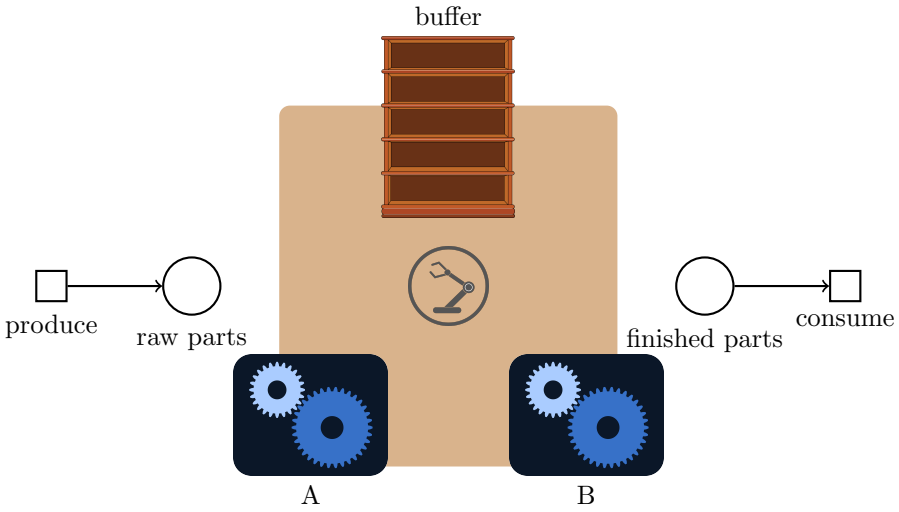
**Technische Universität München, I7**
**Prof. J. Esparza / Ch. Weil-Kennedy / P. Meyer**

**Summer term 2019**
**24.04.2019**

# Petri nets — Exercise sheet 1

Due 30.04.2019

**Exercise 1.1 (adapted from [1, ex. 2.20])**
Consider a simple production system in which raw parts are first processed by a machine $A$, stored into a buffer, and then processed by a machine $B$. The parts are moved around using a single robot arm $R$. The buffer can contain at most five items at a time, and machines $A$ and $B$ can only handle one item at a time.




Model this production system as a Petri net by extending the partial model shown above. The actions of machines $A$ and $B$ are not atomic: they have a beginning and an end. On the other hand, the action of the robot arm can be considered as atomic. There is no need to distinguish between particular buffer places, or between particular items to be processed.

**Exercise 1.2**
Consider Lamport's 1-bit mutual exclusion algorithm:

First process

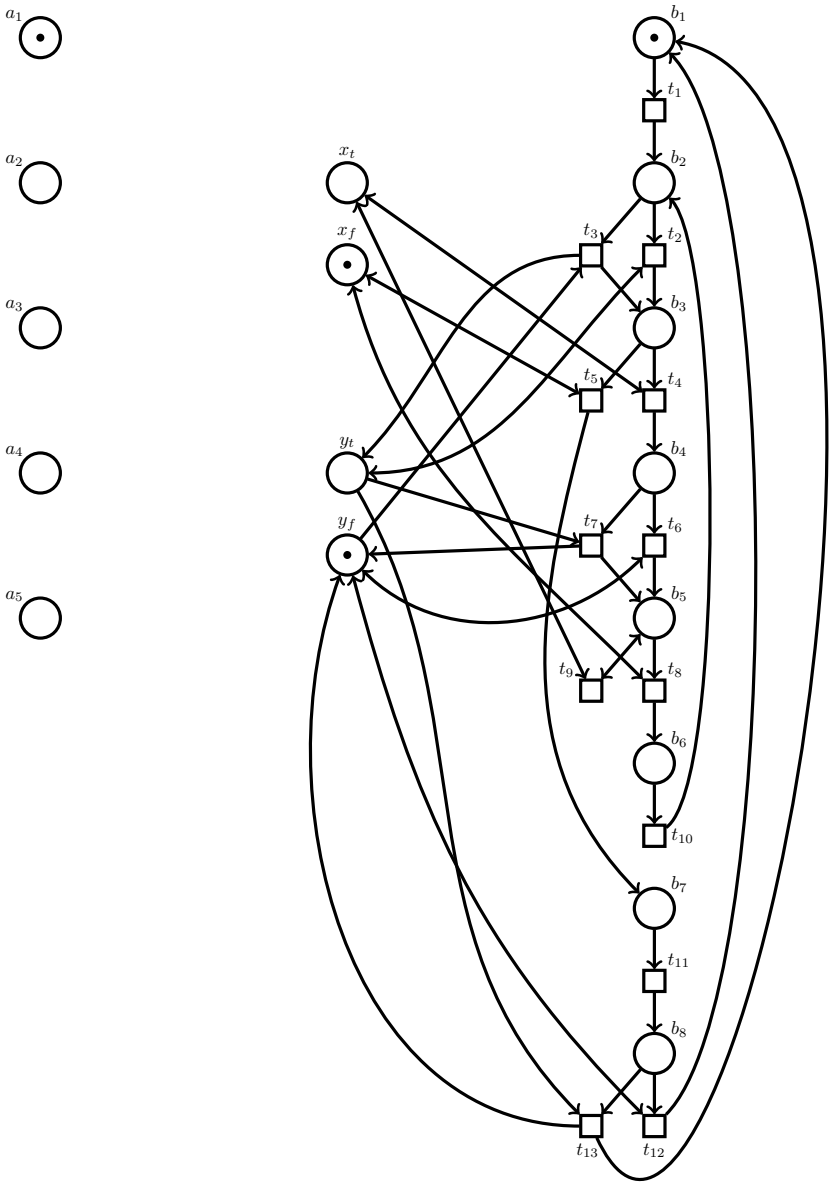
```
1.  while True:
2.    x = True
3.    while y: pass
4.    # critical section
5.    x = False
```

