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

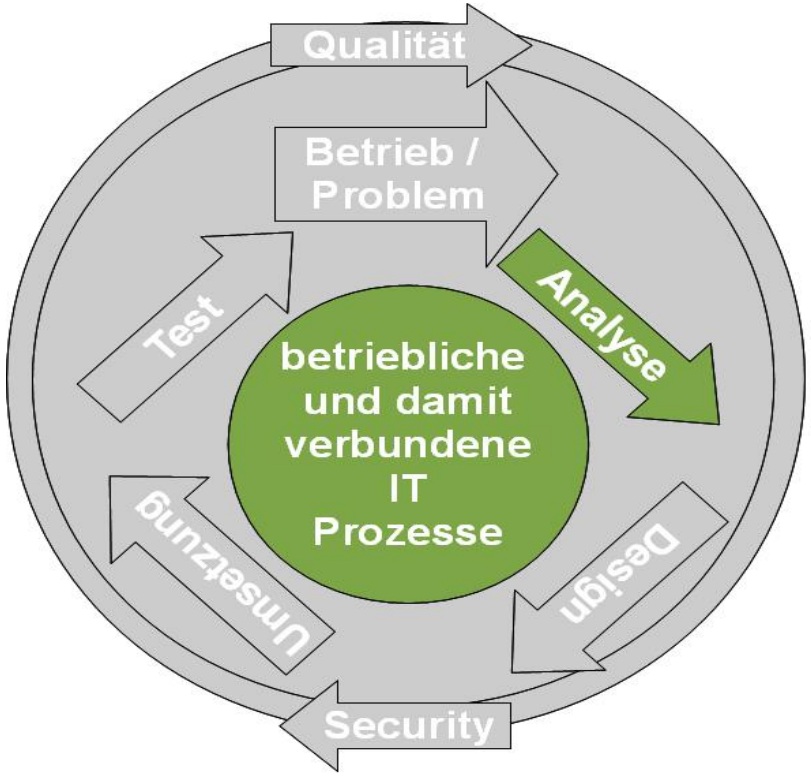
2.1 Petrinetze

**[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.

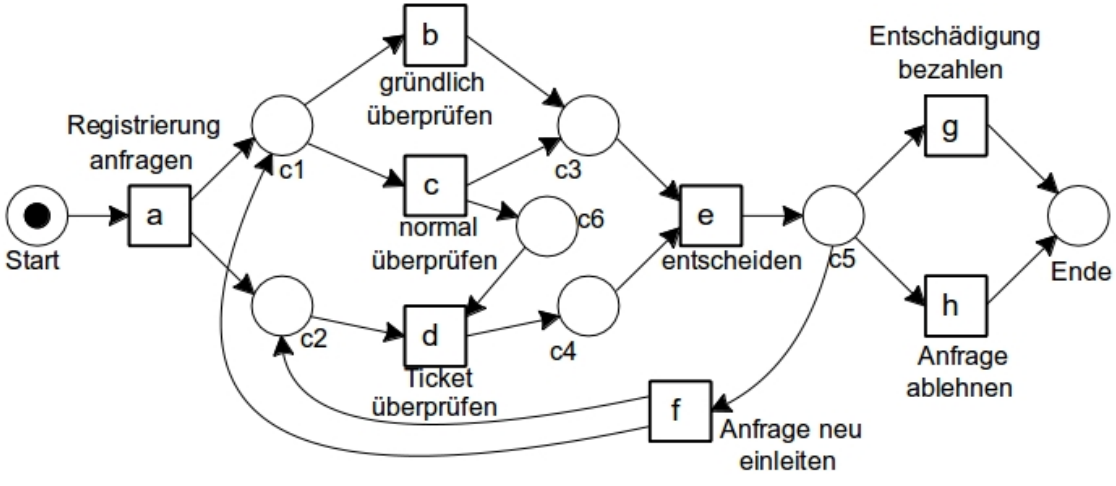
Verschiedene Ansätze dafür vorhanden: Abstract State Machines, Petrinetze, ...

- 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).

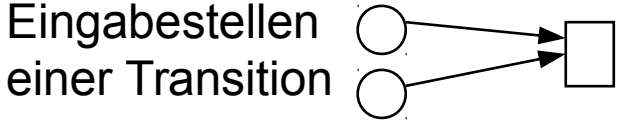


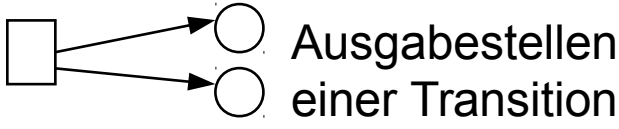
Vorsicht: Existieren heute viele Varianten.

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 !).



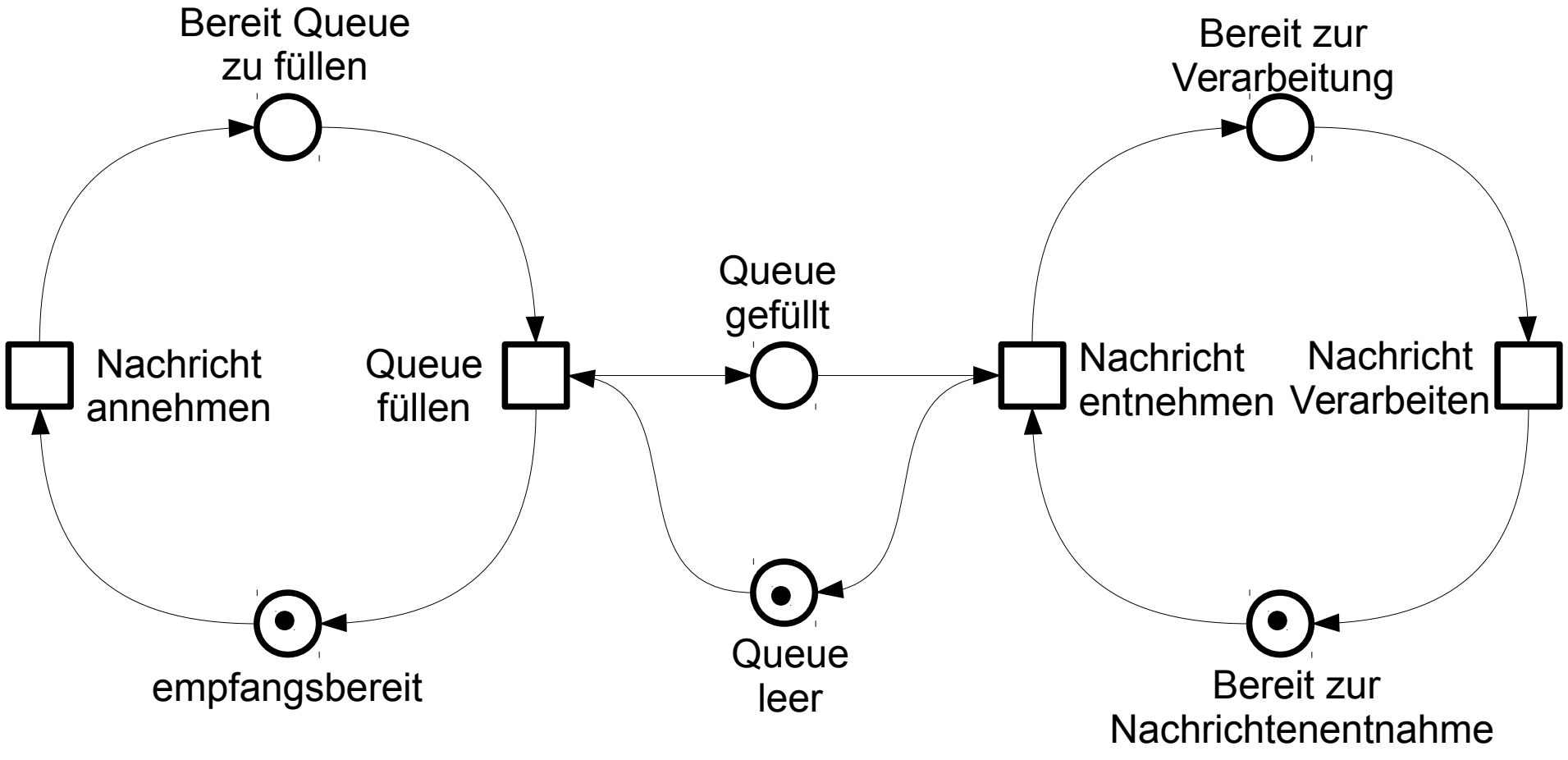


Stellen mit Objekten (sog. **Token** / **Marken**) belegen.

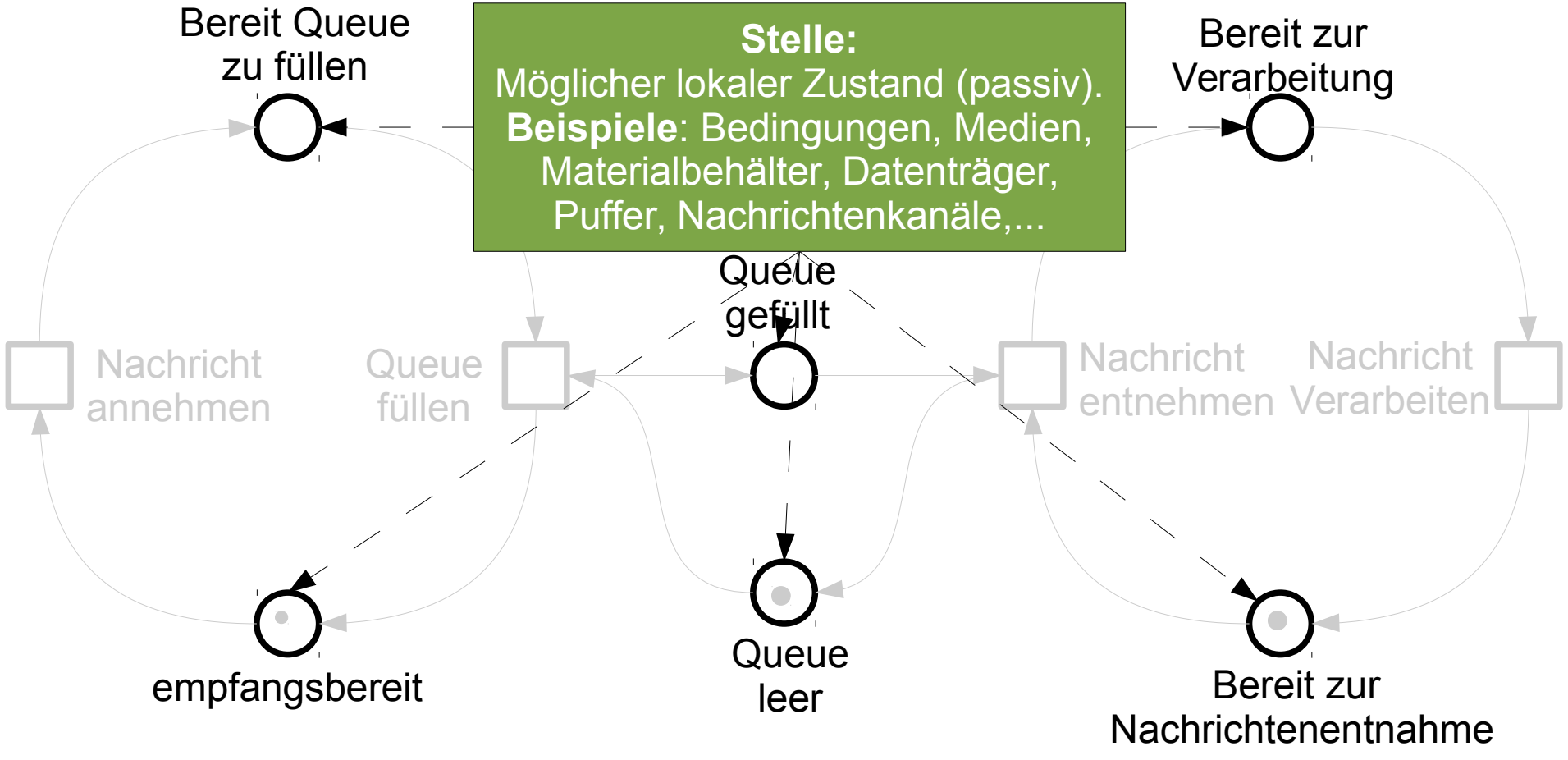
Durchlauf der Token durch Petrinetz beschreibt **dynamisches Verhalten** des Systems.

