Vorlesung

Methodische Grundlagen des Software-Engineering

im Sommersemester 2014

Prof. Dr. Jan Jürjens

TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

Teil 2.1: Petrinetze

v. 12.05.2014

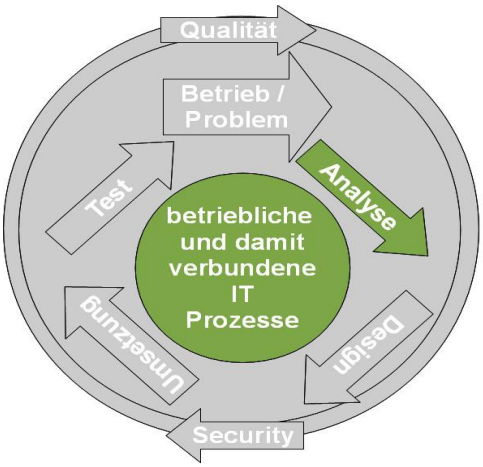
2.1 Petrinetze

[inkl Beiträge von Prof. Volker Gruhn, Jutta Mülle und Dr. Silvia von Stackelberg]

Literatur:
[Rei10] W. Reisig: Petrinetze. Vieweg, 2010. Unibibliothek E-Book:
http://www.ub.tu-dortmund.de/katalog/titel/1305786
Teil I

Einordnung: 2.1 Petrinetze

- Geschäftsprozessmodellierung
- Process-Mining
  - Einführung: Process-Mining
  - Petrinetze
  - Data-Mining
  - Datenbeschaffung
  - Prozessextraktion
  - Konformanzanalyse
  - Mining: Zusätzliche Perspektiven
  - Betriebsunterstützung
  - Werkzeugunterstützung
  - Analysiere „Lasagne Prozesse“
  - Analysiere „Spaghetti Prozesse“
  - Kartographie und Navigation
  - Epilog
- Modellbasierte Entwicklung sicherer Software



Überblick: 2.1 Petrinetze

- Petrinetz-Syntax
- Ausführung
- Analyse von Systemen
- Workflow-Netze

Motivation

Kap. 1: GP-Modellierungsnotationen **EPK**, **BPMN, BPEL**.

- Intendiertes Modellverhalten informell diskutiert.

Automatische Verarbeitung (z.B. Analyse, Simulation) der GP-Modelle benötigt präzise Definition des Ausführungsverhaltens.

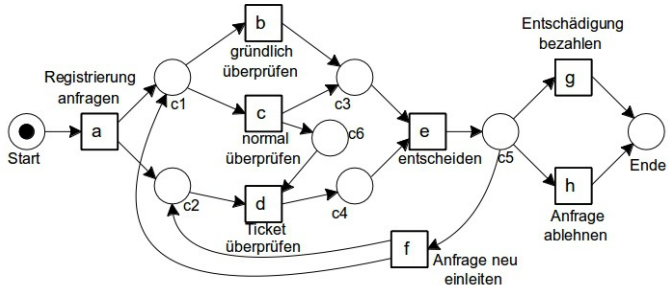
Verschiedene Ansätze dafür vorhanden: Abstract State Machines, Petrinetze, ...

- Z.B.: **Ausführungssemantik** von **UML 2-Aktivitätsdiagrammen** mit Petrinetzen definiert.

Hier: **Petrinetze**.

- Verwendet in zentralem Ansatz für **Process-Mining**.

- Modellierung, Analyse, Simulation von dynamischen Systemen mit **nebenläufigen** und **nichtdeterministischen** Merkmalen.
- Erlauben die Beschreibung von **Kontroll- und Datenfluss**.
- Benannt nach Carl Adam Petri (Dissertation "Kommunikation mit Automaten", 1962).

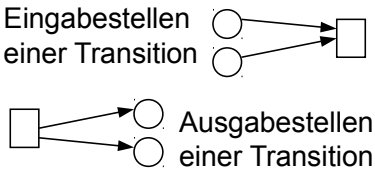


Vorsicht: Existieren heute viele Varianten.

Petrinetz (informell)

Bipartiter gerichteter Graph, besteht aus:

- zwei Sorten von **Knoten**:
  - **Stelle** („S-Elemente“): Zwischenablage von Informationen
  - **Transitionen** („T-Elemente“): Verarbeitung von Informationen
- **Kanten** („Bogen“): verbinden Stellen mit Transitionen (**nie** Stellen mit Stellen oder Transitionen mit Transitionen !).



Stellen mit Objekten (sog. **Token** / **Marken**) belegen.

Durchlauf der Token durch Petrinetz beschreibt **dynamisches Verhalten** des Systems.

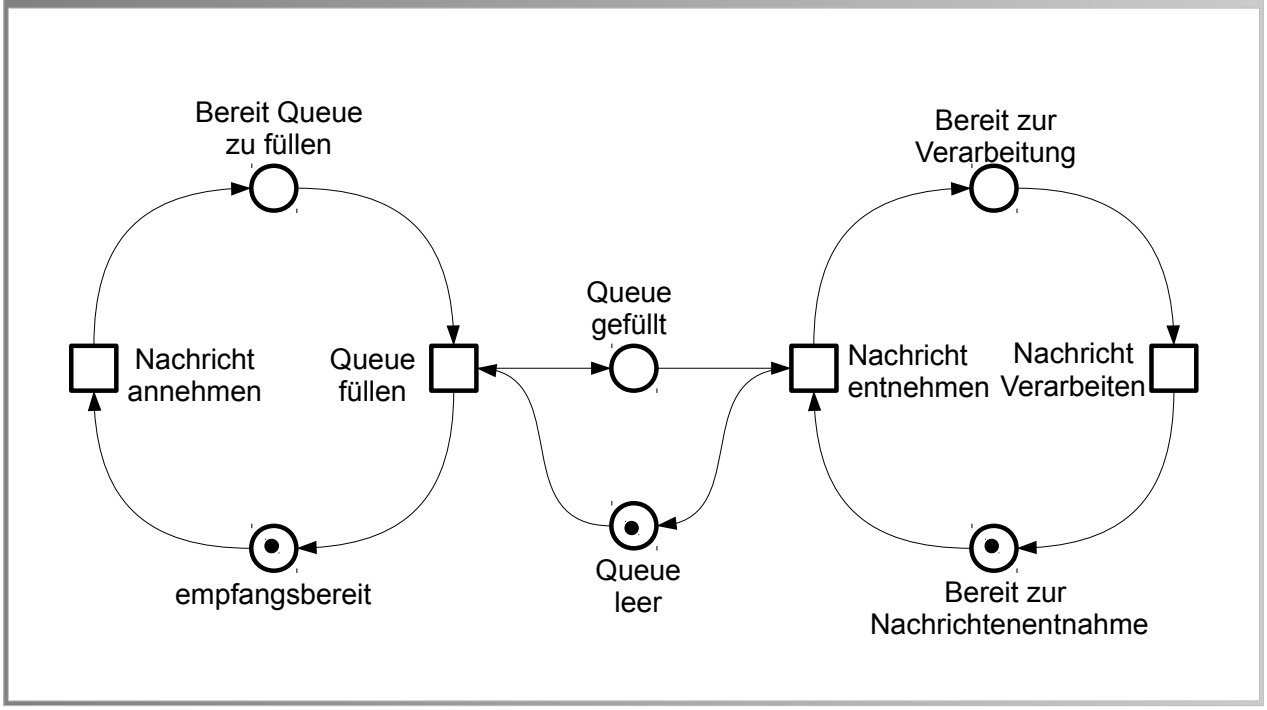

Stelle mit Token

Hier: nur **ungetypte** Marken (=> „**Stellen/Transitions-Netz**“).

Literatur:

W. Reisig: Petrinetze

- Kap. 2.2 (Komponenten eines Netzes), S. 22-23

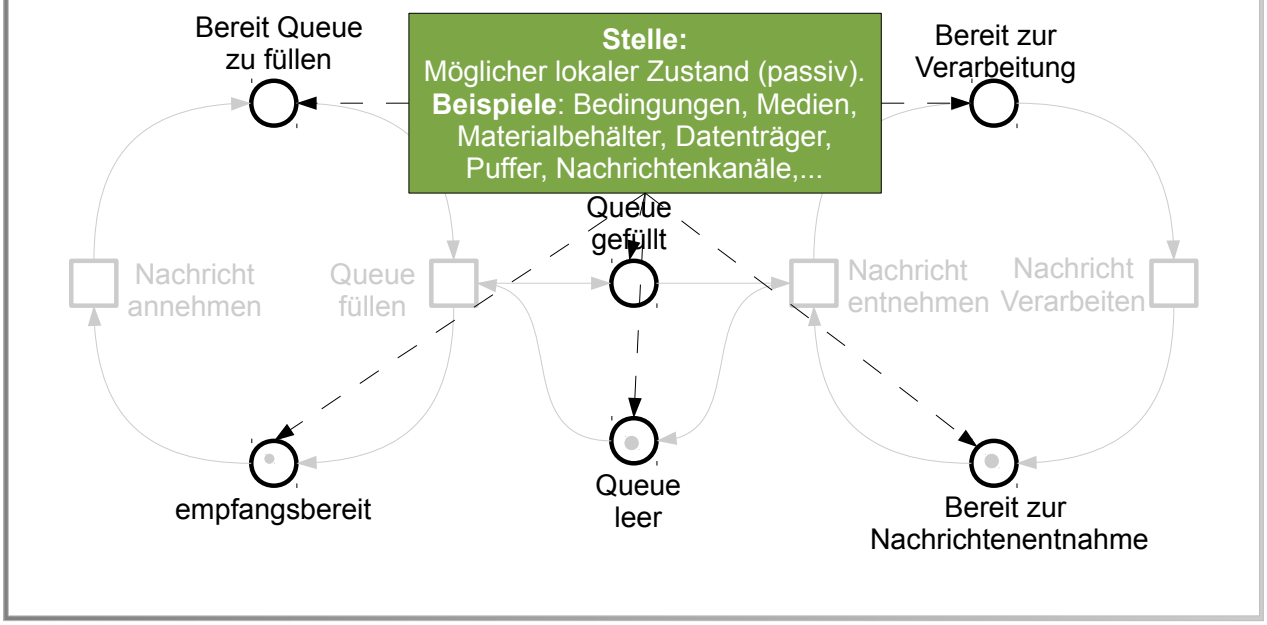


Literatur:

W. Reisig: Petrinetze

- Kap. 2.2 (Komponenten eines Netzes), S. 22-23
- dazu auch Kap. 1 (weiteres einführendes Beispiel), S. 9-18

Petrinetz-Syntax: Stelle

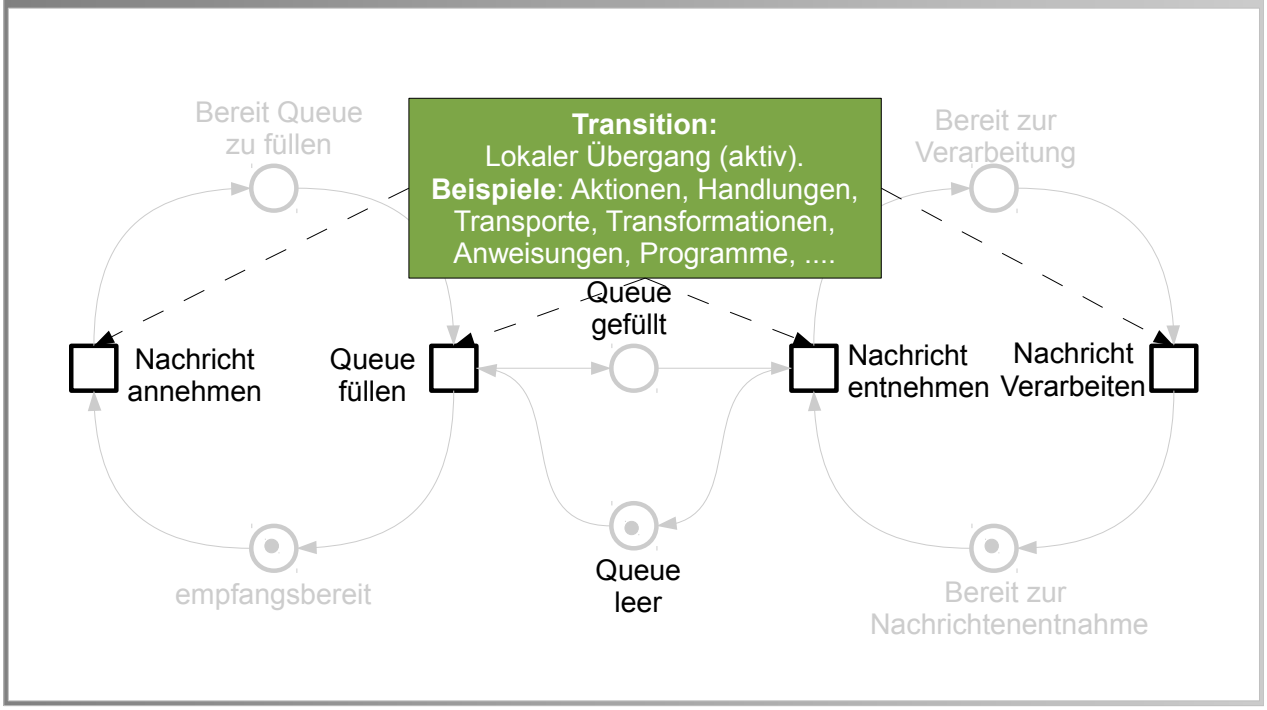


Literatur:

W. Reisig: Petrinetze

- Kap. 2.2 (Komponenten eines Netzes), S. 22-23
- dazu auch Kap. 1 (weiteres einführendes Beispiel), S. 9-18

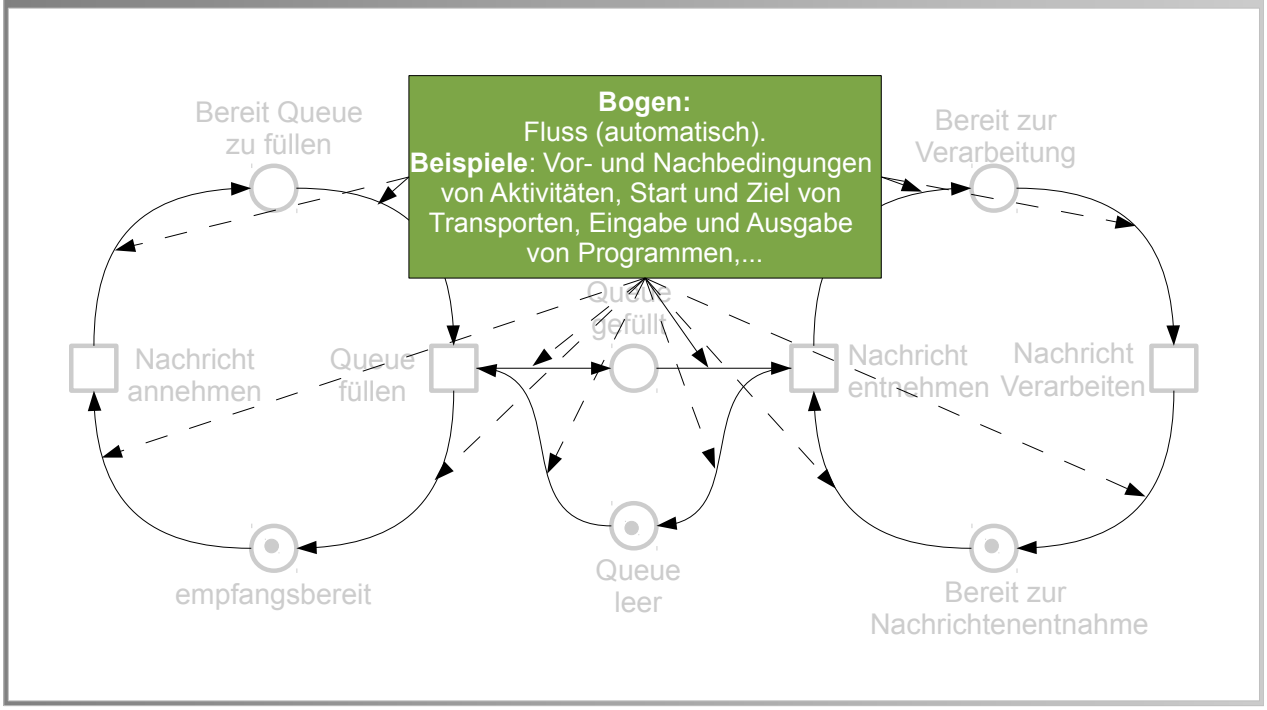
Petrinetz-Syntax: Transition



Literatur:

W. Reisig: Petrinetze

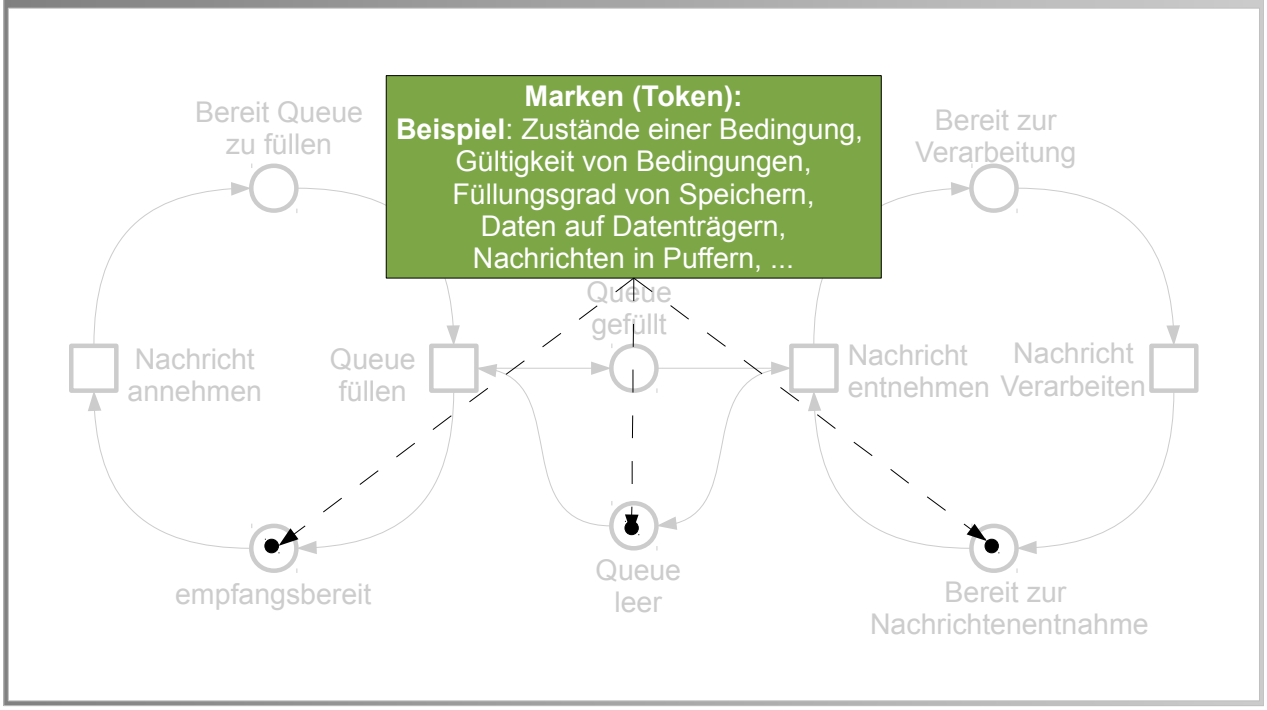
- Kap. 2.2 (Komponenten eines Netzes), S. 22-23
- dazu auch Kap. 1 (weiteres einführendes Beispiel), S. 9-18



Literatur:

W. Reisig: Petrinetze

- Kap. 2.2 (Komponenten eines Netzes), S. 22-23
- dazu auch Kap. 1 (weiteres einführendes Beispiel), S. 9-18