Second process

```
1.  while True:
2.    y = True
3.    if x then:
4.      y = False
5.      while x: pass
6.      goto 2
7.    # critical section
8.    y = False
```

The algorithm can be modeled by a Petri net $\mathcal{N}$ where each program location (i.e. line of code of a process) is associated to a place, and where the shared binary variables `x` and `y` are associated to two places each. In

more details, $\mathcal{N} = (P, T, F)$ where $P = \{a_1, \ldots, a_5, b_1, \ldots, b_8, x_t, x_f, y_t, y_f\}$. A token in $a_i$ (resp. $b_i$) indicates that the first (resp. second) process is at line $i$; a token in $x_t$ (resp. $y_t$) indicates that `x` (resp. `y`) has value `True`; and a token in $x_f$ (resp. $y_f$) indicates that `x` (resp. `y`) has value `False`. The initial marking of $\mathcal{N}$ is $M_0 = \{a_1, b_1, x_f, y_f\}$. We give a partial Petri net that only models the second process:




(a) Complete the above Petri net $\mathcal{N}$ so that it also models the first process. You should not add new places, only transitions and arcs. Note that `pass` is a "no operation", i.e. an operation without any effect.

(b) Complete the given `APT` file for $\mathcal{N}$ accordingly, and verify whether

(i) $(\mathcal{N}, M_0)$ is bounded;

(ii) $(\mathcal{N}, M_0)$ is live.

(c) Complete the given `LoLA` file for $\mathcal{N}$ accordingly, and verify whether

(i) $(\mathcal{N}, M_0)$ is deadlock-free;

(ii) a process can be at multiple program locations at the same time;

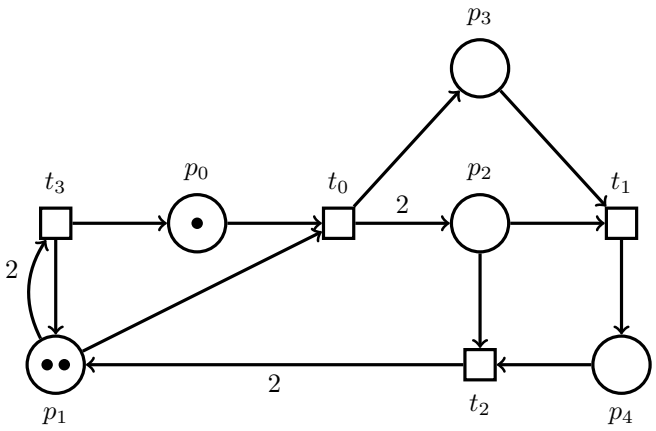
(iii) whether both processes can reach their critical sections simultaneously.