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

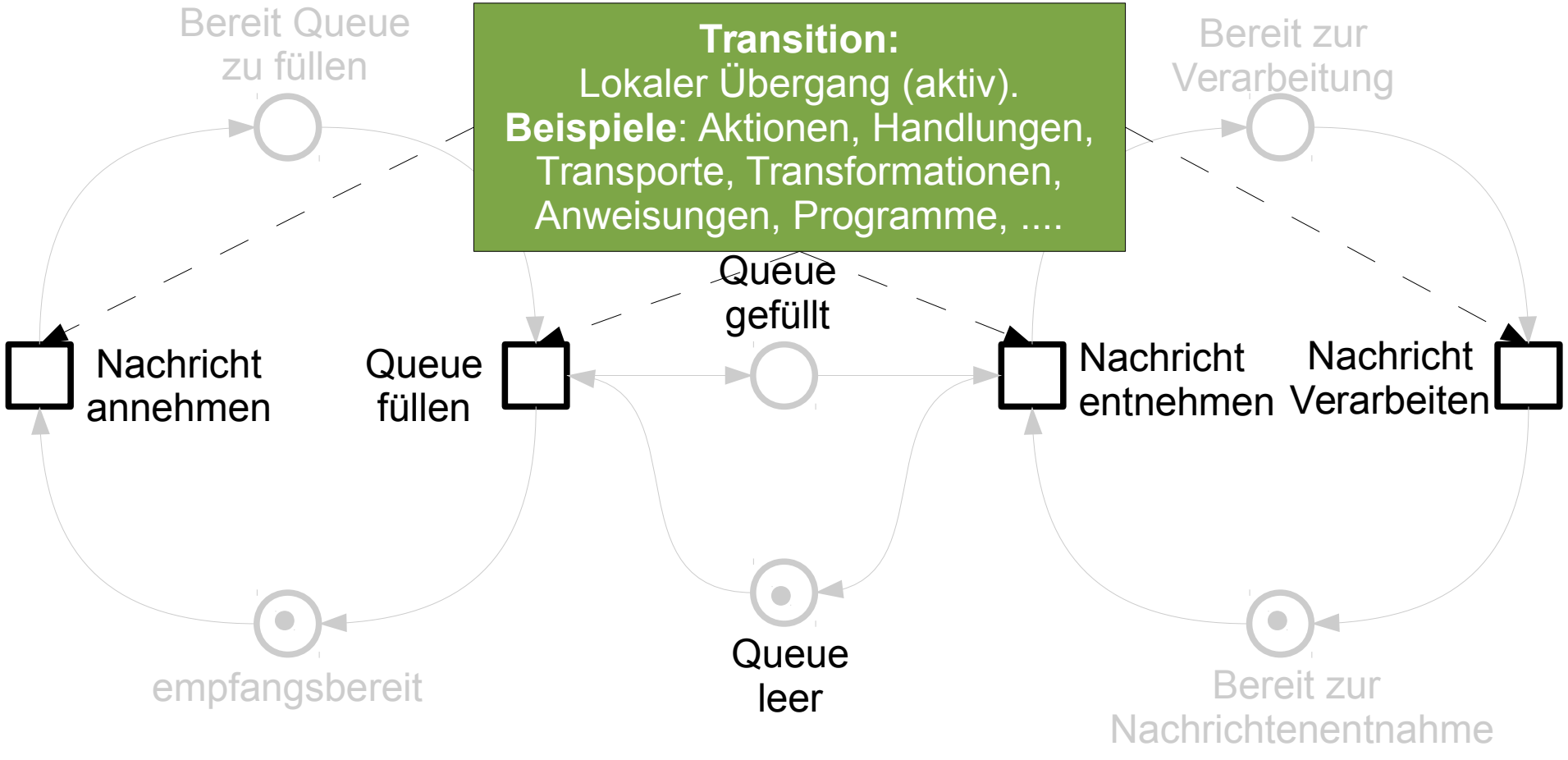
Petrinetz: Beispiel Nachrichten-Queue



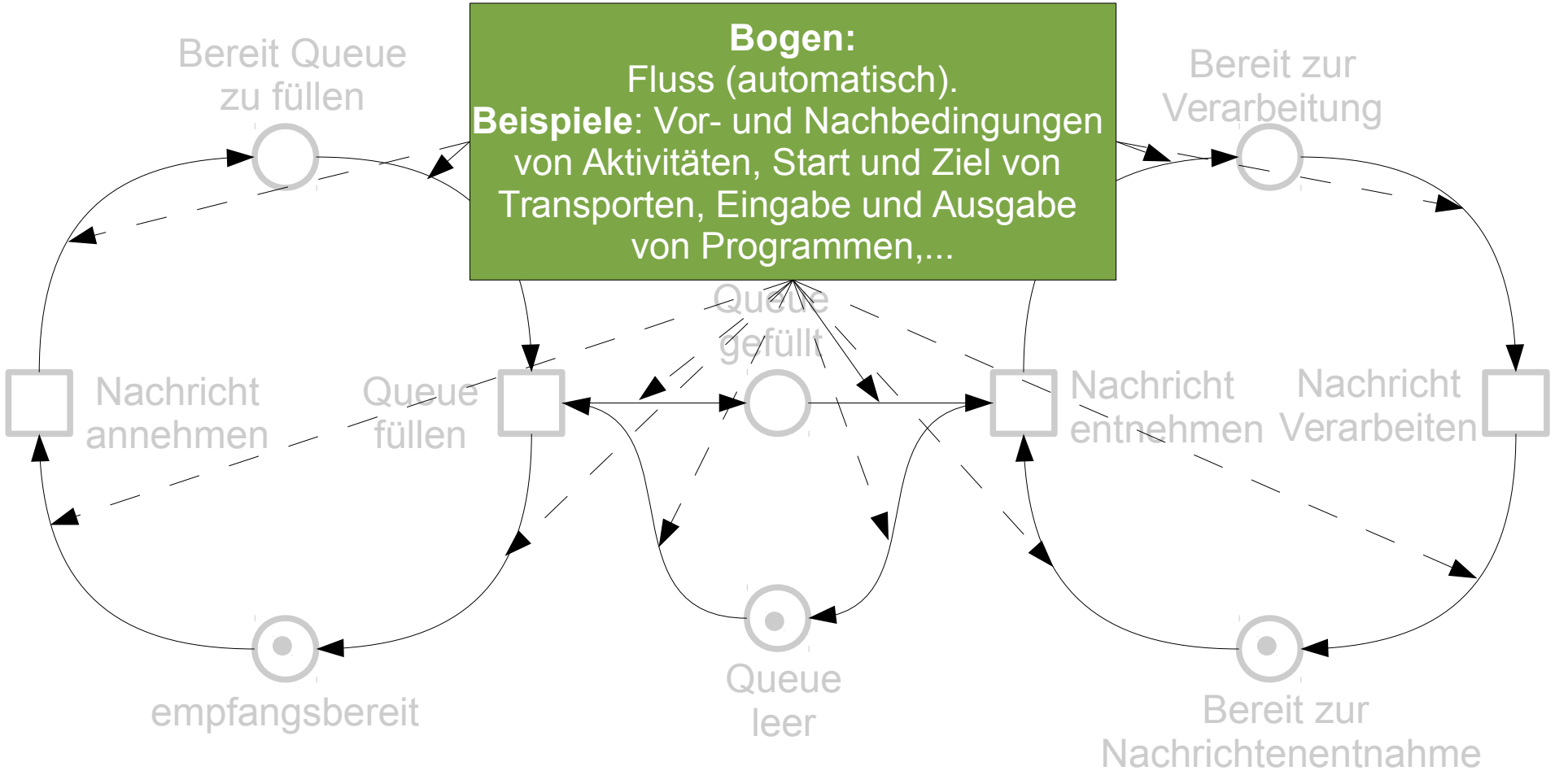
Petrinetz-Syntax: Stelle



Petrinetz-Syntax: Transition



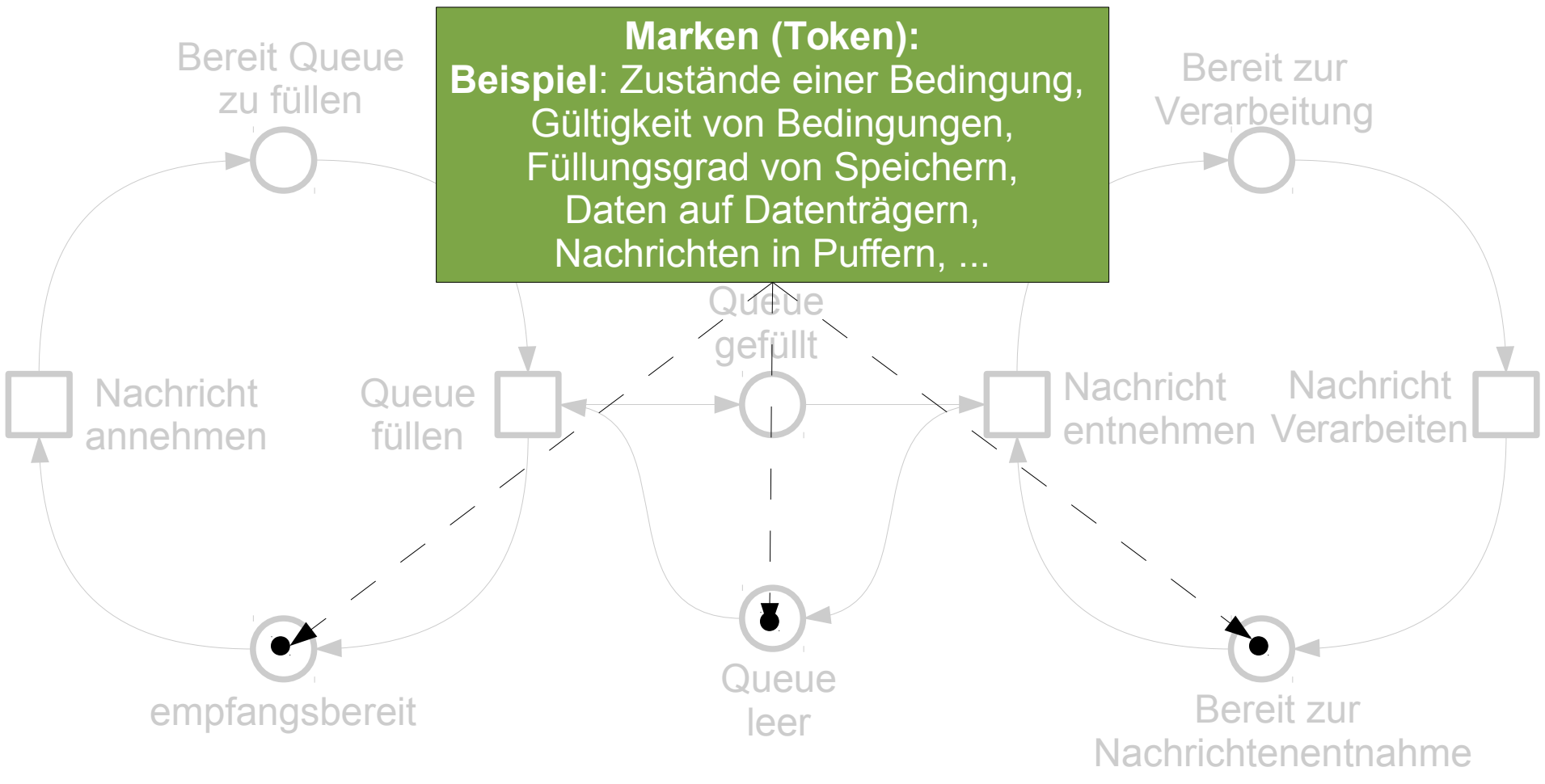
Petrinetz-Syntax: Bogen



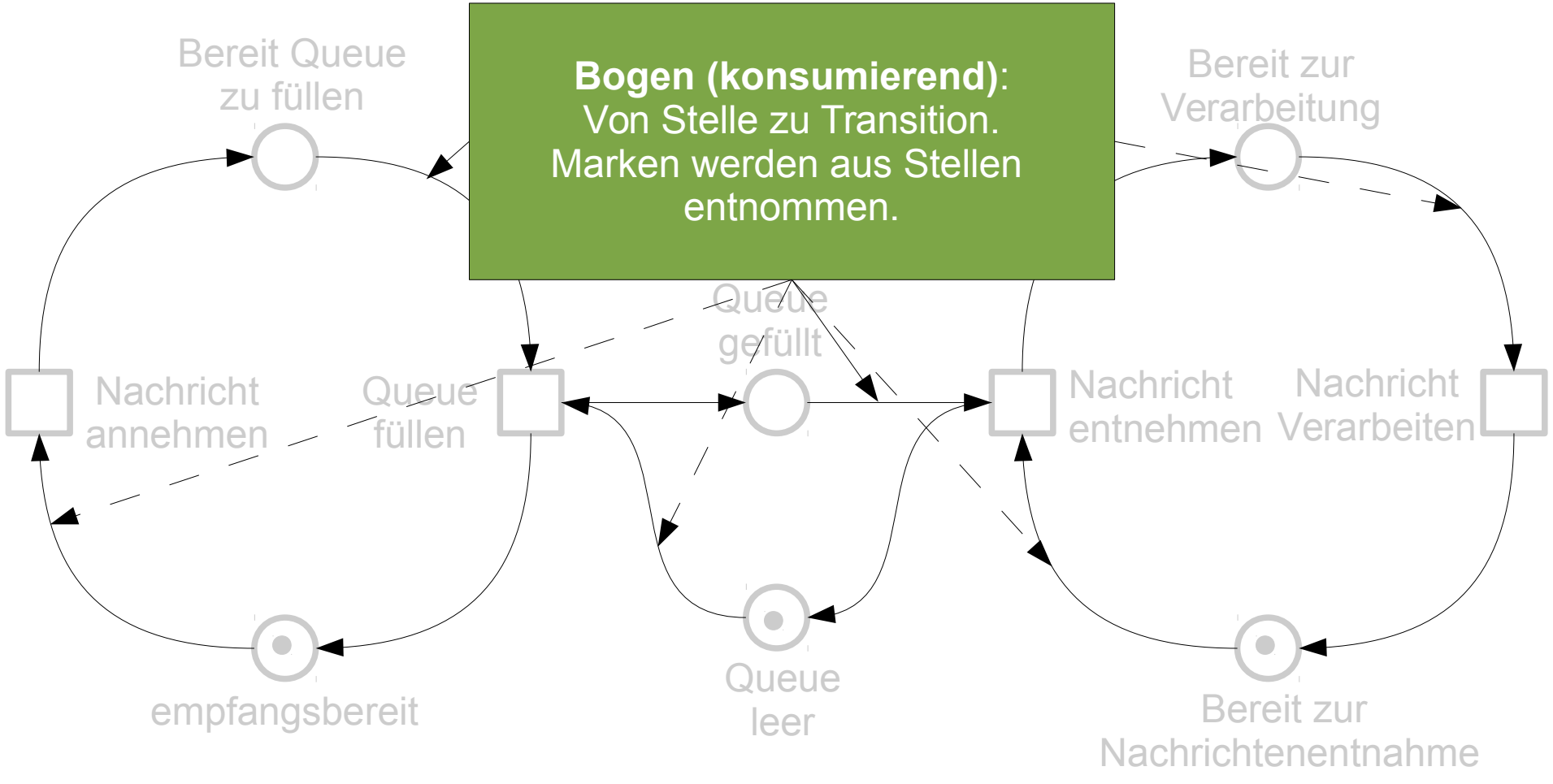
Bogen:
Fluss (automatisch).
Beispiele: Vor- und Nachbedingungen von Aktivitäten, Start und Ziel von Transporten, Eingabe und Ausgabe von Programmen,...

Petrinetz-(Ausführungs-)Syntax: Marken (Token)

Marken (Token):
Beispiel: Zustände einer Bedingung,
Gültigkeit von Bedingungen,
Füllungsgrad von Speichern,
Daten auf Datenträgern,
Nachrichten in Puffern, ...

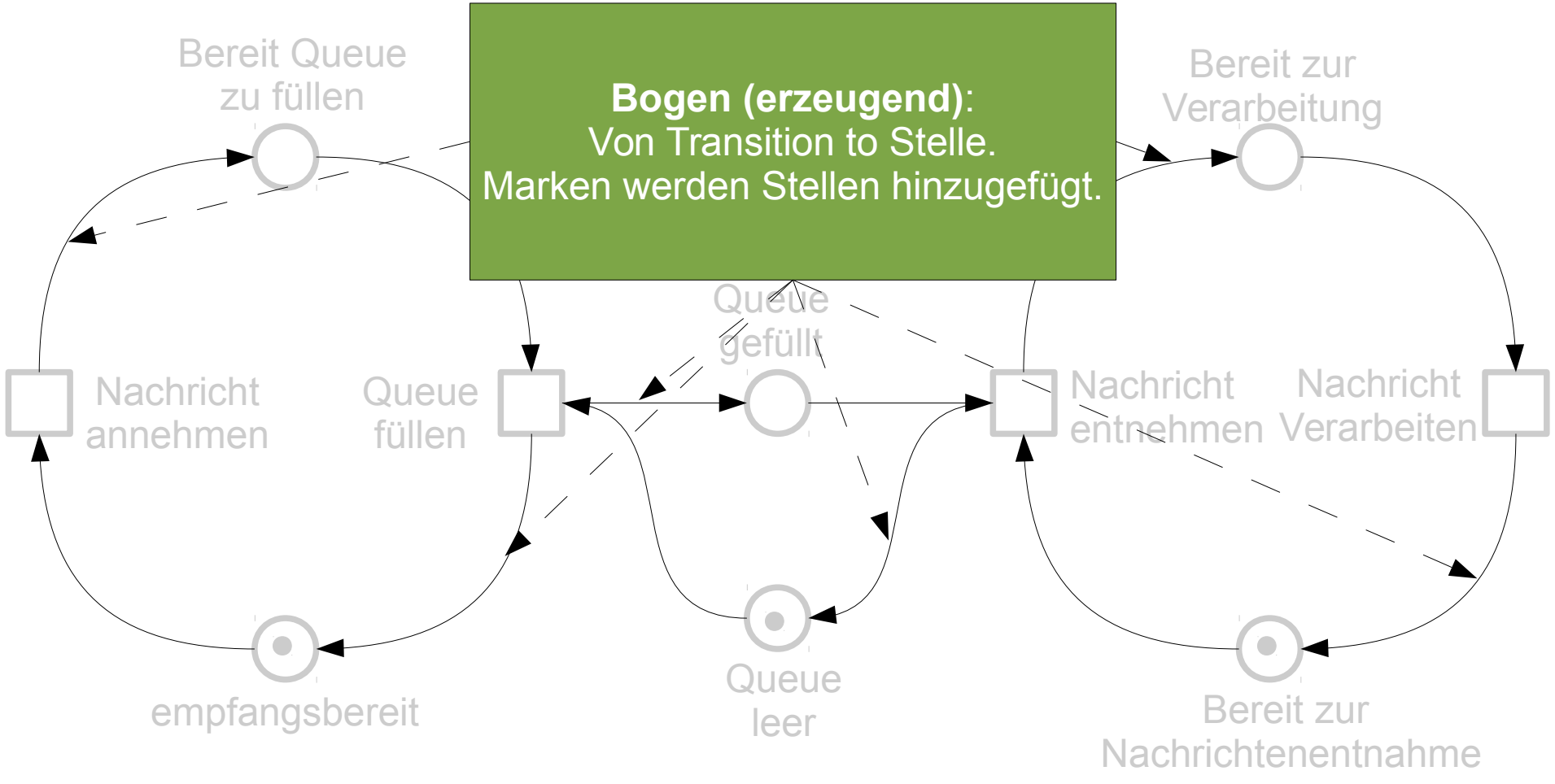


Petrinetz-Syntax: Bogen (konsumierend)



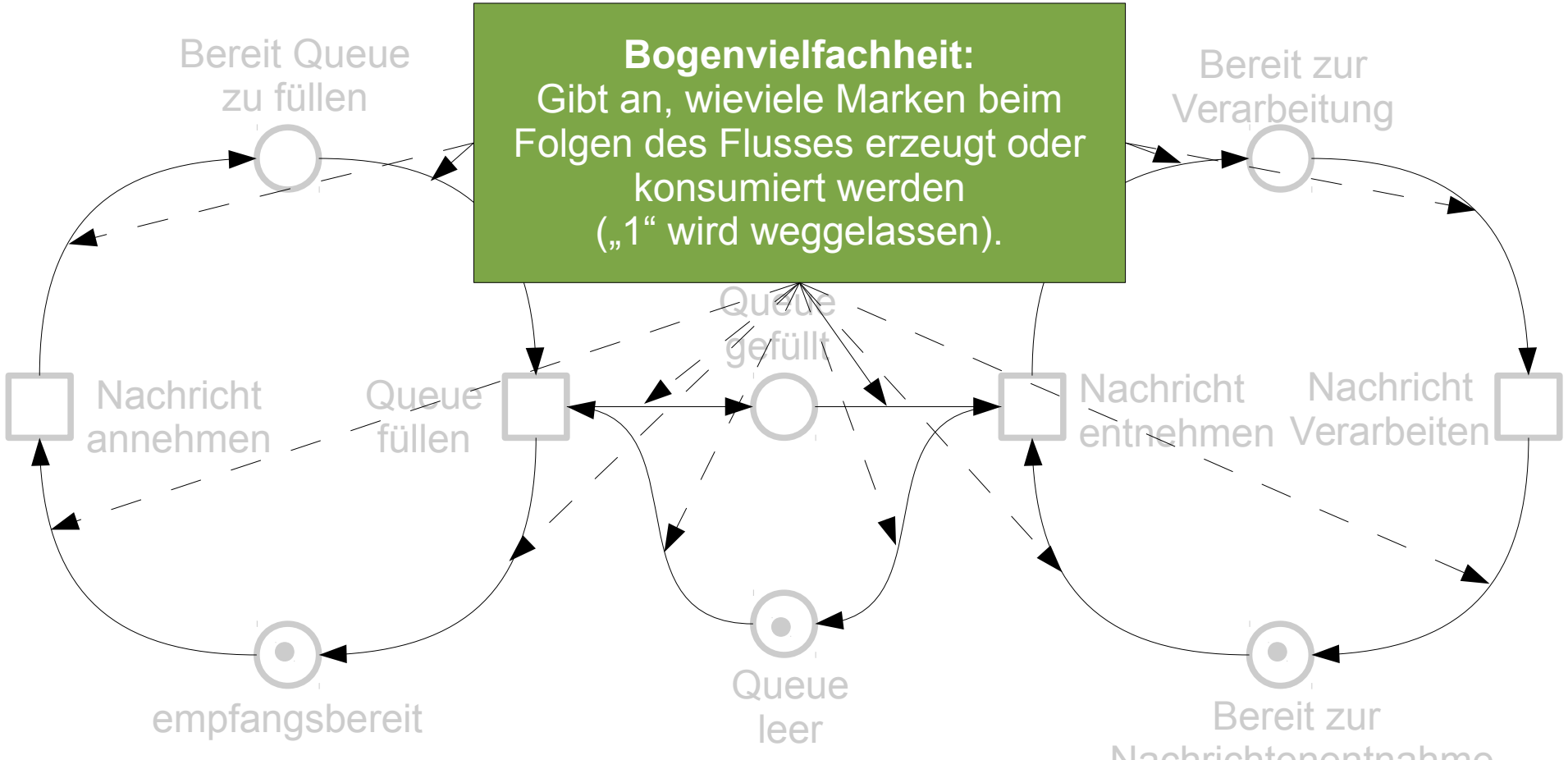
Bogen (konsumierend):
Von Stelle zu Transition.
Marken werden aus Stellen entnommen.

Petrinetz-Syntax: Bogen (erzeugend)



Bogen (erzeugend):
Von Transition to Stelle.
Marken werden Stellen hinzugefügt.

Petrinetz-Syntax: Bogenvielfachheit



Bogenvielfachheit:
Gibt an, wieviele Marken beim Folgen des Flusses erzeugt oder konsumiert werden („1“ wird weggelassen).

Syntaxdefinition Petrinetz

Gegeben:

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

mit: **$S \neq \emptyset$, $T \neq \emptyset$ und $S \cap T = \emptyset$**

- F: Menge von **Bögen:**

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

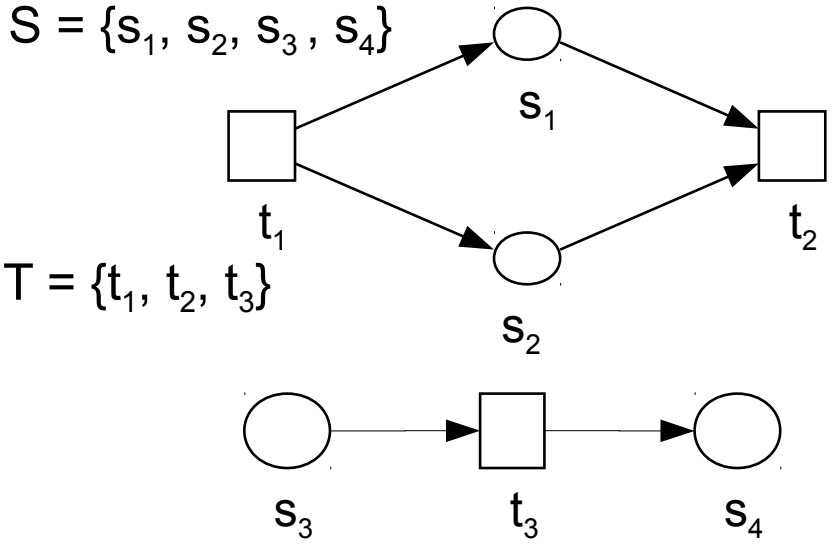
Bogenvielfachheit („Gewicht“) W:

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

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

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

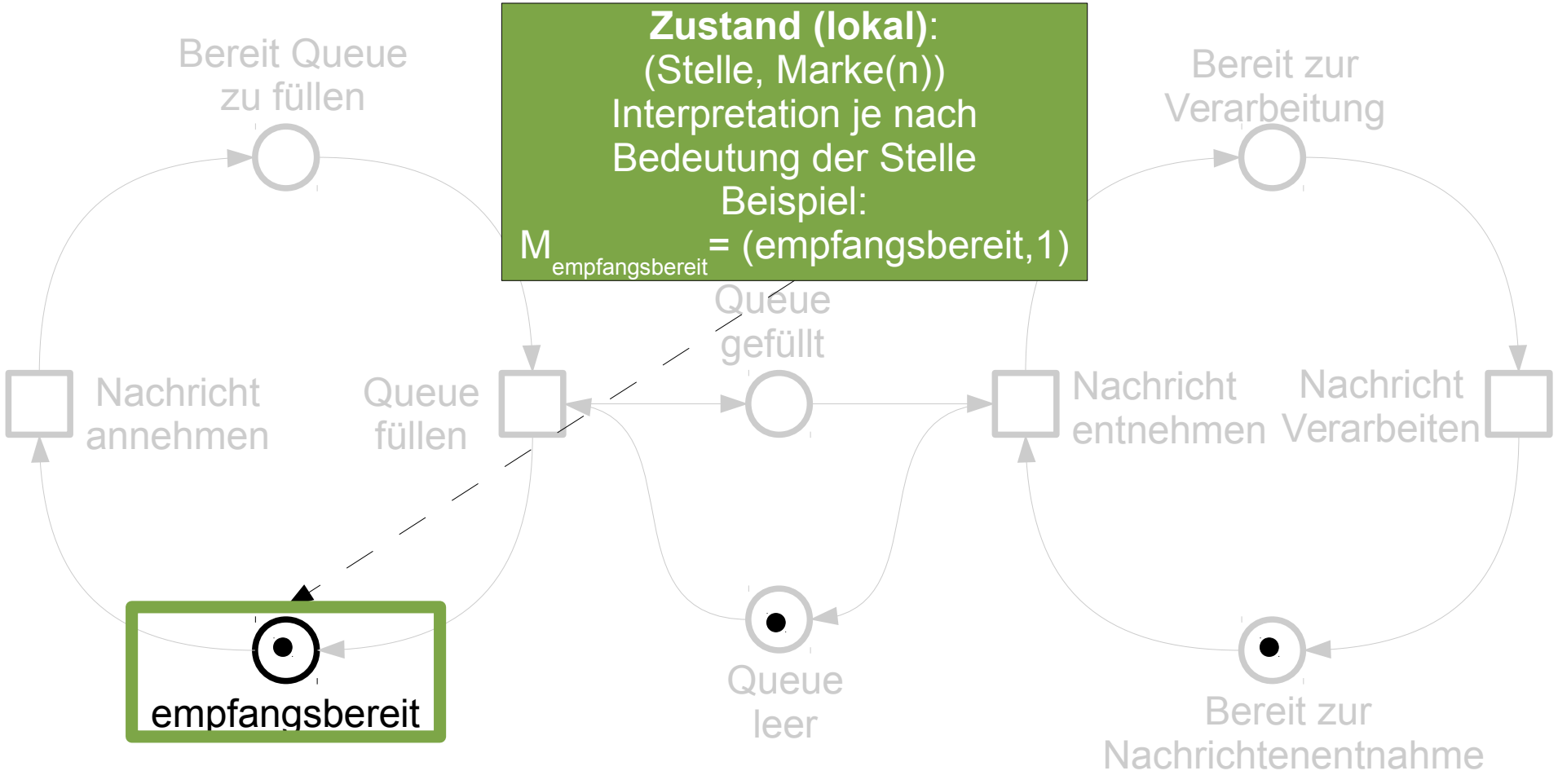
=> (S, T, F, W, M_0): **Petrinetz**.



Überblick Petrinetze

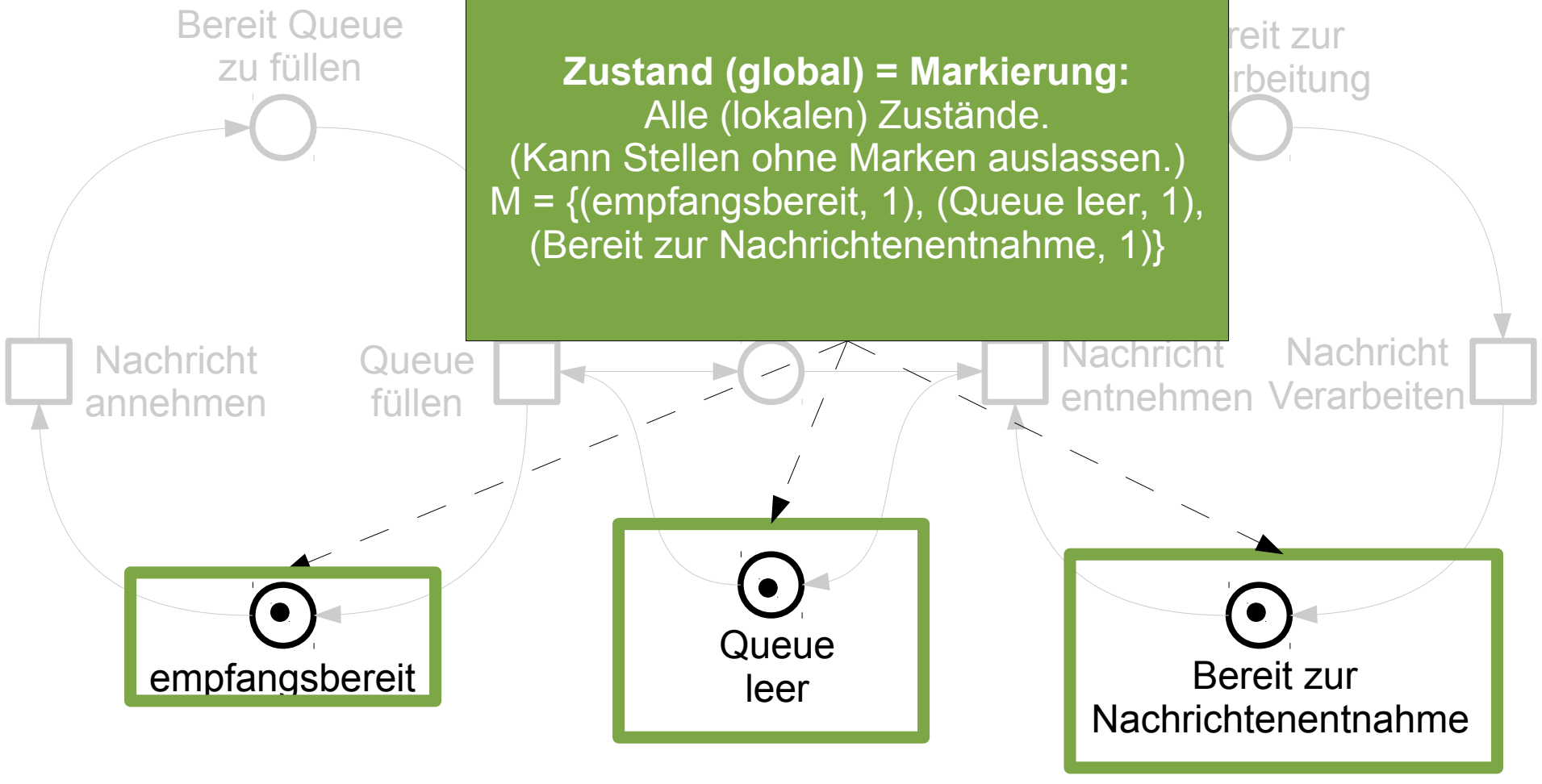
- Petrinetz-Syntax
- Ausführung
- Analyse von Systemen
- Workflow-Netze

Petrinetz-Ausführung: Zustand (lokal)



Petrinetz-Ausführung: Zustand (global)

Zustand (global) = Markierung:
Alle (lokalen) Zustände.
(Kann Stellen ohne Marken auslassen.)
M = {(empfangsbereit, 1), (Queue leer, 1),
(Bereit zur Nachrichtenentnahme, 1)}



Petrinetz: Ausführung

Markierung: Verteilung Marken auf Stellen (aktueller Systemzustand).