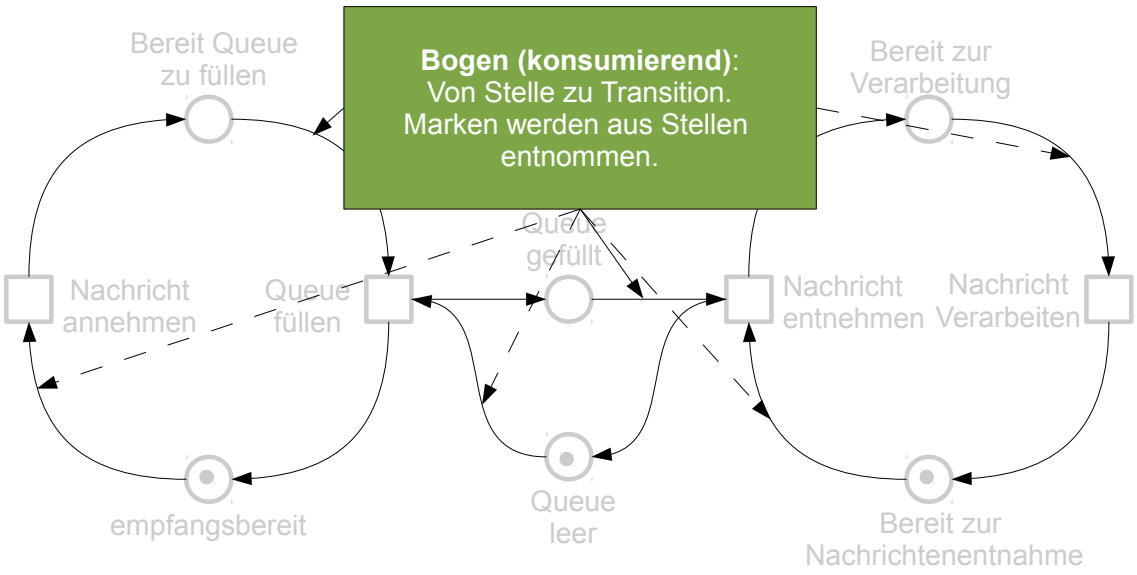


Literatur:

W. Reisig: Petrinetze

- Kap. 2.2 (Komponenten eines Netzes), S. 24
- Weiteres Beispiel Kap. 3.3, S.37-38, Abb. 3.2

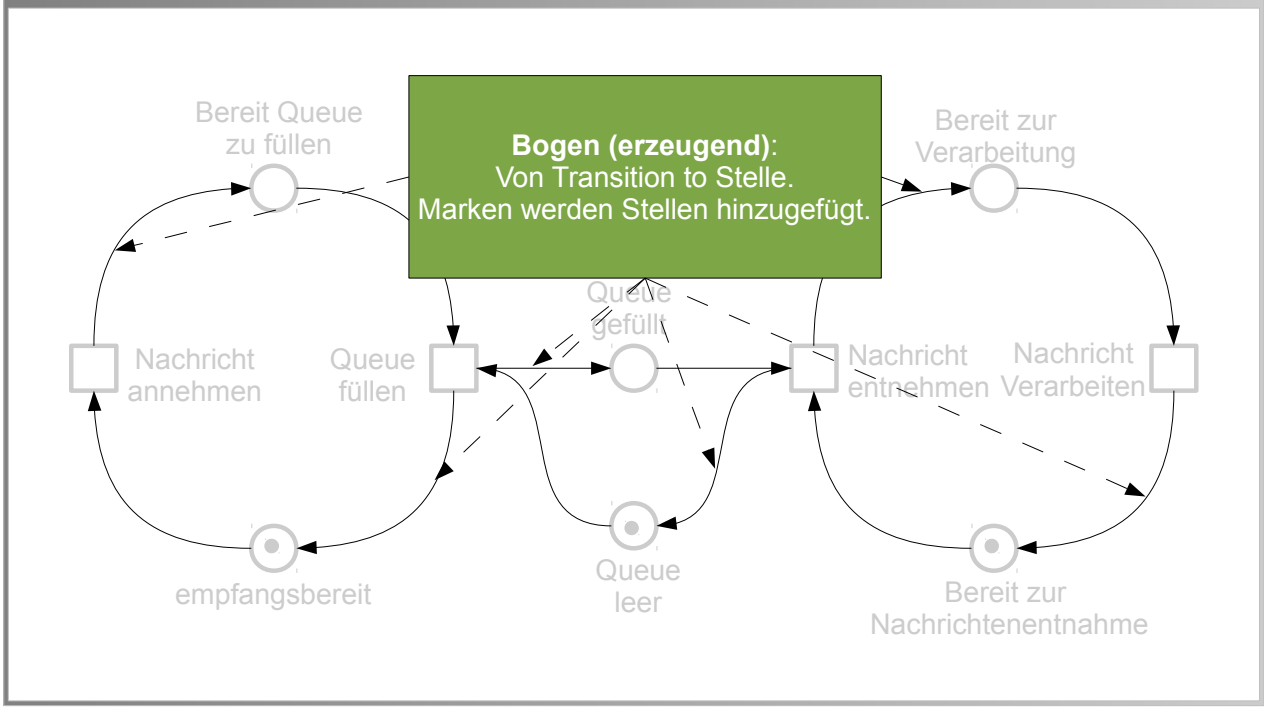
Petrinetz-Syntax: Bogen (konsumierend)



Literatur:

W. Reisig: Petrinetze

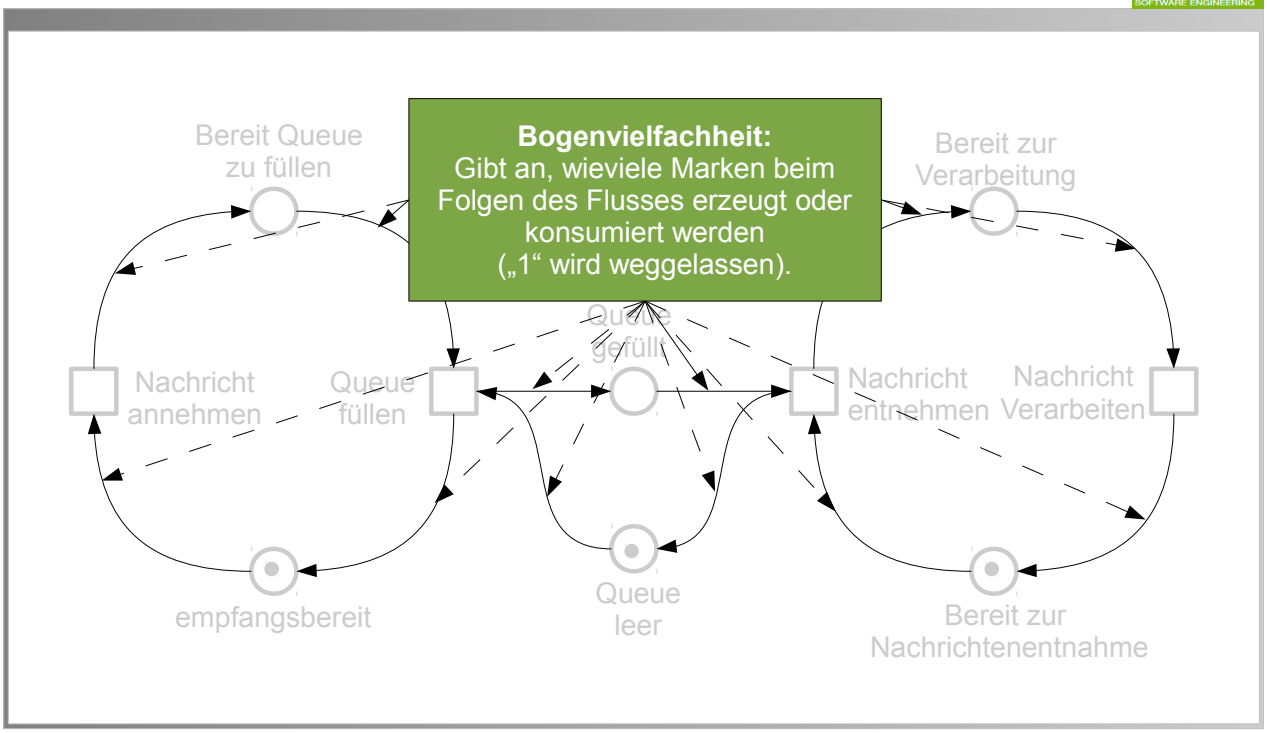
- Weiteres Beispiel Kap. 3.3, S.37-38, Abb. 3.2



Literatur:

W. Reisig: Petrinetze

- Weiteres Beispiel Kap. 3.3, S.37-38, Abb. 3.2



Literatur:

W. Reisig: Petrinetze

- Weiteres Beispiel Kap. 3.3, S.37-38, Abb. 3.2

Syntaxdefinition Petrinetz

Gegeben:

- S: endliche Menge von **Stellen**
- T: endliche Menge von **Transitionen**

mit: **S ≠ ∅, T ≠ ∅ und S ∩ T = ∅**

- F: Menge von **Bögen:**

  $F \subseteq (S \times T) \cup (T \times S)$ binäre Relation

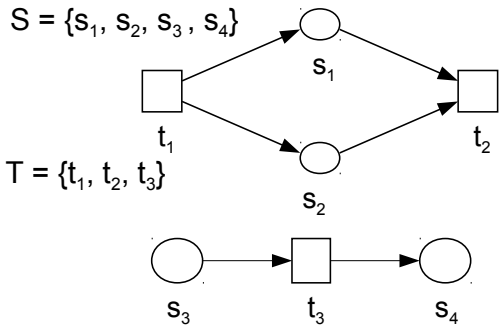
Bogenvielfachheit („Gewicht“) W:

$W : F \to \mathbb{N} \setminus 0$

- **Globaler Startzustand** („Anfangsmarkierung“) M_0:

  $M_0 : S \to \mathbb{N}$

=> (S, T, F, W, M_0): **Petrinetz**.



F = {(t1,s1), (t1,s2), (s1,t2), (s2,t2), (s3,t3), (t3,s4)}

Literatur:

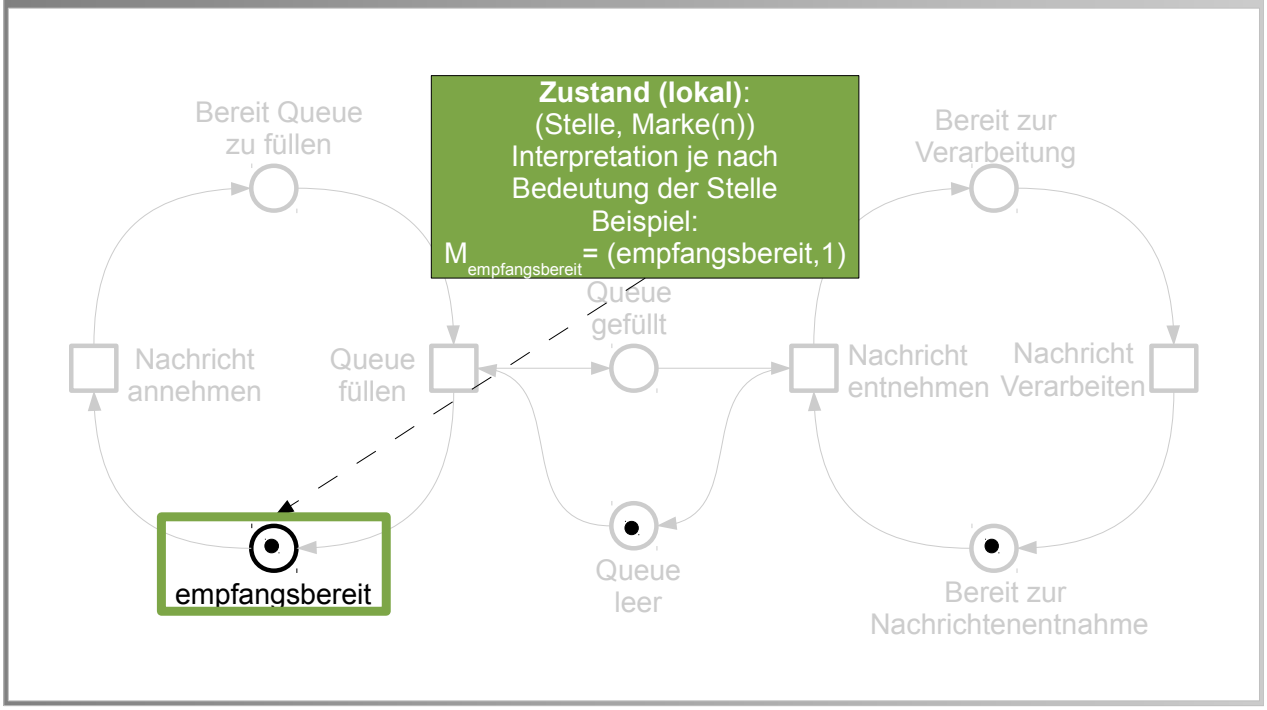
W. Reisig: Petrinetze

- Kap. 2.2 (Komponenten eines Netzes), S. 22-23

Überblick Petrinetze

- Petrinetz-Syntax
- Ausführung
- Analyse von Systemen
- Workflow-Netze

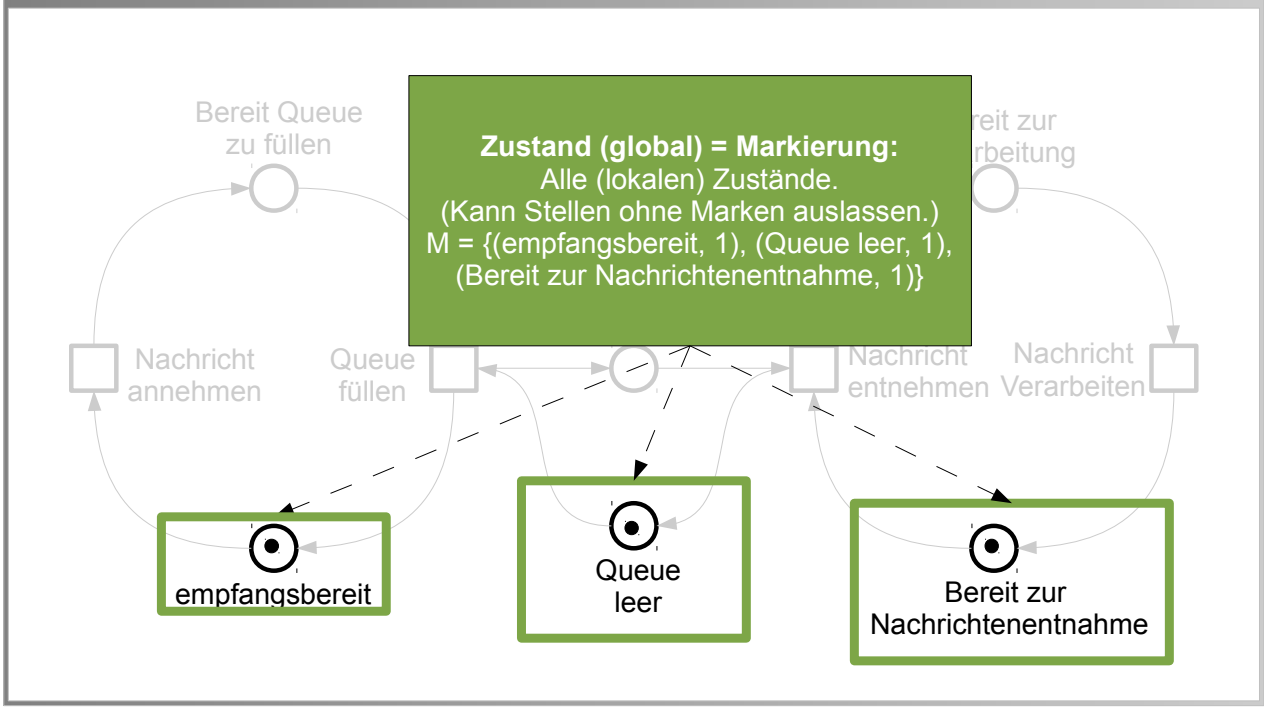
Petrinetz-Ausführung: Zustand (lokal)



Literatur:

W. Reisig: Petrinetze

- Weiteres Beispiel Kap. 3.3, S.37-38, Abb. 3.2



Literatur:

W. Reisig: Petrinetze

- Weiteres Beispiel Kap. 3.3, S.37-38, Abb. 3.2

Markierung: Verteilung Marken auf Stellen (aktueller Systemzustand).

Initiale Markierung: Anfangszustand eines Netzes.

Verhaltenssimulation: evolvierende Anzahl Marken pro Stelle beobachten.

- Basierend auf aktueller Markierung: **aktivierte Transitionen** ermitteln. **Schalten** führt zu Folgemarkierung.
- Unter Folgemarkierung sind (möglicherweise) andere Transitionen aktiviert.
- Solange iterieren, bis keine Transition mehr aktiv (=> „**tote Markierung**“).

Literatur:

W. Reisig: Petrinetze

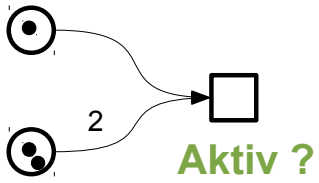
- Kap. 2.5 (Schritt), S. 26-27
- Kap. 2.8 (Erreichbarkeit), S. 31
- Kap. 3.1 (Erreichbarkeit), S. 35-36

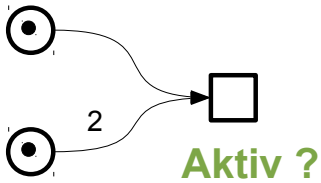
Ablauf: Aktivierung von Transition

Transition t ist **aktiviert**, wenn:

$\forall\ p \in Vorgänger\ von\ t : W(p,t) \leq m_p$

- *W((p,t))*: Gewicht des Bogens von p nach t
- m_p: Anzahl Marken auf p





Literatur:

W. Reisig: Petrinetze

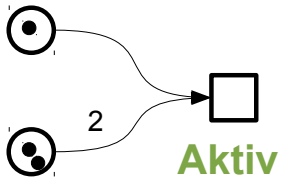
- Kap. 6.1, S. 73-74

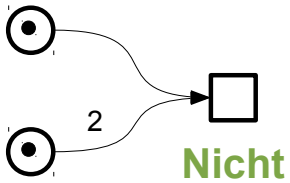
Ablauf: Aktivierung von Transition

Transition t ist **aktiviert**, wenn:

$\forall\ p \in Vorgänger\ von\ t : W(p,t) \leq m_p$

- *W((p,t))*: Gewicht des Bogens von p nach t
- m_p: Anzahl Marken auf p





Literatur:

W. Reisig: Petrinetze

- Kap. 6.1, S. 73-74

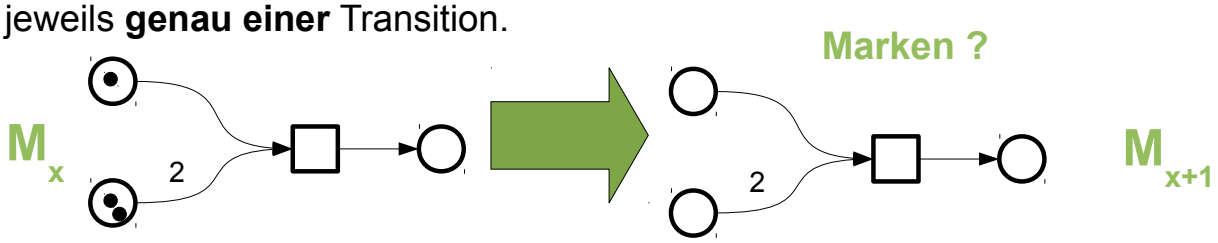
Ablauf: Schalten von Transition

Eine der aktivierten Transitionen wird beim Übergang von Zustand M_x nach Zustand M_{x+1} geschaltet (**nicht-deterministische Auswahl**):

- Marken auf **Vorgänger**-Stellen werden **konsumiert**
- Marken auf **Nachfolger**-Stellen werden **produziert**

Anzahl konsumierter / produzierter Marken jeweils gemäß **Bogenvielfalt**:
=> **Gesamtanzahl** Marken kann sich **ändern**.

