Marcel ZAUDER 16-124-836
Pascal GERIG 16-104-721

11.1 Several Questions: Petri Nets

a) *List and briefly explain all the elements a Petri net consists of.*
A Petri net consists of the following elements:
   (i) A finite set P of places, which can hold tokens.
   (ii) A finite set T of transitions, which generate token when a certain requirement of tokens input from different places is reached.
   (iii) An Input function which lists the requirements for transitions to be able to "fire".
   (iv) An Output function which lists the output for every transition when itis fired.
   (v) (An initial state, called marking, which defines in which places tokens are already available.) - is not part of a PetriNet but should be also available.

b) *What is the reachability set of a net?*
The reachability set of a PetriNet is the set of all markings (different placements of tokens) that can be reached from a certain initial marking state (similar to the number of possible states that can be reached).

c) *How can you compute the reachability set of a net?*
If a Petri net is not k-bounded the reachability set would be undefined, or better said it has infinite states that can be reached.
If it is k-bounded we can construct a "reachability graph" by iteratively adding all markings, step by step, that are reachable from the previous state.

d) *Write a condition in natural language which guarantees that a net is bounded.*
There exists no cycle in the PetriNet which has a net-positive output of tokens, so that when executing the cycle once we have at most the same amount of tokens as before executing it.

11.2 Petri Nets Examples

a) *Provide the definition of the Petri net in figure 1.*

$$P = \{v, w, x, y, z\} \qquad T = \{a, b, c, d\}$$
$$I(a) = \{v, w\} \quad I(b) = \{w, x\} \quad I(c) = \{y\} \quad I(d) = \{z\}$$
$$O(a) = \{y\} \quad O(b) = \{z\} \quad O(c) = \{v, w\} \quad O(d) = \{w, x\}$$
$$m = \{w, x, x, y, y, y\}$$

b) *Provide the definition of the Petri net in figure 2.*

$$P = \{a, b, c, d\} \qquad T = \{x, y\}$$
$$I(x) = \{a, b\} \qquad I(y) = \{d, c\}$$
$$O(x) = \{b, d, c\} \qquad O(y) = \{b\}$$
$$m = \{a, a, b\}$$

c) *Is the Petri net in Figure 2 bounded?*
Yes it is 3-bounded because *x* can fire 2-times, therefore 1 token is in **b**, **d**, and **c** and then *y* can fire 2-times such that in **b** there are 3 tokens and the PetriNet is now deadlocked.

d) *Is the Petri net in Figure 2 safe?*
No, it is not safe, because after *x* was fired 2-times, **a** does not have any tokens left and therefore *x* cannot fire again, therfore the PetriNets will get into a deadlocked state when **d** and **c** are also empty, which will eventually be the case after *y* was fired also 2-times.

e) *Is the Petri net in Figure 2 conservative?*
No, it is not conservative because x requires two tokens in order to fire, but generates 3.

f) *Are all the transitions live in the Petri net in Figure 2?*
At this point in time not all transitions are live because *y* cannot fire as **d** and **c** do not have any tokens. In order to be life *x* must have fired at least once to fill **d** and **c**.

Marcel ZAUDER 16-124-836
Pascal GERIG 16-104-721

11.3 Several Questions: Lock Objects and Threads

a) *How can you enable fairness in the Java class Semaphore?*
When instantiating a semaphore one can give a second argument after the integer which indicates the number of semaphores that exists, which is either *true* or *false*. If it is *true* the fairness for the semaphore is enabled:

```
private final Semaphore semaphore = new Semaphore(MAX_AVAILABLE, true);
```

b) *Which fairness strategy does the Java class Semaphore support?*
When a semaphore/lock is released the thread with the longest waiting time is given the semaphore/lock (FIFO processing).

c) *What is the purpose of daemon threads in Java?*
Daemon threads are low-priority threads whose only role is to provide services to user threads. Because of that daemon threads will not prevent the JVM from exiting once all user threads have finished their execution.

d) *How can you create a daemon thread in Java?*
After a thread was created which should use a *daemon thread*, we can run the following code:

```
NewThread daemonThread = new NewThread();
daemonThread.setDaemon(true);
daemonThread.start();
```