Initiale Markierung: Anfangszustand eines Netzes.

Verhaltenssimulation: evolvierende Anzahl Marken pro Stelle beobachten.

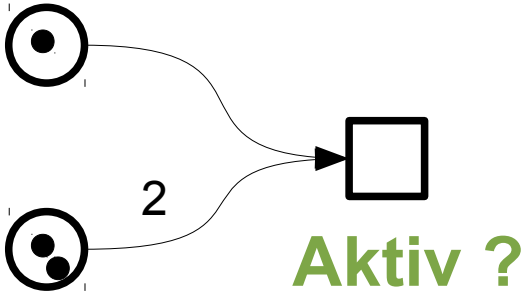
- Basierend auf aktueller Markierung: **aktivierte Transitionen** ermitteln. **Schalten** führt zu Folgemarkierung.
- Unter Folgemarkierung sind (möglicherweise) andere Transitionen aktiviert.
- Solange iterieren, bis keine Transition mehr aktiv (=> „**tote Markierung**“).

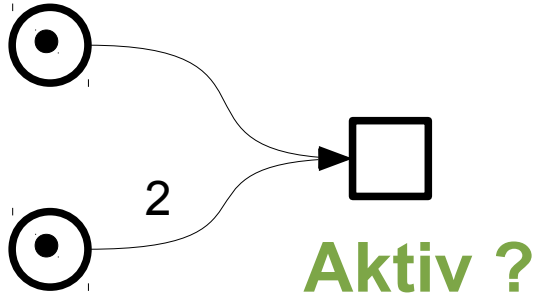
Ablauf: Aktivierung von Transition

Transition t ist **aktiviert**, wenn:

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

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



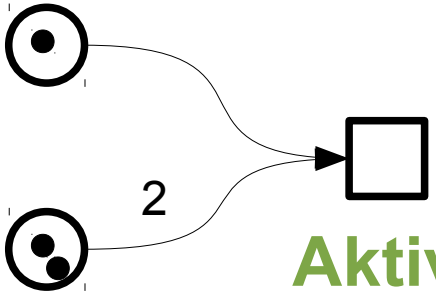


Ablauf: Aktivierung von Transition

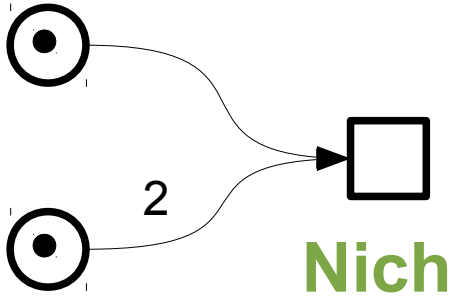
Transition t ist **aktiviert**, wenn:

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

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



Aktiv



Nicht Aktiv

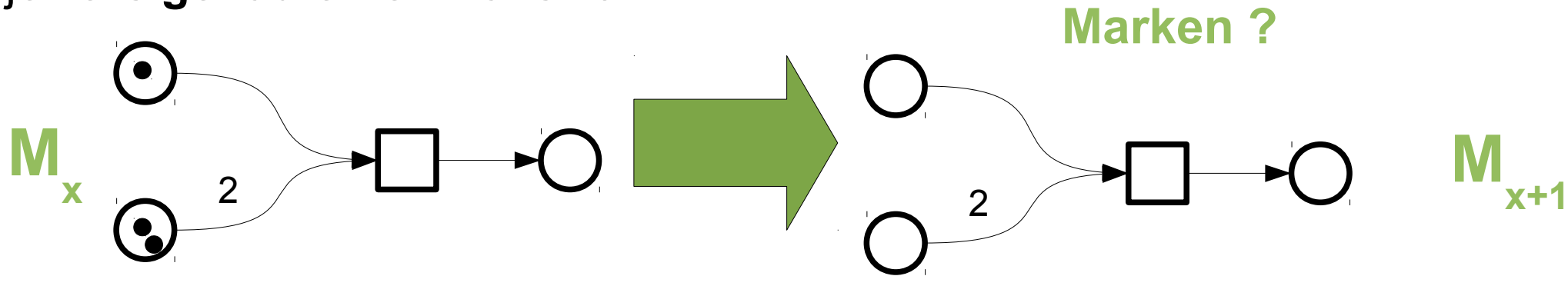
Ablauf: Schalten von Transition

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.



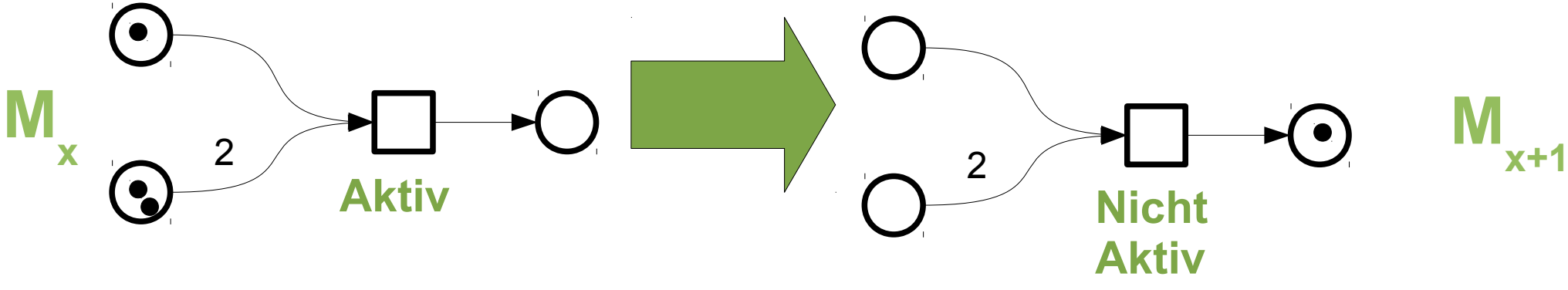
Ablauf: Schalten von Transition

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

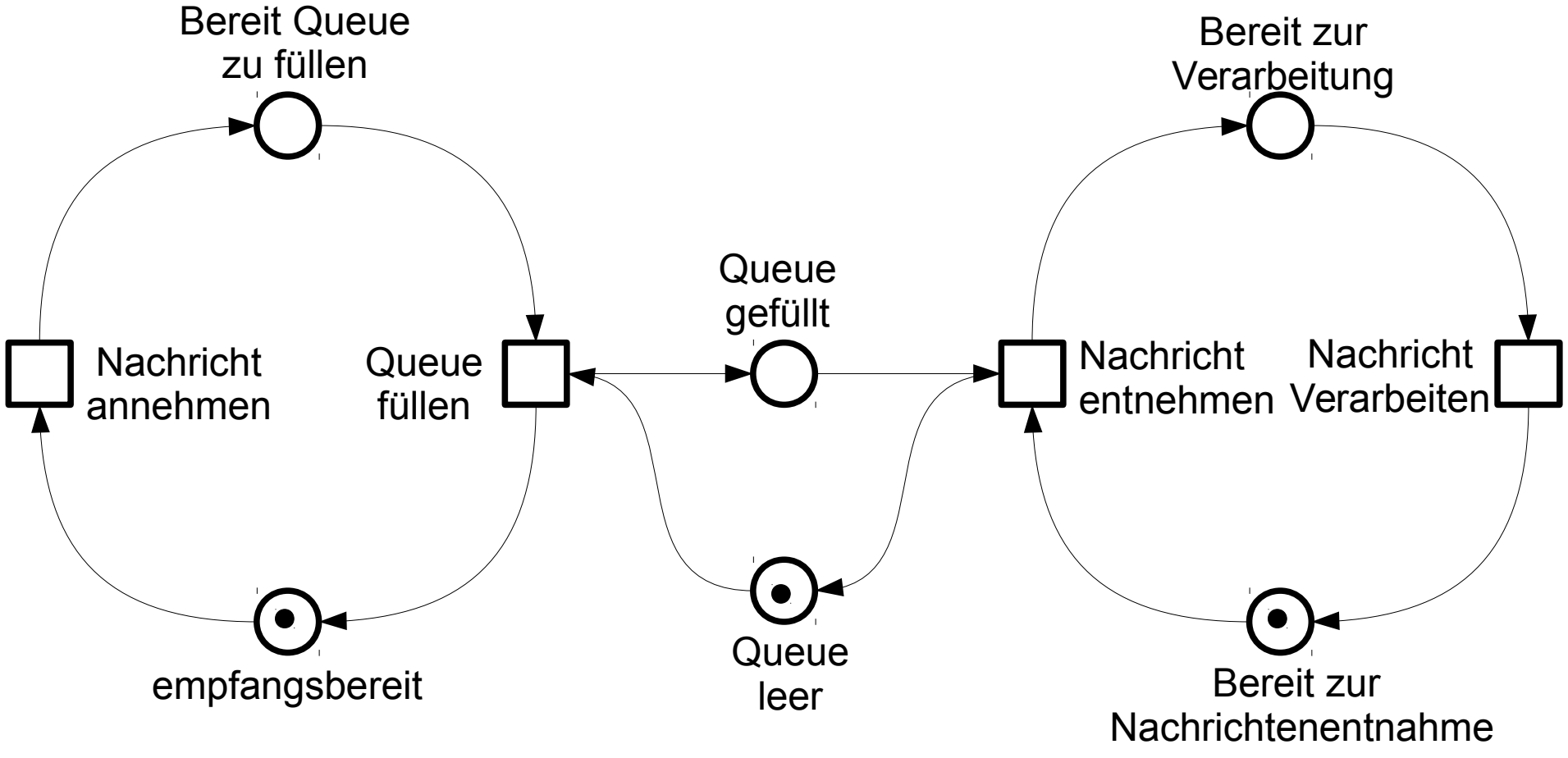
- Marken auf **Vorgänger**-Stellen werden **konsumiert**
- Marken auf **Nachfolger**-Stellen werden **produziert**

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

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

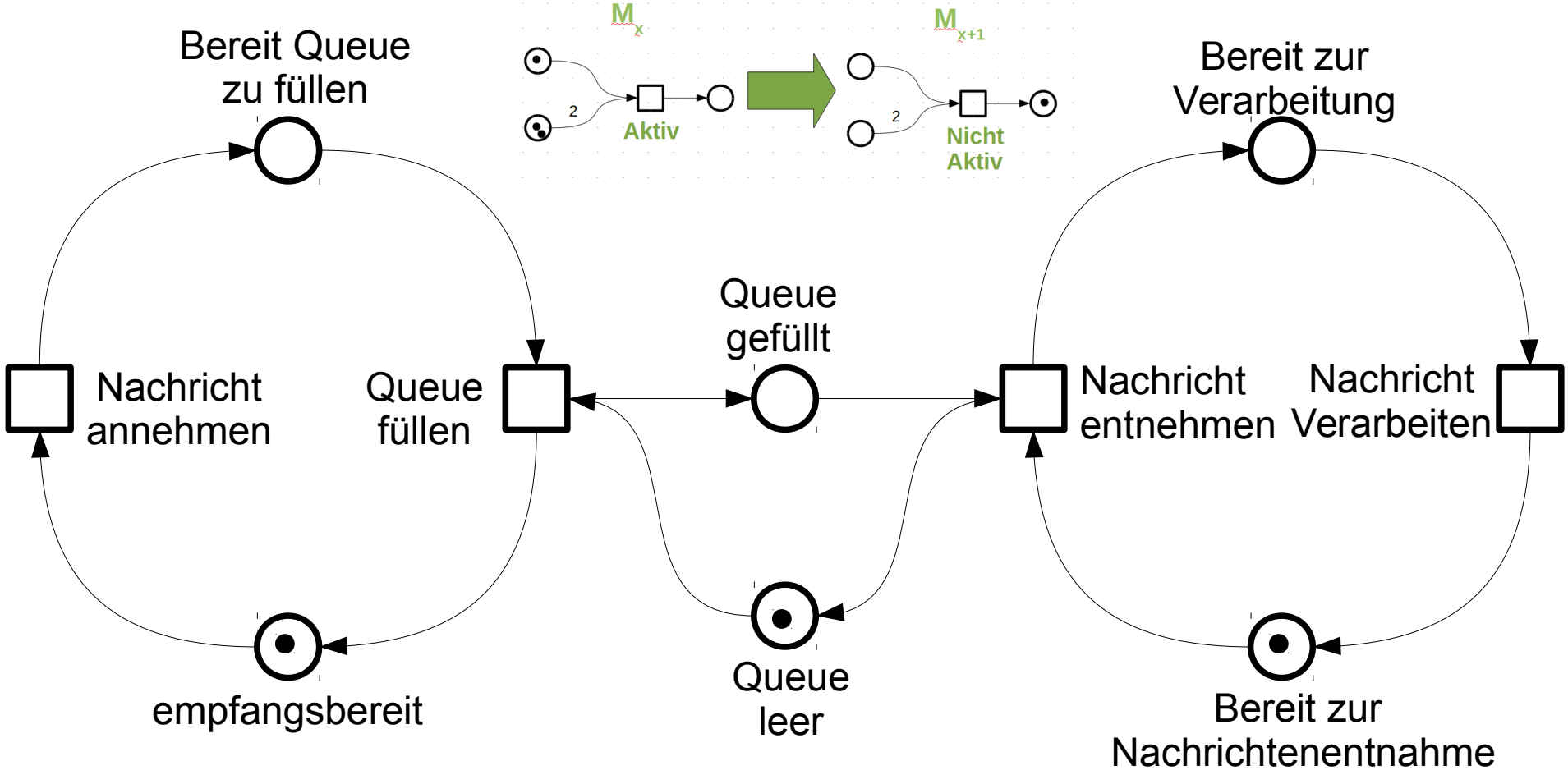


Petrinetz Ablauf: Beispiel Nachrichten-Queue



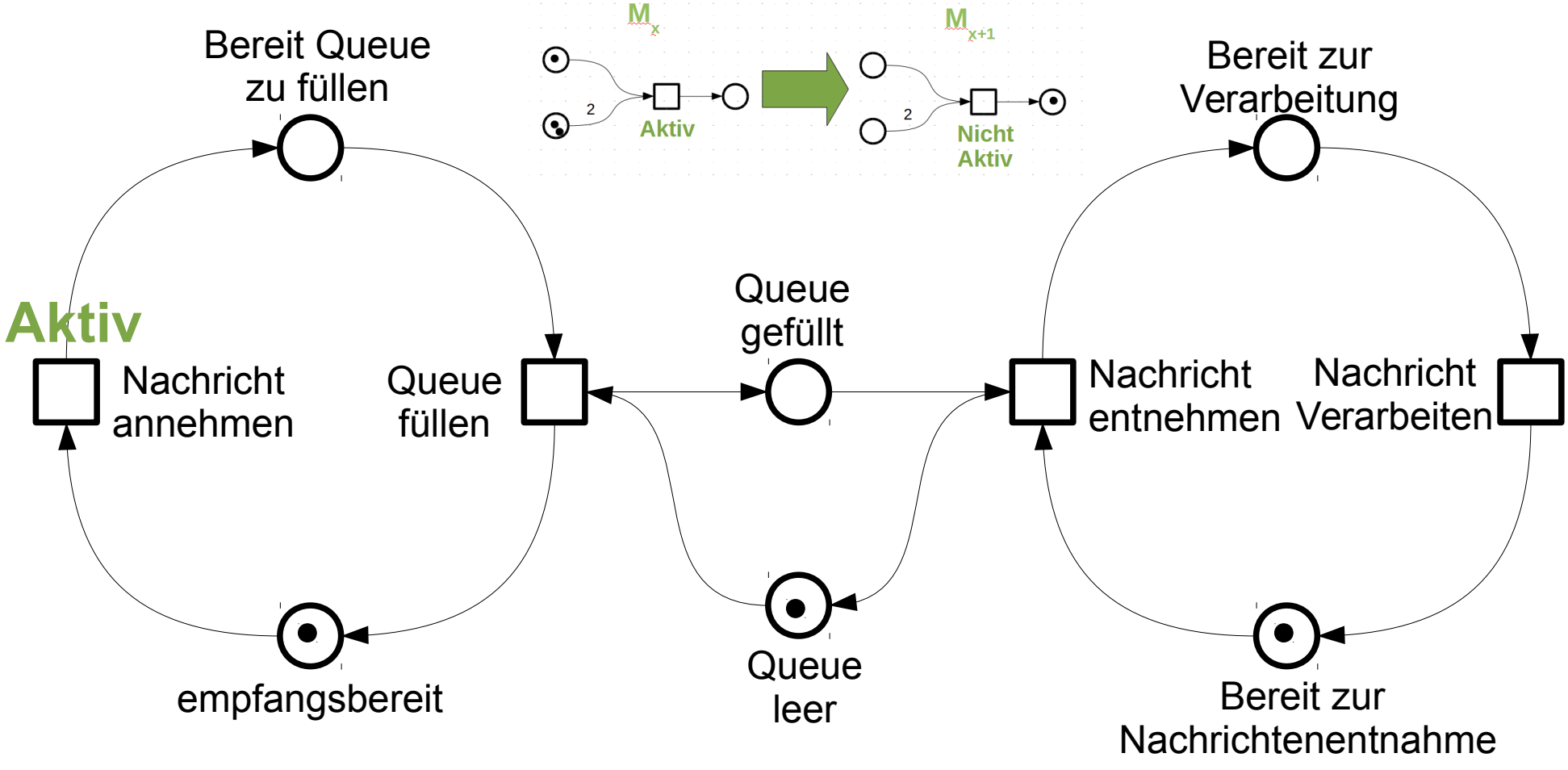
Ablauf: Beispiel (M_0)

Welche Transition(en) aktiviert ?



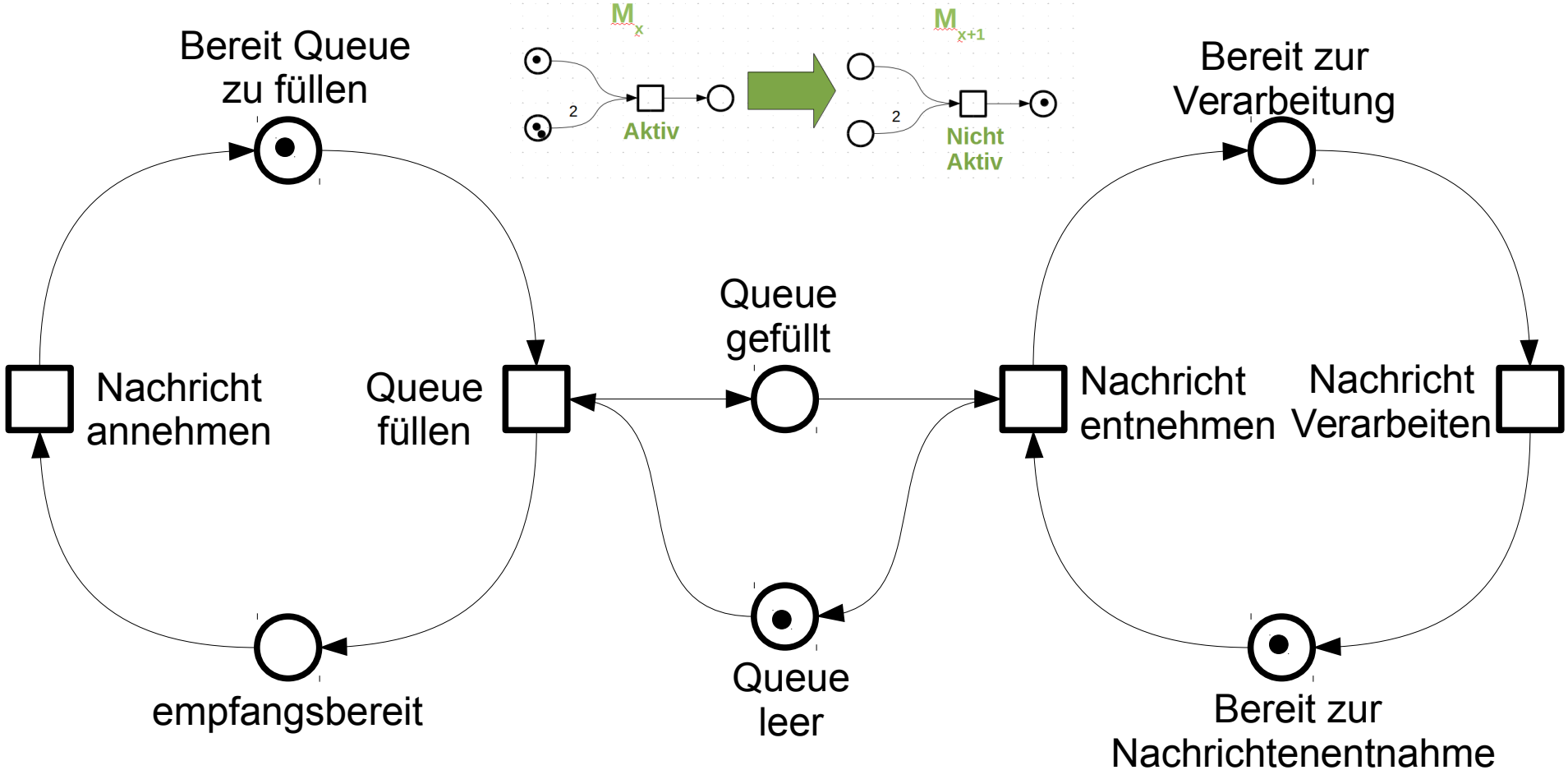
Ablauf: Beispiel (M_0)

Nächster Zustand ?



