Lehrstuhl für Connected Mobility
School of Computation, Information and Technology
Technische Universität München

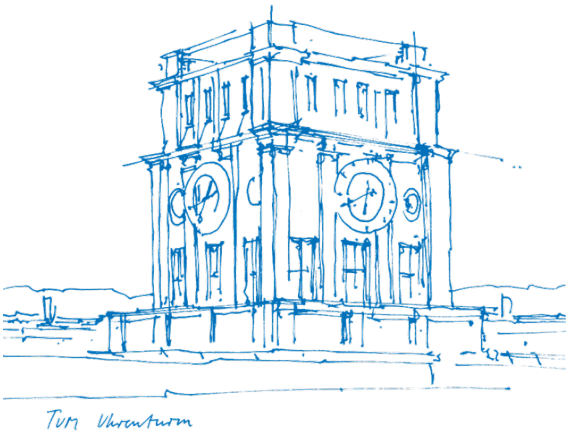
Grundlagen: Betriebssysteme und Systemsoftware

Tutorübung

Mario Delic

Lehrstuhl für Connected Mobility
School of Computation, Information and Technology
Technische Universität München

Übungswoche 6



Petrinetze
Formalitäten

Petrinetz **P = (S,T,F)** is ein Tripel aus Mengen von Stellen S, Transitionen T und Kanten F.

Stellen: Modellieren passive Einheiten (Speicherzellen o.ä.). Dargestellt durch Kreise.

Transitionen: Modellieren aktive Einheiten (Prozesse o.ä.). Dargestellt durch Rechtecke.

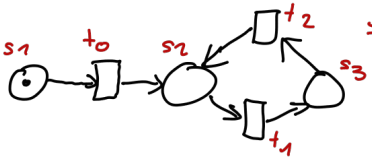
Kanten: $F \subseteq (S \times T) \cup (T \times S) \rightarrow$ Kanten sind gerichtet und führen entweder von einer Stelle S zu einer Transition T, oder von einer T zu einer S. $\rightarrow (S \times S), (T \times T) \notin F$

Kapazität: $c : S \rightarrow \mathbb{N}_0 \cup \{\infty\}$. c(s) = Maximale Anzahl an Tokens, die Stelle s aufnehmen kann.

Belegung: $M : S \rightarrow \mathbb{N}_0 \cup \{\infty\}$. M(s) = Anzahl der Tokens in Stelle s.

Kantengewicht: $w : F \rightarrow \mathbb{N}_0$. w(s,t) = Anzahl der Tokens, die die Kante bzw. Transition t aus der Stelle s erwartet. w(t,s) = Anzahl der Tokens, die die Transition t in Stelle s reinschreibt.

Petrinetze
Eigentschaften



⇒ verklemmungsfrei, fair, aber nicht lebendig da t_0 nie wieder schalten kann, sobald Token s_2 „erreicht"

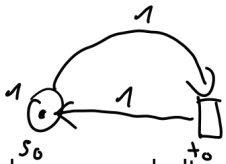
Verklemmt
Ein Petrinetz ist verklemmt, wenn keine Transition mehr schalten kann.

Lebendig
Ein Petrinetz ist lebendig, wenn es keinen Zustand gibts, ab der eine beliebige Transition nie wieder schaltbereit sein wird.

Fair
Ein Petrinetz ist fair, wenn keine Transition verhungern kann.
(Erinnerung Verhungern: Ausführung zwar möglich, könnte aber unendlich lange hinausgezögert/vermieden werden $\rightarrow$ Transition t_x ist schaltbereit, aber es schaltet immer nur $t_y \hookrightarrow t_x$ kann verhungern.)

Nebenläufig
Transitionen sind Nebenläufig, wenn sie unabhängig voneinander Schalten können, ohne sich in der Vor- und Nachbedingung zu beeinflussen.

