Games with perfect information
Exercise sheet 4

Sebastian Muskalla

TU Braunschweig

Summer term 2018

Out: April 25 (**Updated** April 27)

Due: May 2

Submit your solutions on Wednesday, May 2, **at the beginning of** the lecture.
Please submit in groups of three persons.

Exercise 1: Double-reachability games

Consider a finite game arena $G = (V_\Box \uplus V_\bigcirc, R)$ without deadlocks and sets $B_1, B_2 \subseteq V$. In the **double-reachability game** $\mathcal{G}$, $\bigcirc$ wins by enforcing that the play visits first B_1 and later B_2. More formally, the winning condition is given by

$$\begin{aligned} win \;:\; Plays_{inf} &\to \{\bigcirc, \Box\} \\ p &\mapsto \begin{cases} \bigcirc, & \text{if } \exists i \in \mathbb{N}\colon p_i \in B_1 \text{ and } \exists j \in \mathbb{N}, j > i\colon p_j \in B_2 \\ \Box, & \text{else.} \end{cases} \end{aligned}$$

a) Present an algorithm that takes a double-reachability game and computes a reachability game $\mathcal{G}'$, i.e. a game arena $G' = (V', R')$ and a winning set $B' \subseteq V'$, with $V \subseteq V'$, such that for all positions $x \in V$: x is winning for $\bigcirc$ in the double-reachability game $\mathcal{G}$ if and only if it is winning for $\bigcirc$ in the reachability game $\mathcal{G}'$. Argue formally that your algorithm is correct.

b) Present an algorithm that directly computes the winning regions of the double-reachability game. Argue that you algorithm is correct.

Exercise 2: Reach-and-stay games

Consider a finite game arena $G = (V_\Box \uplus V_\bigcirc, R)$ without deadlocks and a winning set $B \subseteq V$. In a reachability game, any play that visits B is winning for $\bigcirc$, no matter how it continues after the visit.

In this exercise, we consider **reach-and-stay games**, in which the goal of player $\bigcirc$ is to make the play visit B and stay there forever. More formally, the winning condition is given by

$$\begin{aligned} win \;:\; Plays_{inf} &\to \{\bigcirc, \Box\} \\ p &\mapsto \begin{cases} \bigcirc, & \text{if } \exists i \in \mathbb{N}\colon \forall k \geqslant i\colon p_k \in B, \\ \Box, & \text{else.} \end{cases} \end{aligned}$$

Present an algorithm that takes a finite game arena without deadlocks and the winning set and computes the winning regions of the reach-and-stay game. Argue that it is correct.

Do uniform positional strategies exist?

Hint: First identify the position form which one stays inside B forever.

Exercise 3: An intricate scheduling problem

Consider the set of tasks $\mathcal{T} = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6\}$, where the computation time C_τ, the relative deadline D_τ, and the minimal interarrival time T_τ are given by the following table.

	C_τ	D_τ	T_τ
τ_1	2	2	5
τ_2	1	1	5
τ_3	1	2	6
τ_4	2	4	100
τ_5	2	6	100
τ_6	4	8	100

We assume that we have 2 processors. Recall that the jobs can be freely migrated between processors after each tick, but they have to be processed sequentially, i.e. not both processors can work on the same job during one tick.

Assume that each task generates a job as soon as the minimal interarrival time has elapsed, i.e. all tasks generate a job at time 0, τ_1 and τ_2 generate a job at time 5, τ_3 generates a job a time 6, and so on.

Consider the time interval $[0, 8]$. Show that there is a scheduling of the jobs for this interval that makes no job miss its deadline. Give a graphic representation of your scheduling.

Exercise 4: An intricate scheduling problem II

Consider again the scheduling problem from Exercise 3. Prove that the input is infeasible for MOFST, i.e. online scheduling where we allow the tasks to delay the generation of jobs.

Hint: Towards a contradiction, assume that an online scheduler exists. Show that by time 8, at least one job has missed its deadline. Structure your proof as follows:

- Assume that all tasks generate a job at time 0. Note that this fixes the jobs for the time interval $[0, 5)$, and since the online scheduler has no knowledge when which job will be generated later, fixes a scheduling on the interval.
- For this fixed scheduling, there are two cases:
  - Case 1: The job generated by task τ_5 is not scheduled on any processor in the time interval $(2, 4]$.
  - Case 2: The job generated by task τ_5 is scheduled for at least one step on a processor in the time interval $(2, 4]$.

  Show that for each of the cases, there is a possible generation of jobs that makes a job miss its deadline.