Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Leonel Merino

Solution
Series 11 — 29.11.2017 – v1.0c
Petri Net

Exercise 1 (3 Points)

Answer the following questions about Petri nets:

a) List *and briefly explain* all the elements a Petri net consists of. **Answer:**

A Petri net C is defined as

$$C = \langle P, T, I, O \rangle \text{ with marking function } m$$

in which:

i. *P is a finite set of places*
ii. *T is a finite set of transitions*
iii. *I is an input function*
$$I : T \to Nat^P$$
iv. *O is an output function*
$$O : T \to Nat^P$$
v. *m is a marking function of C that mapps tokens to individual places*
$$m : P \to Nat$$

Petri nets are directed bigraph representations with two types of vertices, i.e., places and transistions, in combination with tokens. Places can have tokens assigned. Transitions are are able to split, merge and pass tokens. An input function takes as argument a transition and returns a set containing all the incoming edges. An output function takes as argument a transition and returns a set containing all the outgoing edges. The marking function specifies in which places tokens reside, i.e., are assigned to.

b) How can nets model concurrency and synchronization? **Answer:**

Transitions can represent competing processes, and places can represent resources, with tokens (markings) indicating the availability of a resource. But a process may also correspond to a subnet, with places representing the state of a process. Tokens can then represent control flow, or data flow, or synchronization conditions. Semaphores, for example, could be modeled as a transition with as many incoming edges as the semaphore has permits.

c) What is the reachability set of a net? How can you compute this set? **Answer:**

The reachability set R(C, μ) of a net C is the set of all markings μ' reachable from the initial marking m. There exist various solutions to compute that set. For example, one can build a tree of all transitions and collect the subsets of states by layers. A layer can be defined as the subset of elements that are reachable from the root using the same number of transitions.

d) What kinds of Petri nets can be modeled by finite state processes? **Answer:**

Petri nets that are bounded can be modeled using FSP. If it is not bounded you would need an infinite number of states.

e) What are some simple conditions for guaranteeing that a net is bounded? **Answer:**

One simple condition would be "all transitions in the Petri net have the same number of incoming and outgoing edges". Another could be "the number of outgoing edges is lesser that the number of incoming edges".

f) What could you add to Petri nets to make them Turing-complete? **Answer:**

An imperative language is Turing complete if it has conditional branching (e.g., "if" and "goto" statements, or a "branch if zero" instruction. In conclusion, adding zero tests to the definition would be the easiest way to make the Petri nets Turing-complete.

Exercise 2 (3 Points)

Perform some analysis on the provided Petri nets:

a) Provide the definition of the Petri net in figure 2. **Answer:**

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

b) Provide the definition of the Petri net in figure 3. **Answer:**

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

c) Is the Petri net in Figure 3 bounded? Safe? Conservative? Are all the transitions live? **Answer:**

The petri net in figure 3 is 3-bounded, because places never hold more than 3 tokens. This also implies that it is not safe. It's not conservative because the number of tokens is not constant. All transitions are not live, because if you hit x then y twice the net will be deadlocked.

Exercise 3 (3 Points)

Two machines need to interact with a database. The machines can read, write or stay idle. Model the environment using Petri nets ensuring that the machines cannot write at the same time. Use the Petri net editor from the web site of the course[1]. Hand-drawn Petri net graphs are acceptable, but make them readable, please! **Answer:**

See figure 1.

Exercise 4 (1 Points)

Answer the following questions about lock objects and threads:

a) How do the classes ReentrantLock and Semaphore support fairness?
Hint: You may have to look at the Java documentation. **Answer:**

The constructors ReentrantLock(boolean fair) and Semaphore(int permits, boolean fair) provide parameters to enable fairness. This fairness implementation is based on the FIFO concept and notifies the longest waiting thread first.

b) What are daemon threads in Java? What is their purpose? How can you create them? **Answer:**

Daemon threads are very much like traditional threads, besides the fact that they will transition into the TERMINATED state as soon as all user threads terminated. They should perferrably be used for stateless or non-critical operations, e.g., logging and monitoring purposes, because the termination could happen unpredictably. Daemon threads can be created by calling setDaemon(true) on a thread that is still in the NEW state.

[1] http://scg.unibe.ch/download/petitpetri/

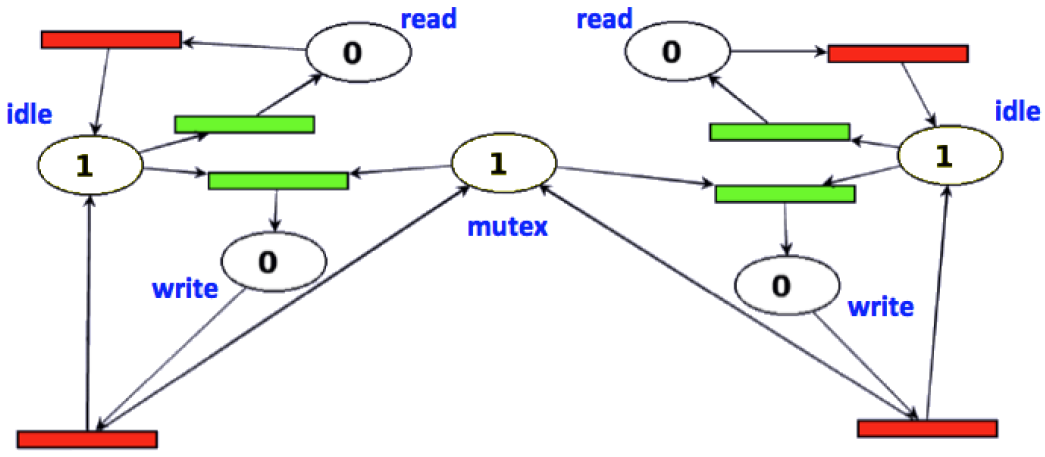

Figure 1: Petri net exercise 3 solution

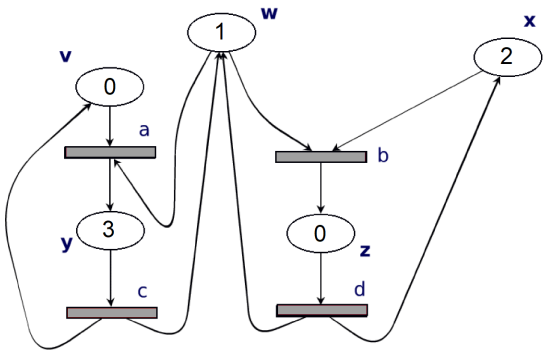

Figure 2: Sample Petri net

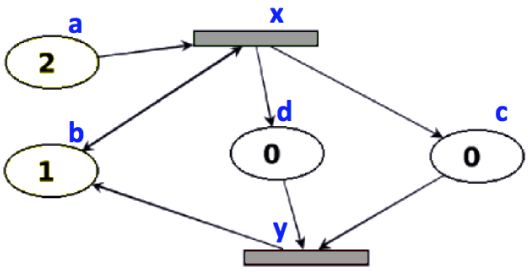

Figure 3: Another sample Petri net