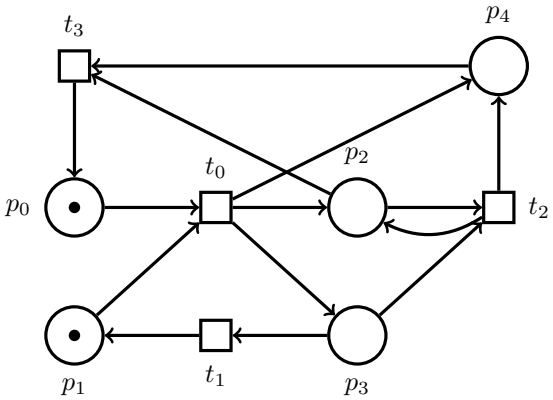
**Exercise 1.3**
For each Petri net $(\mathcal{N}, M_0)$ below:

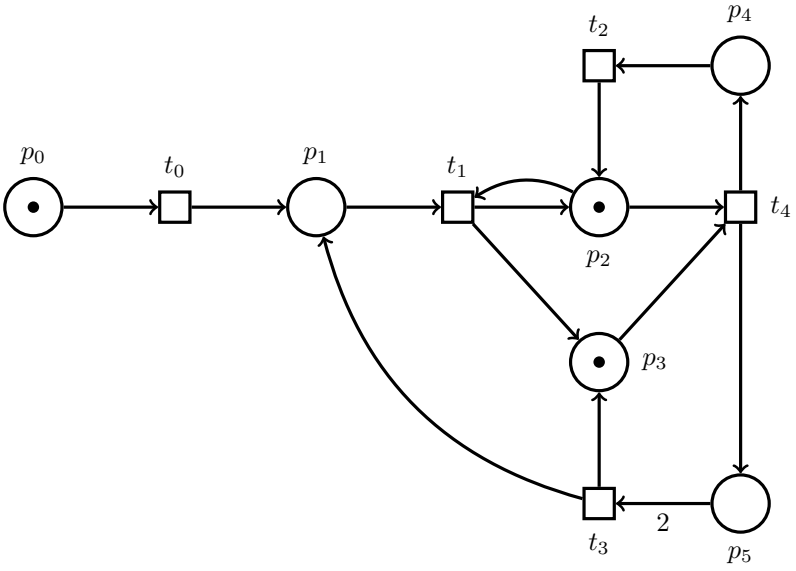
(a) construct the reachability graph of $(\mathcal{N}, M_0)$.

(b) say whether $(\mathcal{N}, M_0)$ is bounded, deadlock-free and/or live. If it is bounded, give the smallest $k$ such that it is $k$-bounded. Justify your answers.

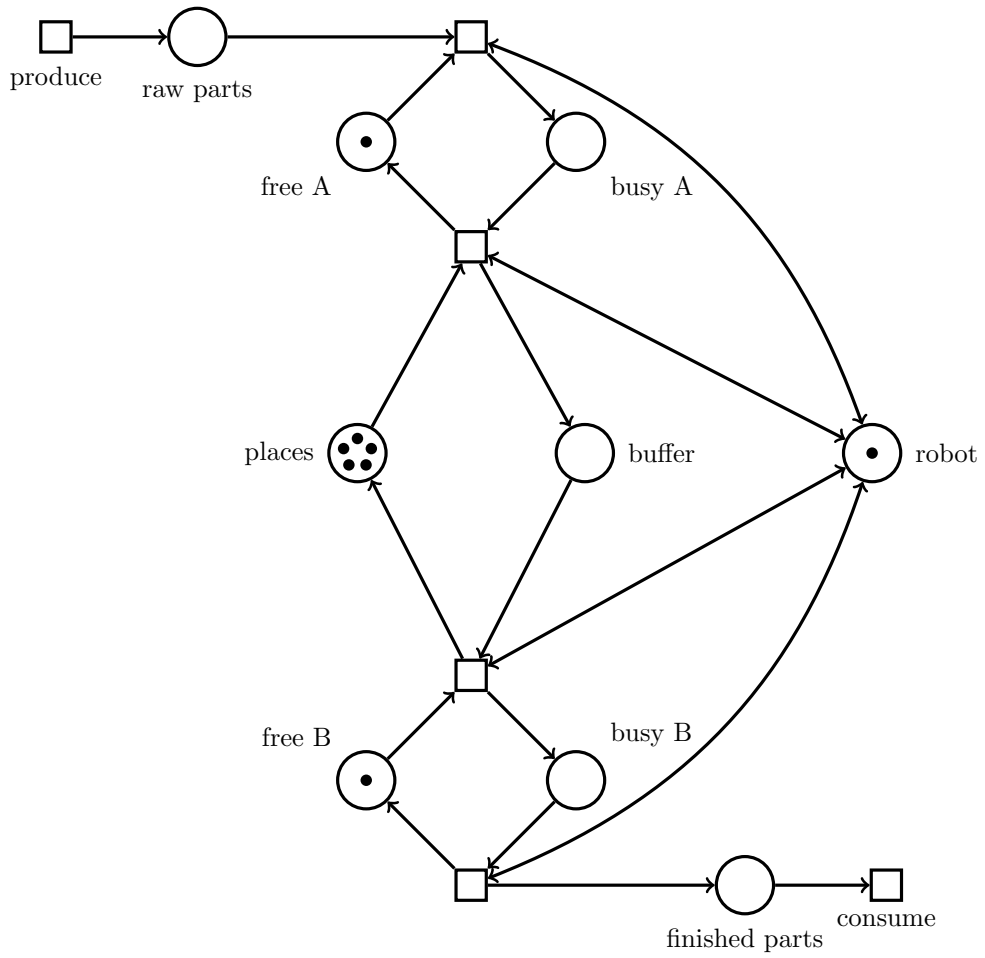
(c) give the subnet $\mathcal{N}' = (P', T', F')$ of $\mathcal{N}$ such that $P' = \{p_0, p_1, p_2, p_4\}$ and $T' = T$.



