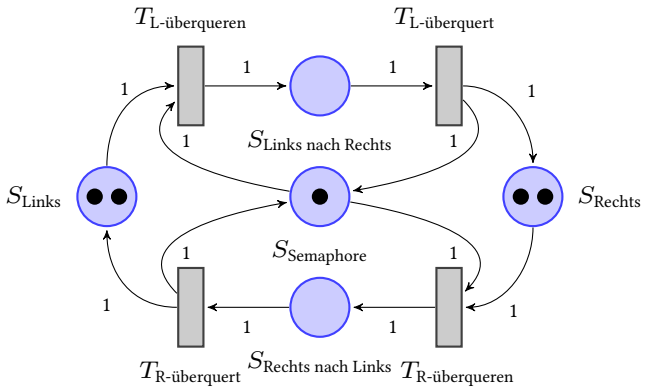
Betriebsysteme (WS19/20)
Übungsblatt 8

Yudong Sun
12141043

16. Dezember 2019

Aufgabe H40 (a)

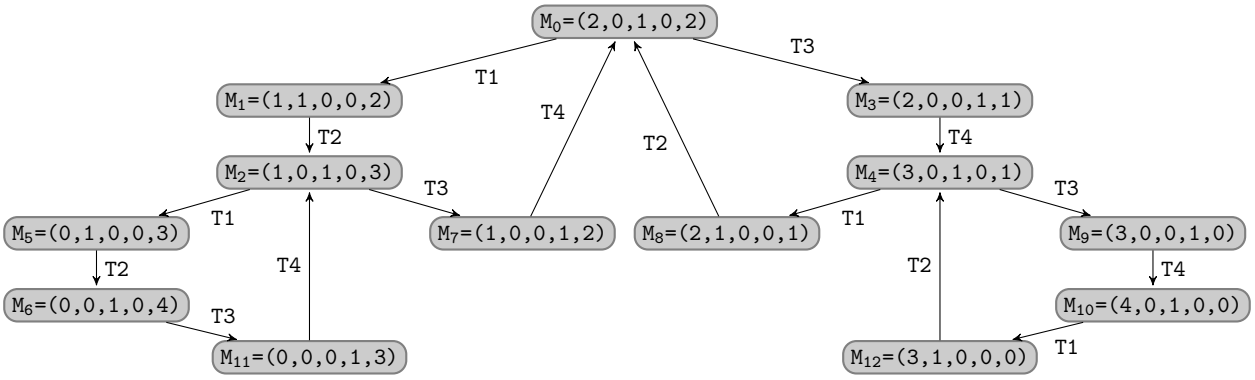


(b) Kennzeichnung:

M_i=(S1,S2,S5,S4,S3)

Petri-Netz	Erreichbarkeitsgraph
S_{Links}	S1
$S_{\text{Links nach Rechts}}$	S2
$S_{\text{Semaphore}}$	S5
S_{Rechts}	S3
$S_{\text{Rechts nach Links}}$	S4

Petri-Netz	Erreichbarkeitsgraph
$T_{\text{L-Überqueren}}$	T1
$T_{\text{L-Überquert}}$	T2
$T_{\text{R-Überqueren}}$	T3
$T_{\text{R-Überquert}}$	T4



(c) Es ist ein faires Petri-Netz. In jedem zyklischen Teilgraphen des Erreichbarkeitsgraphs wird alle Transitionen T_1 bis T_4 bei einem unendlichen Durchlauf unendlich oft geschaltet. Ein Beispiel dafür ist $M_0 \xrightarrow{T_1} M_1 \xrightarrow{T_2} M_2 \xrightarrow{T_3} M_7 \xrightarrow{T_4} M_0$

Aufgabe H41

(a) Wenn es keinen wechselseitigen Ausschluss gibt, kann es zu Inkonsistenzen führen. Beispielsweise können Dateien von P_1 nicht gedruckt werden, wenn P_2 und P_1 das Dateipointer in die Warteschlange (fast) gleichzeitig ändern.

Seien `next = 0` und die Warteschlange leer am Anfang.

aktiver Prozess	ausgeführte Codezeile	Inhalt von `w`	Wert von `next`	Kommentar
P_1	`W[next] = pointer_file1`	`[ptr_1, ...]`	0	P_1 möchte die Datei `ptr_1` drucken.
P_2	`W[next] = pointer_file2`	`[ptr_2, ...]`	0	P_2 möchte die Datei `ptr_2` drucken. w wird überschrieben.
P_2	`next = next + 1`	`[ptr_2, ...]`	1	P_2 incrementiert `next`.
P_1	`next = next + 1`	`[ptr_2, ...]`	2	P_1 incrementiert `next`.

In diesem Ablauf gibt es nur eine Datei in die Warteschlange `w`, obwohl die Variable `next` zeichnet, dass es 2 davon gibt. `w` war von P_2 überschrieben. Die Datei von P_1 wird dann nicht gedruckt.

(b) Wir gehen davon aus, dass es sich nur um 2 Prozessen behandelt.

Prozess P_1

```
...
flags[1] = true;
turn = 2;
while (flags[2] && turn == 2)
    { wait(); }
w[next] = pointer_file1;
next = next + 1
flags[1] = false;
...
```

Prozess P_2

```
...
flags[2] = true;
turn = 1;
while (flags[1] && turn == 1)
    { wait(); }
w[next] = pointer_file2;
next = next + 1
flags[2] = false;
...
```

(c) Es funktioniert bei 2 Prozessen aber skaliert nicht gut bei mehreren Prozessen. Bei mehreren Prozessen ist „Bounded Waiting“ nicht garantiert.

Aufgabe H42 Sehen Sie bitte `u08-h42.txt`