

SOFTWARETECHNIK SS01

ASSIGNMENT 5

**Exercise 8:** (10 Points)

A ferryman is to take a goat, a wolf and a cabbage across a river. The ferry can transport only the ferryman and one other object. Use a Petri net to represent the organization of this transport problem under the constraint that neither the wolf and the goat nor the goat and the cabbage are left alone at a bank of the river.

**Exercise 9:** (10 Points)

*Mutual exclusion* (mutex) is a famous problem from concurrent programming. There is an infinite loop in which  $n$  processes execute a sequence of instructions partitioned into a *critical section* and a *non-critical section*. The program must satisfy the *mutual exclusion property*: instructions from the critical sections of two or more processes must not be interleaved. *Dekker's algorithm* (see next page) solves the mutex problem for two processes. In addition to the critical and non-critical section, each process executes a *pre-protocol* and a *post-protocol* for entering or leaving the critical section.

Use state chart diagrams to model the algorithm. Convince yourself (you don't need to discuss this in your solution) that the mutex-property holds and that every process that tries to enter the critical section will eventually succeed. Assume that no process can halt during the pre- and post-protocol and in the critical section.

**Exercise 10:** (10 Points)

A binary tree whose nodes and leaves are labeled by integers is inductively defined as follows.

`Tree ::= leaf Integer | node Integer (Tree Tree)`

Use the visitor pattern to implement this recursive data type in Java together with a generic visitor operation on trees that is instantiated in two different ways:

- to compute an inorder representation of the tree (using a Java-vector),
- to compute the sum of the values of the labels of the tree.

**Optional Exercise 11:** (10 Points)

Solve exercise 10 in a functional language (e.g. Haskell). Discuss the two different solutions.

Dekker's algorithm:

```
C1,C2: Integer range 0..1 := 1;
Turn : Integer range 1..2 := 1;

task body P1 is
begin
  loop
    Non_Critical_Section_1;
    C1 := 0;
    loop
      exit when C2 = 1;
      if Turn = 2 then
        C1 := 1;
        loop exit when Turn = 1; end loop;
        C1 := 0;
      end if;
    end loop;
    Critical_Section_1;
    C1 := 1;
    Turn := 2;
  end loop;
end P1;

task body P2 is
begin
  loop
    Non_Critical_Section_2;
    C2 := 0;
    loop
      exit when C1 = 1;
      if Turn = 1 then
        C2 := 1;
        loop exit when Turn = 2; end loop;
        C2 := 0;
      end if;
    end loop;
    Critical_Section_2;
    C2 := 1;
    Turn := 1;
  end loop;
end P2;
```

Please check in your solutions either as a postscript or a pdf document by July 05, 2001.