Jede **Folgemarkierung** (= Folgezustand) ergibt sich aus dem Schalten jeweils **genau einer** Transition.



Literatur:
W. Reisig: Petrinetze
Kap. 6.1, S. 73-74

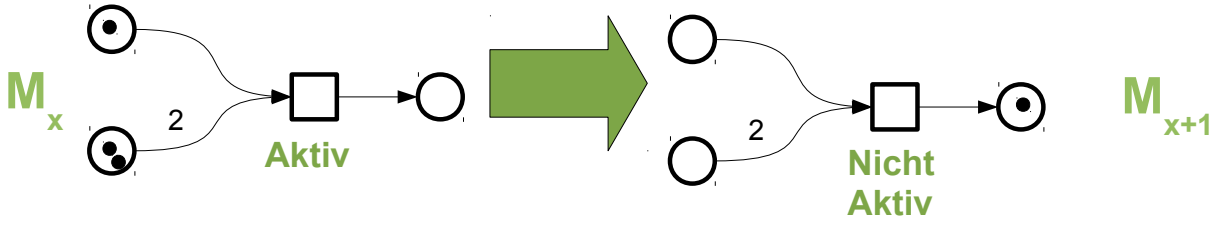
Ablauf: Schalten von Transition

Eine der aktivierten Transitionen wird beim Übergang von Zustand M_x nach Zustand M_{x+1} geschaltet (**nicht-deterministische Auswahl**):

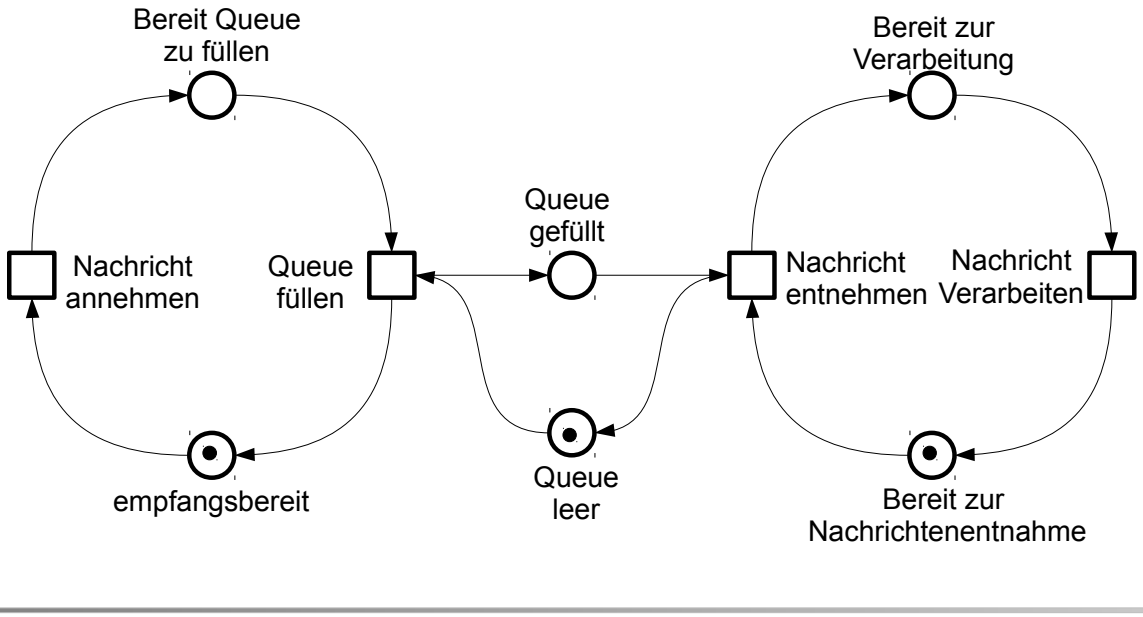
- Marken auf **Vorgänger**-Stellen werden **konsumiert**
- Marken auf **Nachfolger**-Stellen werden **produziert**

Anzahl konsumierter / produzierter Marken jeweils gemäß **Bogenvielfalt**:
=> **Gesamtanzahl** Marken kann sich **ändern**.

Jede **Folgemarkierung** (= Folgezustand) ergibt sich aus dem Schalten jeweils **genau einer** Transition.



Literatur:
W. Reisig: Petrinetze
Kap. 6.1, S. 73-74



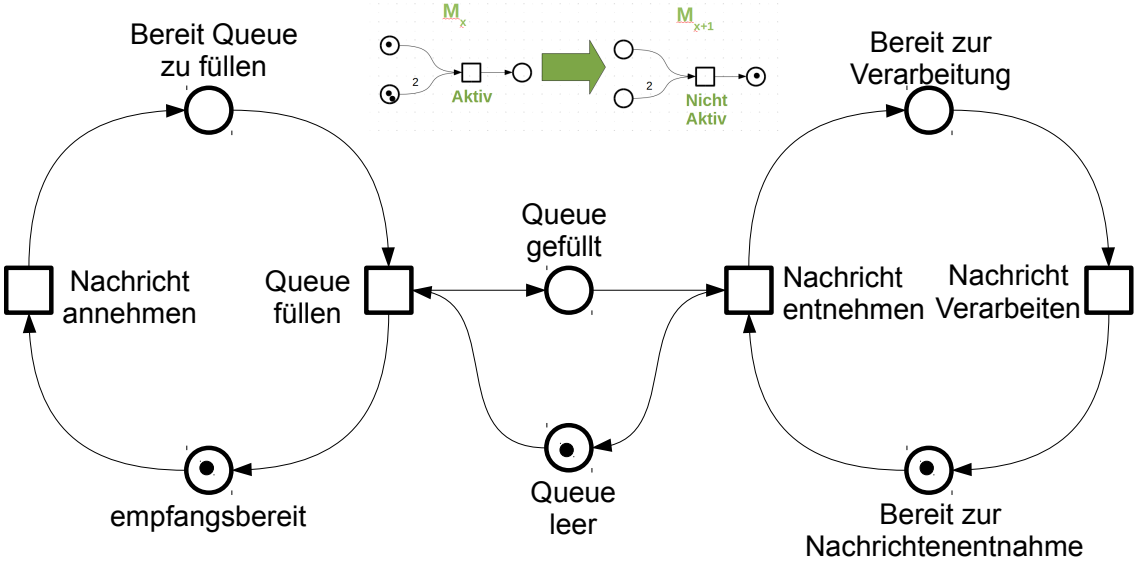
Literatur:

W. Reisig: Petrinetze

- Weiteres Beispiel Kap. 3.3, S.39, Abb. 3.3

Ablauf: Beispiel (M_0)

Welche Transition(en) aktiviert ?



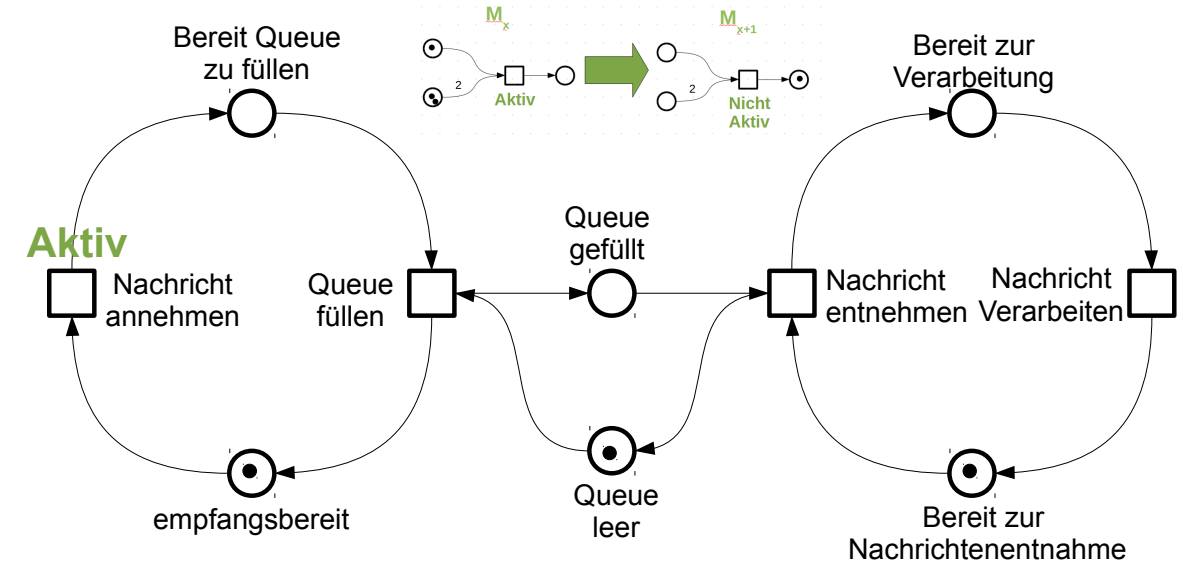
Literatur:

W. Reisig: Petrinetze

- Weiteres Beispiel Kap. 3.4, S.41, Abb. 3.5,3.6

Ablauf: Beispiel (M_0)

Nächster Zustand ?



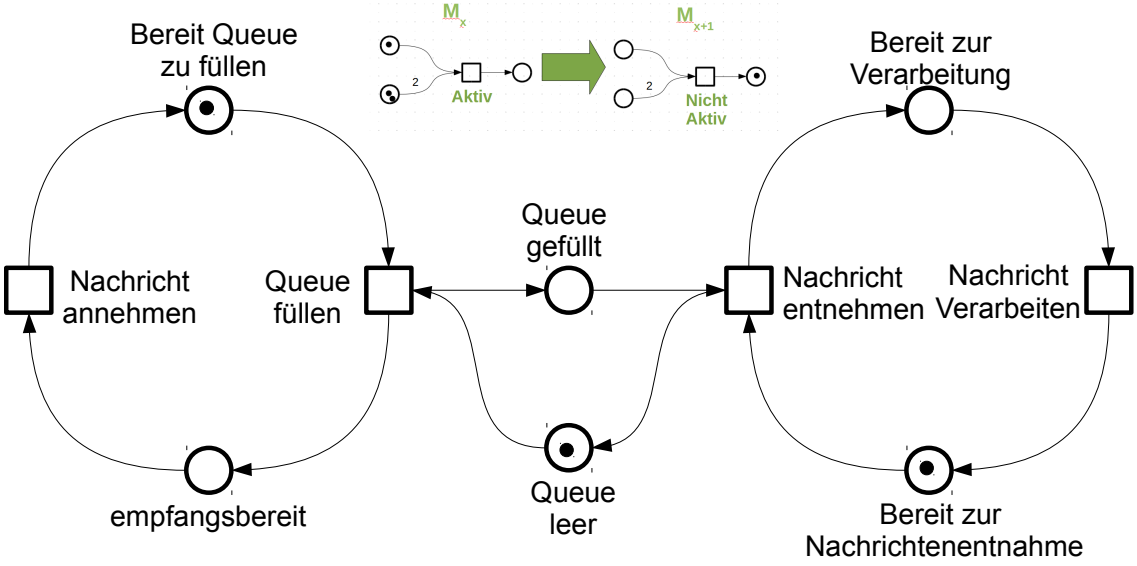
Literatur:

W. Reisig: Petrinetze

- Weiteres Beispiel Kap. 3.4, S.41, Abb. 3.5,3.6

Ablauf: Beispiel (M_1)

Welche Transition(en) aktiviert ?



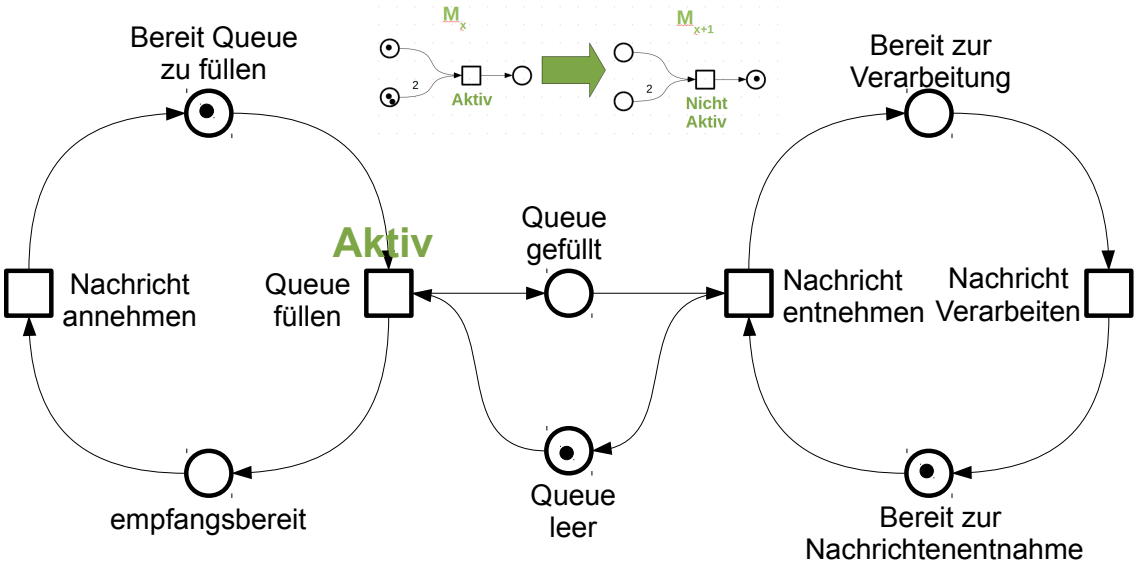
Literatur:

W. Reisig: Petrinetze

- Weiteres Beispiel Kap. 3.4, S.41, Abb. 3.5,3.6

Ablauf: Beispiel (M_1)

Nächster Zustand ?



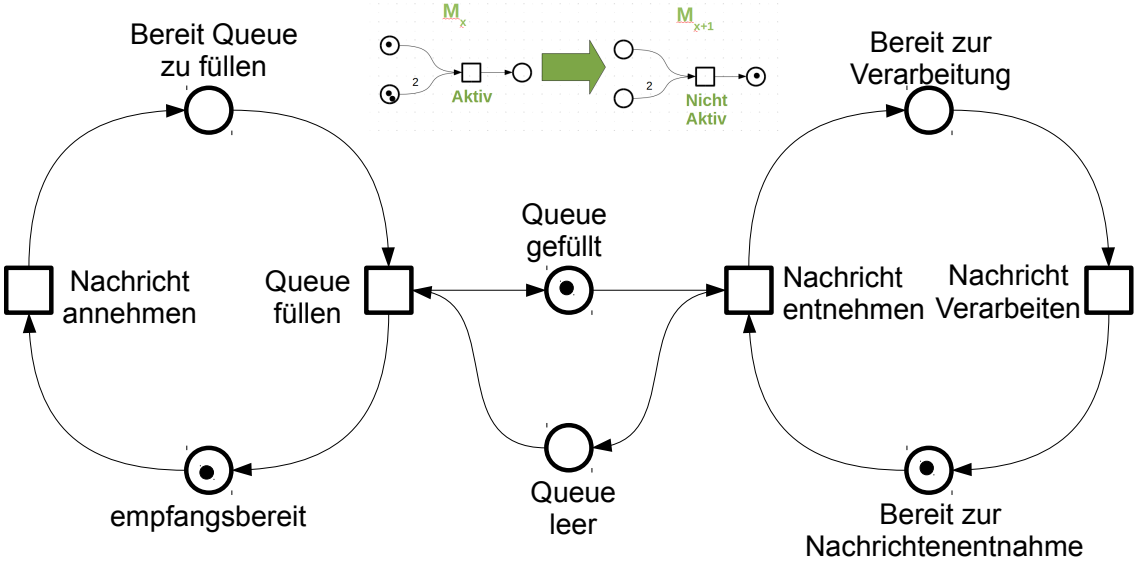
Literatur:

W. Reisig: Petrinetze

- Weiteres Beispiel Kap. 3.4, S.41, Abb. 3.5,3.6

Ablauf: Beispiel (M_2)

Welche Transition(en) aktiviert ?



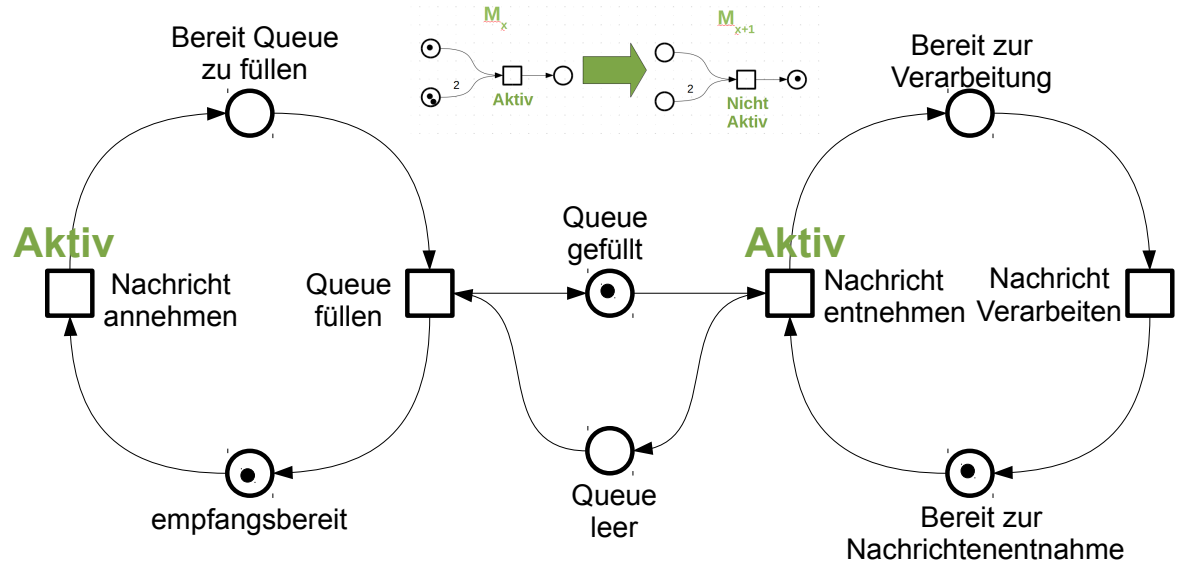
Literatur:

W. Reisig: Petrinetze

- Weiteres Beispiel Kap. 3.4, S.41, Abb. 3.5,3.6

Ablauf: Beispiel (M_2)

Nächster Zustand ?

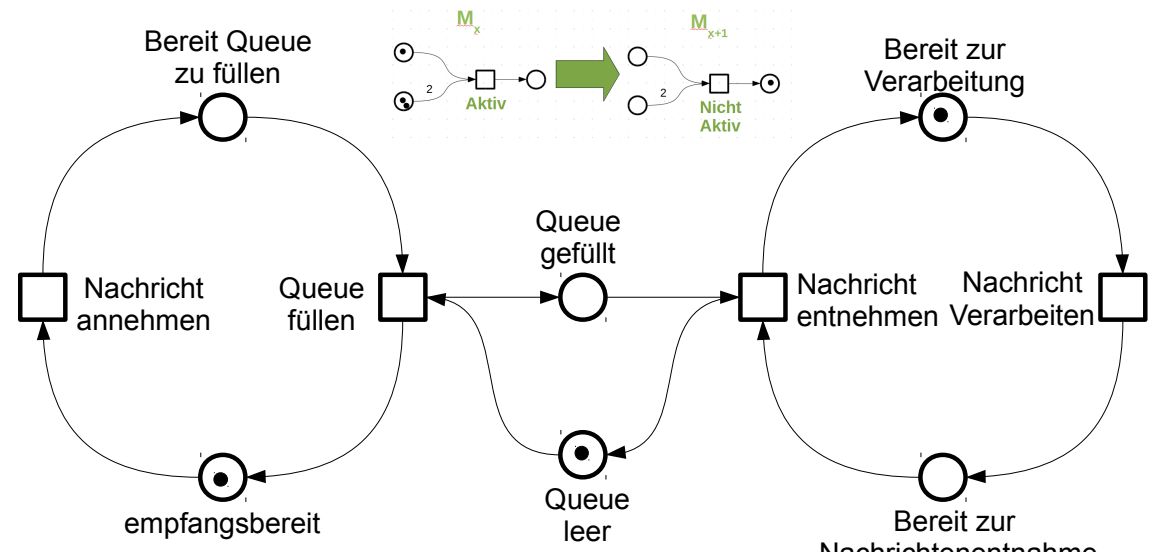


Literatur:

W. Reisig: Petrinetze

- Weiteres Beispiel Kap. 3.4, S.41, Abb. 3.5,3.6

Ablauf: Beispiel (M_3)



(Von M_2 wurde die Transition „Nachricht entnehmen“ ausgeführt;
alternativ hätte die Transition „Nachricht annehmen“ ausgeführt werden können.)