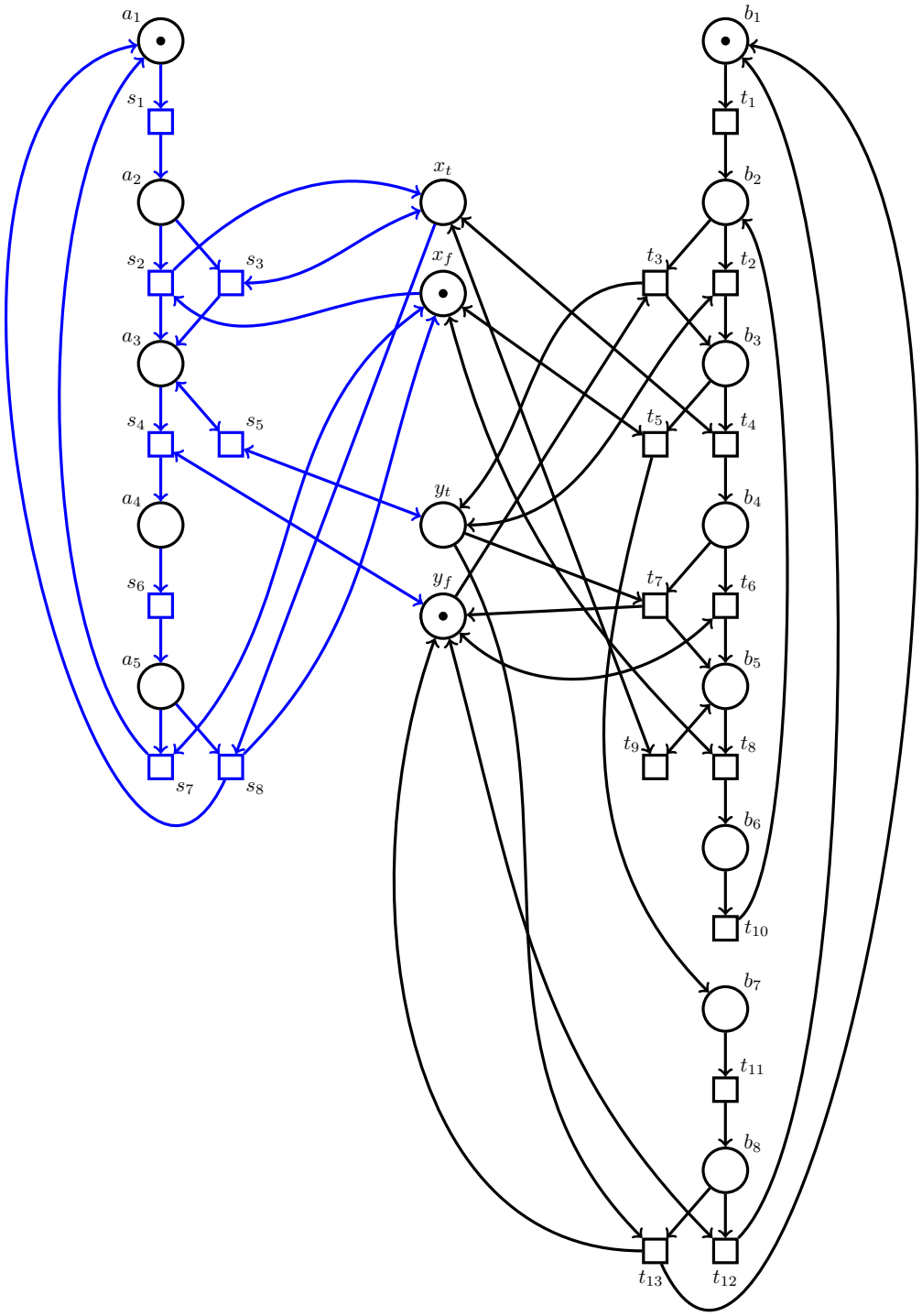
**Solution 1.1 (adapted from [1, ex. 2.20])**
The production system can be modelled as follows:




Notice that since the robot place is always marked, it could also be omitted.

**Solution 1.2**

(a)



Notice that variable $x$ will always have value false when the first process enters place $a_2$, so transition $s_3$ will never be used and can be omitted. Similarly with $s_7$, $t_2$ and $t_{12}$.

(b) (i)
```
> java -jar apt.jar bounded lamport.apt
bounded:  Yes
smallest_K: 1
```

(ii)
```
> java -jar apt.jar strongly_live lamport.apt
strongly_live:  No
sample_witness_transition:  s3
sample_witness_firing_sequence:  [s1]
```

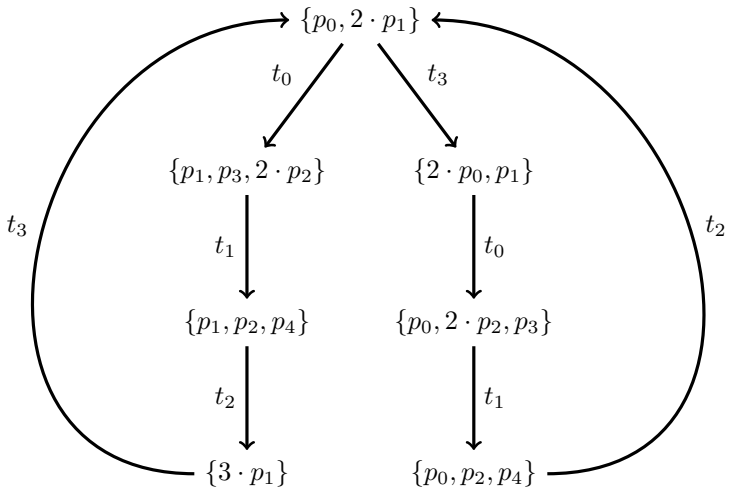
(c) (i)
```
> lola lamport.lola -f "REACHABLE DEADLOCK"
lola:  result:  no
lola:  The net does not have deadlocks.
```

(ii)
```
> lola lamport.lola -f "REACHABLE (a1 + a2 + a3 + a4 + a5 > 1) OR (b1 + b2 + b3 + b4
+ b5 + b6 + b7 + b8 > 1)"
lola:  result:  no
lola:  The predicate is unreachable.
```

(iii)
```
> lola lamport.lola -f "REACHABLE (a4 > 0 AND b7 > 0)"
lola:  result:  no
lola:  The predicate is unreachable.
```

**Solution 1.3**

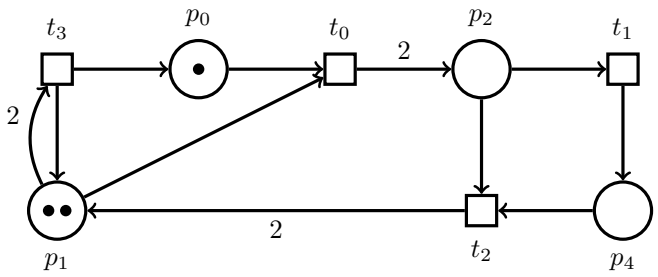
1. (a)