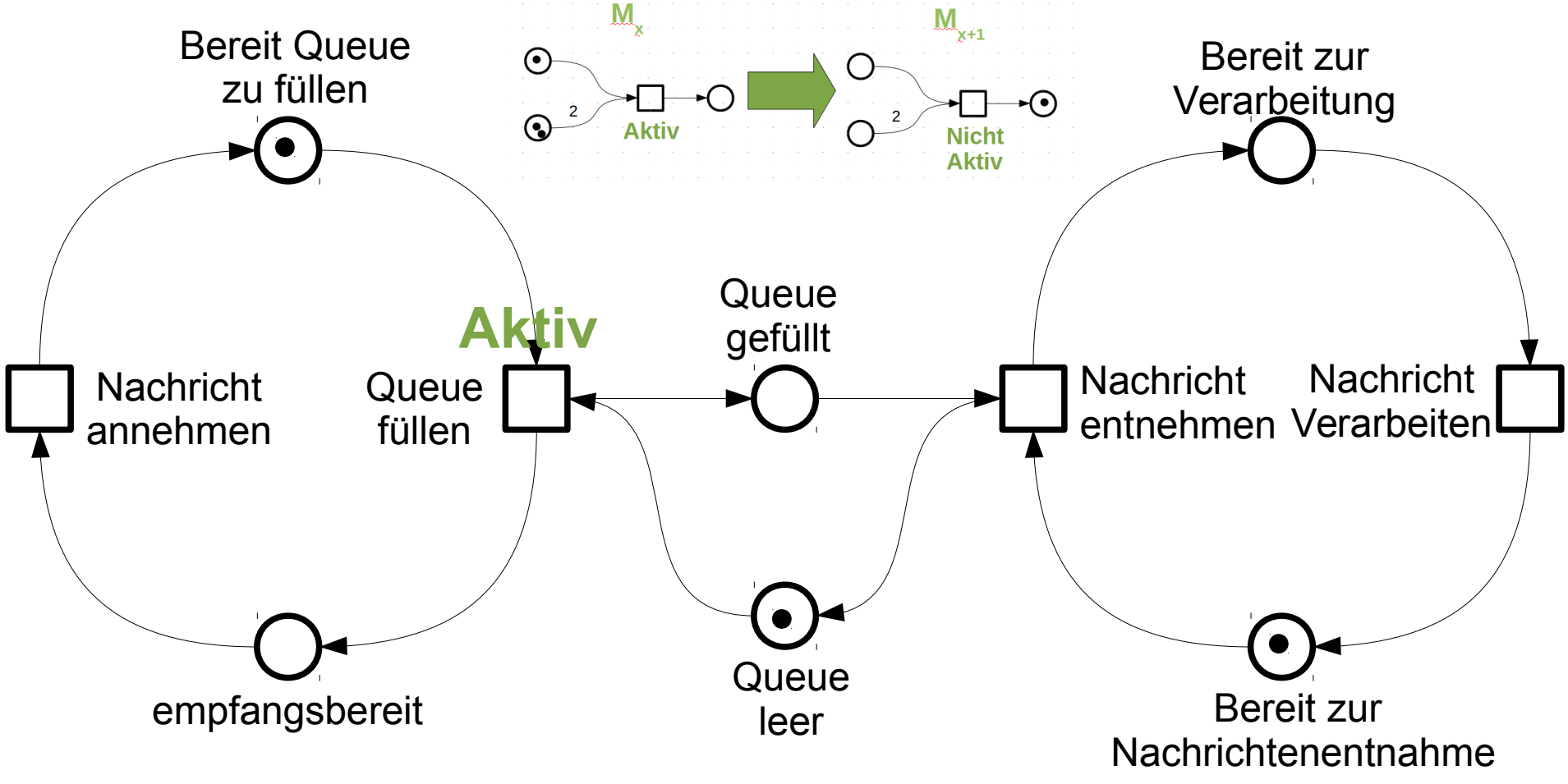
Ablauf: Beispiel (M_1)

Welche Transition(en) aktiviert ?



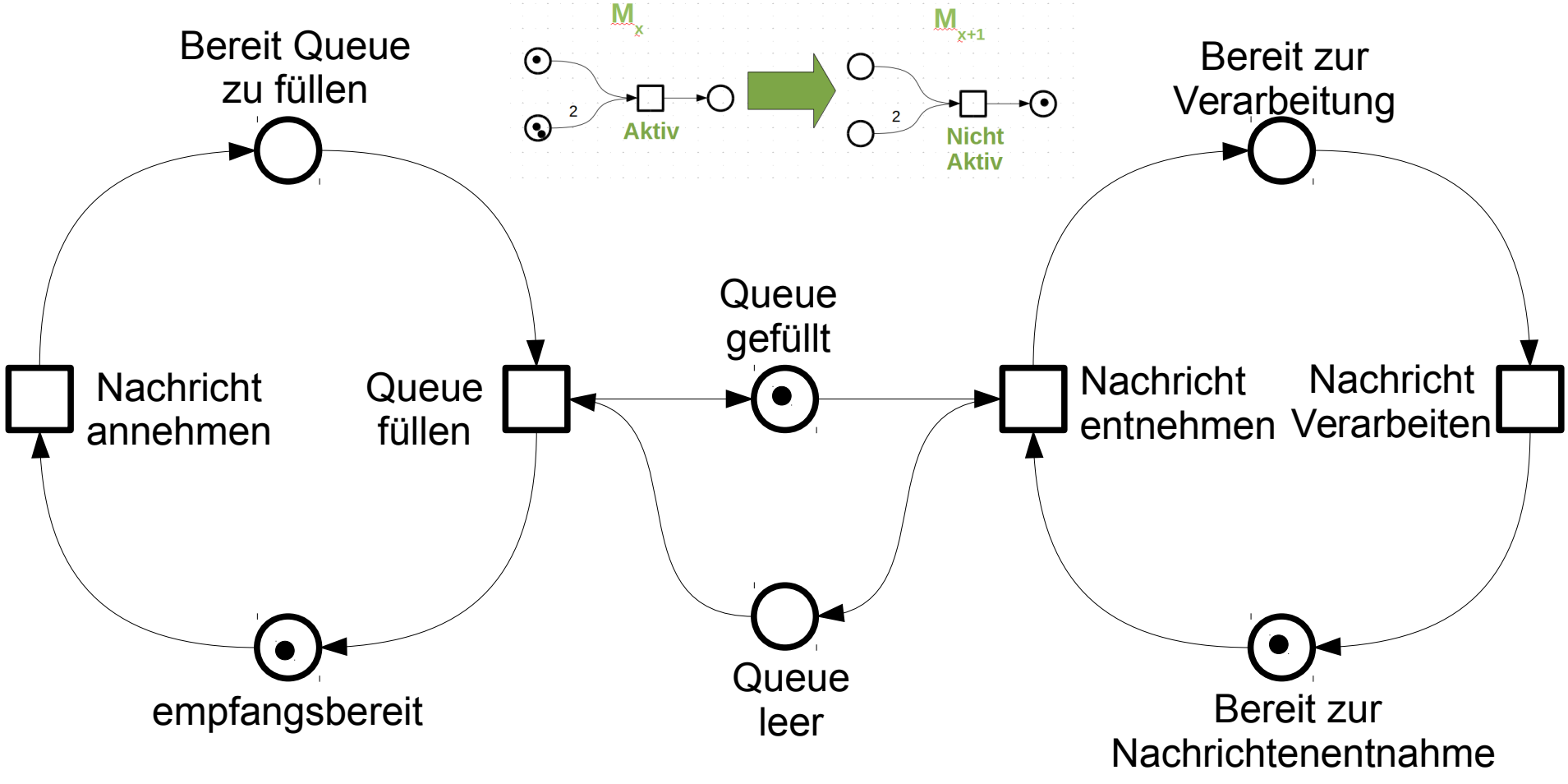
Ablauf: Beispiel (M_1)

Nächster Zustand ?



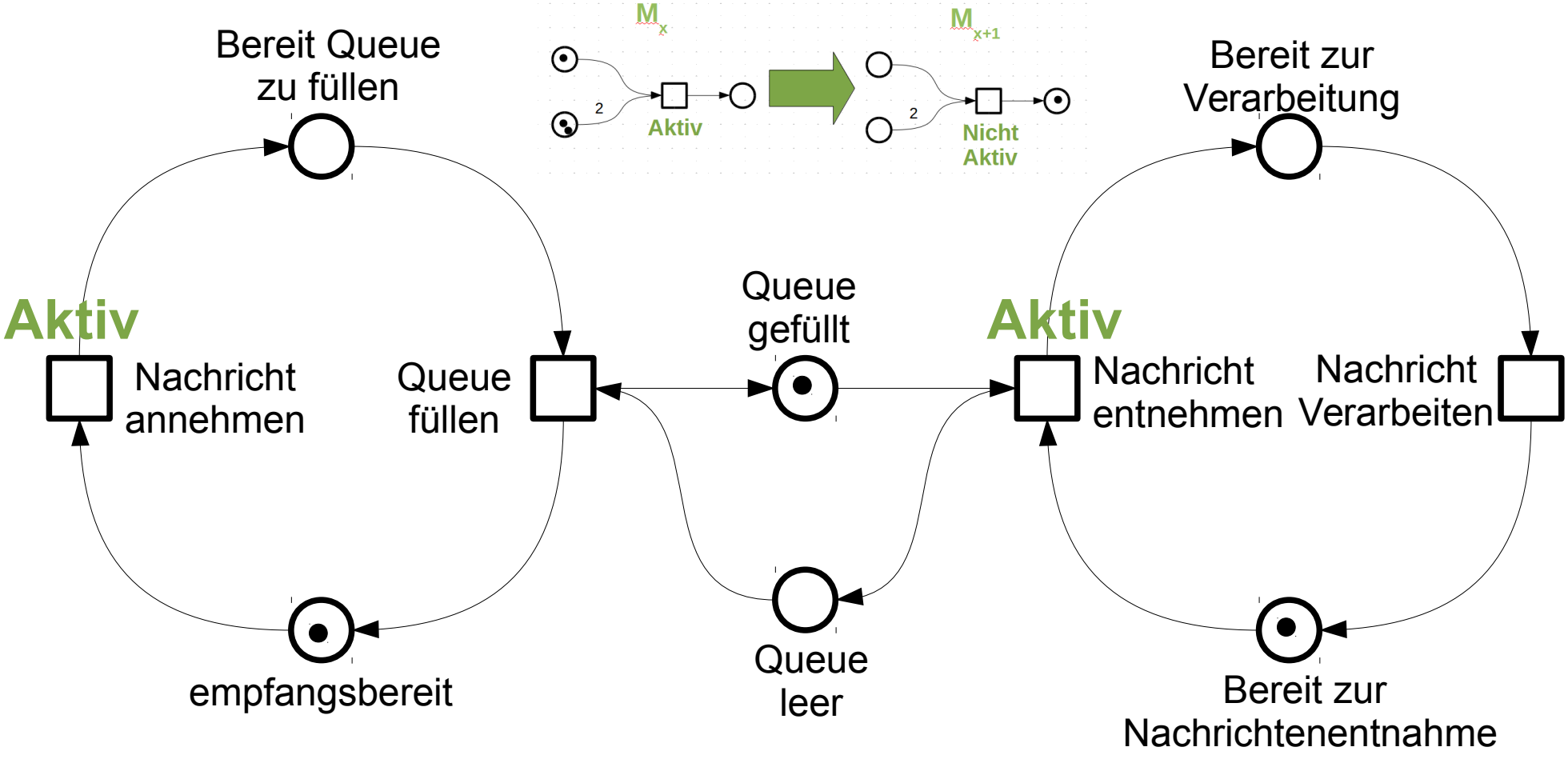
Ablauf: Beispiel (M_2)

Welche Transition(en) aktiviert ?

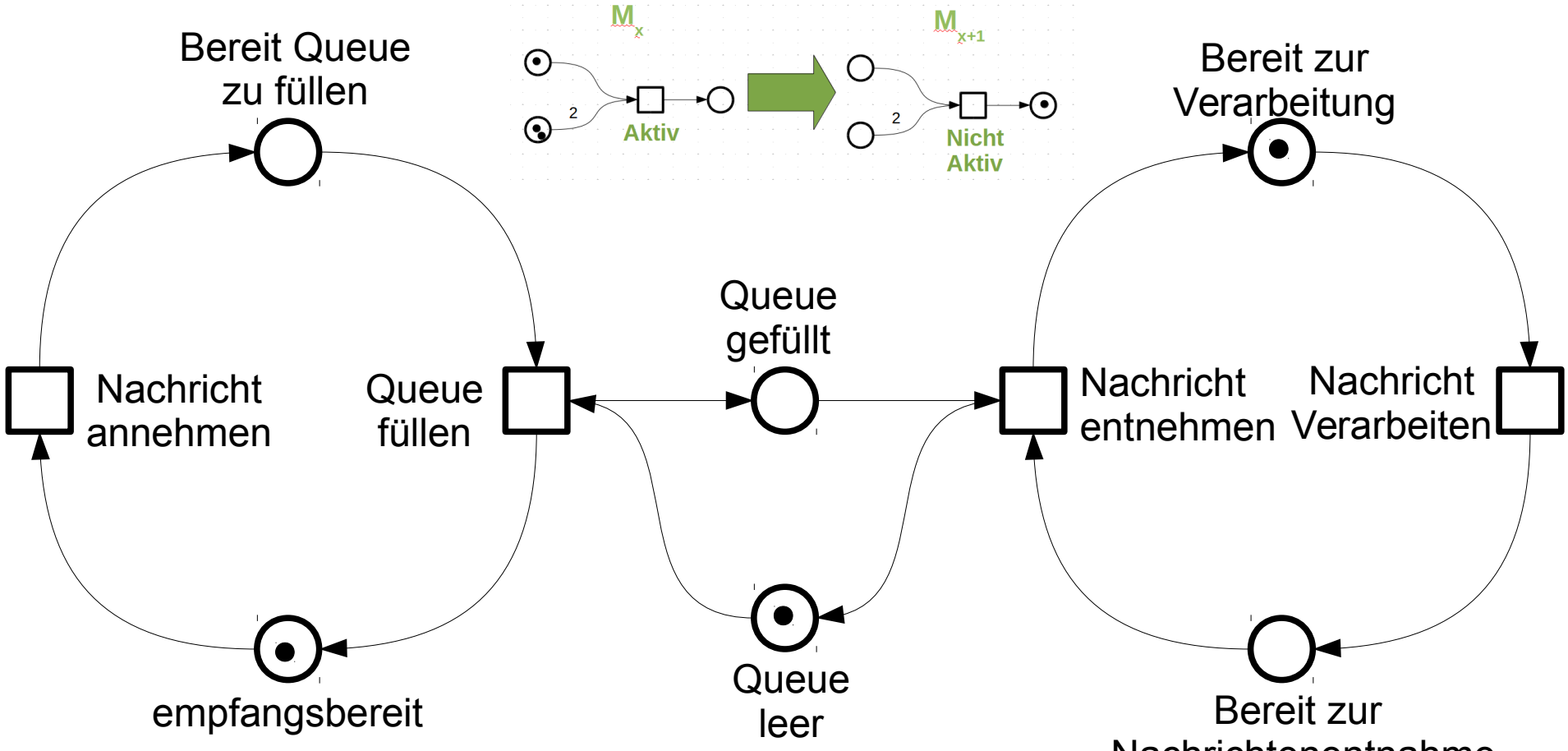


Ablauf: Beispiel (M_2)

Nächster Zustand ?



Ablauf: Beispiel (M_3)



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

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 ?

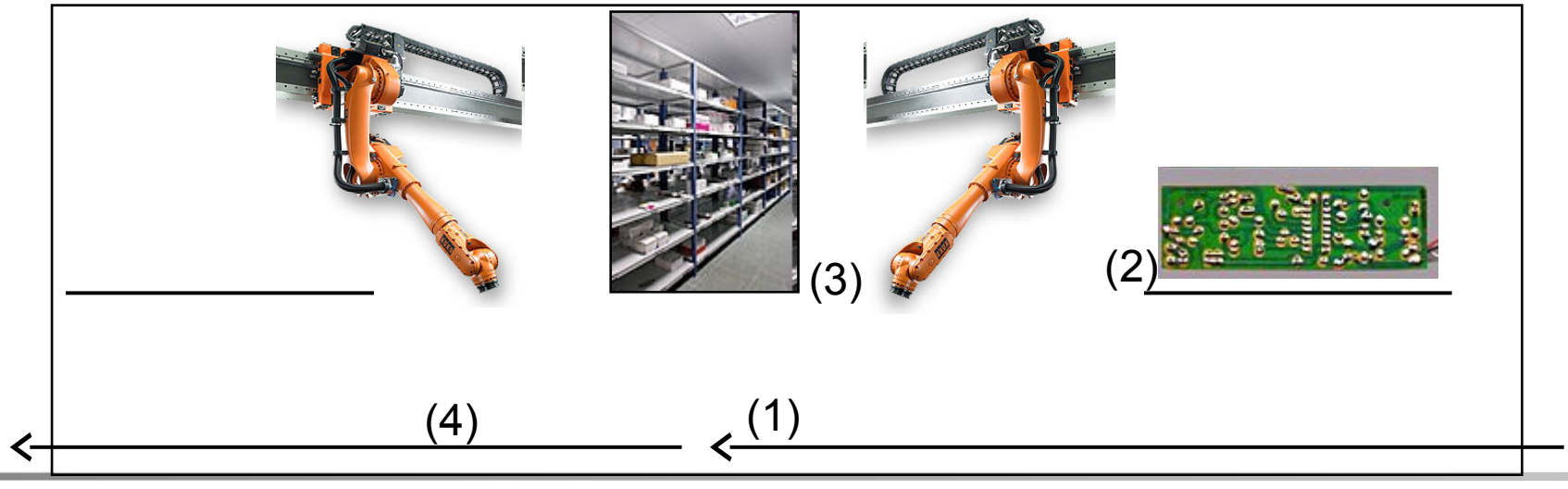
Antwort:

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

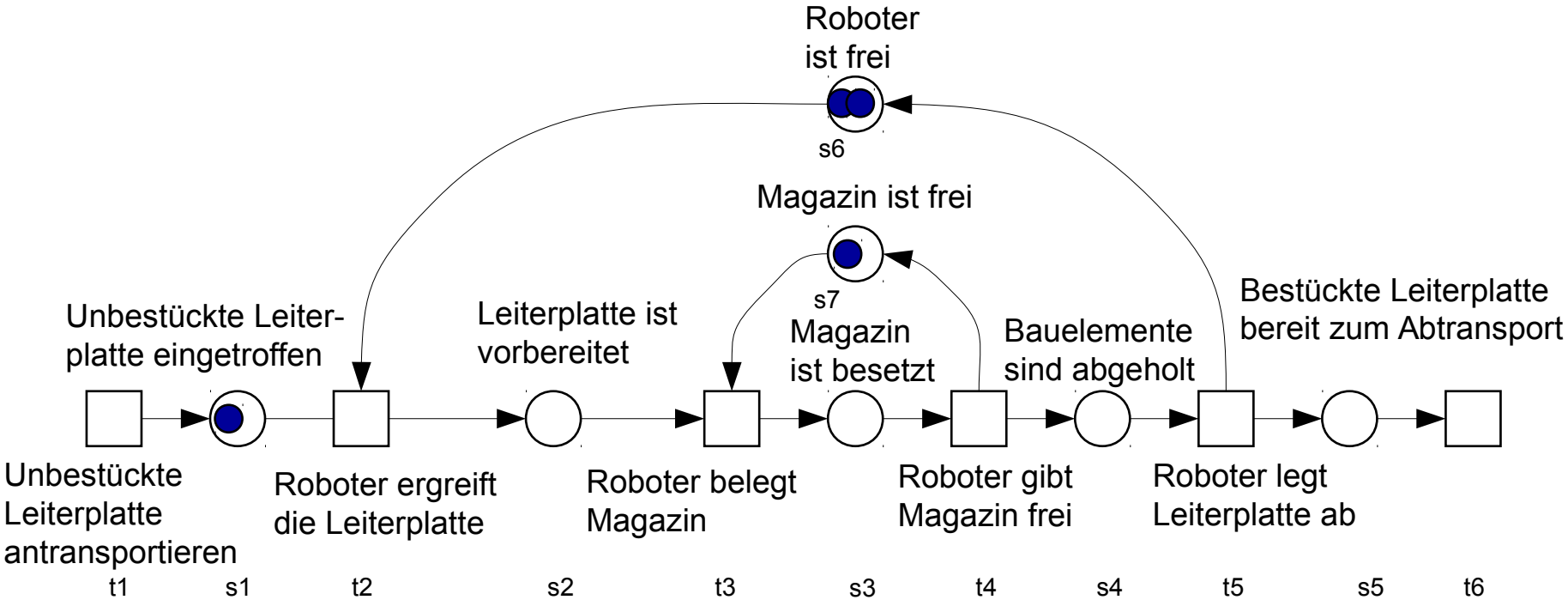
Beispiel: Bestückungsroboter

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).