Literatur:

W. Reisig: Petrinetze

- Weiteres Beispiel Kap. 3.4, S.41, Abb. 3.5,3.6

Frage

Gibt es eine obere Grenze, wieviele Nachrichten gleichzeitig in dieser Queue enthalten sein können ?

Frage

Gibt es eine obere Grenze, wieviele Nachrichten gleichzeitig in dieser Queue enthalten sein können ?

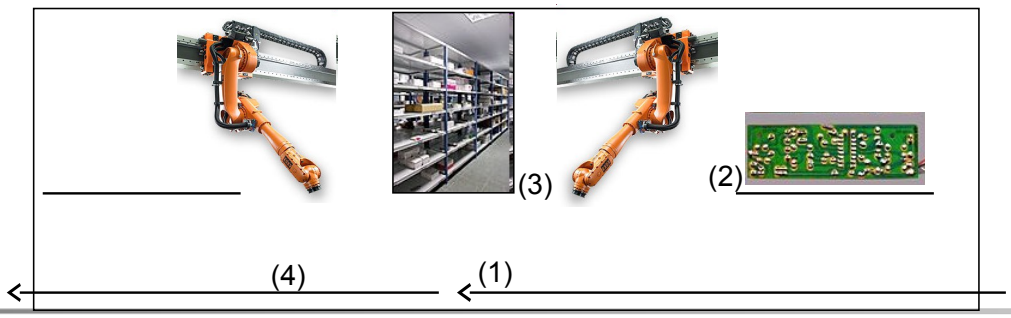
Antwort:

- In der Queue befindet sich höchstens **eine** Nachricht .
- Ausführung der Transition „Queue füllen“, nachdem die Stellen „Nachricht empfangen“ und „Queue leer“ einen Marker besitzen.

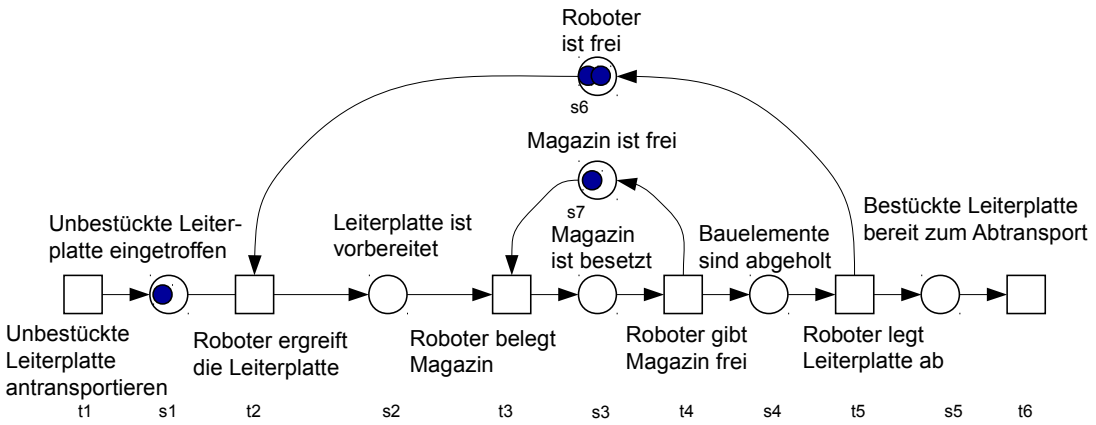
Beispiel: Bestückungsroboter

Zwei Roboter bestücken **Leiterplatten** mit elektronischen **Bauelementen**:

- Leiterplatten auf Fließband antransportiert (1).
- Freier Roboter nimmt Leiterplatte vom Fließband (2).
- Beide Roboter frei: nichtdeterministisch entschieden, wer Leiterplatte nimmt.
- Jeweils ein Roboter darf auf Bauelemente-Magazin zugreifen (3), um Leiterplatte mit Bauelementen zu bestücken.
- Jeweils eine Leiterplatte zu einem Zeitpunkt abtransportierbar (4).

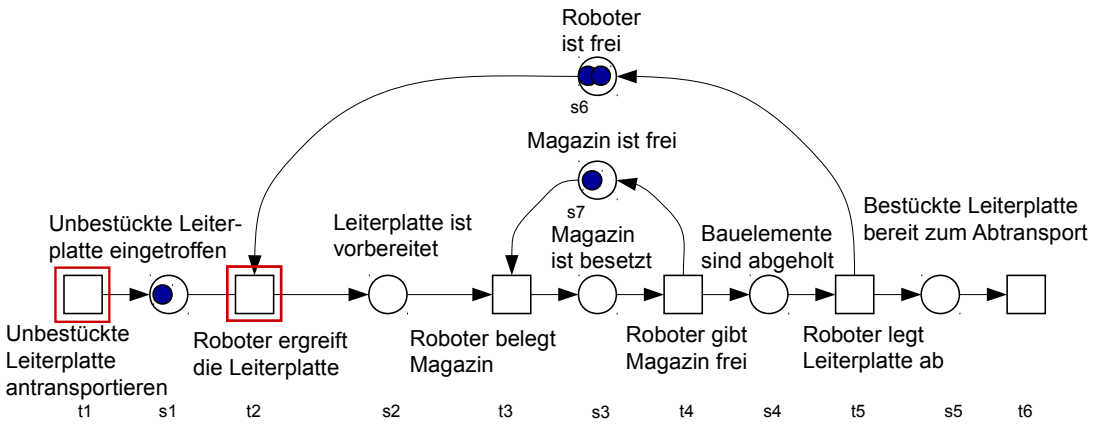


Beispiel: Bestückungsroboter (M0)



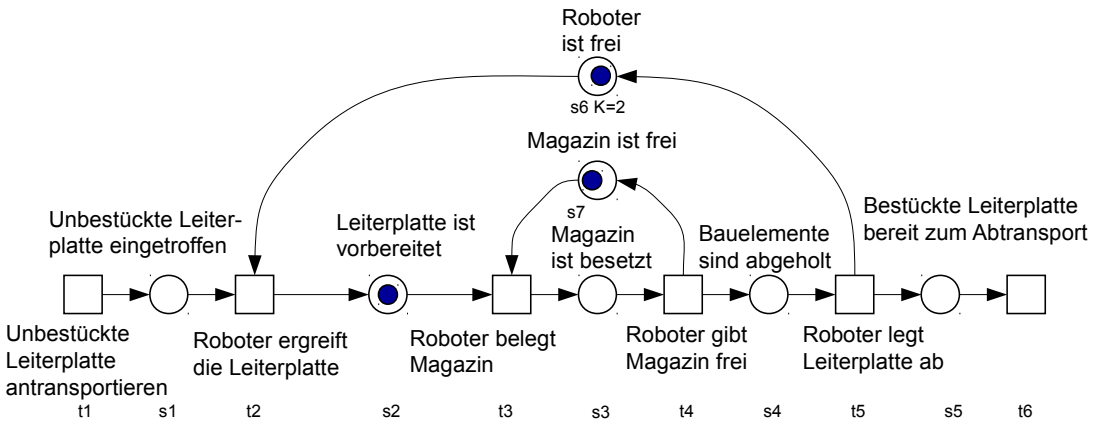
Welche Transition(en) aktiviert ?

Beispiel: Bestückungsroboter (M0)



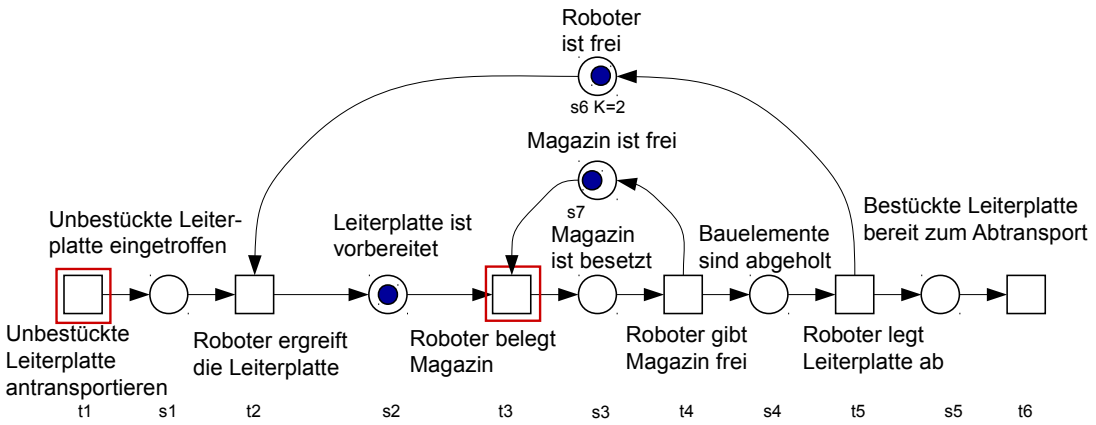
bezeichnet aktivierte Transitionen. Möglicher nächster Zustand ?

Beispiel: Bestückungsroboter (M1)



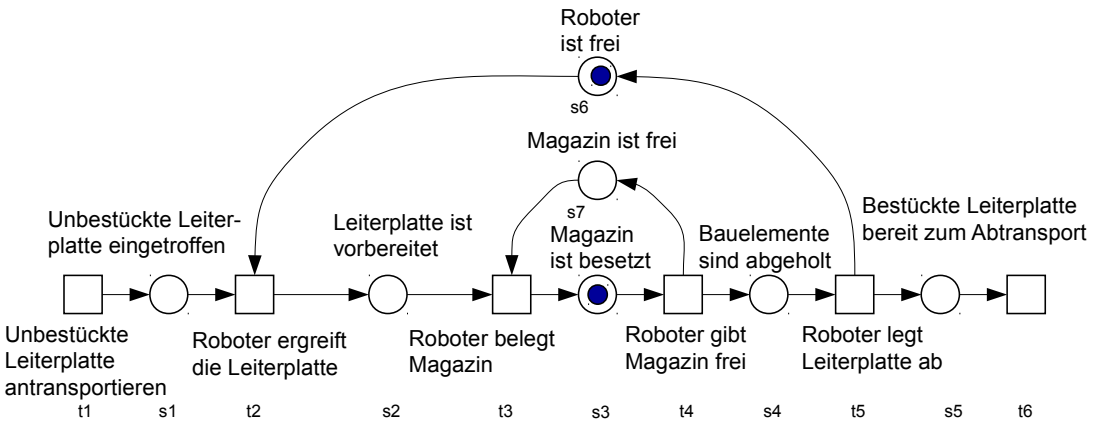
Welche Transition(en) aktiviert ?

Beispiel: Bestückungsroboter (M1)



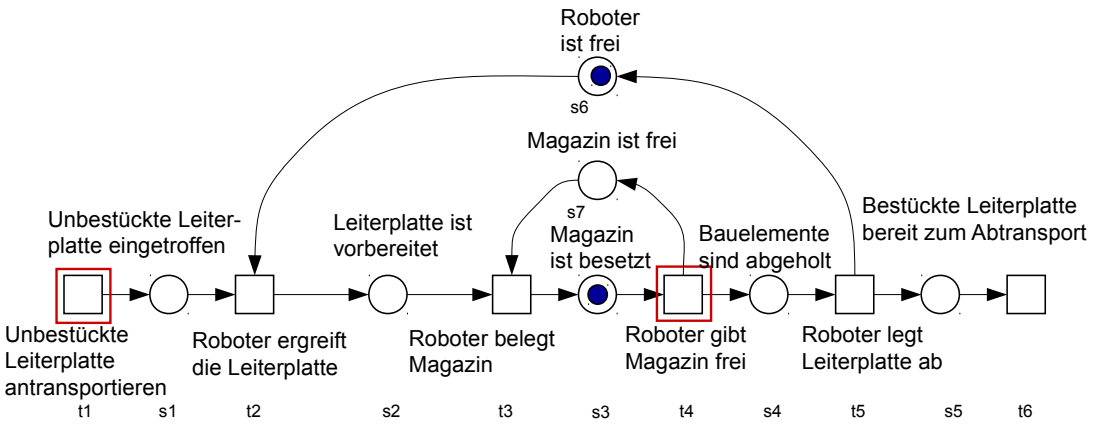
Möglicher nächster Zustand ?

Beispiel: Bestückungsroboter (M2)



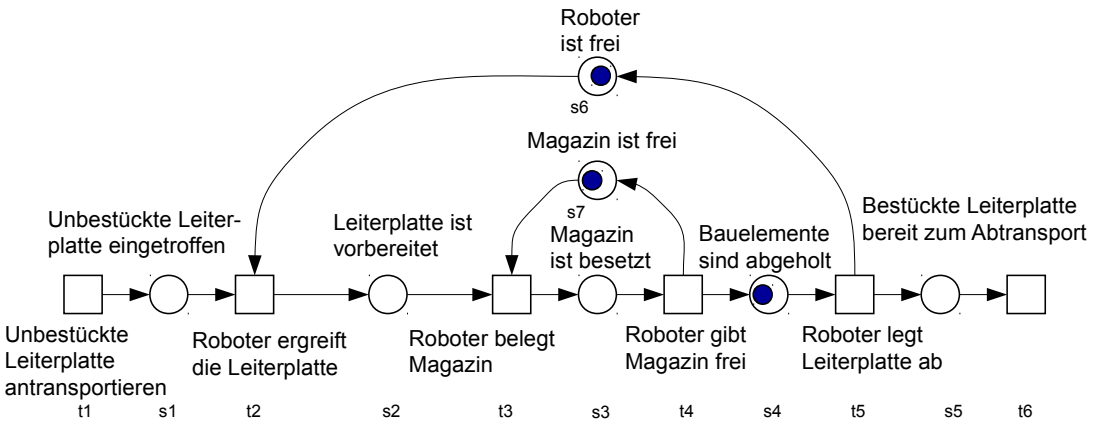
Welche Transition(en) aktiviert ?

Beispiel: Bestückungsroboter (M2)



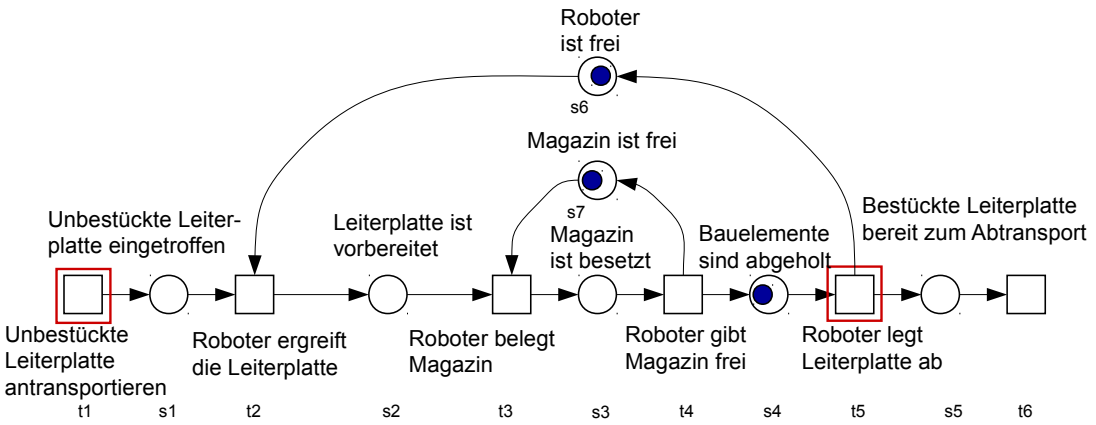
Möglicher nächster Zustand ?

Beispiel: Bestückungsroboter (M3)



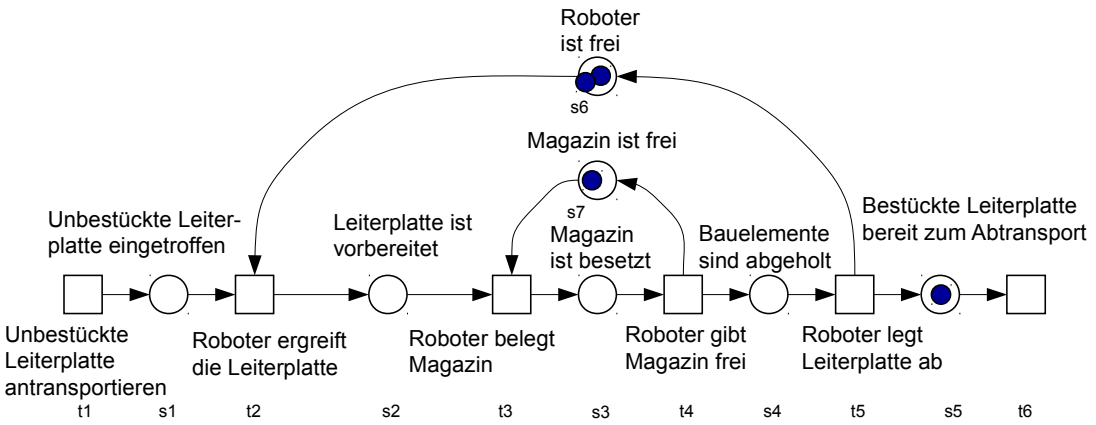
Welche Transition(en) aktiviert ?

Beispiel: Bestückungsroboter (M3)



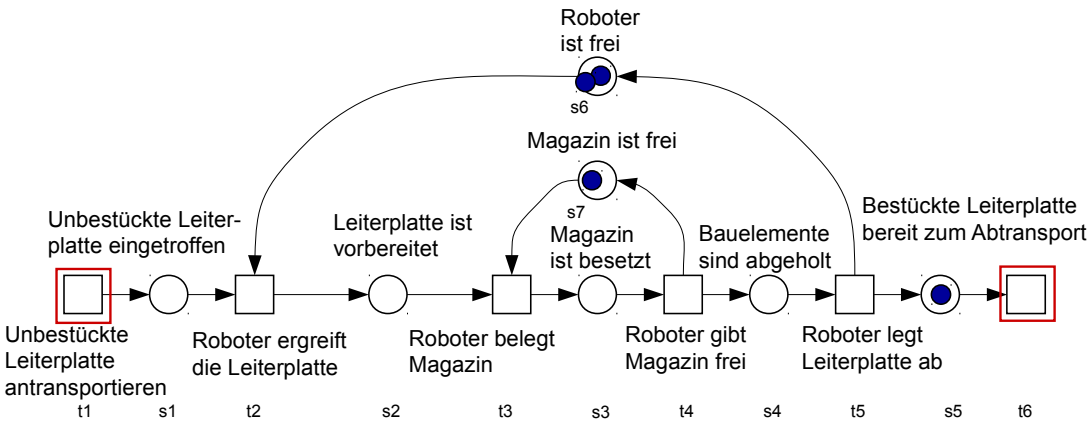
Möglicher nächster Zustand ?