Petrinetze
Funktionalität



kann nicht schalten!
t0• erfüllt: 1 Token entnehmbar
t0• nicht erfüllt: zu schreibende Stelle voll!
(wenn C(s0) ≤ 2, dann schalten möglich.)
•t0
•t0

- Eine Transition kann nur schalten, wenn alle ihre Vor- und Nachbedingungen erfüllbar sind!

→ Die Stellen aller eingehenden Kanten haben genügend Tokens && die Stellen aller ausgehenden Kanten haben genug freien Platz.

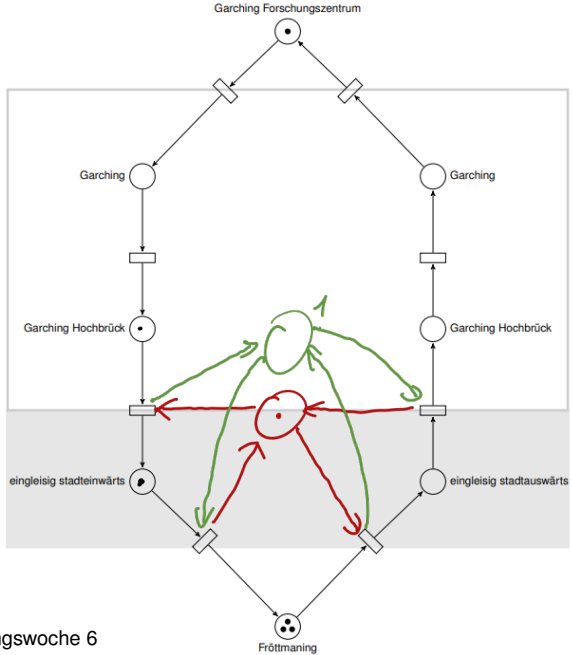
- Es schaltet immer nur eine Transition zu einer Zeit → Mehrere Transitionen schalten nie gleichzeitig.
- Sonderfall Bool'sches Netz: Alle Kapazitäten und Kantengewichte = 1!

- Synchronisation in Petrinetzen:

Kapazitäten und Belegungen reichen i.d.R. nicht aus um ein bestehendes Netz effektiv zu Synchronisieren.

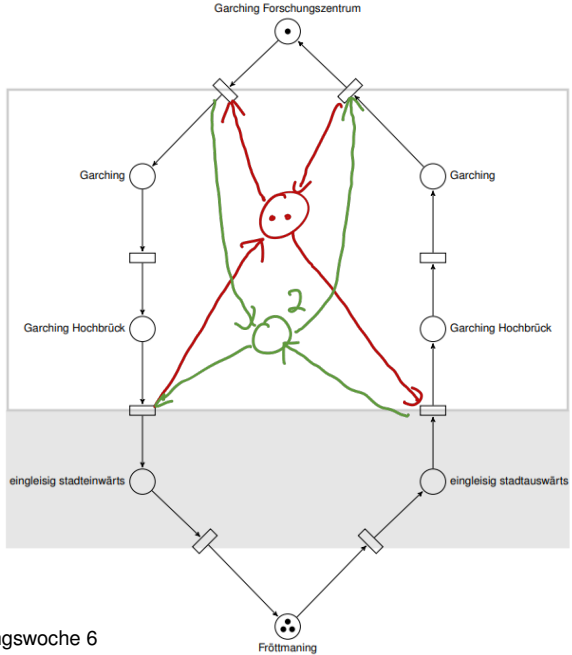
Stattdessen einfacher: 'Deklarieren' einer extra Mutex/Semaphor-Stelle + Verbinden mit allen relevanten Transitionen (um den kritischen Bereich herum). Anschließendes Synchronisieren durch Anpassung der Kapazität/Belegung und der Kantenrichtungen.

3a) Lösung mit Belegung in rot



alternativ mit Kapazität in grün

3b) Lösung mit Beleg.



alternative mit Kap.

Aufgabe 3 a) b) c) (c alternativ auch durch Vorbelegung + Pfeile drehen)