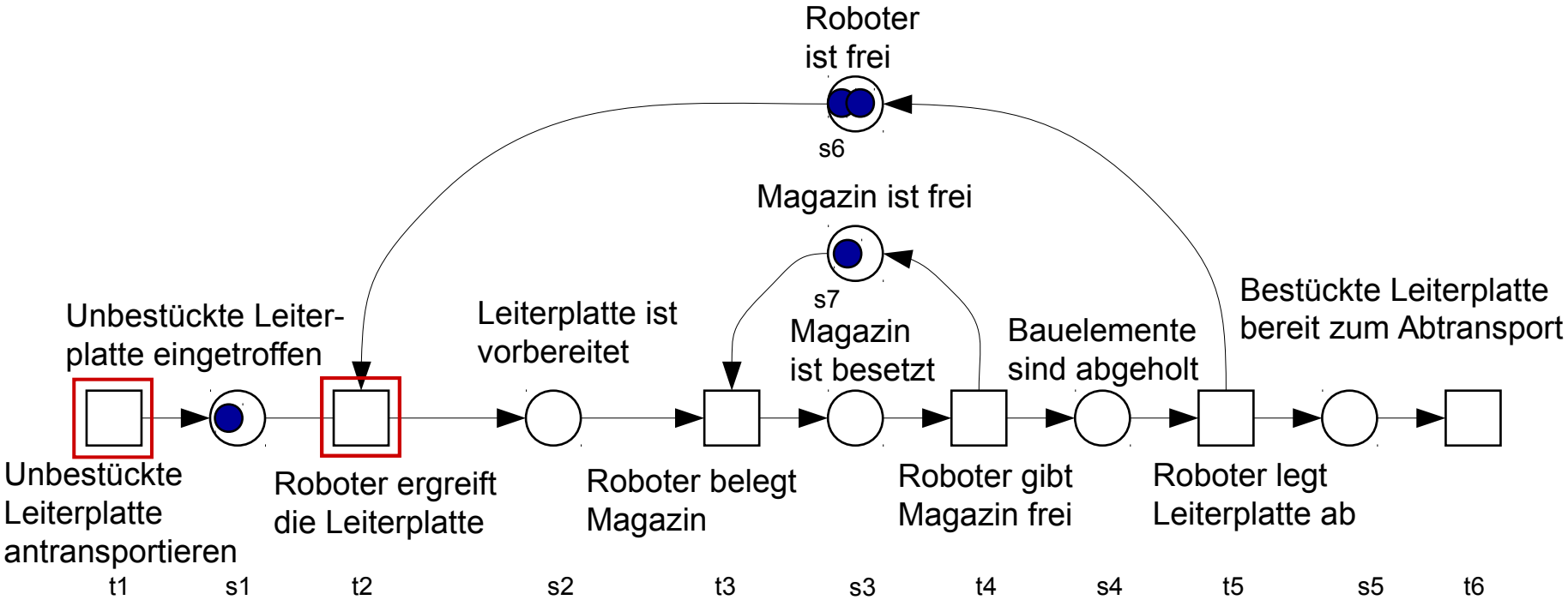


Beispiel: Bestückungsroboter (M0)



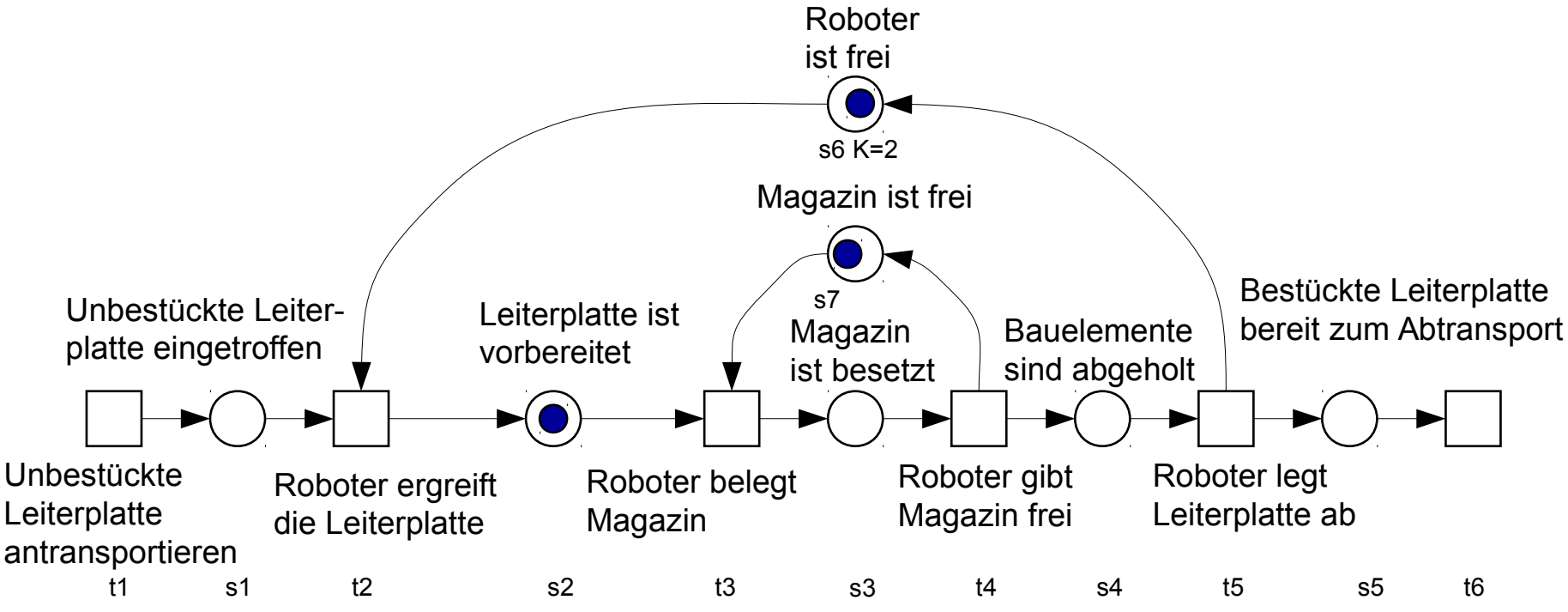
Welche Transition(en) aktiviert ?

Beispiel: Bestückungsroboter (M0)



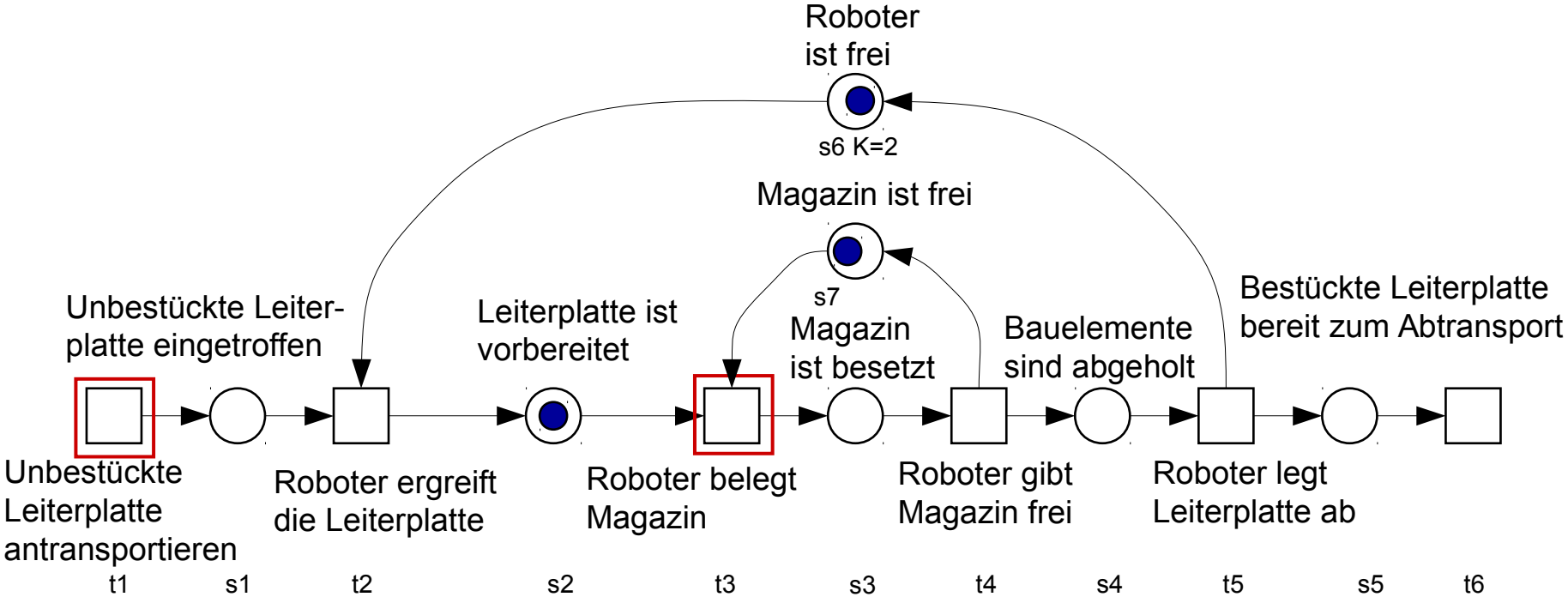
bezeichnet aktivierte Transitionen. Möglicher nächster Zustand ?

Beispiel: Bestückungsroboter (M1)



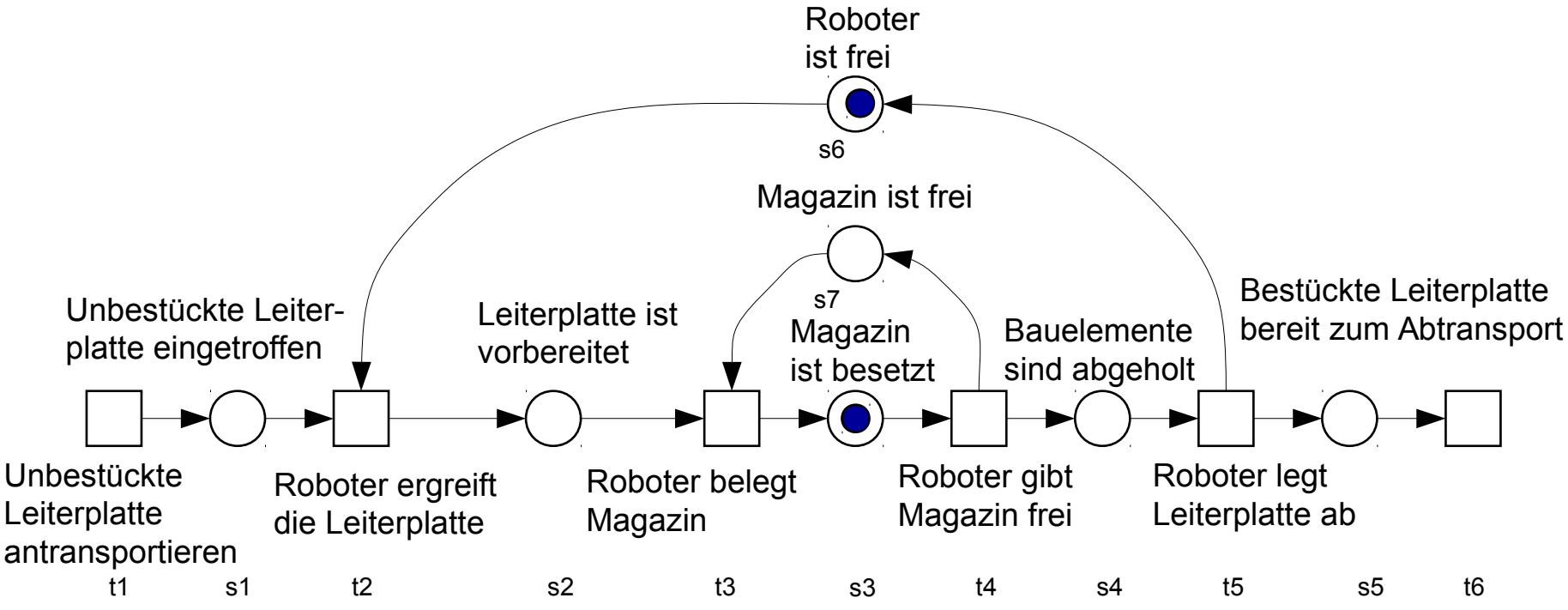
Welche Transition(en) aktiviert ?

Beispiel: Bestückungsroboter (M1)



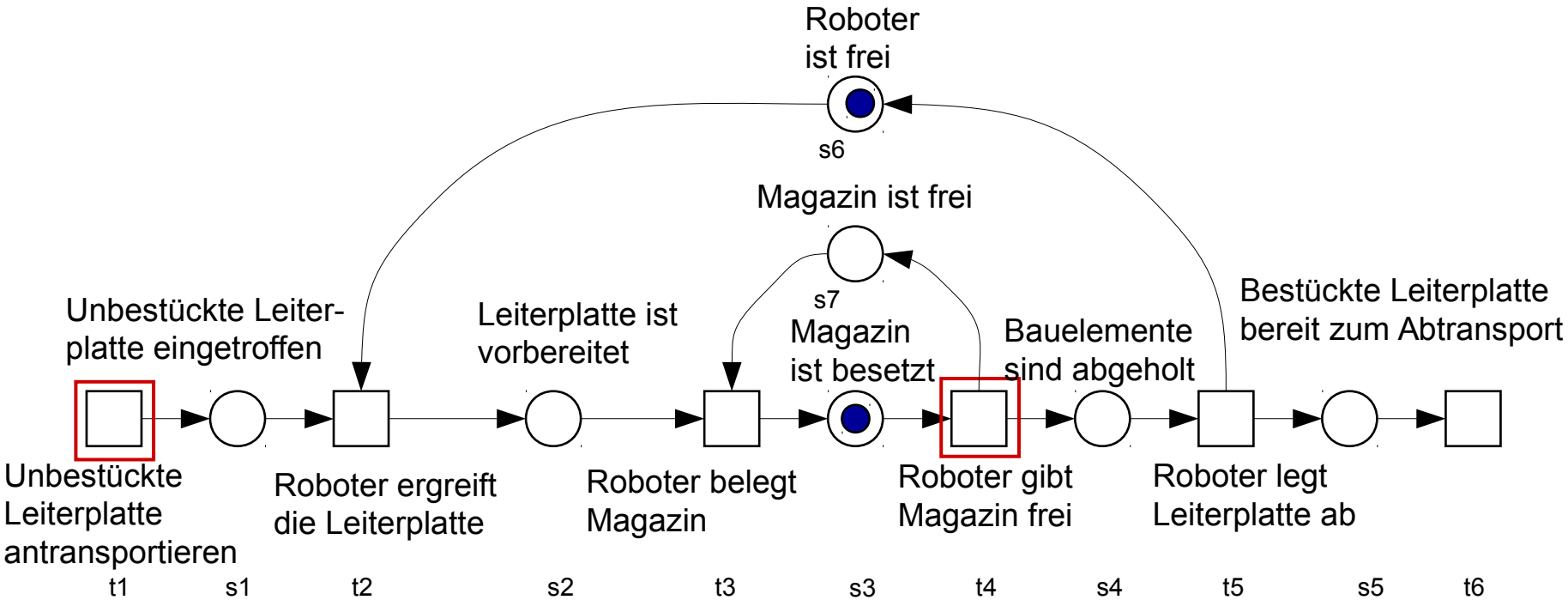
Möglicher nächster Zustand ?

Beispiel: Bestückungsroboter (M2)



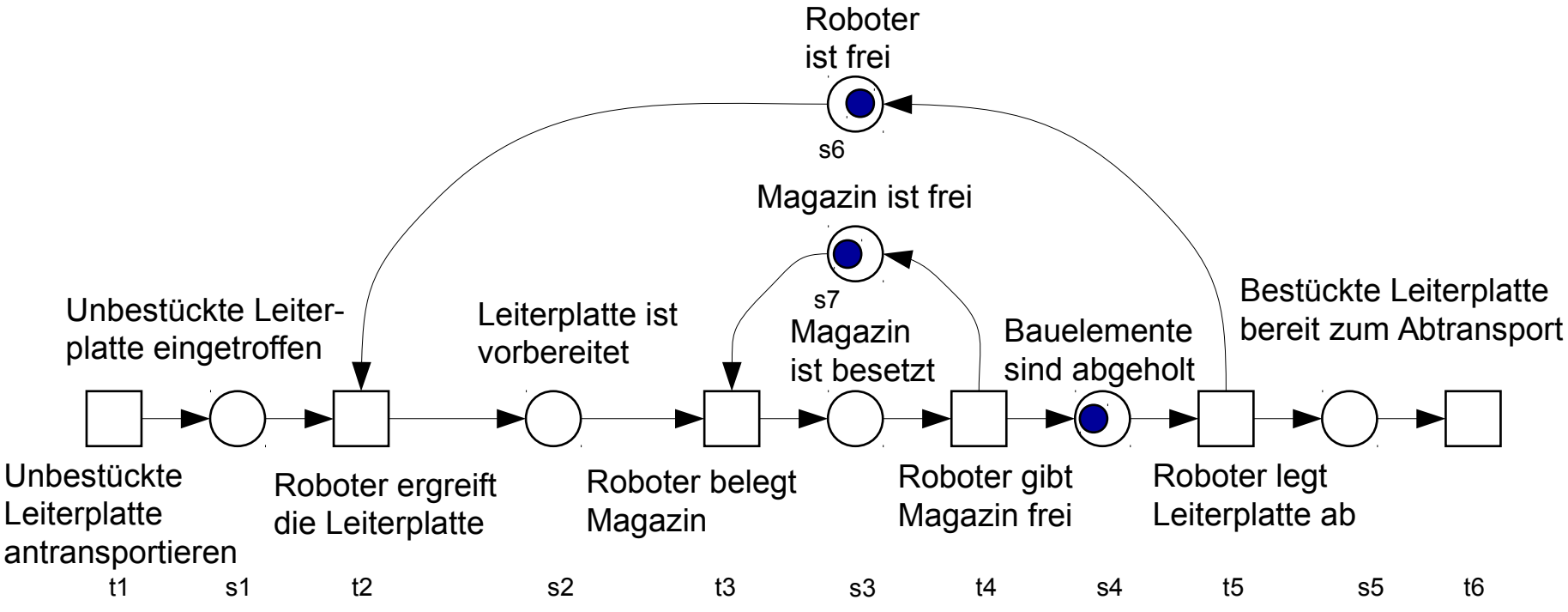
Welche Transition(en) aktiviert ?

Beispiel: Bestückungsroboter (M2)



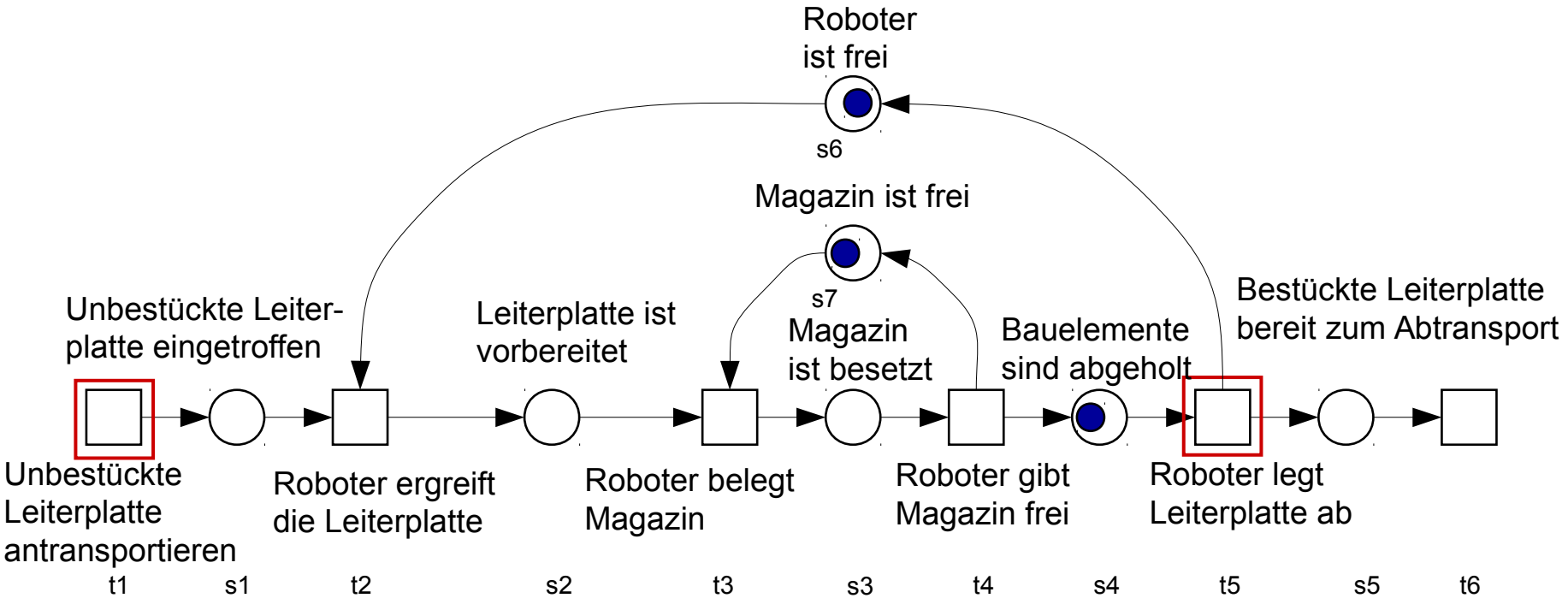
Möglicher nächster Zustand ?

