Software Validation and Verification
First Exercise Sheet

Exercise 1

In the following, whenever transition systems are compared via $=$ or $\neq$, this means (in)equality up to renaming of states (i.e. isomorphism). You can therefore safely assume that pairs made of a pair and an element are equal to pairs made of an element and a pair: $\langle\langle x, y\rangle, z\rangle = \langle x, \langle y, z\rangle\rangle = \langle x, y, z\rangle$.

1. Show that the handshaking operator $\|_\_$ is **not** associative, i.e. it is not true that for any sets of actions H, H', and for any transition systems $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$, the following holds.
$$(\mathcal{T}_1 \|_H \mathcal{T}_2) \|_{H'} \mathcal{T}_3 = \mathcal{T}_1 \|_H (\mathcal{T}_2 \|_{H'} \mathcal{T}_3)$$
2. Show that the handshaking operator $\|_\_$ is associative when the synchronization set is the same for both occurrences, i.e. that for any set of actions H, and for any transition systems $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$, the following holds.
$$(\mathcal{T}_1 \|_H \mathcal{T}_2) \|_H \mathcal{T}_3 = \mathcal{T}_1 \|_H (\mathcal{T}_2 \|_H \mathcal{T}_3)$$
3. Show that the handshaking operator $\|$ that forces transition systems to synchronize over their common actions is associative, i.e. that for any transition systems $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$, the following holds.
$$(\mathcal{T}_1 \| \mathcal{T}_2) \| \mathcal{T}_3 = \mathcal{T}_1 \| (\mathcal{T}_2 \| \mathcal{T}_3)$$
(Recall that $\mathcal{T} \| \mathcal{T}'$ is defined as $\mathcal{T} \|_{Act \cap Act'} \mathcal{T}'$, with Act and Act' the actions of $\mathcal{T}$ and $\mathcal{T}'$ respectively)

Exercise 2

Consider the following mutual exclusion algorithm with shared variables y1 and y2 (both initially at 0).

Process P1

```
while true do
  ... noncritical section ...
  y1 := y2 + 1;
  wait until (y2 = 0) or (y1 < y2)
  ... critical section ...
  y1 := 0;
od
```

Process P2

```
while true do
  ... noncritical section ...
  y2 := y1 + 1;
  wait until (y1 = 0) or (y2 < y1)
  ... critical section ...
  y2 := 0;
od
```

1. Give the program graphs $\mathcal{P}_1$ and $\mathcal{P}_2$ representing the processes. (A pictorial representation suffices, and you can use a single node for representing each of the critical and noncritical sections.)
2. Give the reachable part of the transition system of $\mathcal{P}_1 \mathbin{|||} \mathcal{P}_2$ where $\texttt{y1} \leq 2$ and $\texttt{y2} \leq 2$.
3. Does the algorithm ensures mutual exclusion?

Exercise 3

In the following, we denote with $\mathcal{T}_\mathcal{P}$ the transition system of the program graph $\mathcal{P}$. Moreover, we will say that a transition system is infinite if the set of states reachable from the initial ones is an infinite set.

Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two program graphs, discuss the validity of the following statements:

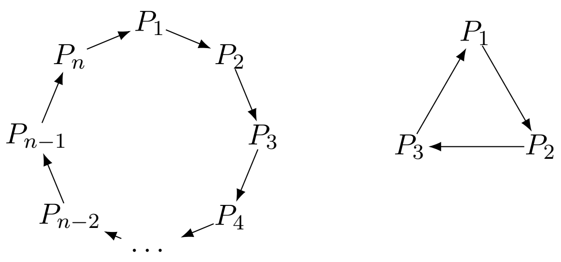
1. if $\mathcal{T}_{\mathcal{P}_1} \mathbin{|||} \mathcal{T}_{\mathcal{P}_2}$ is infinite then also $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$ is infinite;
2. if $\mathcal{T}_{\mathcal{P}_1 ||| \mathcal{P}_2}$ is infinite then also $\mathcal{T}_{\mathcal{P}_1} \mathbin{|||} \mathcal{T}_{\mathcal{P}_2}$ is infinite.

Hint: For the first point recall that the full definition of *program graph* has more than just states and transitions.

Exercise 4

Consider the following leader election algorithm: For $n \in \mathbb{N}$, n processes $P_1, \ldots, P_n$ are located in a ring topology where each process is connected by an unidirectional, asynchronous channel to its neighbour as outlined below.

In General **For Three Processes**



Each process P_i is assigned a unique identifier $id(P_i) \in \mathbb{N}$ and has a private variable containing the identifier of the process currently assumed to be the leader. We name this variable `l1` for the process P_1, `l2` for P_2, and so on, and we assume that each process initially considers itself the leader, thus each `li` is initialized to $id(P_i)$. The aim of the algorithm is to elect the process with the highest identifier as the (unique) leader within the ring, i.e. all the variables `l1`, `l2`, ..., `ln` must converge to the maximum $id(P_i)$. Each process P_i executes the same algorithm and it continuously performs two operations: (i) it sends its current leader (stored in `li`) on its output channel; and (ii) upon receiving messages over its input channel, the program stores the received value into another private variable `xi` (initially set to 0), and updates `li` if the received id is higher.

1. Model the protocol described above with three processes as a channel system $[\mathcal{P}_1|\mathcal{P}_2|\mathcal{P}_3]$;

2. Write an *initial execution* of the transition system $\mathcal{T}_{[\mathcal{P}_1|\mathcal{P}_2|\mathcal{P}_3]}$ where the three processes converge to a common leader, assuming channels have capacity 1 and $id(P_i) = i$ for each process P_i;

3. Modify the channel system so that all channels are faulty (i.e. they may nondeterministically discard a message instead of delivering it), to do so, define a program graph $\mathcal{P}_f$ such that $[\mathcal{P}_1|\mathcal{P}_2|\mathcal{P}_3|\mathcal{P}_f]$ models a system similar to the previous one but where the channels are not reliable.

Recall: An *initial execution* for a transition system $\mathcal{T}$ is an alternating sequence of states and actions $s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \ldots \xrightarrow{\alpha_n} s_n$ with s_0 an initial state and $\rightarrow$ the transition relation of $\mathcal{T}$.