Concurrency Theory (WS 2016) Out: Thu, 03 Nov Due: Wed, 09 Nov

Exercise Sheet 2

D'Osualdo, Lederer, Schneider Technische Universität Kaiserslautern

Please submit your solution in groups of 2 to 3 people until Wed, 09 Nov at 10h into the mailbox next to 34-401.1.
Please note: in the graphical representation of Petri net we use here boxes $\boxed{t}$ for transitions.

Problem 1: Family of Generating Traps

Add arcs to the Petri net N below so that its family of generating traps contains exponentially (in N's size) many traps. Once added, describe $N = (S, T, W)$ formally and prove that the family of generating traps is exponential in N's size.

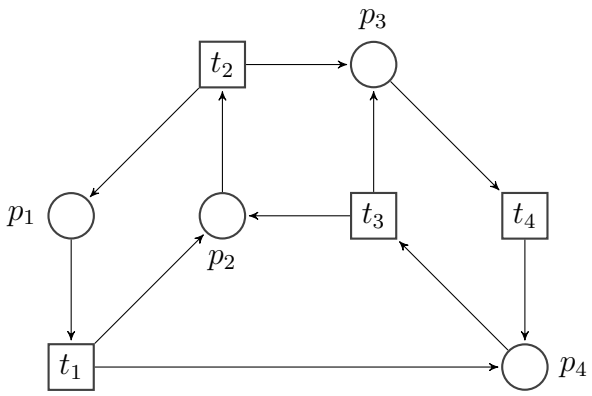
Does the generating family of traps for N contain only minimal traps? Argument your answer.

Problem 2: Invariants for Petri Nets

Let $N = (S, T, W)$ be a Petri net.

a) prove that if I and J are structural invariants of N, so are $I + J$ and $k \cdot I$ ($\forall k \in \mathbb{Z}$).

b) compute a basis of structural invariants for the following net:



Problem 3: Concurrent Boolean Programs

Consider the following syntax definition

$$\begin{aligned} P ::= {} & \mathbf{skip} \mid x := b \mid P_1; P_2 \mid \mathbf{if}(x)\{P\}\mathbf{else}\{P\} \mid \\ & \mathbf{while}(x)\{P\} \mid \mathbf{fork}\{P\} \mid \mathbf{acquire}(\ell) \mid \mathbf{release}(\ell) \\ b ::= {} & 0 \mid 1 \mid x \mid * \mid x \,\mathbf{and}\, y \mid \mathbf{not}\, x \end{aligned}$$

Programs P can make use of boolean variables $x, y, \ldots$, where 1 stands for true and 0 for false; a separate set of variables $\ell_1, \ell_2, \ldots$ represent locks: they have no value and can only be used by $\mathbf{acquire}$ and $\mathbf{release}$. Initially, all the variables are 0 and the locks are released. The semantics of the language is the expected one, we describe the non standard constructs:

- A program $\mathbf{fork}\{P\}; P'$ will start a new 'thread' running P and continue with P' without waiting for P to terminate.
- A program $\mathbf{acquire}(\ell); P$ will try to acquire the lock ℓ: if the lock is acquired by some other thread the program will block. If at some point the lock is released then the acquire may proceed and P can be executed.
- Similarly, executing $\mathbf{release}(\ell)$ should make the lock available again; you can assume the program never releases a released lock.

As demonstrated in the lecture, we can associate a Petri net semantics to the language.

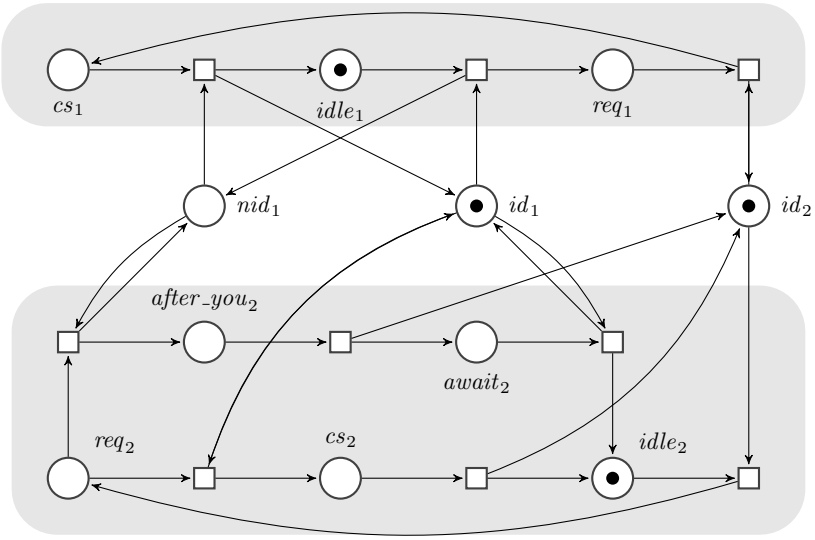
a) Provide a translation to Petri nets of the syntax constructs not seen in the lecture.

b) Explain why we are unable to construct a Petri net if we introduce the concept of variables that are private to a single thread.

c) Propose an extension of the language (and its translation to Petri nets) where we have variables $c_1, c_2, \ldots$ that can take values in $\mathbb{N}$. How can you represent them? Which operations are easy to support? Which operations are problematic for the translation to Petri nets?

Problem 4: Lamport's Mutual Exclusion Algorithm

Consider the Petri net below, describing Lamport's 1-bit mutual exclusion algorithm.



In this exercise, we want to show that both processes (delimited by the gray regions) never enter their critical section (cs_1 and cs_2) at the same time, i.e. that a marking M with $M(cs_1) = M(cs_2) = M(nid_1) = 1$ is not reachable.

a) State the corresponding BVS and prove that it is feasible by finding a marking that is a solution.

b) Find a trap that is initially marked but not marked by the solution found in 4a).

The Enhanced Verification System incorporates the trap property check for all traps at once. A simpler approach could be to solve the BVS and, in case it is feasible, find a trap Q that is initially marked but not marked by some solution. Then we can add some inequalities to the BVS that ensure the trap property only for Q.

c) Show how to refine the BVS of item 4a) with the trap of item 4b). Is it feasible now? What does this imply for mutual exclusion?

d) You can imagine this process can be iterated until the system becomes infeasible or we are inconclusive. What is the drawback of this approach in comparison to the Enhanced Verification System seen in the lecture?