Exercise 1 (Traffic Lights). Consider again the traffic lights example from the lecture notes:

1. How can you correct this Petri net to avert unwanted behaviours (like $r \rightarrow ry \rightarrow rr$) in a 1-safe manner?

2. Extend your Petri net to model two traffic lights handling a street intersection.

Exercise 2 (Producer/Consumer). A producer/consumer system gathers two types of processes:

- producers who can make the actions produce ($p$) or deliver ($d$), and
- consumers with the actions receive ($r$) and consume ($c$).

All the producers and consumers communicate through a single unordered channel.

1. Model a producer/consumer system with two producers and three consumers. How can you modify this system to enforce a maximal capacity of ten simultaneous items in the channel?

2. An inhibitor arc between a place $p$ and a transition $t$ makes $t$ firable only if the current marking at $p$ is zero. In the following example, there is such an inhibitor arc between $p_1$ and $t$. A marking $(0, 2, 1)$ allows to fire $t$ to reach $(0, 1, 2)$, but $(1, 1, 1)$ does not allow to fire $t$. 
Using inhibitor arcs, enforce a priority for the first producer and the first consumer on the channel: the other processes can use the channel only if it is not currently used by the first producer and the first consumer.

Exercise 3 (Model Checking Petri Nets). Let us fix a Petri net $\mathcal{N} = \langle P, T, F, W, m_0 \rangle$. We consider as usual propositional LTL, with a set of atomic propositions AP equal to $P$ the set of places of the Petri net. We define proposition $p$ to hold in a marking $m$ in $\mathbb{N}^P$ if $m(p) > 0$.

The models of our LTL formulae are computations $m_0 m_1 \cdots$ in $(\mathbb{N}^P)^\omega$ such that, for all $i \in \mathbb{N}$, $m_i \xrightarrow{\mathcal{N}} m_{i+1}$ is a transition step of the Petri net $\mathcal{N}$.

1. We want to prove that state-based LTL model checking can be performed in polynomial space for 1-safe Petri nets. For this, prove that one can construct an exponential-sized B"uchi automaton $B_\mathcal{N}$ from a 1-safe Petri net that recognizes all the infinite computations of $\mathcal{N}$ starting in $m_0$.

2. In the general case, state-based LTL model checking is undecidable. Prove it for Petri nets with at least two unbounded places, by a reduction from the halting problem for 2-counter Minsky machines.

3. We consider now a different set of atomic propositions, such that $\Sigma = 2^{AP}$, and a labeled Petri net, with a labeling homomorphism $\lambda : T \rightarrow \Sigma$. The models of our LTL formulae are infinite words $a_0 a_1 \cdots$ in $\Sigma^\omega$ such that $m_0 \xrightarrow{t_0} \mathcal{N} m_1 \xrightarrow{t_1} \mathcal{N} m_2 \cdots$ is an execution of $\mathcal{N}$ and $\lambda(t_i) = a_i$ for all $i$.

Prove that action-based LTL model checking can be performed in polynomial space for labeled 1-safe Petri nets.

Exercise 4 (VASS). An $n$-dimensional vector addition system with states (VASS) is a tuple $\mathcal{V} = \langle Q, \delta, q_0 \rangle$ where $Q$ is a finite set of states, $q_0 \in Q$ the initial state, and $\delta \subseteq Q \times \mathbb{Z}^n \times Q$ the transition relation. A configuration of $\mathcal{V}$ is a pair $(q, v)$ in $Q \times \mathbb{N}^n$. An execution of $\mathcal{V}$ is a sequence of configurations $(q_0, v_0)(q_1, v_1) \cdots (q_m, v_m)$ such that $v_0 = 0$, and for $0 < i \leq m$, $(q_{i-1}, v_i - v_{i-1}, q_i)$ is in $\delta$.

1. Show that any VASS can be simulated by a Petri net.
2. Show that, conversely, any Petri net can be simulated by a VASS.

**Exercise 5** (VAS). An $n$-dimensional vector addition system (VAS) is a pair $(v_0, W)$ where $v_0 \in \mathbb{N}^n$ is the initial vector and $W \subseteq \mathbb{Z}^n$ is the set of transition vectors. An execution of $(v_0, W)$ is a sequence $v_0 v_1 \cdots v_m$ where $v_i \in \mathbb{N}$ for all $0 \leq i \leq m$ and $v_i - v_{i-1} \in W$ for all $0 < i \leq m$.

We want to show that any $n$-dimensional VASS $V = \langle Q, \delta, q_0 \rangle$ can be simulated by an $(n+3)$-dimensional VAS $(v_0, W)$. Let $k = |Q|$, and $q_0, \ldots, q_{k-1}$ the states of $V$. We define two functions $a(i) = i + 1$ and $b(i) = (k + 1)(k - i)$. We encode a configuration $(q_i, v)$ of $V$ as the vector $(v(1), \ldots, v(n), a(i), b(i), 0)$. For every state $q_i$, $0 \leq i < k$, we add two transition vectors to $W$:

$$
t_i = (0, \ldots, 0, -a(i), a(k - 1 - i) - b(i), b(k - 1 - i))
$$

$$
t'_i = (0, \ldots, 0, b(i), -a(k - 1 - i), a(i) - b(k - 1 - i))
$$

For every transition $d = (q_i, w, q_j)$ of $V$, we add one transition vector to $W$:

$$
t_d = (w(1), \ldots, w(n), a(j) - b(i), b(j), -a(i))
$$

1. Show that any execution of $V$ can be simulated by $(v_0, W)$ for a suitable $v_0$.
2. Conversely, show that this VAS $(v_0, W)$ simulates $V$ faithfully.