have?

(c) How can this solution be generalized to any number of $n$ agents?

(d) How does the number of reachable markings grow as $n$ increases?

## Exercise 1.2 Reachability graph

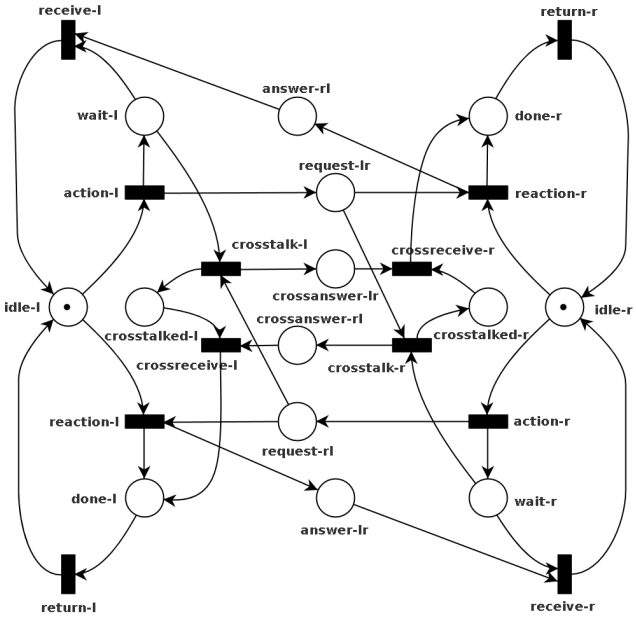
Construct the reachability graph of the following Petri net. Is the empty marking $M = (0, 0, 0)$ reachable? If yes, give a firing sequence $M_0 \xrightarrow{\sigma} M$.



**Exercise 1.3** **Petri net analysis with tools**

Recall the third attempt for the solution of the action/reaction protocol (Figure 2.11 in the script). This system has a deadlock, and we would like to find the dead marking and a firing sequence leading to it. Due to the large size of the system and its reachability graph, we will use a tool to find them.

On the course website in the exercises section, you can find this net in suitable formats for the tools PIPE, APT and LoLA. Below the Petri net is shown as modelled in PIPE.




Install and use a tool of your choice to find a dead marking $M$ and a firing sequence $M_0 \xrightarrow{\sigma} M$ leading to it.

- **PIPE** (`http://pipe2.sourceforge.net/`)

  To install PIPE, download the zip file from SourceForge and extract it. You can start it by executing `launch.sh` (Linux/OS X) or `launch.bat` (Windows).

  In PIPE, after opening the file, a reachability graph can be constructed by selecting the module "Reachability/Coverability Graph" on the left. Dead markings are colored red in the graph and a firing sequence can be read off the arcs from the initial marking (S0) to the dead marking.

  To verify this sequence, the animation mode in PIPE can be used. It is activated with the flag icon on the top. Enabled transitions are marked in red and can be fired by clicking on them. After firing the found sequence, no transition should be enabled.

- **APT** (`https://github.com/CvO-Theory/apt`)

  To install APT, clone the repository from GitHub or download the zip file and extract it. APT needs the JDK and Apache Ant to build it, for instructions refer to the readme on GitHub. It can then be run by executing `java -jar apt.jar`.

  APT can not directly find deadlocks. However, it can generate the reachability graph with the following command:
  `java -jar coverability_graph action-reaction-third-attempt.apt`
  The reachability graph can then be checked for a marking which has no outgoing arcs.

  APT can also check if the net is live and, if it is not, print a firing sequence after which some transition is not enabled any more. While this is not a sufficient condition for a deadlock, it is an indicator for one. The command for this is
  `java -jar apt.jar strongly_live action-reaction-third-attempt.apt`
  and then the sequence $\sigma$ can be fired with
  `java -jar apt.jar fire_sequence "`$\sigma$`" action-reaction-third-attempt.apt`
  to give a marking which can be checked if it is dead.

- **LoLA** (`http://service-technology.org/lola/`)

  For installing LoLA, you need a working C++ compiler such as GCC or Clang. On Linux and OS X, it can be compiled with `./configure` and `make`. On Windows, you might need a Unix-like environment via Cygwin.

  LoLA can answer reachability queries and can print a witness marking and firing sequence when a marking is reachable. To find a deadlock, one can use the command:
  `lola action-reaction-third-attempt.lola -f "EF DEADLOCK" -s -p`

## Exercise 1.4 Boundedness, liveness and deadlock freedom

For each of the three Petri nets $A$, $B$ and $C$, check if the following properties hold. This can be done by constructing the reachability graphs of the Petri nets.