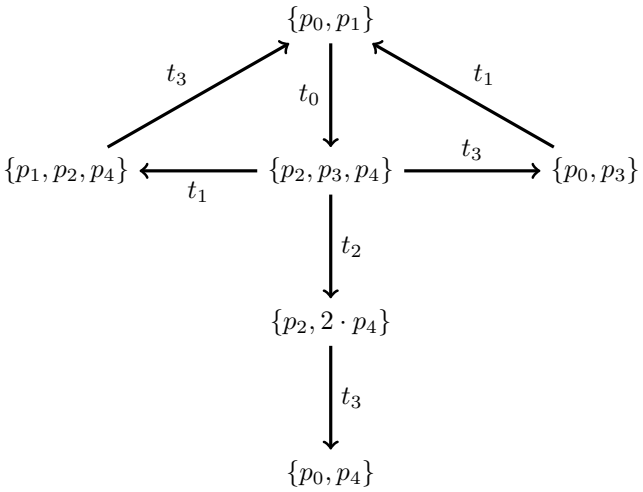

(b) It is 3-*bounded* since all markings of the reachability graph have at most three tokens in each place. It is *deadlock-free* since every marking of the reachability graph has an outgoing arc. It is *live* because for every transition $t$, every marking $M$ of the reachability graph leads to a marking $M'$ with an outgoing arc labeled by $t$.

★ Alternatively, liveness follows from the fact that the reachability graph is strongly connected and has an occurrence of every transition.

(c)

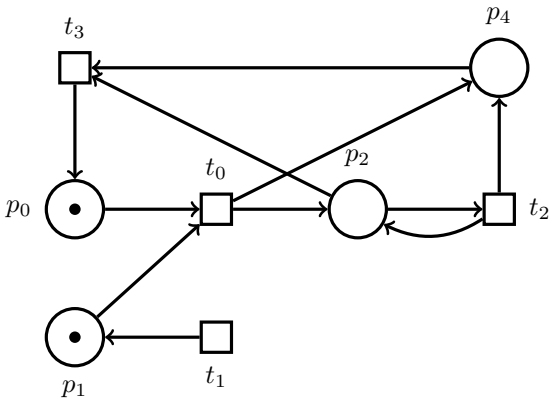

$t_3$
$p_0$
$t_0$
$p_2$
$t_1$
2
2
2
$p_1$
$t_2$
$p_4$

2. (a)