Beispiel: Bestückungsroboter (M3)



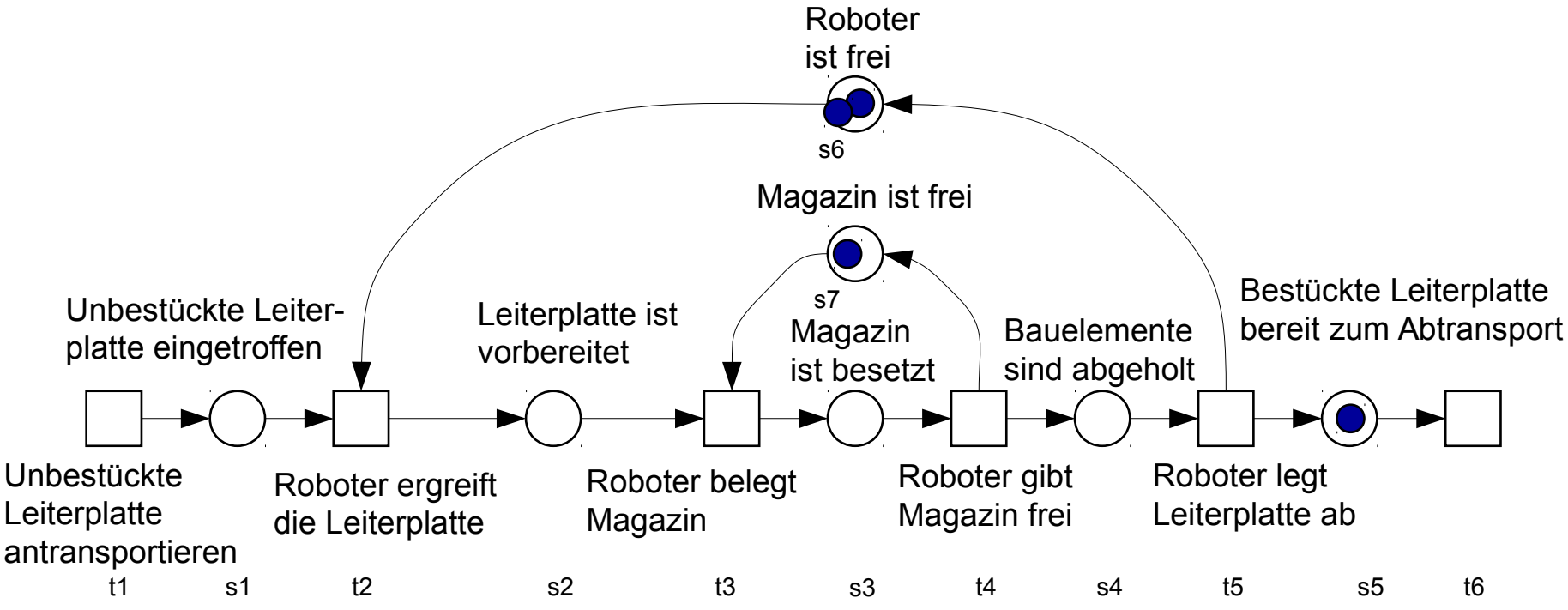
Welche Transition(en) aktiviert ?

Beispiel: Bestückungsroboter (M3)



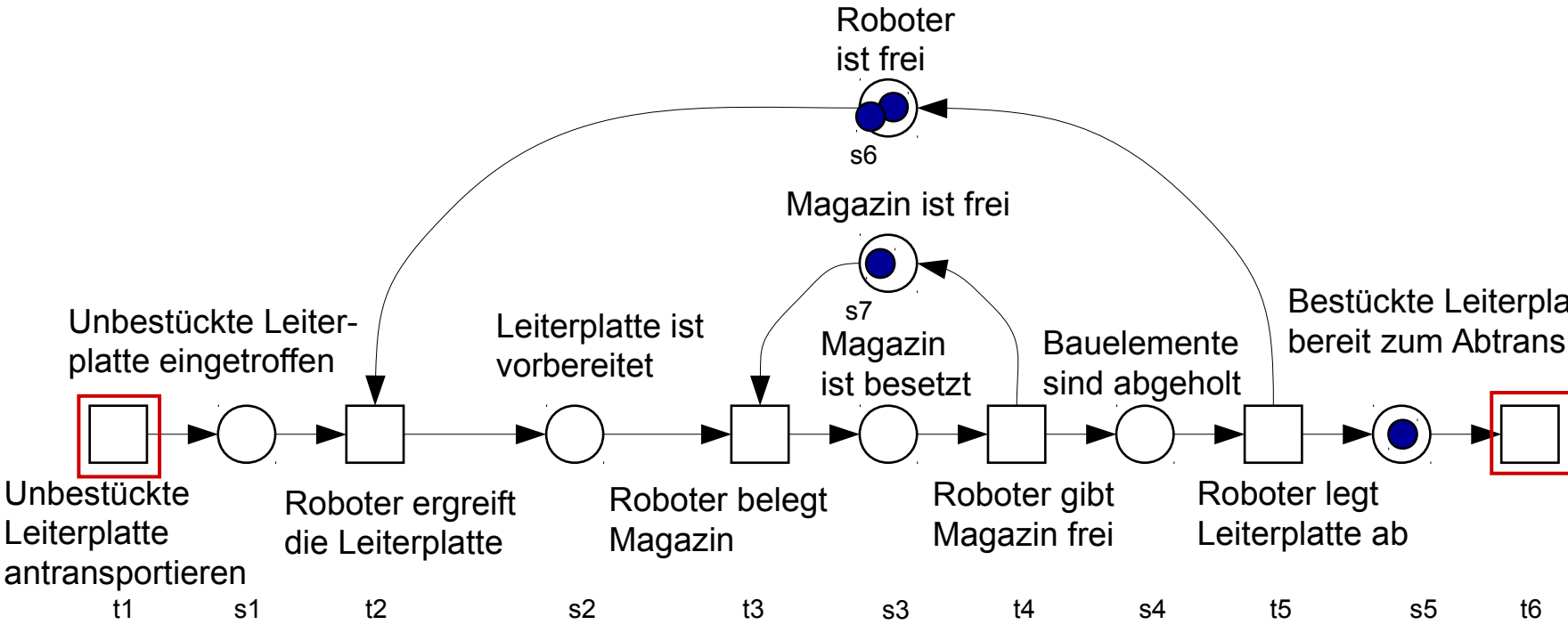
Möglicher nächster Zustand ?

Beispiel: Bestückungsroboter (M4)



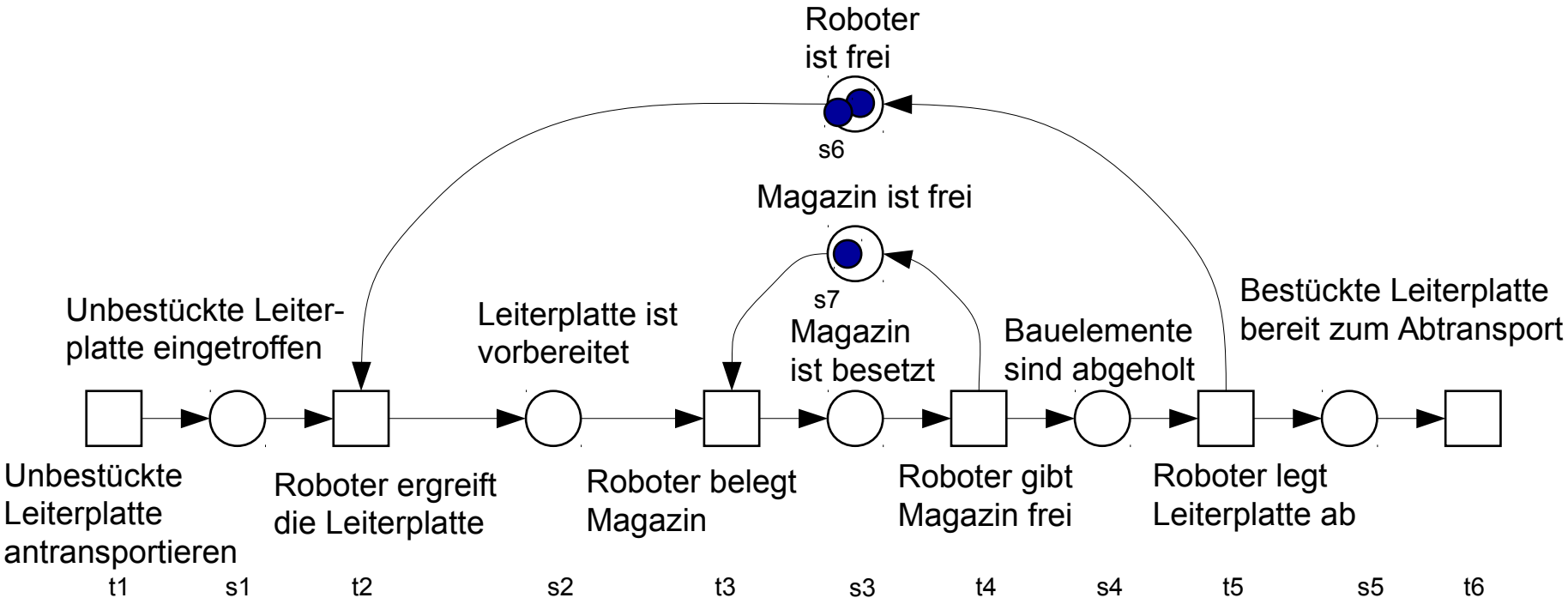
Welche Transition(en) aktiviert ?

Beispiel: Bestückungsroboter (M4)



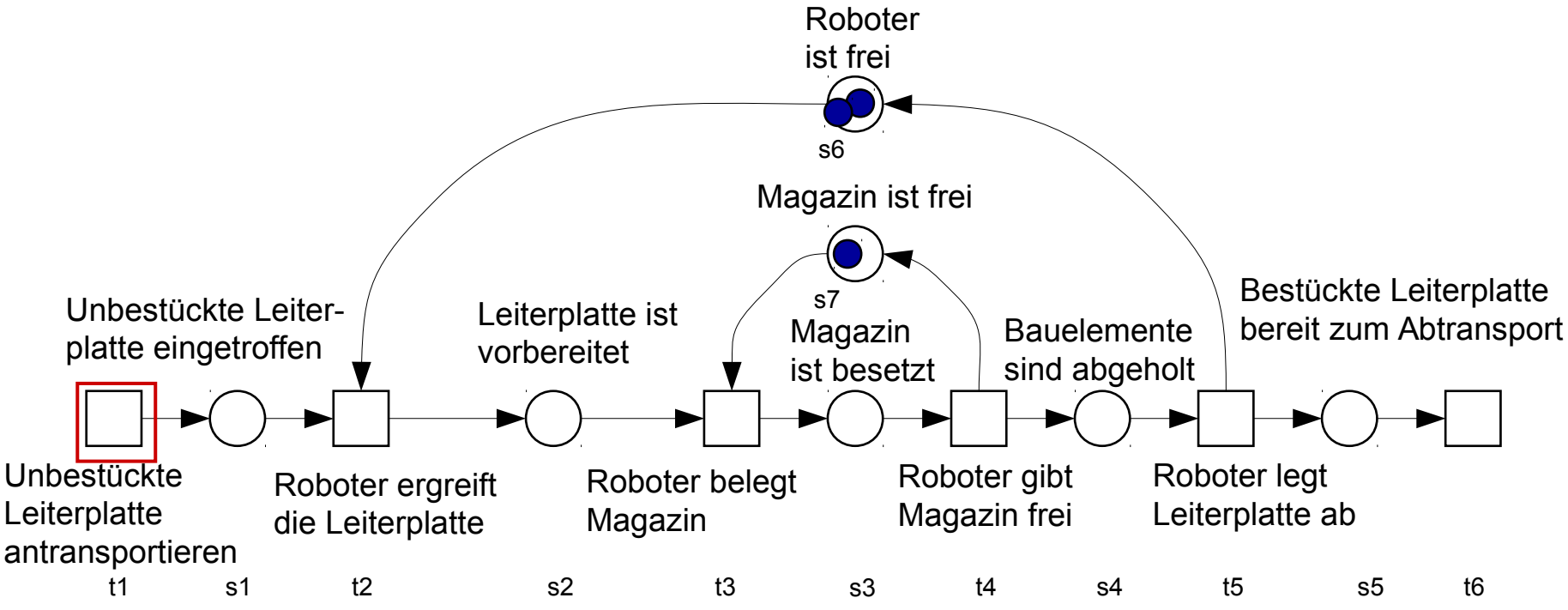
Möglicher nächster Zustand ?

Beispiel: Bestückungsroboter (M5)



Welche Transition(en) aktiviert ?

Beispiel: Bestückungsroboter (M5)



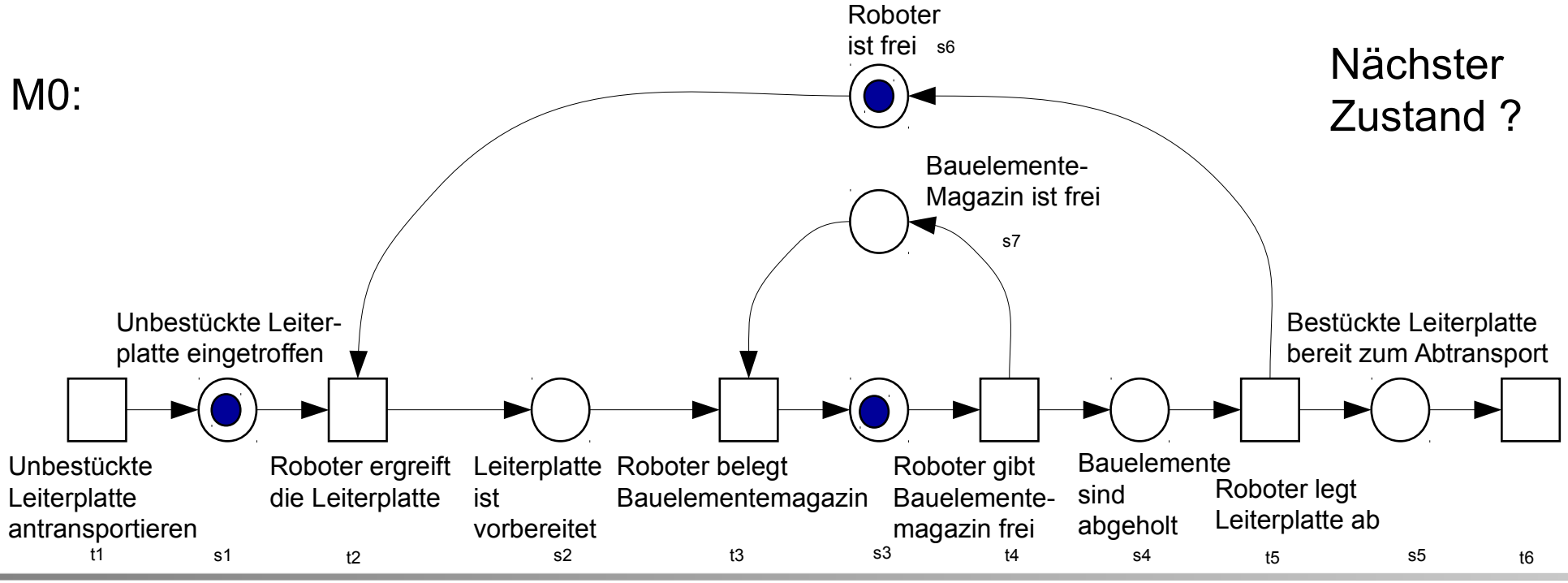
Usw. ...

Nichtdeterministische Auswahl von Transition: Beispiel

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:



Nächster Zustand ?

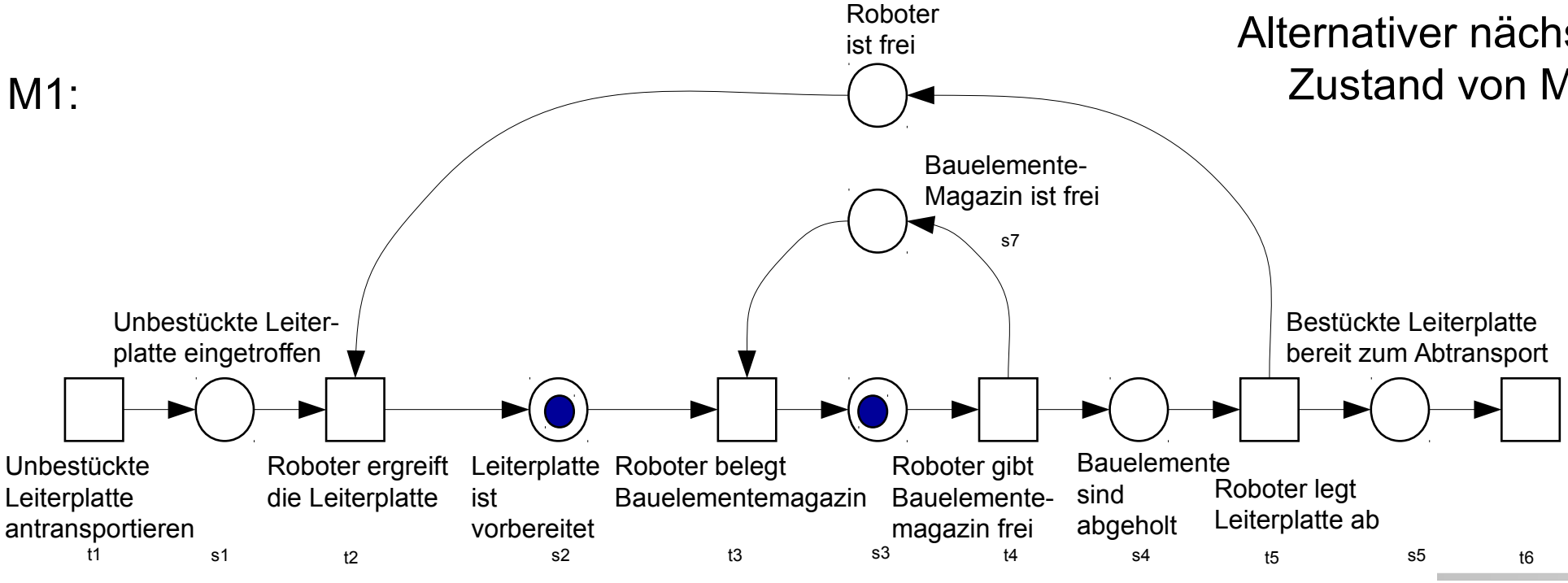
Bauelemente-Magazin ist besetzt

Nichtdeterministische Auswahl von Transition: Beispiel

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 ?

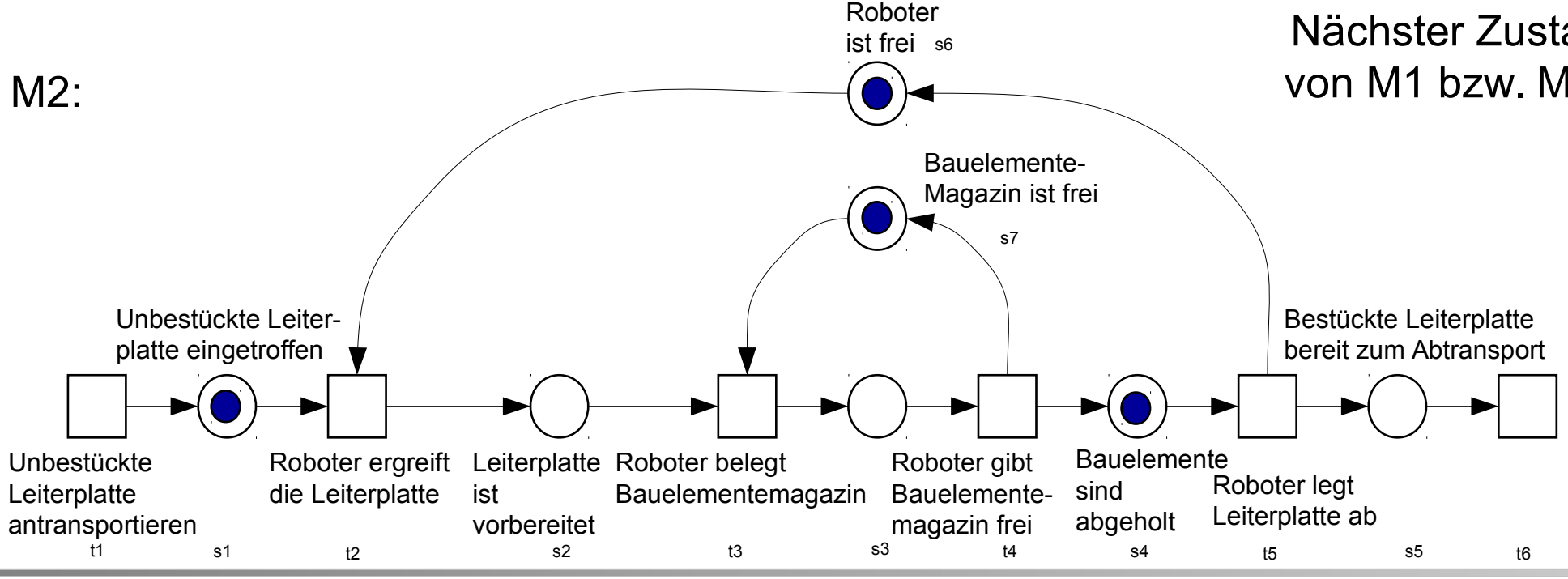


Bauelemente-Magazin ist besetzt

Nichtdeterministische Auswahl von Transition: Beispiel

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:



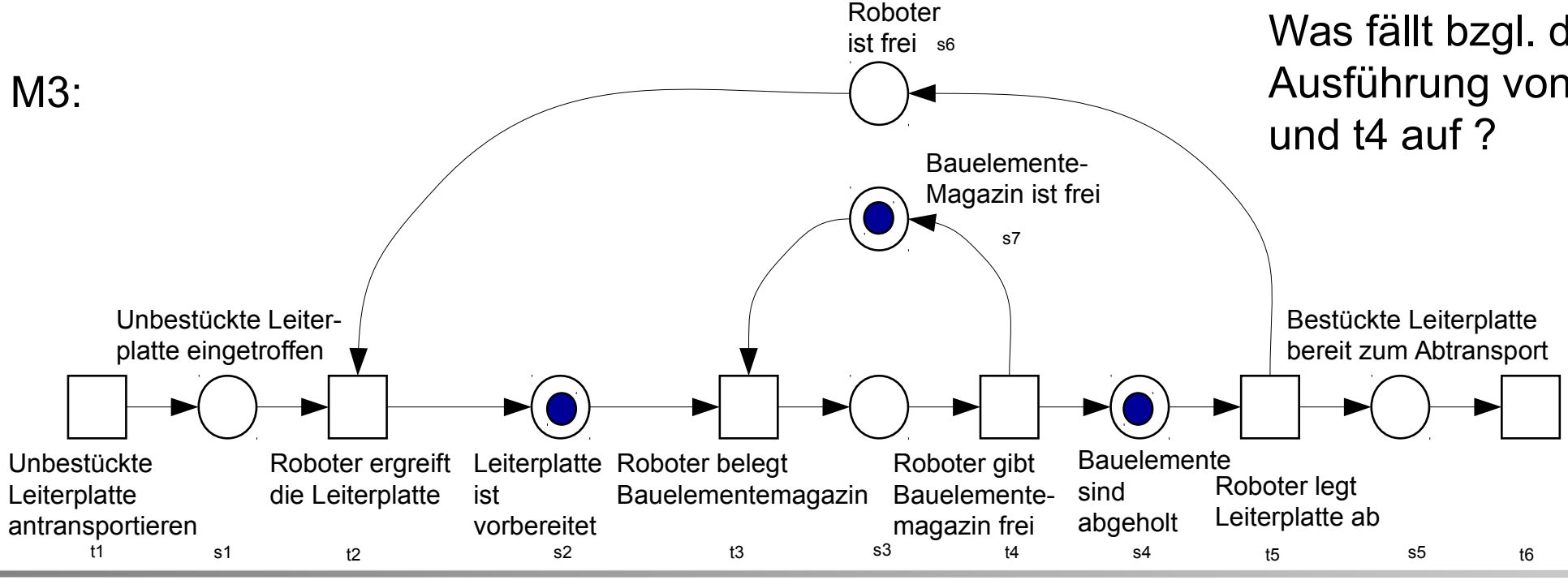
Nächster Zustand von M1 bzw. M2 ?