Beispiel: Bestückungsroboter (M4)



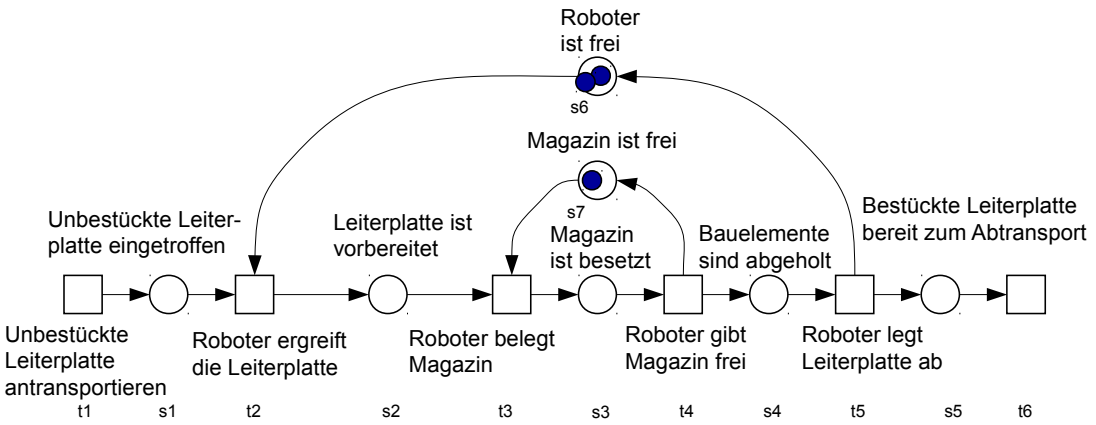
Welche Transition(en) aktiviert ?

Beispiel: Bestückungsroboter (M4)



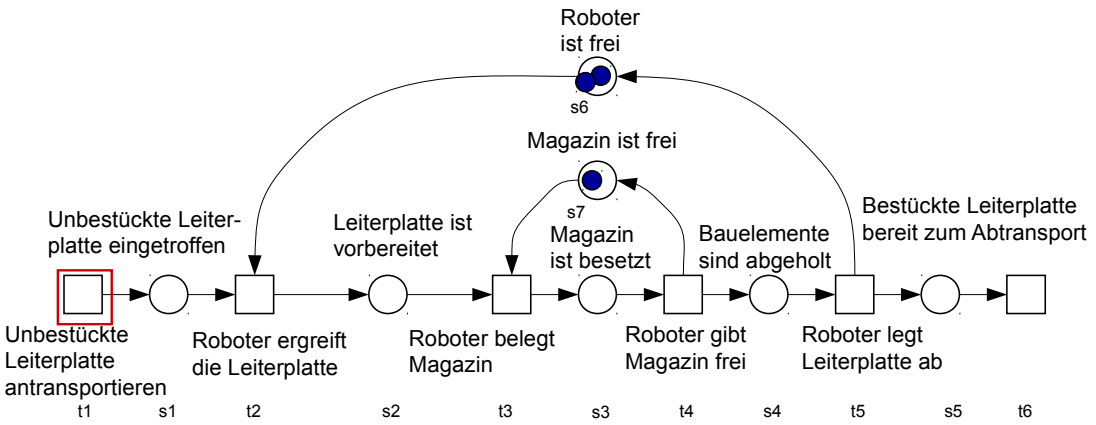
Möglicher nächster Zustand ?

Beispiel: Bestückungsroboter (M5)



Welche Transition(en) aktiviert ?

Beispiel: Bestückungsroboter (M5)

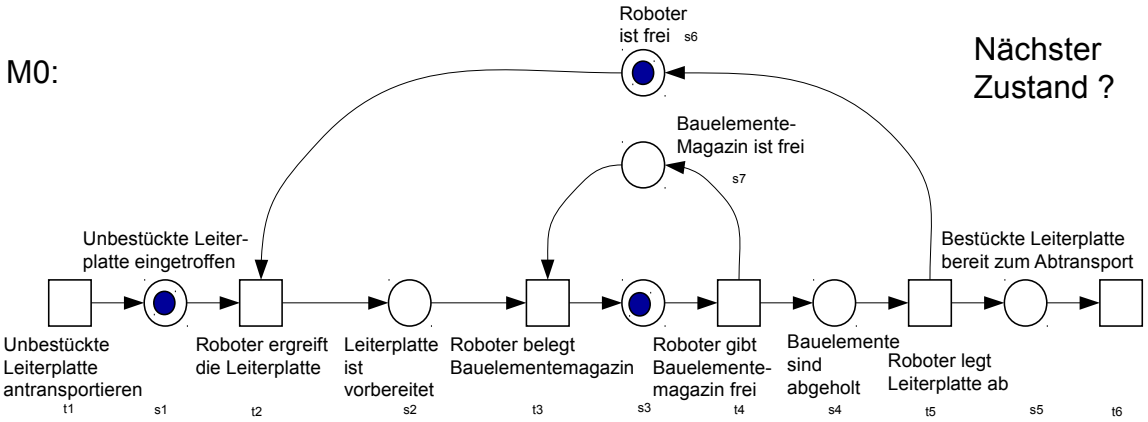


Usw. ...

Andere Startmarkierung M0 für nicht-deterministische Verzweigung:

Nr.	s1	s2	s3	s4	s5	s6	s7	Schaltungen
M0	1	0	1	0	0	1	0	
M1								
M2								

M0:



Bauelemente-Magazin ist besetzt

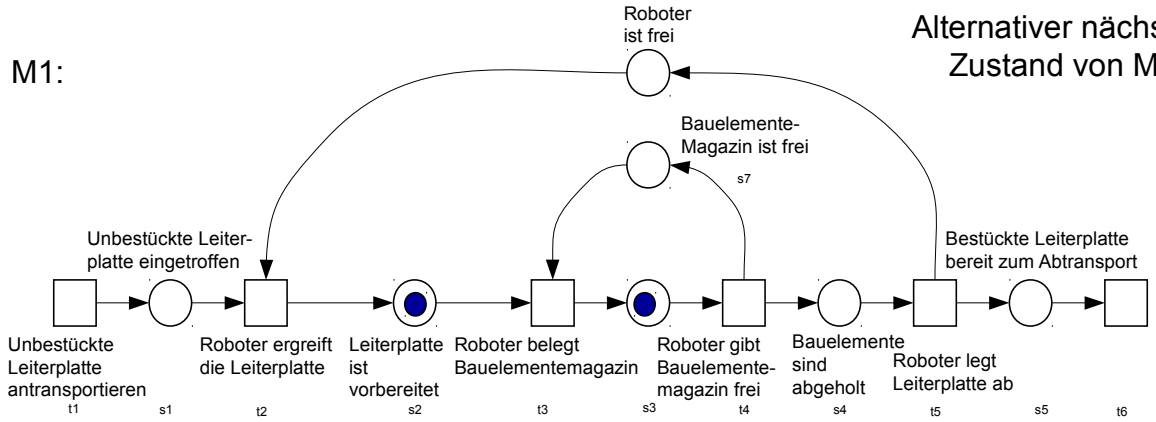
Literatur:

W. Reisig: Petrinetze

- Kap. 2.5 (Schritt), S. 26-27
- Kap. 2.8 (Erreichbarkeit), S. 31
- Kap. 3.1 (Erreichbarkeit), S. 35-36

Nr.	s1	s2	s3	s4	s5	s6	s7	Schaltungen
M0	1	0	1	0	0	1	0	t2->M1
M1	0	1	1	0	0	0	0	
M2								

M1:



Alternativer nächster Zustand von M0 ?

Bauelemente-Magazin ist besetzt

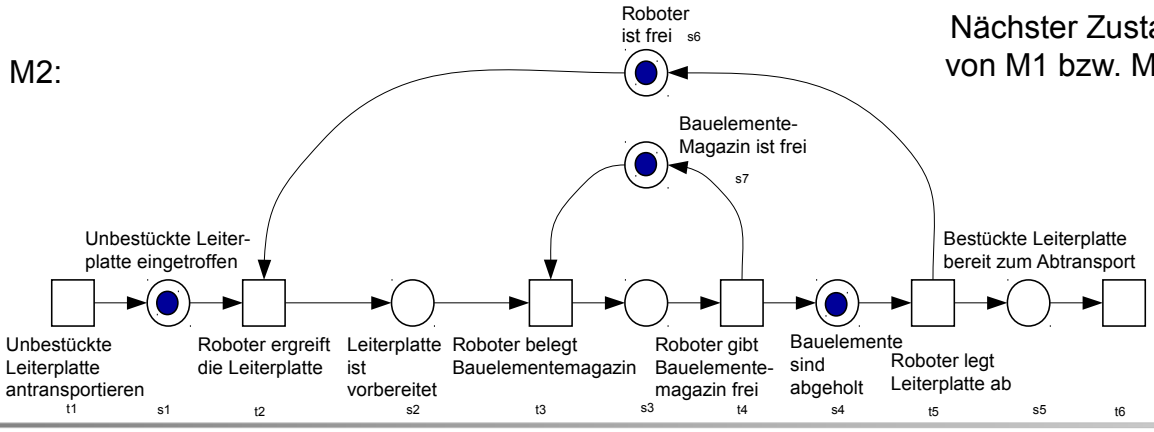
Literatur:

W. Reisig: Petrinetze

- Kap. 2.5 (Schritt), S. 26-27
- Kap. 2.8 (Erreichbarkeit), S. 31
- Kap. 3.1 (Erreichbarkeit), S. 35-36

Nr.	s1	s2	s3	s4	s5	s6	s7	Schaltungen
M0	1	0	1	0	0	1	0	t2->M1 t4->M2
M1	0	1	1	0	0	0	0	
M2	1	0	0	1	0	1	1	

M2:



Nächster Zustand von M1 bzw. M2 ?

Bauelemente-Magazin ist besetzt

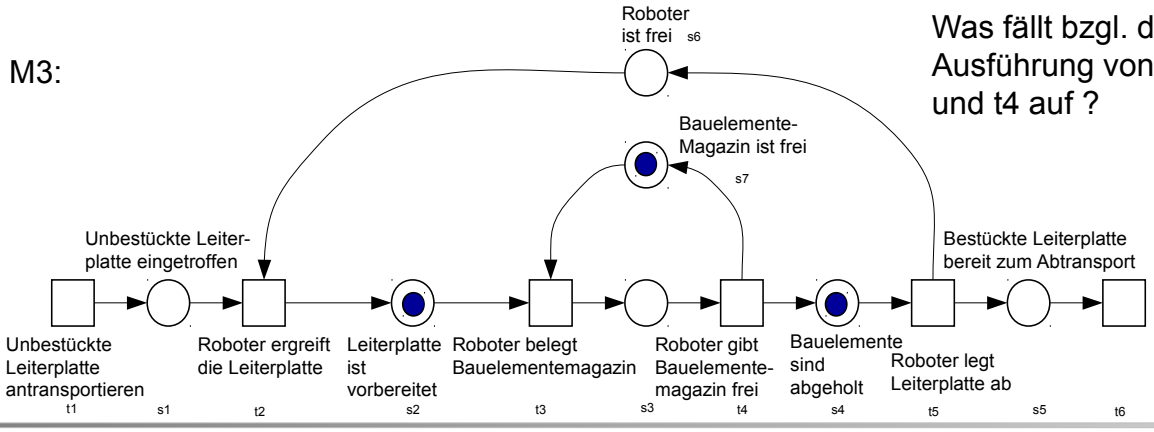
Literatur:

W. Reisig: Petrinetze

- Kap. 2.5 (Schritt), S. 26-27
- Kap. 2.8 (Erreichbarkeit), S. 31
- Kap. 3.1 (Erreichbarkeit), S. 35-36

Nr.	s1	s2	s3	s4	s5	s6	s7	Schaltungen
M0	1	0	1	0	0	1	0	t2->M1 t4->M2
M1	0	1	1	0	0	0	0	t4->M3
M2	1	0	0	1	0	1	1	t2->M3 t5->M4

M3:



Was fällt bzgl. der Ausführung von t2 und t4 auf ?

Bauelemente-Magazin ist besetzt

Literatur:

W. Reisig: Petrinetze

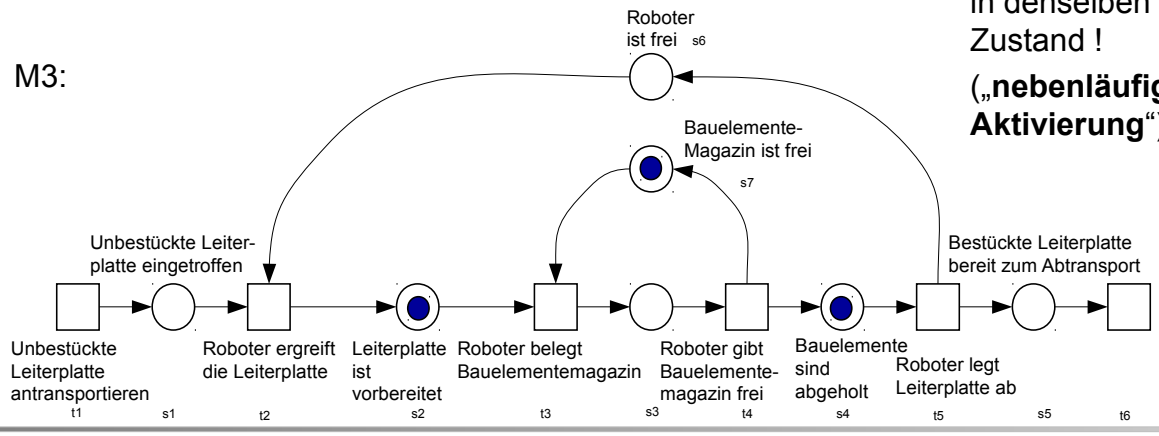
- Kap. 2.5 (Schritt), S. 26-27
- Kap. 2.8 (Erreichbarkeit), S. 31
- Kap. 3.1 (Erreichbarkeit), S. 35-36

Nr.	s1	s2	s3	s4	s5	s6	s7	Schaltungen
M0	1	0	1	0	0	1	0	t2->M1 t4->M2
M1	0	1	1	0	0	0	0	t4->M3
M2	1	0	0	1	0	1	1	t2->M3 t5->M4

t2 und t4 können in beliebiger Reihenfolge ausgeführt werden und resultieren in denselben Zustand !

(„**nebenläufige Aktivierung**“)

M3:



Bauelemente-Magazin ist besetzt

Literatur:

W. Reisig: Petrinetze

- Kap. 2.5 (Schritt), S. 26-27
- Kap. 2.8 (Erreichbarkeit), S. 31
- Kap. 3.1 (Erreichbarkeit), S. 35-36

Erreichbarkeit: Notation und Definition

- **M [t>** : bei Markierung M ist Transition t aktiviert ([> symbolisiert Pfeil)
- **M [t> M' :** M' ist direkte Folgemarkierung zur Markierung M nach Schaltung von Transition t
- **M [w>** : Liste von Transitionen w=[t1,t2,...,tn] ist iterativ aktiviert unter Markierung M, d.h.: M [t1> M1 [t2> M2 … [tn> Mn
- **M [{t1, t2, ..., tn}>** : Liste von Transitionen [t1,t2,...,tn] ist in beliebiger Schaltungsreihenfolge iterativ aktiviert unter Markierung M (= alle Permutationen als Schaltfolgen aktiviert; genannt "**nebenläufig aktiviert**")
- **$[M_0>$** := {M | ∃ w ∈ T* mit M_0 [w> M} (**Erreichbarkeitsmenge** des Systems; die Markierungen M ∈ $[M_0>$ heißen **erreichbar**)

Literatur:

W. Reisig: Petrinetze

- Kap. 2.5 (Schritt), S. 26-27
- Kap. 2.8 (Erreichbarkeit), S. 31
- Kap. 3.1 (Erreichbarkeit), S. 35-36

Erreichbarkeitsalgorithmus (breadth-first; vgl. obiges Beispiel)

Eingabe: Petrinetz. Ausgabe: Erreichbarkeitstabelle (vgl. vorletzte Folie).

1. Trage in ein Schema mit Spalten „Markierungsnummer“, „Markierung“ und „Schaltungen“ **Anfangsmarkierung M_0** ein.
2. In **aktueller Markierung M_i** für jede **Transition t**: aktiviert ?
   - Falls **t** aktiviert: Berechne Folgemarkierung.
     - Folgemarkierung bereits eine **Markierung M_j** ?
     - Wenn nicht: Benenne Folgemarkierung M_j (für ein neues j>i) und lege neue Zeile in der Tabelle für M_j an.
   - In beiden Fällen: Trage M_i **[t>** M_j in Zeile M_i, Spalte „Schaltungen“ ein.
3. M_i erledigt, falls **alle** Transitionen überprüft.
4. Alle eingetragenen Markierungen erledigt ?
   - **Ja**: Erreichbarkeitsanalyse abgeschlossen
   - **Nein**: Überprüfe die nächste Markierung und fahre bei 2 fort.

Literatur:

W. Reisig: Petrinetze

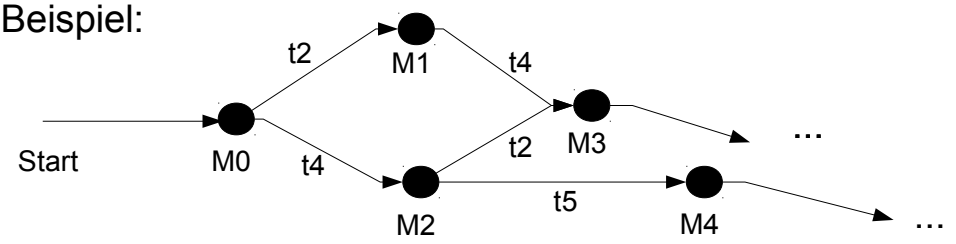
- Kap. 2.5 (Schritt), S. 26-27
- Kap. 2.8 (Erreichbarkeit), S. 31
- Kap. 3.1 (Erreichbarkeit), S. 35-36

Nr.	s1	s2	s3	s4	s5	s6	s7	Schaltungen
M0	1	0	1	0	0	1	0	t2->M1 t4->M2
M1	0	1	1	0	0	0	0	t4->M3
M2	1	0	0	1	0	1	1	t2->M3 t5->M4

Erreichbarkeitstabelle oft als Graph dargestellt:

- Knoten: Zustände (linke Spalte; ggf. inkl. Markierungsbelegungen)
- Kanten: Schaltungen (rechte Spalte)

Obiges Beispiel:

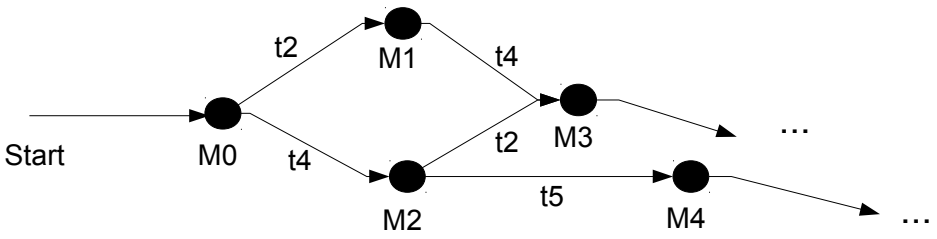


Literatur:

W. Reisig: Petrinetze

- Kap. 2.5 (Schritt), S. 26-27
- Kap. 2.8 (Erreichbarkeit), S. 31
- Kap. 3.1 (Erreichbarkeit), S. 35-36

Nr.	s1	s2	s3	s4	s5	s6	s7	Schaltungen
M0	1	0	1	0	0	1	0	t2->M1 t4->M2
M1	0	1	1	0	0	0	0	t4->M3
M2	1	0	0	1	0	1	1	t2->M3 t5->M4



Erzeugtes Event-Log
(= Menge der Folgen der ausgeführten Transitionen):

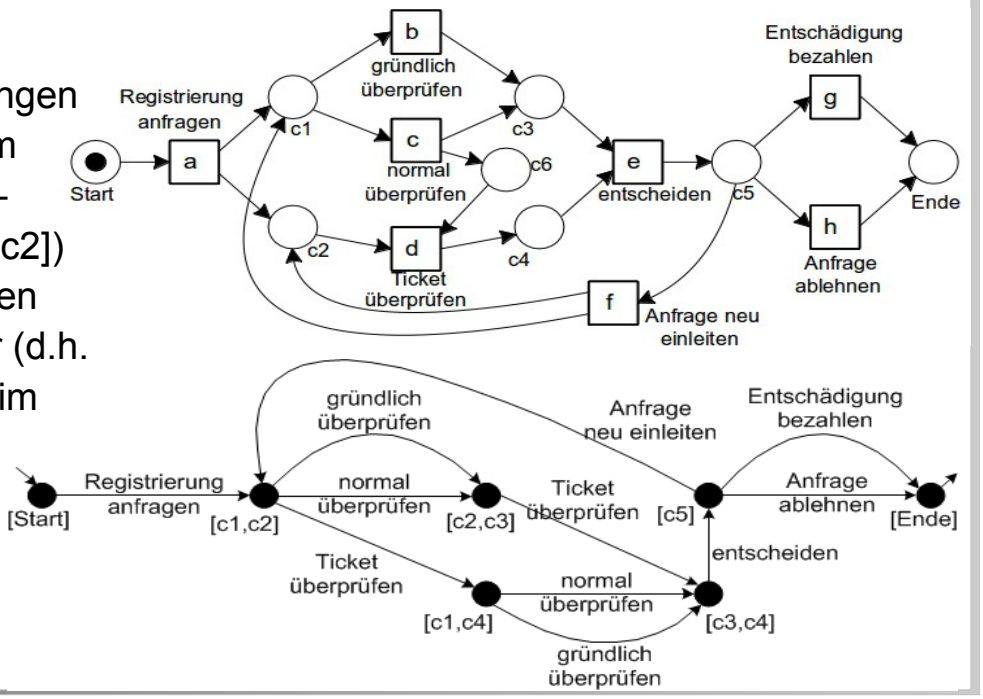
{[t2,t4,...],[t4,t2,...],[t4,t5,...], ...}

Literatur:

W. Reisig: Petrinetze

- Kap. 2.5 (Schritt), S. 26-27
- Kap. 2.8 (Erreichbarkeit), S. 31
- Kap. 3.1 (Erreichbarkeit), S. 35-36

NB: Bezeichnungen der Zustände im Erreichbarkeits-graph (z.B. [c1,c2]) spiegeln globalen Zustand wieder (d.h. welche Stellen im Petrinetz mit Markern belegt sind).