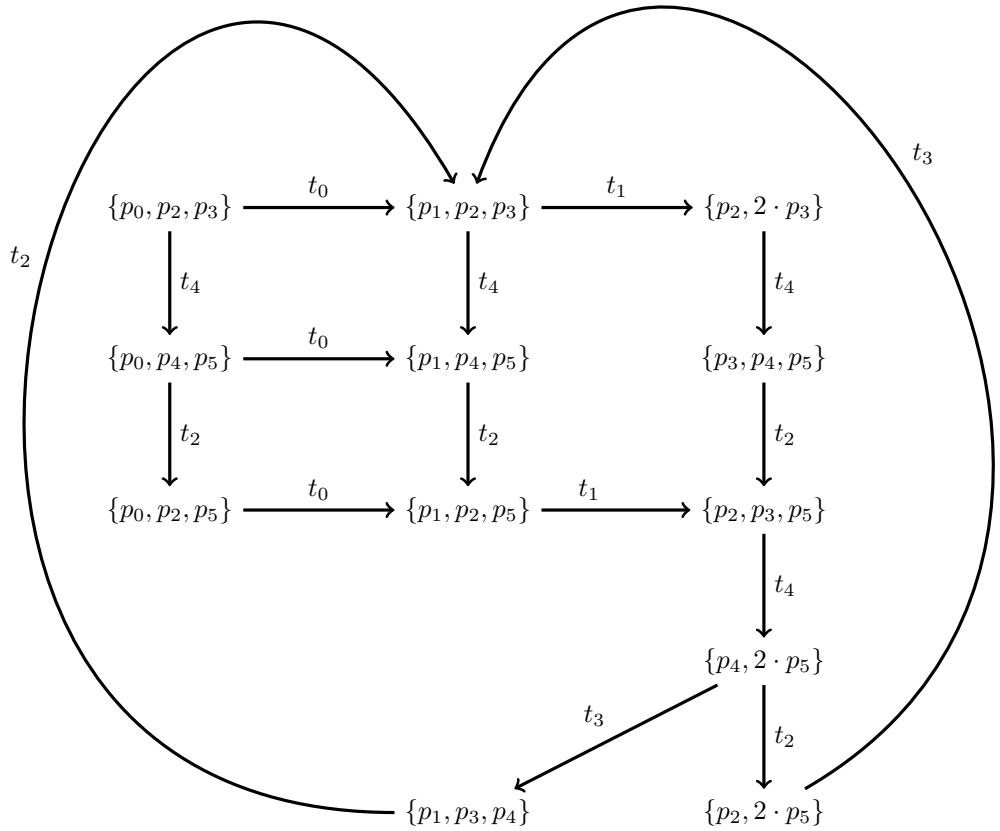



(b) It is 2-*bounded* since all markings of the reachability graph have at most two tokens in each place. It is *not deadlock-free* since $\{p_0, p_4\}$ has no successor. It is *not live* since it is not deadlock-free.

(c)



3. (a)




(b) It is *not live* since in the reachability graph has no path from $\{p_1, p_2, p_3\}$ that contains $t_0$. It is 2-*bounded* since all markings of the reachability graph have at most 2 tokens in each place. It is *deadlock-free* since every marking of the reachability graph has an outgoing arc.

★ It is possible to show that the net is not live without inspecting the reachability graph. Note that $M_0 \xrightarrow{t_0} \{p_1, p_2, p_3\}$. Moreover, $\mathcal{N}$ has no transition that produces a token in $p_0$. Therefore, $\{p_1, p_2, p_3\}$ cannot reach any marking from which $t_0$ is enabled.

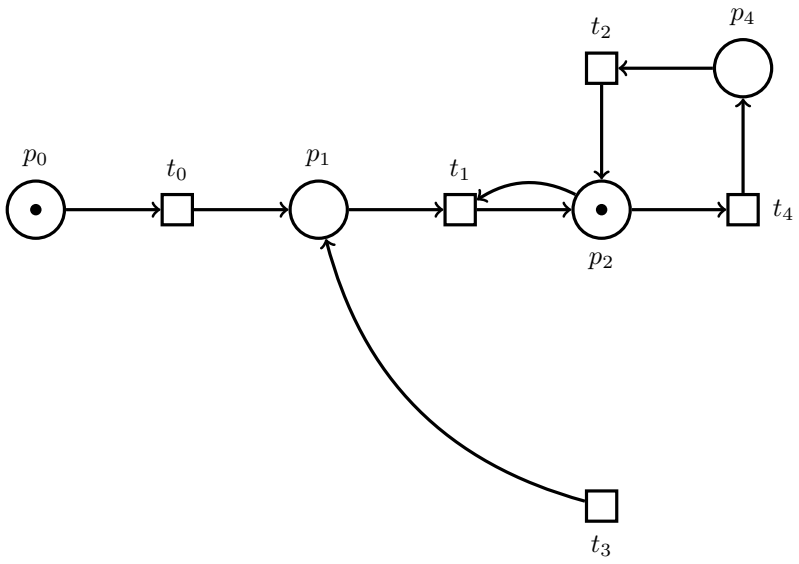
★ There is an alternative way to prove 2-boundness and deadlock-freedom without inspecting the reachability graph. Let $Q = \{p_0, p_1, p_3, p_5\}$ and $R = \{p_2, p_4\}$. We claim that $M(Q) = 2$ and $M(R) = 1$ for every reachable marking $M$. The claim clearly holds for $M_0$. Moreover, every transition of $\mathcal{N}$ consumes and produces the same amount of tokens from both $Q$ and $R$, which proves the claim. Now, for the sake of contradiction, assume there exists a deadlock, e.g. there exists some reachable marking $M$ from which no transition is enabled. By definition of transitions $t_0$, $t_2$ and $t_3$,

this implies that

$$M(p_0) = 0,$$
$$M(p_4) = 0,$$
$$M(p_5) \leq 1,$$

and hence, by the claim, that $M(p_1) + M(p_3) \geq 1$ and $M(p_2) = 1$. In particular $M(p_1) > 0$ or $M(p_3) > 0$. If the former holds, then $t_1$ is enabled, if the latter holds, then $t_4$ is enabled. Both cases yield contradictions.

(c)




# References

[1] Wil van der Aalst, Massimiliano de Leoni, Boudewijn van Dongen, and Christian Stahl. Course business information systems: Exercises. Available at `http://wwwis.win.tue.nl/~wvdaalst/old/courses/BIScourse/exercise-bundle-BIS-2015.pdf`, 2015.