Exercise 1 True or False? (2 points)

1. There are transition systems with finite number of states, but have countably infinite number of executions? Justify your answer.

2. There are transition systems with finite number of states, but have uncountably infinite number of executions? Justify your answer.

Exercise 2 (4 points)
There are three light bulbs in a room and there are three toggle switches outside the room. Each switch operates exactly one light bulb. Initially all bulbs are switched off. Toggling a switch either turns on a bulb if it was off or turns it off if it was on.

1. Define the behaviour of the three light bulbs and their switches as a transition system \((S, \text{Act}, \rightarrow, s_0, \text{AP}, L)\). (You can define the set of states and transition relation in a set builder notation, instead of drawing the entire transition system.)

2. How many states are reachable from the initial state?

3. Does every execution of your transition system defines a valid behaviour? Justify your answer.

Exercise 3 (4 points)
A concurrent system comprises of competing processes \(P_1, \ldots, P_n\) (without shared memory) that access common resources within their critical sections. We assume that the resources may only be accessed exclusively and that \(k\) equivalent instances are available.

Further, let \(n, k \in \mathbb{N}\) with \(2 \leq k \leq n\).

Process \(P_i\) can be described by a transition system \(T_i\) (Figure 1) with three states and the actions request, enter and release as indicated on the right.

a) Develop a transition system representation of an arbiter that communicates with the processes using actions request and release. The arbiter should assure that there are no more than \(k\) processes within their critical section at the same time.

![Figure 1: The process \(T_i\).](image-url)
b) Sketch the transition system of the parallel composition

\[(T_1 ||| T_2 ||| T_3) \parallel_{Syn} \text{Arbiter}\]

with \(Syn = \{\text{request, release}\}\) for \(k = 2\). You need not consider the states \(wait_i\).