Literatur:

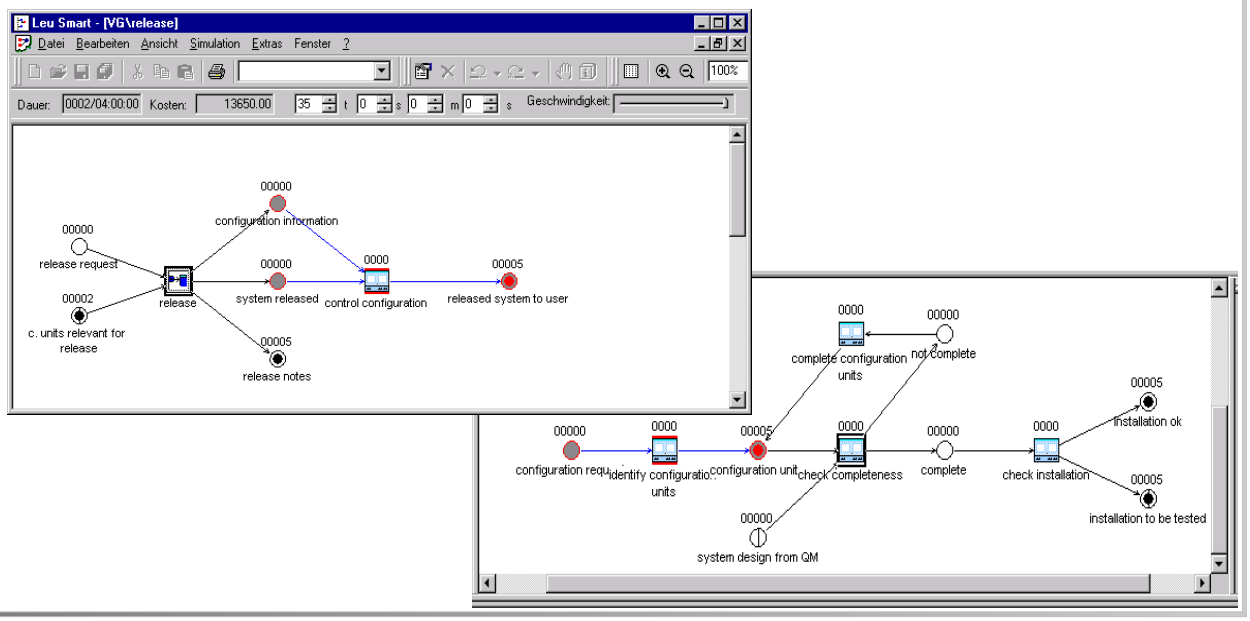
Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

- Kap. 2.2: S. 32 Fig. 2.1

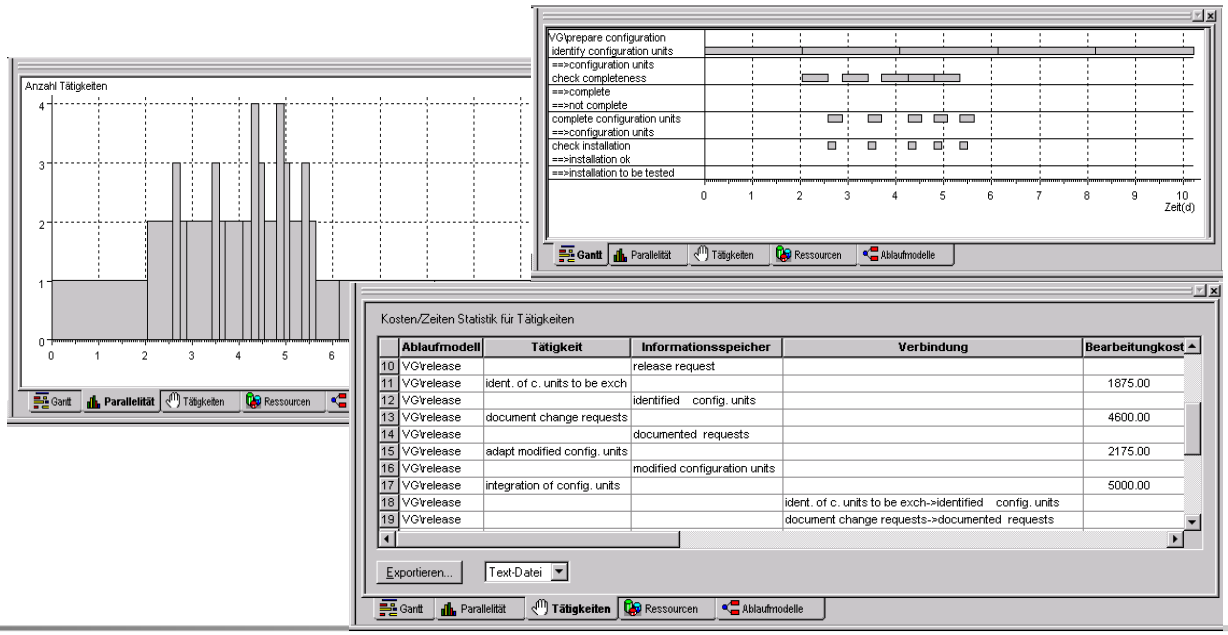
Überblick Petrinetze

- Petrinetz-Syntax
- Ausführung
- Analyse von Systemen
- Workflow-Netze

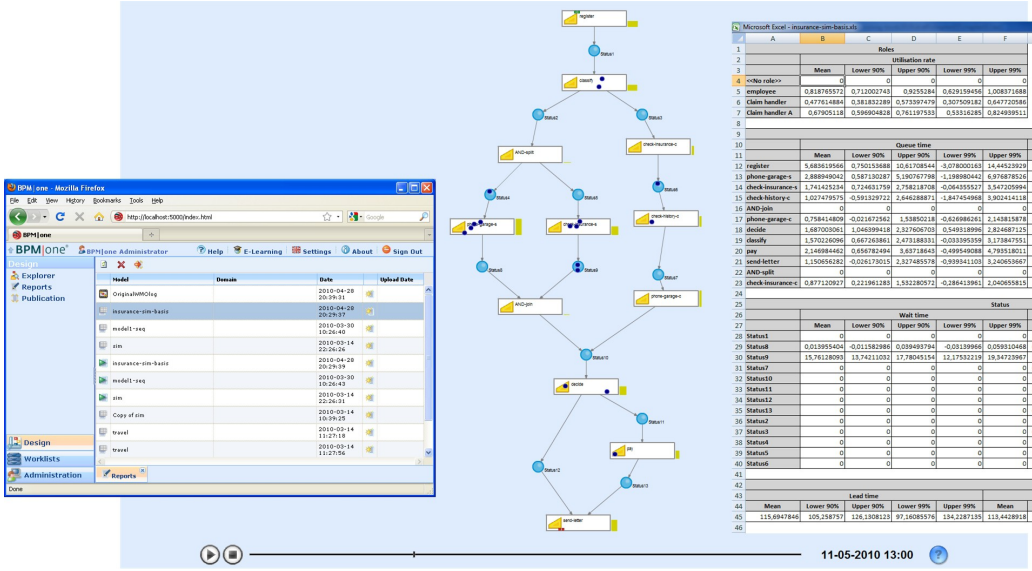
Simulation von Petrinetzen: Animation



Simulation von Petrinetzen: Analyse von Simulationsläufen



	Ablaufmodell	Tätigkeit	Informationsspeicher	Verbindung	Bearbeitungkost
10	VG\release		release request		
11	VG\release	ident. of c. units to be exch			1875.00
12	VG\release		identified config. units		
13	VG\release	document change requests			4600.00
14	VG\release		documented requests		
15	VG\release	adapt modified config. units			2175.00
16	VG\release		modified configuration units		
17	VG\release	integration of config. units			5000.00
18	VG\release			ident. of c. units to be exch->identified config. units	
19	VG\release			document change requests->documented requests	



Literatur:

Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

- Kap. 2.3: S. 56 Fig. 2.19

Simulation:

- Kann zeigen, dass bestimmte Situationen auftreten können.
- Kann **nicht** zeigen, dass bestimmte Situationen **nicht** auftreten.
- **Ausschnitt** aus Menge aller möglichen Verhalten.
- Keine Aussage über deren **Eintrittswahrscheinlichkeit**.

Verifikation: Beweis von Eigenschaften:

- **Statische Eigenschaften:** Unabhängig von Markierungen, nur von Netztopologie abhängig.
  - z.B. Verklemmungen / Deadlocks
- **Dynamische Eigenschaften:** Abhängig von der Menge erreichbarer Markierungen.
  - Standardhilfsmittel: Erreichbarkeitsgraphen (s.o.)

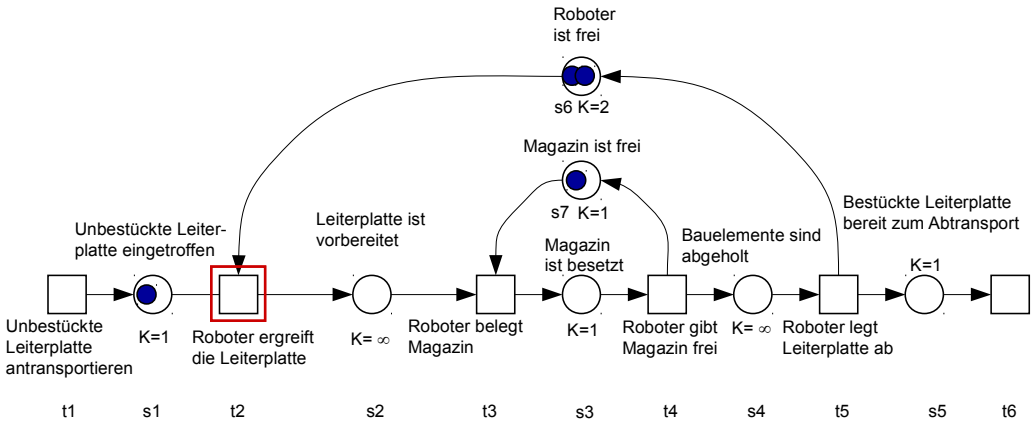
- K: $S \rightarrow \mathbb{N} \cup \{\infty\}$ erklärt eine (möglicherweise unbeschränkte) **Kapazität** für jede Stelle.
- **Markierungen** M: $S \rightarrow \mathbb{N}_0$ müssen Kapazitäten respektieren, d.h. für jede Stelle $s \in S$ gilt: $M(s) \leq K(s)$.
- Transitionen sind bei Verwendung von Kapazitäten **nur dann** aktiviert, wenn **Folgemarkierung** Kapazitäten **respektiert.**

Literatur:

W. Reisig: Petrinetze

- Kap. 6.1 (Platzkapazität, Platz=Stelle), S. 73-74

Beispiel: Petrinetz Bestückungsroboter



bezeichnet die aktivierte Transition.

Sei P=(S,T,F,K,M_0) Petrinetz.

Abbildung B: S $\rightarrow \mathbb{N}_0 \cup \{\infty\}$ ordnet jeder Stelle eine „kritische Markenzahl“ zu.

Petrinetz P heißt:

- **B-sicher** (oder **B-beschränkt**), wenn für alle erreichbaren Markierungen Anzahl der Markierungen pro Stelle durch B begrenzt, d.h.:
  für alle M $\in$ [M_0> und s $\in$S gilt: M(s) $\leq$ B(s).
- **1-sicher**, **2-sicher** usw., wenn B=1, B=2 usw.
- **beschränkt**, wenn es natürliche Zahl b gibt, für die P b-sicher.

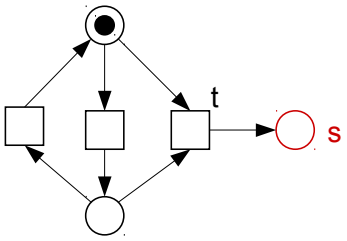
Stelle s heisst **b-sicher**, wenn P B-sicher mit B(s)=b, und B(s‘)= ∞ für s'#s.

Unterschied zwischen Kapazität und Sicherheit:

- **Kapazität begrenzt** Stellenmarkierung (a priori-Begrenzung).
- **Sicherheit beobachtet** Stellenmarkierung (a posteriori-Begrenzung).

Sicherheit: Beispiele

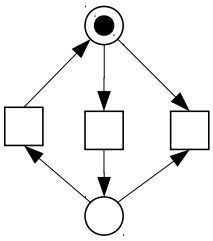
- *Beispiel:* Verkehrsplaner modelliert **Ampelsystem**.
  An bestimmter Stelle s darf sich nur ein Auto aufhalten.
  Für K(s)=1 keine Aussage über Korrektheit der Modellierung.
  Für K(s)=∞ in der Analyse prüfbar, ob B(s)=1.
- *Beispiel:* Transition t soll nie schalten dürfen.
  Durch Hinzufügen einer Beobachtungsstelle s und der Bedingung B(s)=0 kann diese Sicherheitseigenschaft ausgedrückt werden.



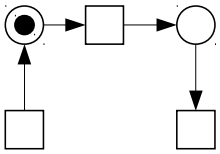
Transition t eines Petrinetz $P=(N,M_0)$ heißt:

- **aktivierbar**: In mindestens einer erreichbaren Markierung aktiviert:
  existiert $M_1 \in [M_0>$ mit: $M_1[t>$
- **lebendig**: In allen erreichbaren Markierung **aktivierbar**:
  für alle $M_1 \in [M_0>$ gilt: existiert $M_2 \in [M_1>$ mit: $M_2[t>$
- **tot:** In keiner erreichbaren Markierung aktiviert:
  für alle $M \in [M_0>$ gilt: $\neg M[t>$

Tot ist nicht logische Negation von lebendig sondern von aktivierbar !



Transition
ist … ?

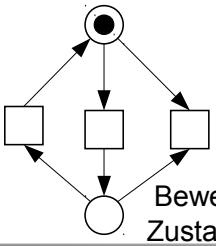


Transition
ist … ?

Transition t eines Petrinetz $P=(N,M_0)$ heißt:

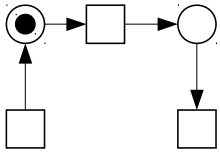
- **aktivierbar**: In mindestens einer erreichbaren Markierung aktiviert:
  existiert $M_1 \in [M_0>$ mit: $M_1[t>$
- **lebendig**: In allen erreichbaren Markierung **aktivierbar**:
  für alle $M_1 \in [M_0>$ gilt: existiert $M_2 \in [M_1>$ mit: $M_2[t>$
- **tot:** In keiner erreichbaren Markierung aktiviert:
  für alle $M \in [M_0>$ gilt: $\neg M[t>$

Tot ist nicht logische Negation von lebendig sondern von aktivierbar !



Tote Transition

Beweis: Gibt in jedem Zustand nur eine Marke, Transition braucht aber zwei !

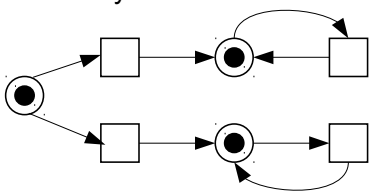


Lebendige Transition

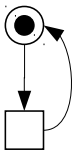
Petrinetz $P=(S,T,F,K,W,M_0)$ heißt:

- **lebendig**: In jeder erreichbaren Markierung ist jede Transition **aktivierbar**:
  für alle $M_1 \in [M_0>$ und $t \in T$ gilt: existiert $M_2 \in [M_1>$ mit: $M_2[t>$
- **deadlockfrei**: In jeder erreichbaren Markierung ist mindestens eine Transition aktiviert:
  für alle $M_1 \in [M_0>$ gilt: existiert $t \in T$ mit: $M_1[t>$
- **tot:** Keine Transition aktiviert:
  $\forall t \in T: \neg M_0\,[t>$

System ist … ?



System ist … ?

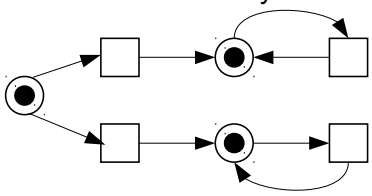


System ist … ?

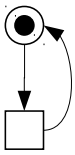
Petrinetz $P=(S,T,F,K,W,M_0)$ heißt:

- **lebendig**: In jeder erreichbaren Markierung ist jede Transition **aktivierbar**:
  für alle $M_1 \in [M_0>$ und $t \in T$ gilt: existiert $M_2 \in [M_1>$ mit: $M_2[t>$
- **deadlockfrei**: In jeder erreichbaren Markierung ist mindestens eine Transition aktiviert:
  für alle $M_1 \in [M_0>$ gilt: existiert $t \in T$ mit: $M_1[t>$
- **tot:** Keine Transition aktiviert:
  $\forall t \in T: \neg M_0\,[t>$

Deadlockfreies System



Lebendiges System



Totes System

tot: keine Transition aktivierbar, d.h. alle Transitionen tot.

(**Achtung**: Nicht **aktivierte** Transition kann trotzdem **aktivierbar** sein, aber wenn keine Transition im Petrinetz **aktiviert** ist, ist auch keine **aktivierbar** !)

D.h. im Erreichbarkeitsgraph geht von der initialen Markierung keine Kante aus.

- Bedeutet häufig einen Deadlock

deadlockfrei: keine erreichbare Markierung tot.

- Berücksichtigung partieller Ausfälle („graceful degradation“).

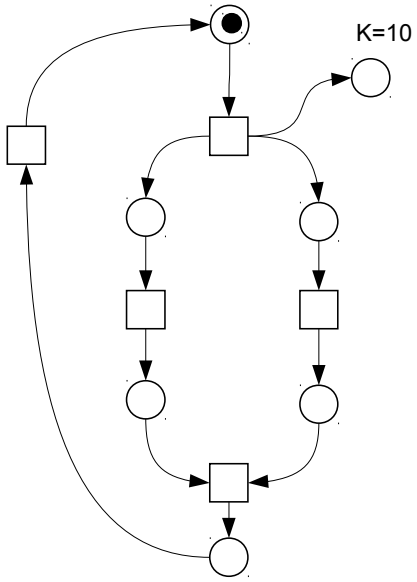
lebendig: alle Transitionen lebendig.

