Technische Universität München
Institut für Informatik
Dr. Ralf-Peter Mundani
Michael Lieb, M. Sc.

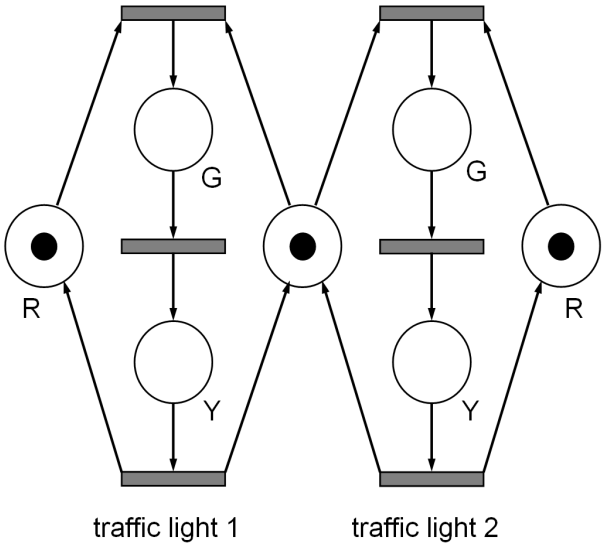
SS 2013

Parallel Programming and HPC

Exercise Sheet 4: Loop Dependencies, OpenMP

1 Petri Networks

Both lights have three separate places R(ed), G(reen), Y(ellow) and share another place in the middle. Initially, there are three tokens in the places shown, hence both traffic lights are in state Red. This Petri network assures that a) never both lights are in state Green and b) both lights switch in alternating fashion from Red to Green and (via Yellow) back.



i) Sketch one full switching cycle (i. e. both traffic lights were in state Green) until the initial state is reached again. Are there any configurations that are not reachable? Show your answer in a formal way by using the Parikh vector!

ii) Give an implementation in pseudo code of this Petri network using semaphores. Show that your solution is fair and free of deadlocks!

2 Parallel Computation of π

With

$$\phi(x) = \frac{1}{1+x^2} \qquad \text{and} \qquad \int \phi(x)dx = arctan(x)$$

one could compute π via the integration of $\phi(x)$ over $[0, 1]$. The following code fragment shows how to compute π sequentially, subdividing the unit interval into N stripes.

```
int    i, N;
double  h, x, sum, PI;
h = 1/N;
sum = 0;
for( i=1 ; i<=N ; i++ ) {
     x = h * (i - 0.5);
     sum = sum + 4/(1 + x*x);
}
PI = h*sum;
```

Extend the program with valid OpenMP directives to compute π in parallel and think about sufficient synchronisation of the threads!

3 Parallel Min-Max-Search

To find the minimal and maximal elements **`min`** and **`max`**, resp., of a 3-dimensional integer array **`A`** of size 3×1000×1000, the following sequential code is used:

```
int   i, j, k, A, min, max;
min = A[1][1][1];
max = A[1][1][1];
for( i=1 ; i<=3 ; i++ ) {
     for( j=1 ; j<=1000 ; j++ ) {
          for( k=1 ; k<=1000 ; k++ ) {
               if( A[i][j][k] < min ) { min = A[i][j][k]; }
               if( A[i][j][k] > max ) { max = A[i][j][k]; }
          }
     }
}
```

Extend the program with valid OpenMP directives for the parallelisation of <u>one</u> loop and – if necessary – think about sufficient synchronisation of the threads!