Bauelemente-Magazin ist besetzt

Nichtdeterministische Auswahl von Transition: Beispiel

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

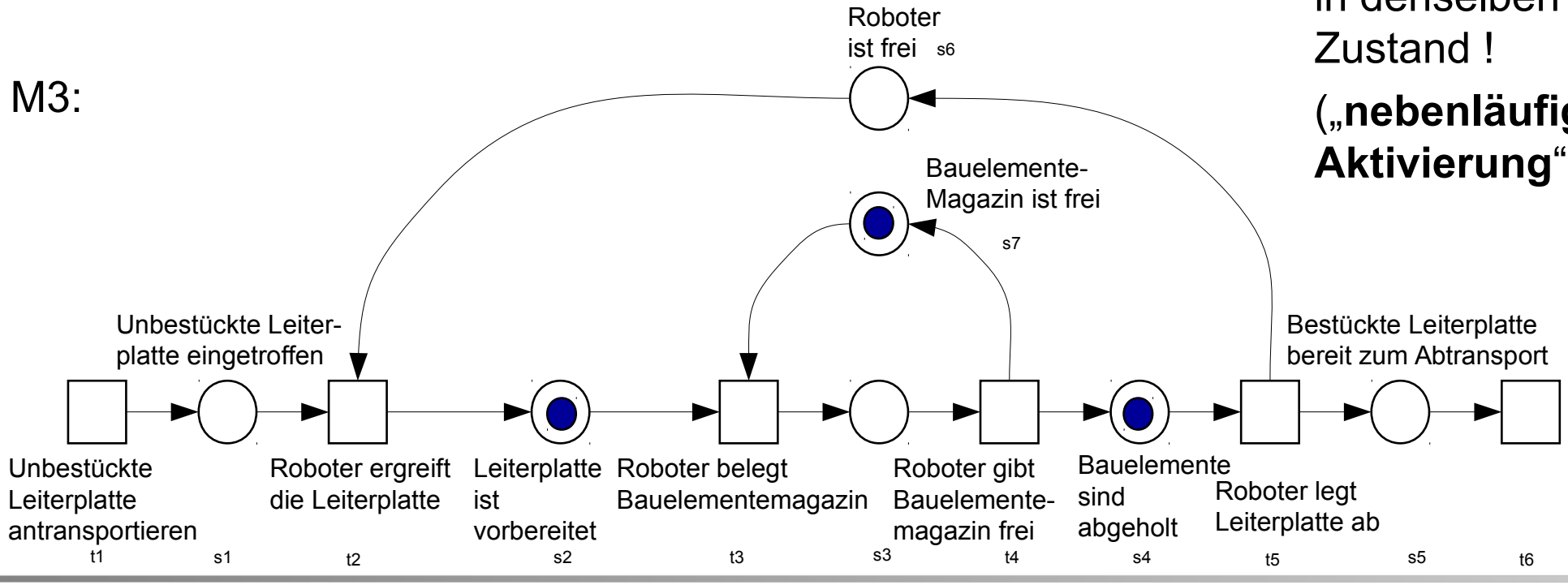
Nichtdeterministische Auswahl von Transition: Beispiel

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

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**")
- **$[M_0>$** := $\{M \mid \exists\, w \in T^*$ mit $M_0\,[w>\, M\}$ (**Erreichbarkeitsmenge** des Systems; die Markierungen $M \in [M_0>$ heißen **erreichbar**)

Erreichbarkeitsalgorithmus (breadth-first; vgl. obiges Beispiel)

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

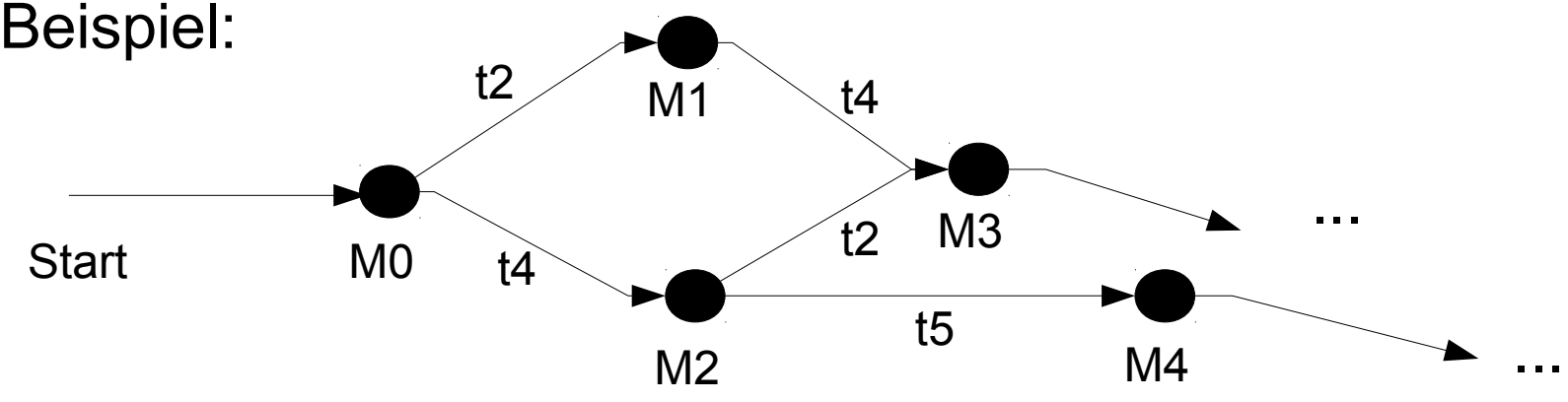
1. Trage in ein Schema mit Spalten „Markierungsnummer“, „Markierung“ und „Schaltungen“ **Anfangsmarkierung M_0** ein.
2. In **aktueller Markierung M_i** für jede **Transition t**: aktiviert ?
   - Falls **t** aktiviert: Berechne Folgemarkierung.
     - Folgemarkierung bereits eine **Markierung M_j** ?
     - 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.

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:

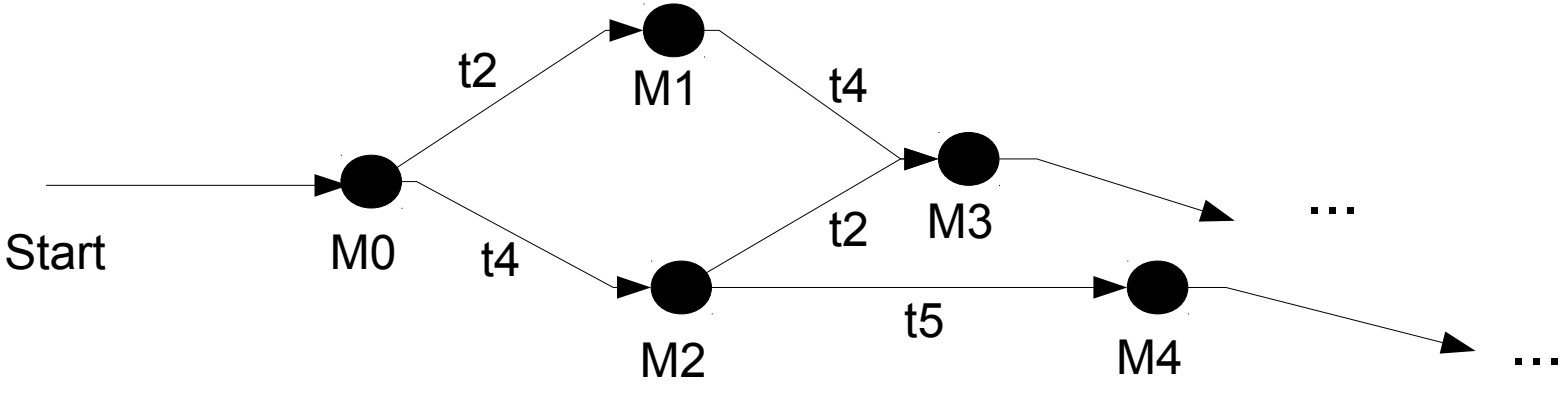
- Knoten: Zustände (linke Spalte; ggf. inkl. Markierungsbelegungen)
- Kanten: Schaltungen (rechte Spalte)

Obiges Beispiel:



Von Erreichbarkeitstabelle zu Event-Log

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

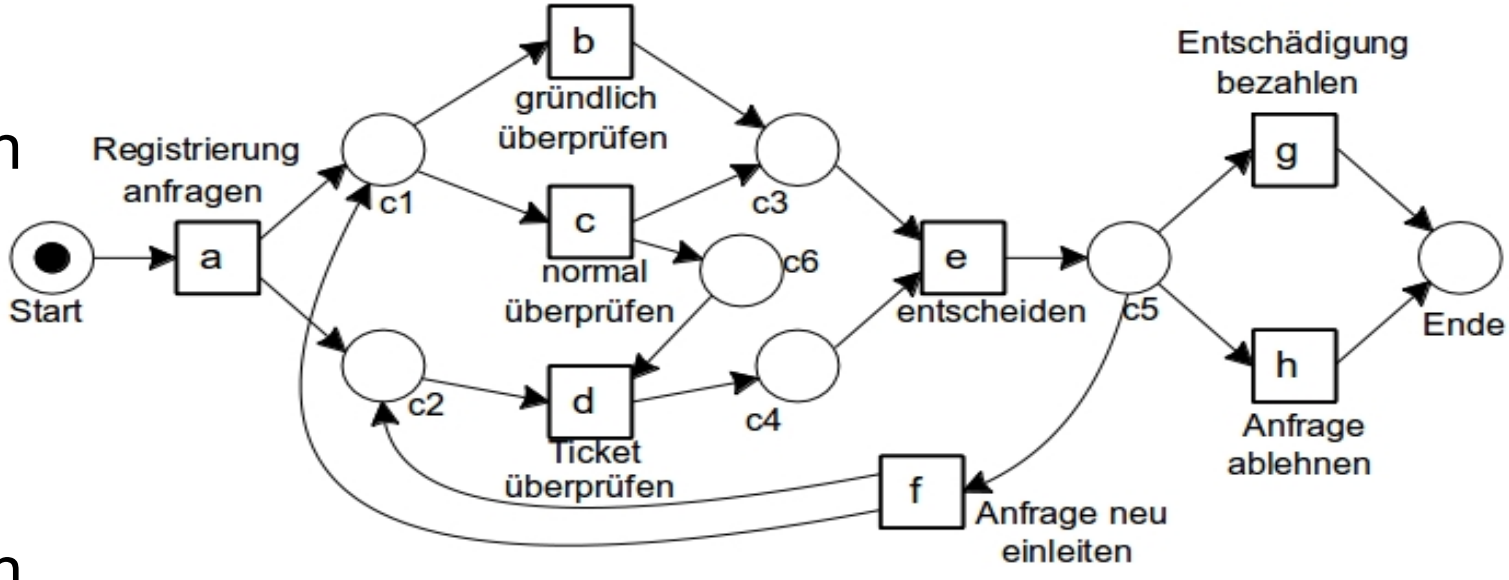


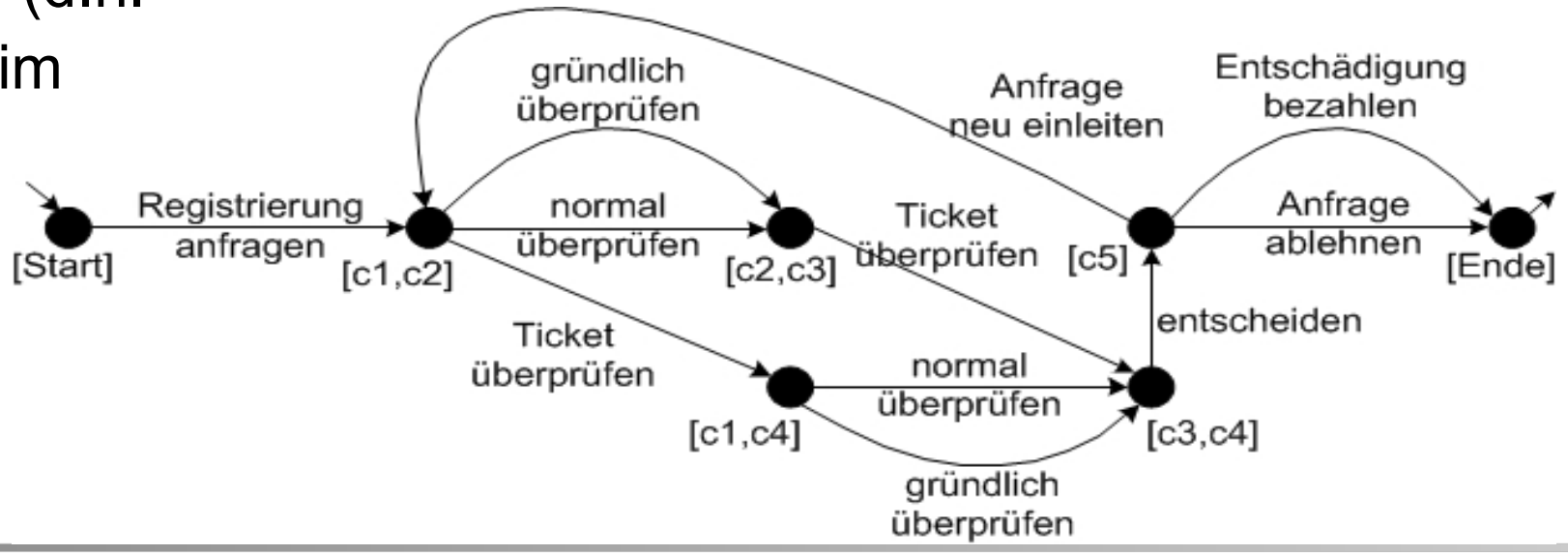
Erzeugtes Event-Log
(= Menge der Folgen der ausgeführten Transitionen):

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

Erreichbarkeitsgraph: Weiteres Beispiel

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).

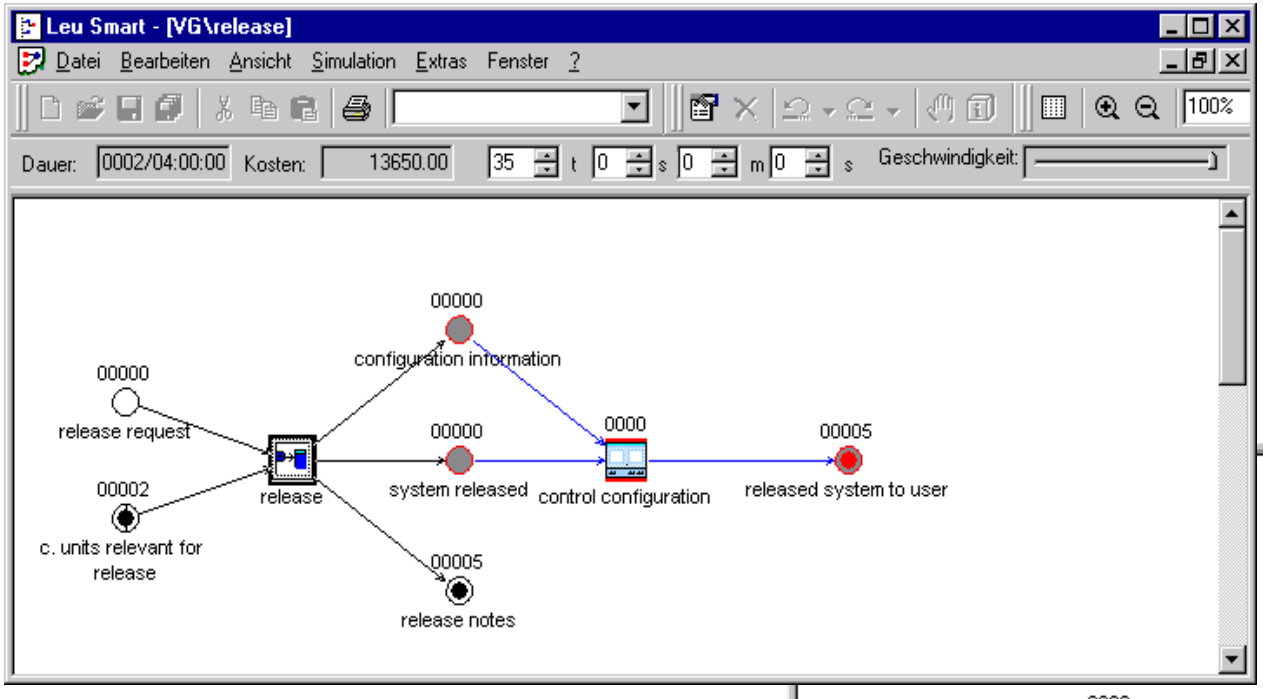


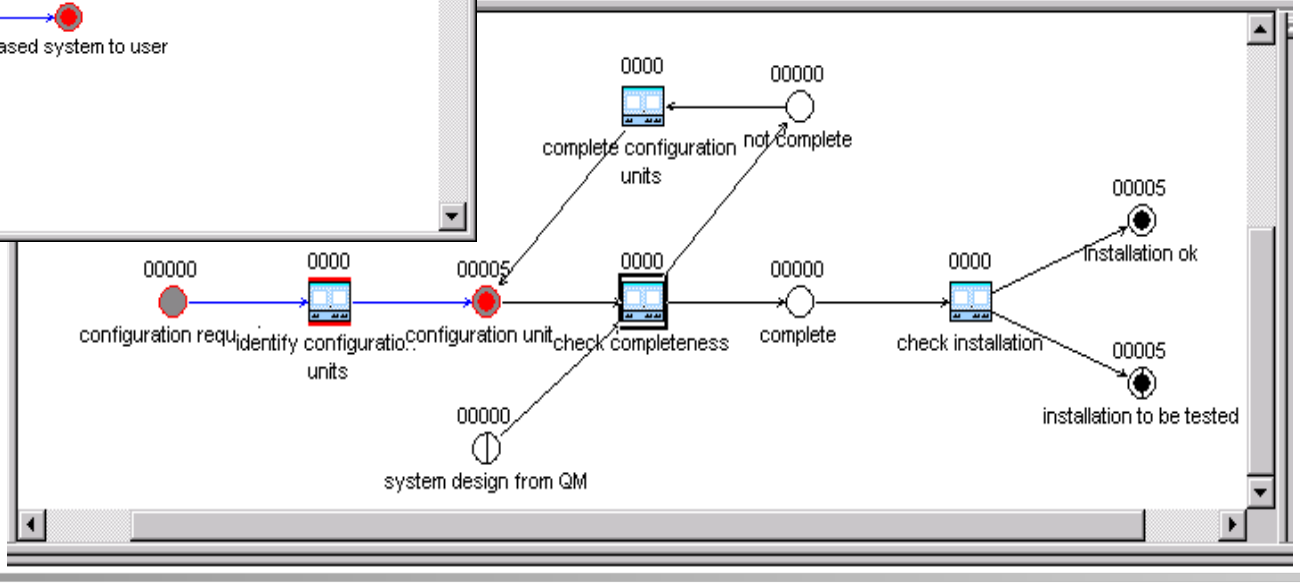


Überblick Petrinetze

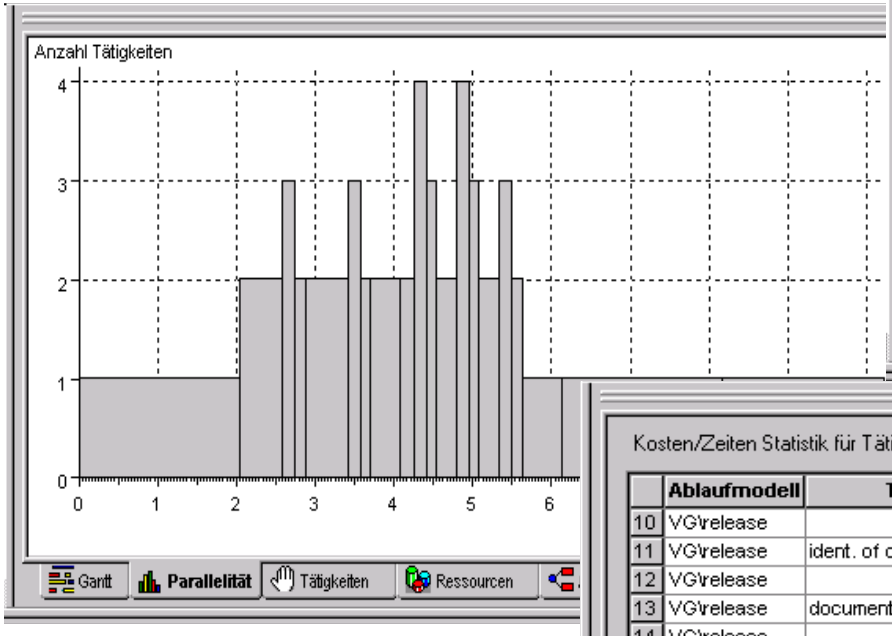
- Petrinetz-Syntax
- Ausführung
- Analyse von Systemen
- Workflow-Netze