D.h.: in Erreichbarkeitsgraph existiert von jedem Knoten ausgehend für jedes t aus T einen Pfad, in dem t als Label vorkommt.

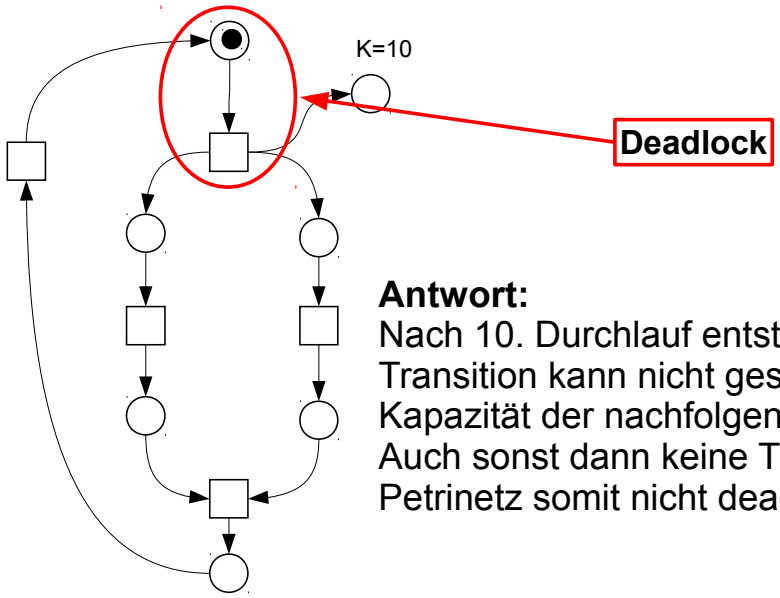
Lebendig ist nicht logische Negation von tot (sondern stärker).

Gibt viele Varianten dieser Definitionen.

Welche Eigenschaft ist hier erfüllt bzw. verletzt ?



Welche Eigenschaft ist hier erfüllt bzw. verletzt ?



Antwort:
Nach 10. Durchlauf entsteht Deadlock:
Transition kann nicht geschaltet werden, da Kapazität der nachfolgenden Stelle erschöpft.
Auch sonst dann keine Transition aktiviert.
Petrinetz somit nicht deadlockfrei.

Überblick Petrinetze

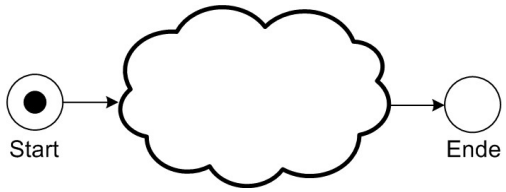
- Petrinetz-Syntax
- Ausführung
- Analyse von Systemen
- Workflow-Netze

Workflow-Netz: Petrinetz mit:

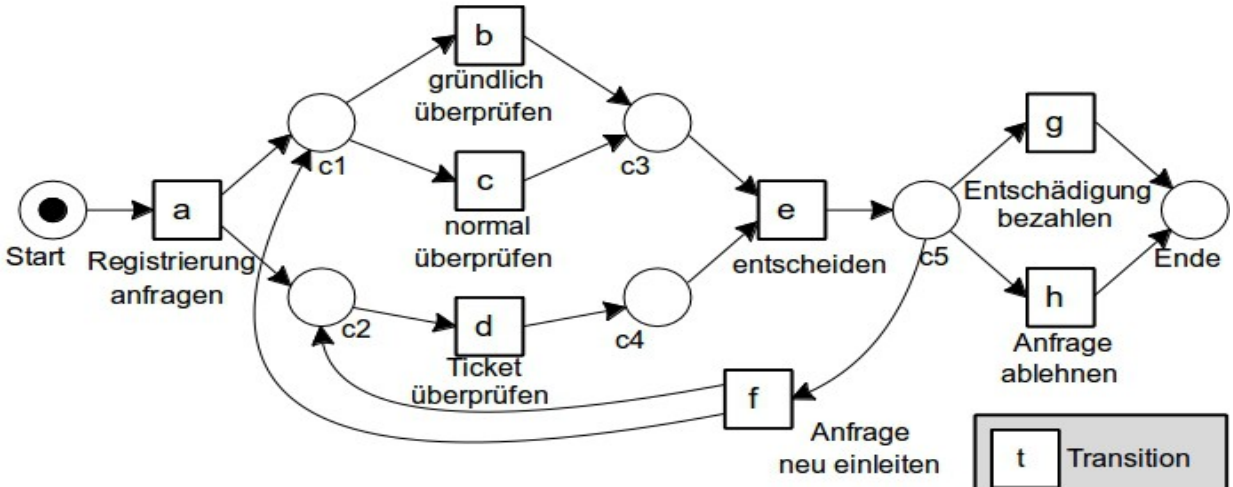
- genau eine Stelle ohne hineinkommenden Bogen („**Start-Stelle**“)
- genau eine Stelle ohne hinausgehenden Bogen („**End-Stelle**“)
- jede Stelle und Transition auf **Pfad** von Start-Stelle zu End-Stelle.

Initiale Markierung bei Ausführung Workflow-Netz:

- **Start-Stelle: genau eine** Marke
- Alle anderen Stellen: keine Marke.



Annahme für Rest des Kapitels:
alle ab nun betrachteten Petrinetze sind **Workflow-Netze**.



Welche **Boolsche Verzweigung** in Workflow-Netzen bei Verzweigung an **Stellen** / **Transitionen** realisiert ?

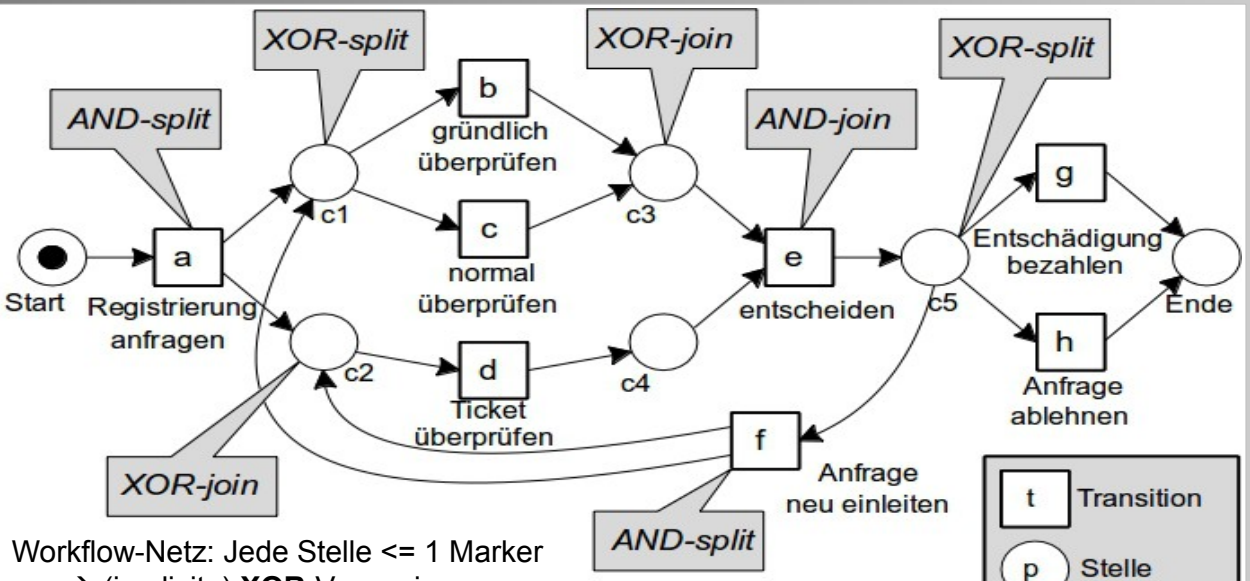
Gilt dies auch in allgemeinen Petrinetzen ?

Literatur:

Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

- Kap. 2.2: S. 35 Fig. 2.2 ohne split/join Angabe

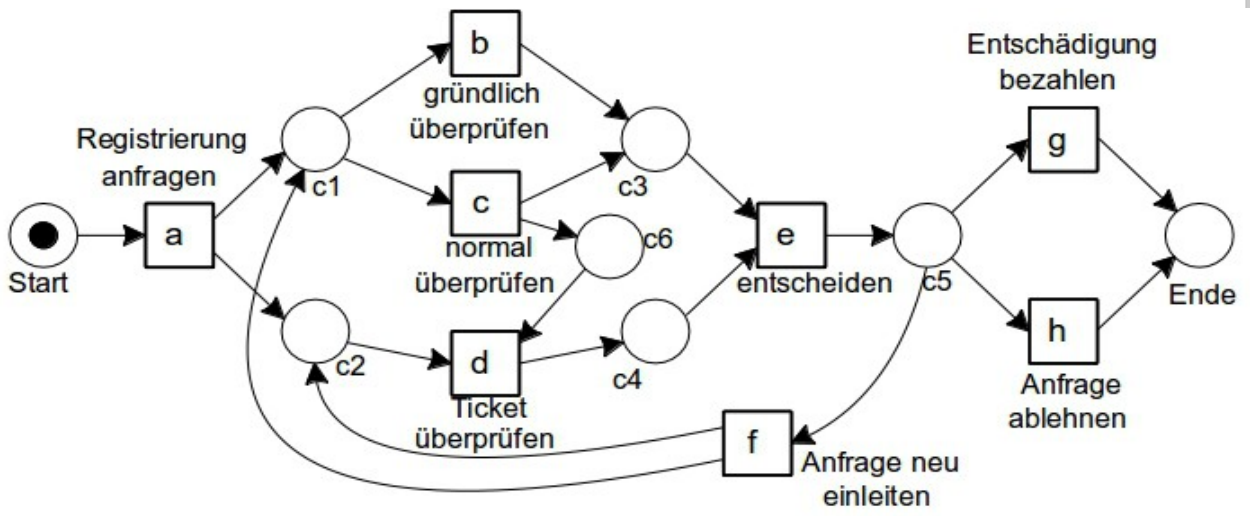
Verzweigungen an Stellen / Transitionen



Workflow-Netz: Jede Stelle <= 1 Marker
→ (implizite) **XOR**-Verzweigung.
[Allgemeines Petrinetz: ggf. > 1 Marker, auch inklusives OR möglich.]
Transitionen: implizite AND-Verzweigung. [Auch allgemeine Petrinetze.]

Literatur:
Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

- Kap. 2.2: S. 35 Fig. 2.2

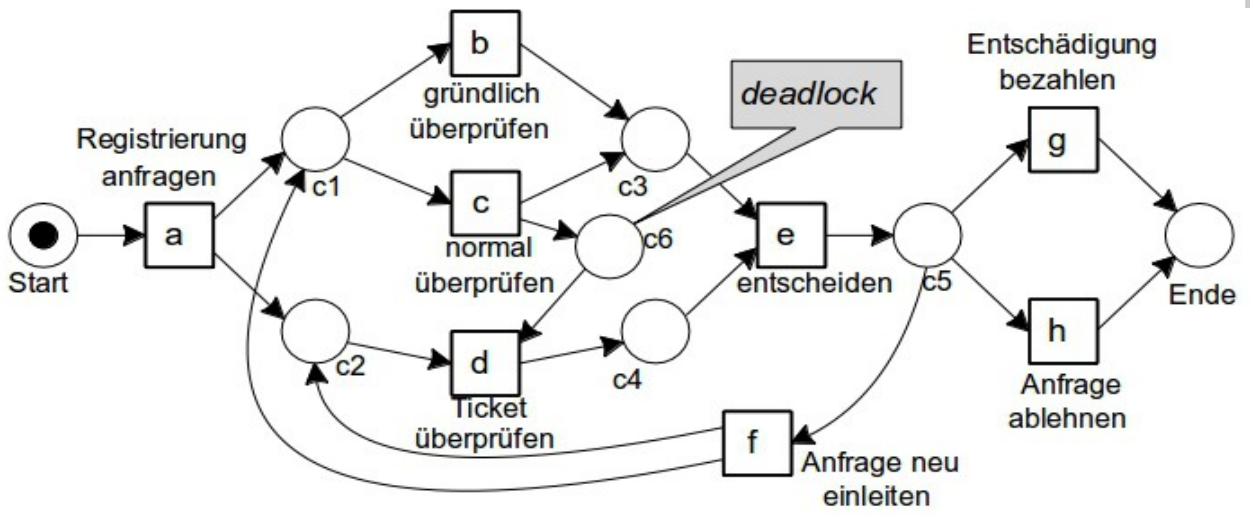


Wo ist das Problem ?

Literatur:

Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

- Kap. 1.4: S. 14 Fig. 1.5

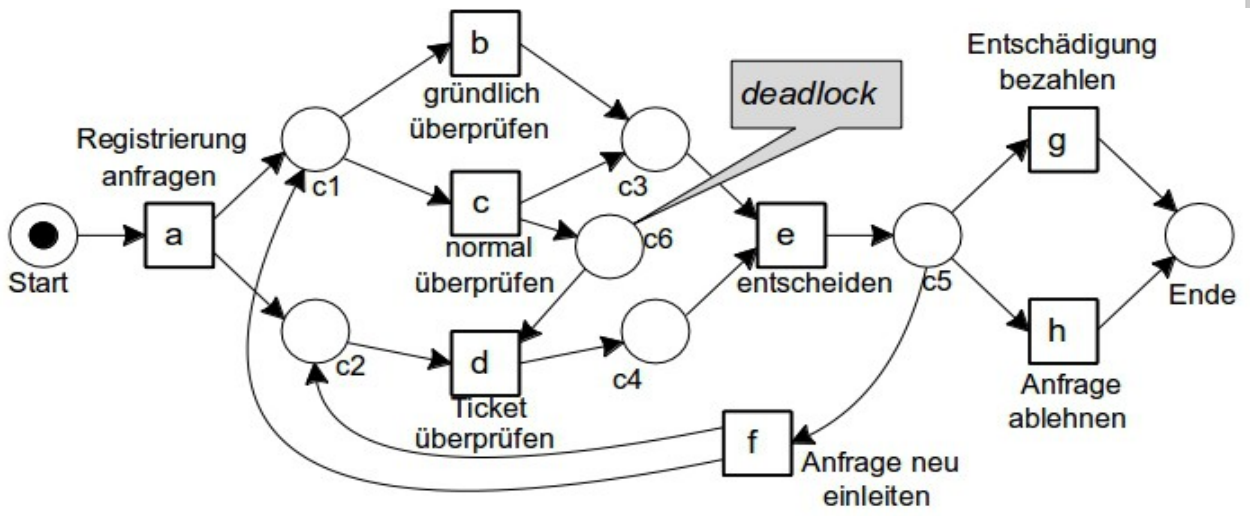


Warum ist hier ein Problem ?

Literatur:

Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

- Kap. 2.3: S. 53 Fig. 2.17

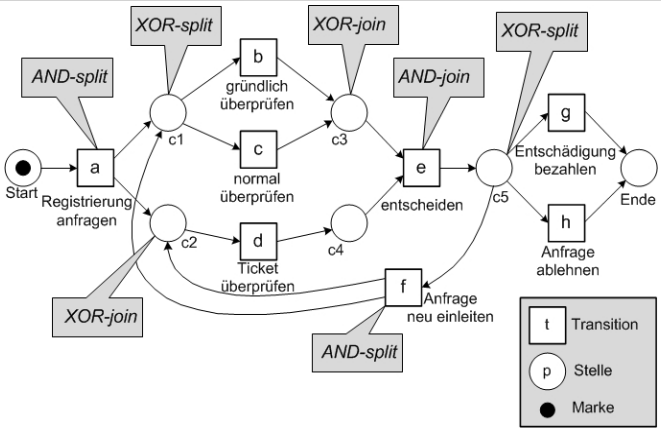


Falls **XOR**-Verzweigung c1 Zweig b wählt, kann d nie schalten, weil c6 kein Token erhält (auch da AND-Verknüpfung e nicht schalten kann).

Literatur:

Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

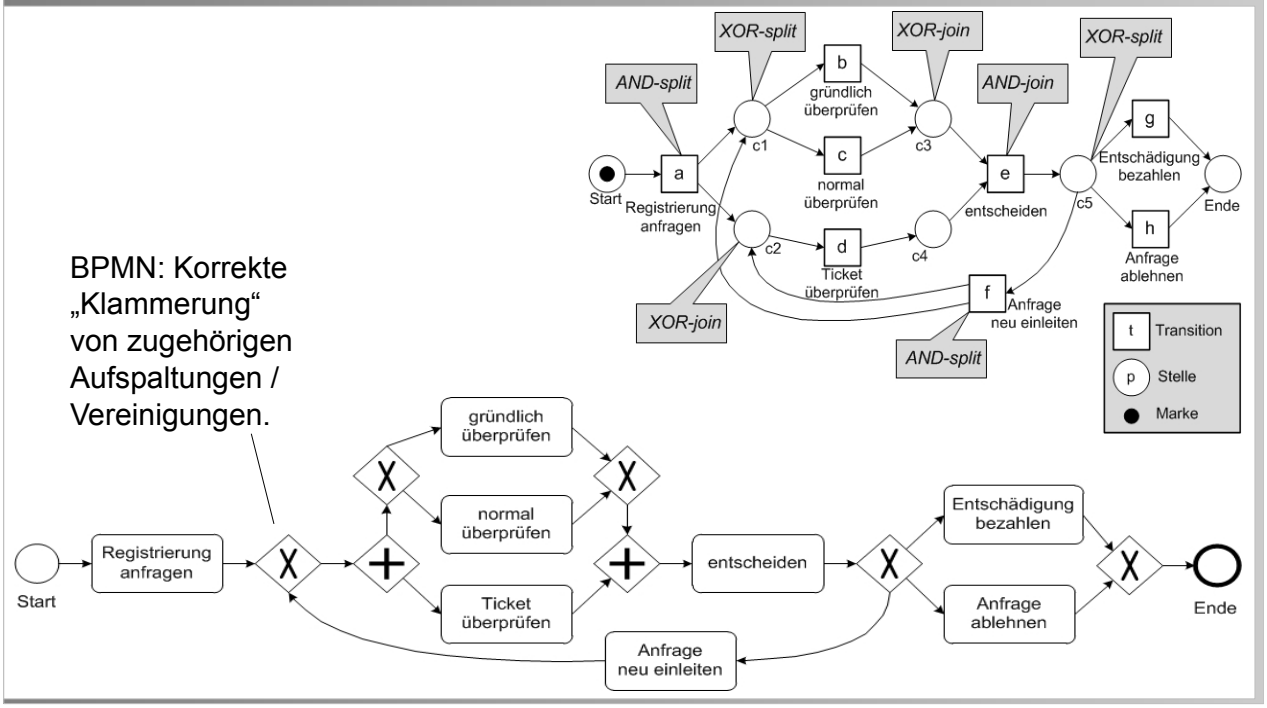
- Kap. 2.3: S. 53 Fig. 2.17



Literatur:

Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

- Kap. 2.2: S. 35 Fig. 2.2



BPMN: Korrekte „Klammerung“ von zugehörigen Aufspaltungen / Vereinigungen.

Literatur:

Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

- Obere Abbildung: Kap. 2.2: S. 35 Fig. 2.2
- Untere Abbildung: Kap.1.2: S.5 Fig.1.2

$\{\langle a,b,d,e,h\rangle, \langle a,d,b,e,h\rangle\}$

Zugehöriges Petrinetz ?