(a) **Boundedness**: Is the Petri net bounded, i.e., for every place $s$, is there a number $b \geq 0$ such that $M(s) \leq b$ for every reachable marking $M$?

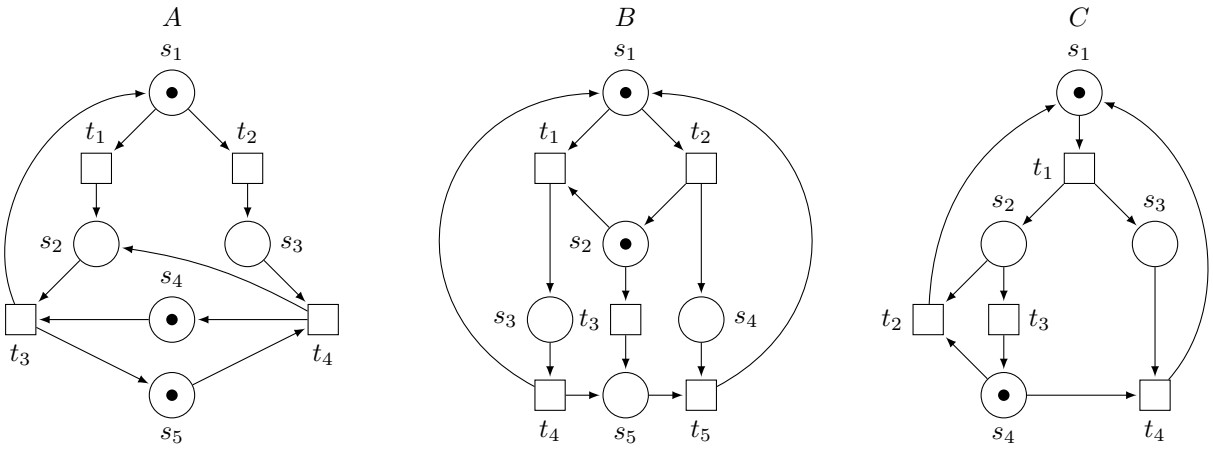
If the Petri net is bounded, give the bound of each place $s$, that is, the maximal number of tokens in that place in all reachable markings.

(b) **Liveness**: Is the net live, i.e., for every reachable marking $M$ and every transition $t$, is there a marking $M'$ reachable from $M$ that enables $t$?

If the Petri net is not live, give a firing sequence $\sigma$ and a transition $t$ such that after firing $\sigma$ from the initial marking $M_0$, we can never fire $t$ again, that is, $M_0 \xrightarrow{\sigma} M$ and $M \not\xrightarrow{t}$ for all $M' \in [M\rangle$.

(c) **Deadlock freedom**: Is the Petri net deadlock-free, i.e., is there a reachable marking $M$ that enables no transitions?

If the Petri net has a deadlock, give a firing sequence $\sigma$ that leads to a dead marking, that is, $M_0 \xrightarrow{\sigma} M$ and $M \not\xrightarrow{t}$ for all transitions $t$.




## Exercise 1.5 Independence of boundedness, liveness and cyclicity

**Definition 1.5.1** (Cyclic Petri nets)**.** A Petri net $(N, M_0)$ is *cyclic* if, loosely speaking, it is always possible to return to the initial marking. Formally: $\forall M \in [M_0\rangle : M_0 \in [M\rangle$.

Show that the properties liveness, boundedness and cyclicity are independent of eath other by exhibiting eight Petri nets, one for each possible combination of the three properties and their negations.

*Remark:* Especially the live, bounded, but not cyclic Petri net is hard to find. However, it can be done with only 4 places and 3 transitions.

## Exercise 1.6 Strong Connectedness Theorem

Let $(N, M_0)$ be a live and bounded Petri net. Show that $N$ is strongly connected.

*Hint:* To show that the net is strongly connected, you need to show that for every arc $(x, y) \in F$, there is a path from $y$ to $x$. Use liveness to construct a firing sequence containing the transition of the arc often enough and then use boundedness on the place of the arc to show that there needs to be a path back. You may also use the following lemma:

**Lemma 1.6.1** (Exchange Lemma)**.** Let $u$ and $v$ be transitions of a net satisfying ${}^\bullet u \cap v^\bullet = \emptyset$. If $M \xrightarrow{vu} M'$ then $M \xrightarrow{uv} M'$.