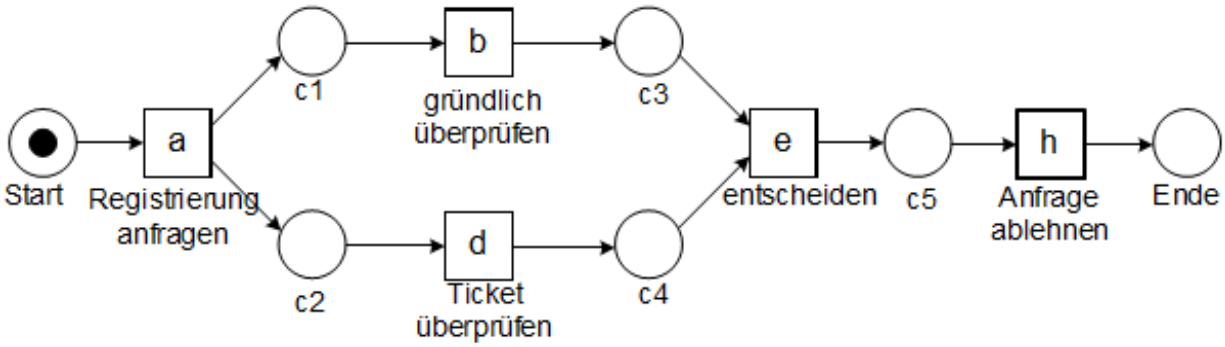
Literatur:

Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

- Kap. 1: S. 14 Fig. 1.2 & 1.5

$$\{\langle a,b,d,e,h\rangle, \langle a,d,b,e,h\rangle\}$$

Zugehöriges Petrinetz ?



Literatur:

Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

- Kap. 1: S. 14 Fig. 1.2 & 1.5

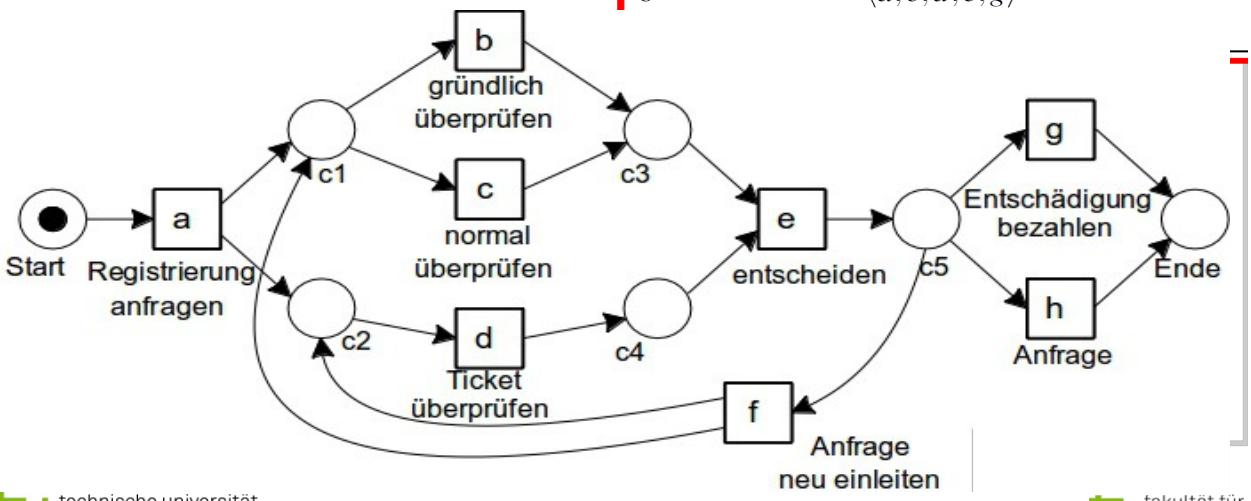
case id	trace
1	$\langle a,b,d,e,h \rangle$
2	$\langle a,d,c,e,g \rangle$
3	$\langle a,c,d,e,f,b,d,e,g \rangle$
4	$\langle a,d,b,e,h \rangle$
5	$\langle a,c,d,e,f,d,c,e,f,c,d,e,h \rangle$
6	$\langle a,c,d,e,g \rangle$
...	...

Literatur:

Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

- Kap. 1: S. 13 Tabelle 1.2

case id	trace
1	$\langle a,b,d,e,h \rangle$
2	$\langle a,d,c,e,g \rangle$
3	$\langle a,c,d,e,f,b,d,e,g \rangle$
4	$\langle a,d,b,e,h \rangle$
5	$\langle a,c,d,e,f,d,c,e,f,c,d,e,h \rangle$
6	$\langle a,c,d,e,g \rangle$



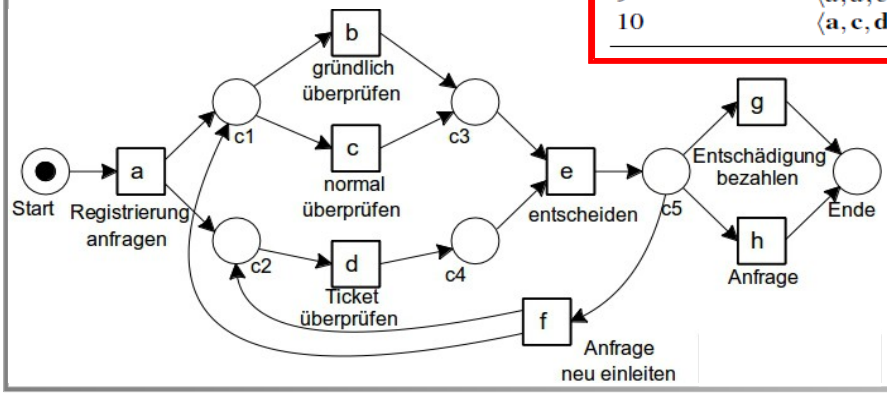
Literatur:

Wil van der Aalst: Process Mining: Discovery, Conformance and Enhancement of Business Processes

- Kap. 1: S. 14 Tabelle 1.2, Fig. 1.5

Sind die zusätzlichen Event-Folgen 7, 8, 10 konform zum extrahierten Modell ?

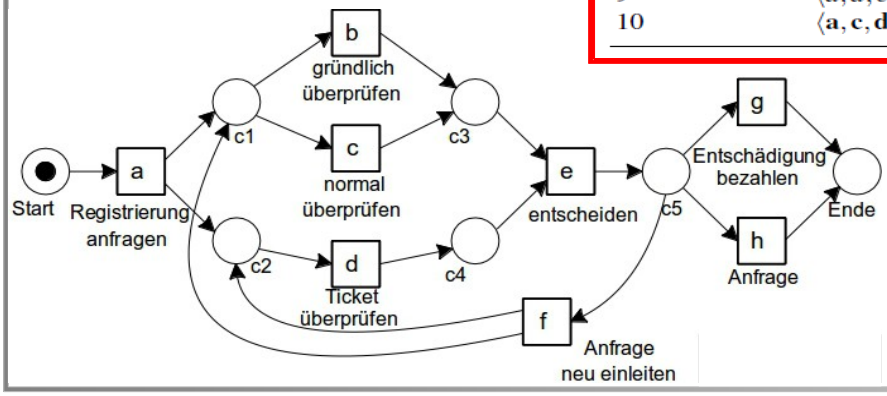
case id	trace
1	$\langle a,b,d,e,h\rangle$
2	$\langle a,d,c,e,g\rangle$
3	$\langle a,c,d,e,f,b,d,e,g\rangle$
4	$\langle a,d,b,e,h\rangle$
5	$\langle a,c,d,e,f,d,c,e,f,c,d,e,h\rangle$
6	$\langle a,c,d,e,g\rangle$
7	$\langle \mathbf{a},\mathbf{b},\mathbf{e},\mathbf{g}\rangle$
8	$\langle \mathbf{a},\mathbf{b},\mathbf{d},\mathbf{e}\rangle$
9	$\langle a,d,c,e,f,d,c,e,f,b,d,e,h\rangle$
10	$\langle \mathbf{a},\mathbf{c},\mathbf{d},\mathbf{e},\mathbf{f},\mathbf{b},\mathbf{d},\mathbf{g}\rangle$



Sind die zusätzlichen Event-Folgen 7, 8, 10 konform zum extrahierten Modell ?

=> **Nein !**

case id	trace
1	$\langle a,b,d,e,h\rangle$
2	$\langle a,d,c,e,g\rangle$
3	$\langle a,c,d,e,f,b,d,e,g\rangle$
4	$\langle a,d,b,e,h\rangle$
5	$\langle a,c,d,e,f,d,c,e,f,c,d,e,h\rangle$
6	$\langle a,c,d,e,g\rangle$
7	$\langle \mathbf{a},\mathbf{b},\mathbf{e},\mathbf{g}\rangle$
8	$\langle \mathbf{a},\mathbf{b},\mathbf{d},\mathbf{e}\rangle$
9	$\langle a,d,c,e,f,d,c,e,f,b,d,e,h\rangle$
10	$\langle \mathbf{a},\mathbf{c},\mathbf{d},\mathbf{e},\mathbf{f},\mathbf{b},\mathbf{d},\mathbf{g}\rangle$



Bewertung: Petrinetze

+ **Einfache** und **wenige** Notationselemente.
+ **Graphisch** gut darstellbar.
+ Marken: übersichtliche **Visualisierung** des **Systemzustands**.
+ Syntax und Semantik **formal** definiert.
+ **Werkzeuge** zur Erstellung, Analyse, Simulation, Code-Generierung vorhanden (z.B. Process Mining).
+ Gut geeignet für **kooperierende** Prozesse.
- Zunächst keine **Datenmodellierung** (kann aber dahin erweitern).

Zusammenfassung

In diesem Abschnitt:

Petri-Netze als Grundlage für Geschäftsprozessmodellierung und für Process-Mining.

- Petrinetze-Syntax
- Ausführung
- Analyse von Systemen
- Workflow-Netze

Im nächsten Abschnitt:

- **Data-Mining** (im Vergleich zu Process-Mining).

Definition Netz: Beispiel

Nachrichten-Queue: **Netz (S,T,F)** mit

- **S = {**empfangsbereit, Bereit Queue zu füllen, Queue gefüllt, Queue leer, Bereit zur Verarbeitung, Bereit zur Nachrichtenentnahme**}**
- **T = {**Nachricht annehmen, Queue füllen, Nachricht entnehmen, Nachricht verarbeiten**}**
- **F = {**(empfangsbereit, Nachricht annehmen), (Nachricht annehmen, Bereit Queue zu füllen), (Bereit Queue zu füllen, Queue füllen), (Queue füllen, empfangsbereit), (Queue füllen, Queue gefüllt), (Queue gefüllt, Nachricht entnehmen), (Nachricht entnehmen, Queue leer), (Queue leer, Queue füllen), (Bereit zur Nachrichtenentnahme, Nachricht entnehmen), (Nachricht entnehmen, Bereit zur Verarbeitung), (Bereit zur Verarbeitung, Nachricht verarbeiten), (Nachricht verarbeiten, Bereit zur Nachrichtenentnahme)**}**

Beispiel: Ampelschaltung

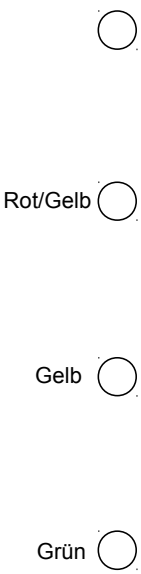
Wir wollen eine Ampelschaltung modellieren.

- Zwei Ampeln an einer Kreuzung
- Nord-Süd-Ampel und Ost-West-Ampel
- Abbieger-Ampel vernachlässigen wir
- Folgende Stellen gibt es:
  - Rot
  - Rot/Gelb
  - Gelb
  - Grün

Beispiel: Ampelschaltung

Wir wollen eine Ampelschaltung modellieren.

- Zwei Ampeln an einer Kreuzung
- Nord-Süd-Ampel und Ost-West-Ampel
- Abbieger-Ampel vernachlässigen wir
- Folgende Stellen gibt es:
  - Rot
  - Rot/Gelb
  - Gelb
  - Grün

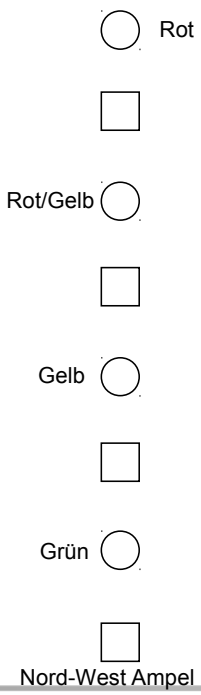


Nord-West Ampel

Beispiel: Ampelschaltung

Wir wollen eine Ampelschaltung modellieren.

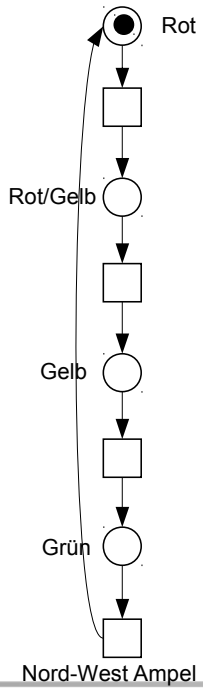
- Zwei Ampeln an einer Kreuzung
- Nord-Süd-Ampel und Ost-West-Ampel
- Abbieger-Ampel vernachlässigen wir
- Folgende Stellen gibt es:
  - Rot
  - Rot/Gelb
  - Gelb
  - Grün



Beispiel: Ampelschaltung

Wir wollen eine Ampelschaltung modellieren.

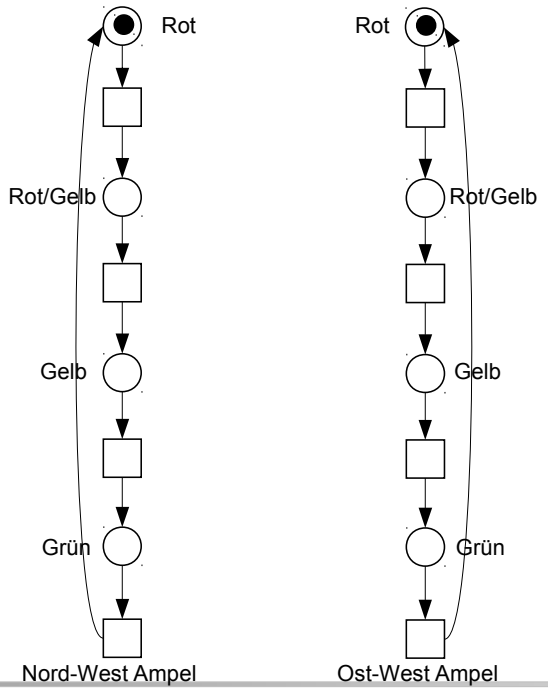
- Zwei Ampeln an einer Kreuzung
- Nord-Süd-Ampel und Ost-West-Ampel
- Abbieger-Ampel vernachlässigen wir
- Folgende Stellen gibt es:
  - Rot
  - Rot/Gelb
  - Gelb
  - Grün



Beispiel: Ampelschaltung

Wir wollen eine Ampelschaltung modellieren.

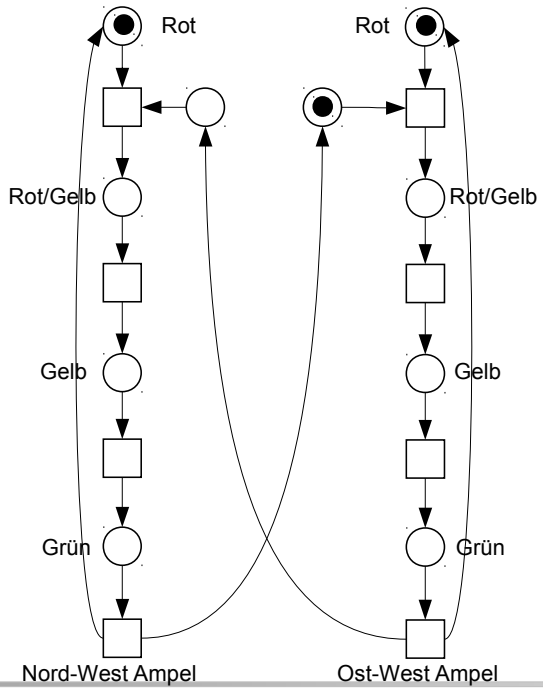
- Zwei Ampeln an einer Kreuzung
- Nord-Süd-Ampel und Ost-West-Ampel
- Abbieger-Ampel vernachlässigen wir
- Folgende Stellen gibt es:
  - Rot
  - Rot/Gelb
  - Gelb
  - Grün



Beispiel: Ampelschaltung

Wir wollen eine Ampelschaltung modellieren.

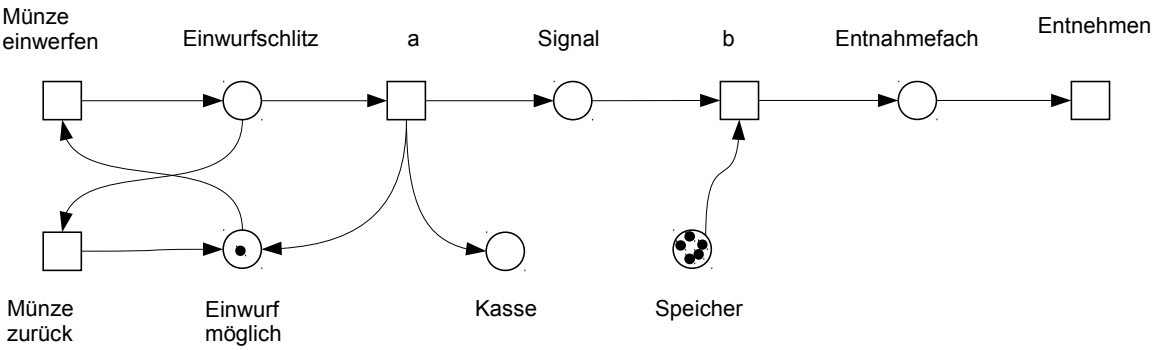
- Zwei Ampeln an einer Kreuzung
- Nord-Süd-Ampel und Ost-West-Ampel
- Abbieger-Ampel vernachlässigen wir
- Folgende Stellen gibt es:
  - Rot
  - Rot/Gelb
  - Gelb
  - Grün



Zweites Beispiel

Was könnte folgendes Petri-Netz modellieren?

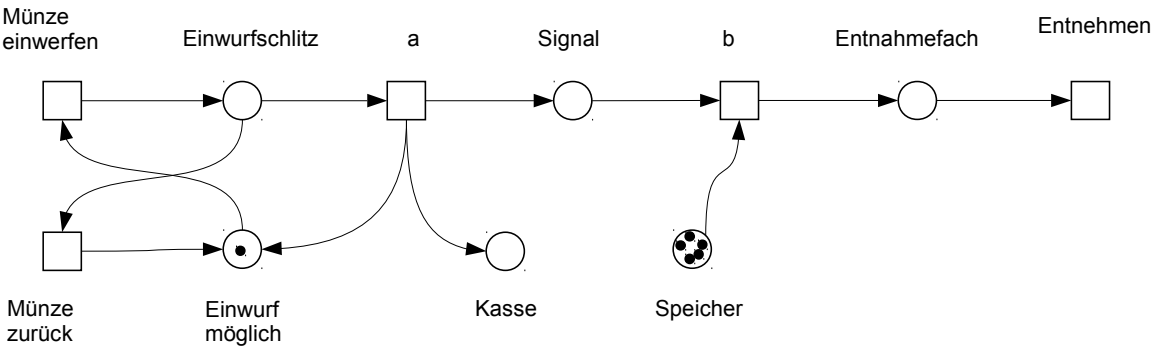
Was könnten die einzelnen Token repräsentieren?



Zweites Beispiel

Was könnte folgendes Petri-Netz modellieren?

Was könnten die einzelnen Token repräsentieren?



Antwort:
Es könnte sich z.B. um einen Getränkeautomaten handeln. Dabei können die Token einerseits „Münzen“ oder auch „Getränkeflaschen“ repräsentieren.

Kein Standard zu Erstellung von Petri-Netzen vorhanden.

Vorgeschlagenes Vorgehen (aus [Bal00]):

1. Stellen und Transitionen auf hohem Abstraktionsniveau identifizieren.
2. Beziehungen ermitteln.
3. Verfeinerung und Ergänzung.
4. Festlegung der Objekte.
5. Schaltregeln identifizieren.
6. Netztyp festlegen.
7. Anfangsmarkierung festlegen.
8. Analyse, Simulation.