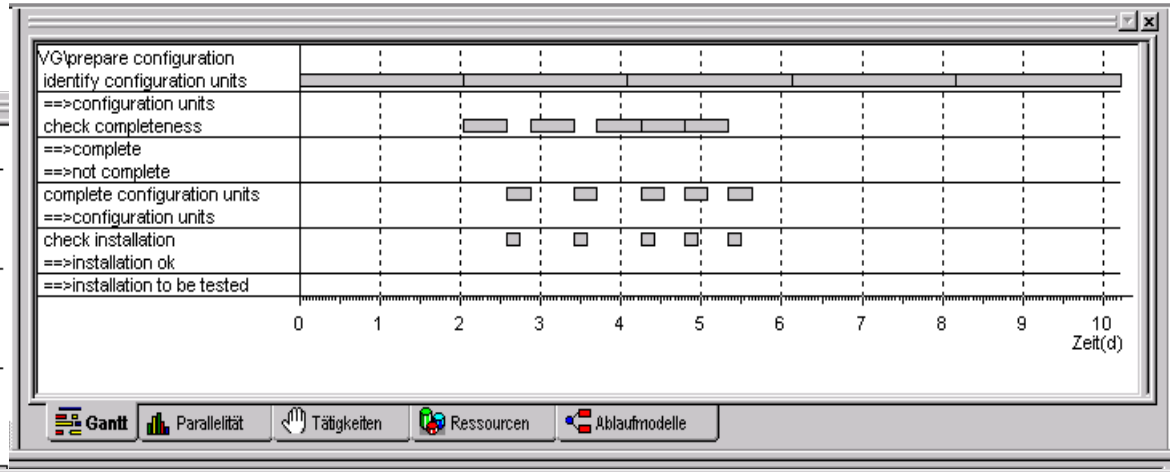
Simulation von Petrinetzen: Animation

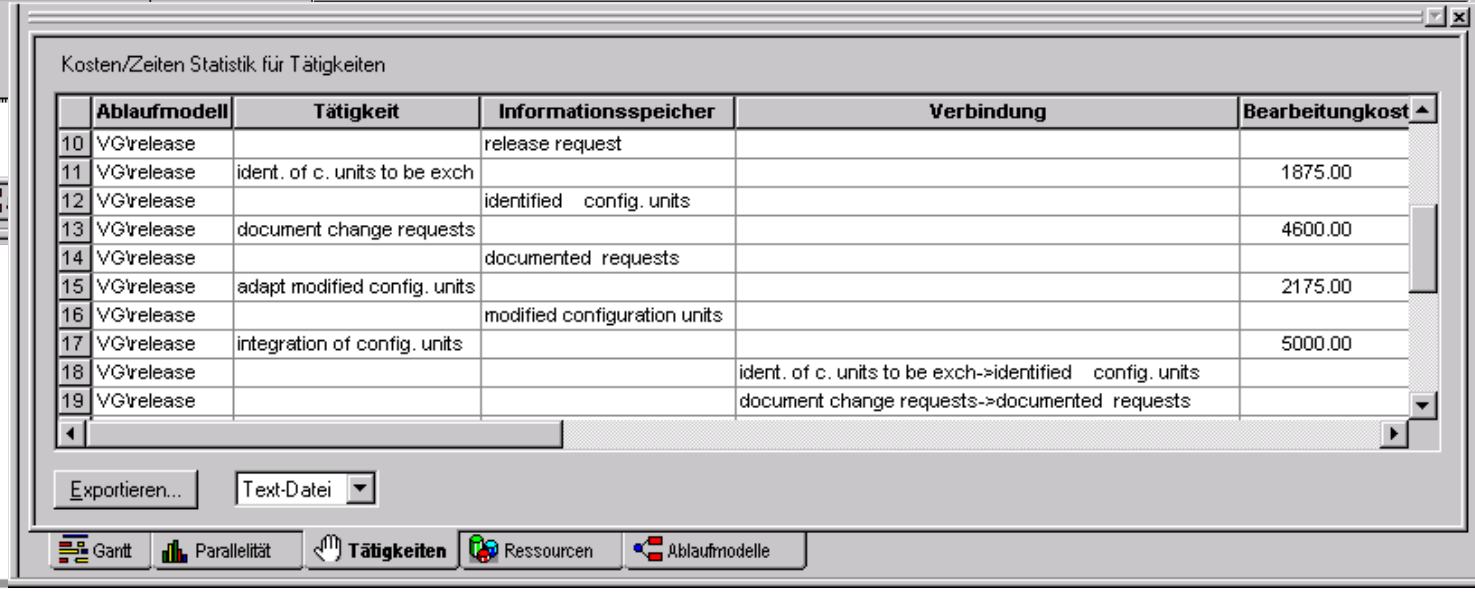




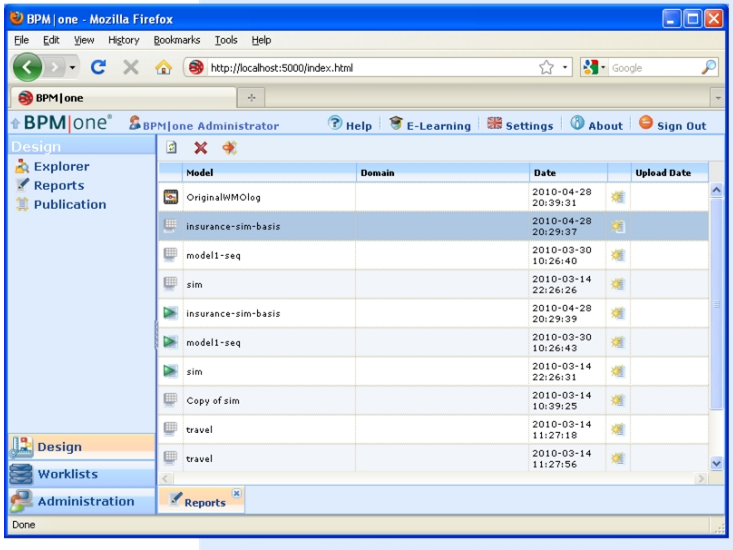
Simulation von Petrinetzen: Analyse von Simulationsläufen

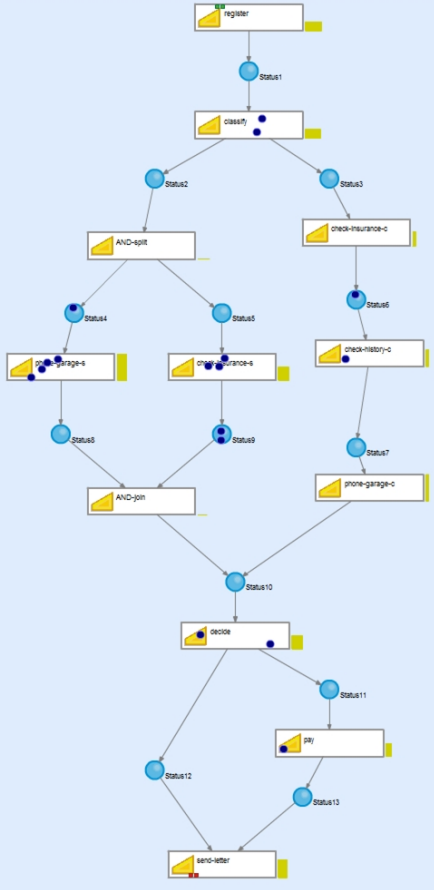






Performanzanalyse, z.B.: Simulation in BPM|one





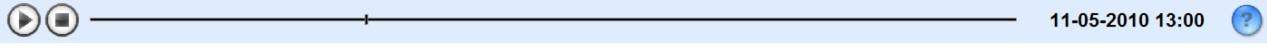
Microsoft Excel - insurance-sim-basis.xls

	Roles					
	Utilisation rate					
	Mean	Lower 90%	Upper 90%	Lower 99%	Upper 99%	
<<No role>>	0	0	0	0	0	
employee	0,818765572	0,712002743	0,9255284	0,629159456	1,008371688	
Claim handler	0,477614884	0,381832289	0,573397479	0,307509182	0,647720586	
Claim handler A	0,67905118	0,596904828	0,761197533	0,53316285	0,824939511	

	Activities									
	Queue time					Work time				
	Mean	Lower 90%	Upper 90%	Lower 99%	Upper 99%	Mean	Lower 90%	Upper 90%	Lower 99%	Upper 99
register	5,683619566	0,750153688	10,61708544	-3,078000163	14,44523929	10,12339217	9,696126225	10,55065811	9,364586531	10,88219
phone-garage-s	2,888949042	0,587130287	5,190767798	-1,198980442	6,976878526	29,95317042	29,69568701	30,21065383	29,49589115	30,41044
check-insurance-s	1,741425234	0,724631759	2,758218708	-0,064355527	3,547205994	14,85471567	14,58402909	15,12540225	14,37398814	15,3354
check-history-c	1,027479575	-0,591329722	2,646288871	-1,847454968	3,902414118	19,80467549	19,20910654	20,40024445	18,74697106	20,86237
AND-join	0	0	0	0	0	0	0	0	0	
phone-garage-c	0,758414809	-0,021672562	1,53850218	-0,626986261	2,143815878	29,97848491	29,60740688	30,34956294	29,31946655	30,63750
decide	1,687003061	1,046399418	2,327606703	0,549318996	2,824687125	15,24293539	15,04371202	15,44215877	14,8891234	15,59674
classify	1,570226096	0,667263861	2,473188331	-0,033395359	3,173847551	9,87474254	9,696300158	10,05318492	9,55783667	10,19164
pay	2,146984462	0,656782494	3,63718643	-0,499549088	4,793518011	15,17529939	14,73446303	15,61613575	14,39239329	15,95820
send-letter	1,150656282	-0,026173015	2,327485578	-0,939341103	3,240653667	20,25803097	20,02110823	20,4949537	19,83726655	20,67879
AND-split	0	0	0	0	0	0	0	0	0	
check-insurance-c	0,877120927	0,221961283	1,532280572	-0,286413961	2,040655815	14,84696658	14,12850766	15,56542549	13,57101495	16,1229

	Status									
	Wait time					Wait + queue time				
	Mean	Lower 90%	Upper 90%	Lower 99%	Upper 99%	Mean	Lower 90%	Upper 90%	Lower 99%	Upper 99
Status1	0	0	0	0	0	1,570226096	0,667263861	2,473188331	-0,033395359	3,173847
Status8	0,013955404	-0,011582986	0,039493794	-0,03139966	0,059310468	0,013955404	-0,011582986	0,039493794	-0,03139966	0,059310
Status9	15,76128093	13,74211032	17,78045154	12,17532219	19,34723967	15,76128093	13,74211032	17,78045154	12,17532219	19,34723
Status7	0	0	0	0	0	0,758414809	-0,021672562	1,53850218	-0,626986261	2,143815
Status10	0	0	0	0	0	1,687003061	1,046399418	2,327606703	0,549318996	2,824687
Status11	0	0	0	0	0	2,146984462	0,656782494	3,63718643	-0,499549088	4,793518
Status12	0	0	0	0	0	1,377037517	-0,240102843	2,994177877	-1,494933068	4,249008
Status13	0	0	0	0	0	0,923838331	0,081404857	1,766271806	-0,572286692	2,419963
Status2	0	0	0	0	0	0	0	0	0	
Status3	0	0	0	0	0	0,877120927	0,221961283	1,532280572	-0,286413961	2,040655
Status4	0	0	0	0	0	2,888949042	0,587130287	5,190767798	-1,198980442	6,976878
Status5	0	0	0	0	0	1,741425234	0,724631759	2,758218708	-0,064355527	3,547205
Status6	0	0	0	0	0	1,027479575	-0,591329722	2,646288871	-1,847454968	3,902414

	Total									
	Lead time					Work time				
Mean	Lower 90%	Upper 90%	Lower 99%	Upper 99%	Mean	Lower 90%	Upper 90%	Lower 99%	Upper 99%	Mean
115,6947846	105,258757	126,1308123	97,16085576	134,2287135	113,4428918	110,4228233	116,4629603	108,0793822	118,8064014	



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**.

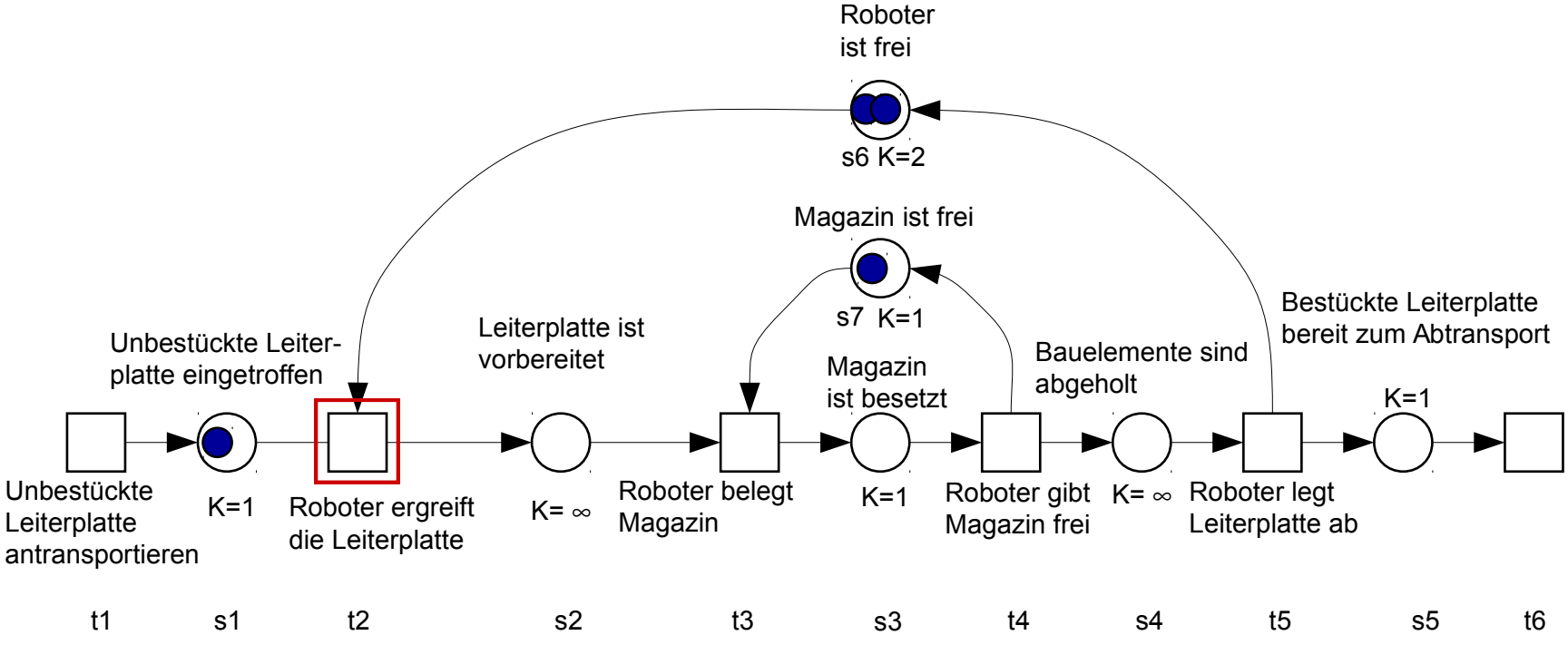
Verifikation: Beweis von Eigenschaften:

- **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.)

Kapazitäten

- 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.**

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.

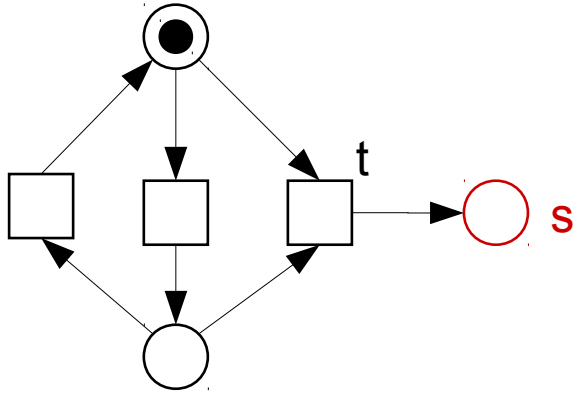
Stelle s heisst **b-sicher**, wenn P B-sicher mit B(s)=b, und B(s‘)= ∞ für s'#s.

Unterschied zwischen Kapazität und Sicherheit:

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

Sicherheit: Beispiele

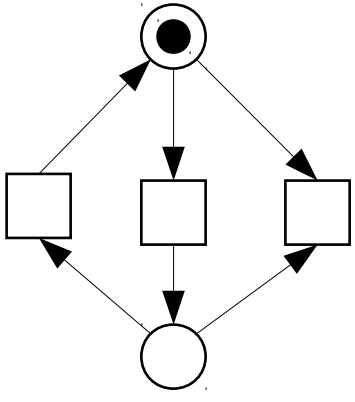
- *Beispiel:* Verkehrsplaner modelliert **Ampelsystem**.
  An bestimmter Stelle s darf sich nur ein Auto aufhalten.
  Für K(s)=1 keine Aussage über Korrektheit der Modellierung.
  Für K(s)=∞ in der Analyse prüfbar, ob B(s)=1.
- *Beispiel:* Transition t soll nie schalten dürfen.
  Durch Hinzufügen einer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



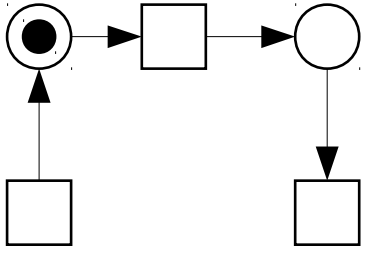
Transition t eines Petrinetz $P=(N,M_0)$ heißt:

- **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 … ?

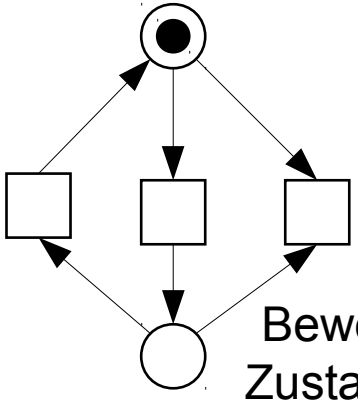


Transition
ist … ?

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

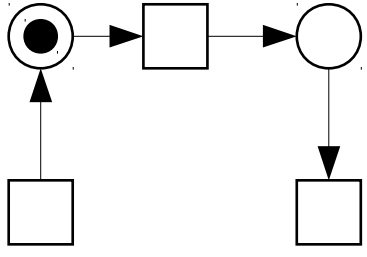
- **aktivierbar**: In mindestens einer erreichbaren Markierung aktiviert:
  existiert $M_1 \in [M_0>$ mit: $M_1[t>$
- **lebendig**: In allen erreichbaren Markierung **aktivierbar**:
  für alle $M_1 \in [M_0>$ gilt: existiert $M_2 \in [M_1>$ mit: $M_2[t>$
- **tot:** In keiner erreichbaren Markierung aktiviert: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

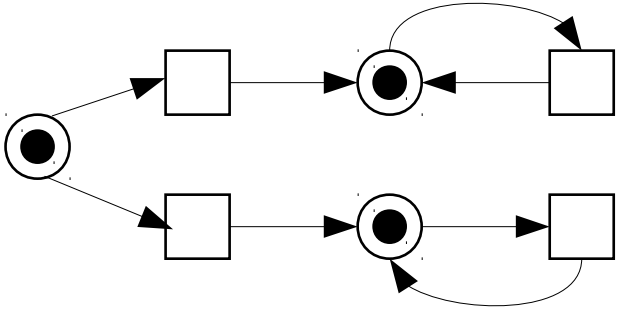


Lebendige Transition

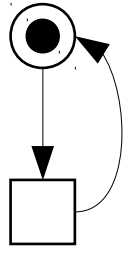
Petrinetz $P=(S,T,F,K,W,M_0)$ heißt:

- **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:
  $\forall t \in T: \neg M_0[t>$

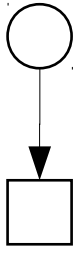
System ist … ?



System ist … ?



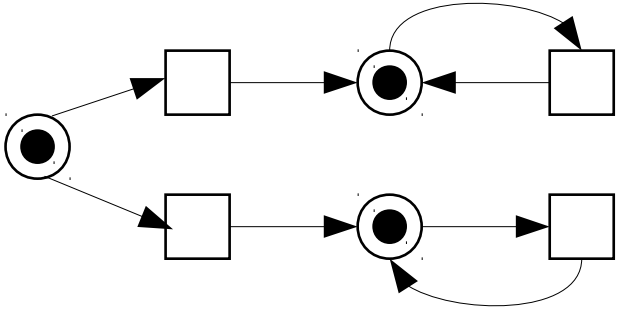
System ist … ?



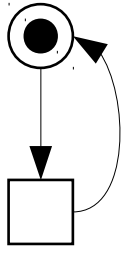
Petrinetz $P=(S,T,F,K,W,M_0)$ heißt:

- **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:
  $\forall t \in T: \neg M_0[t>$

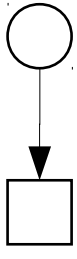
Deadlockfreies System



Lebendiges System



Totes System



Lebendigkeit von Petrinetzen: Anmerkungen

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“).

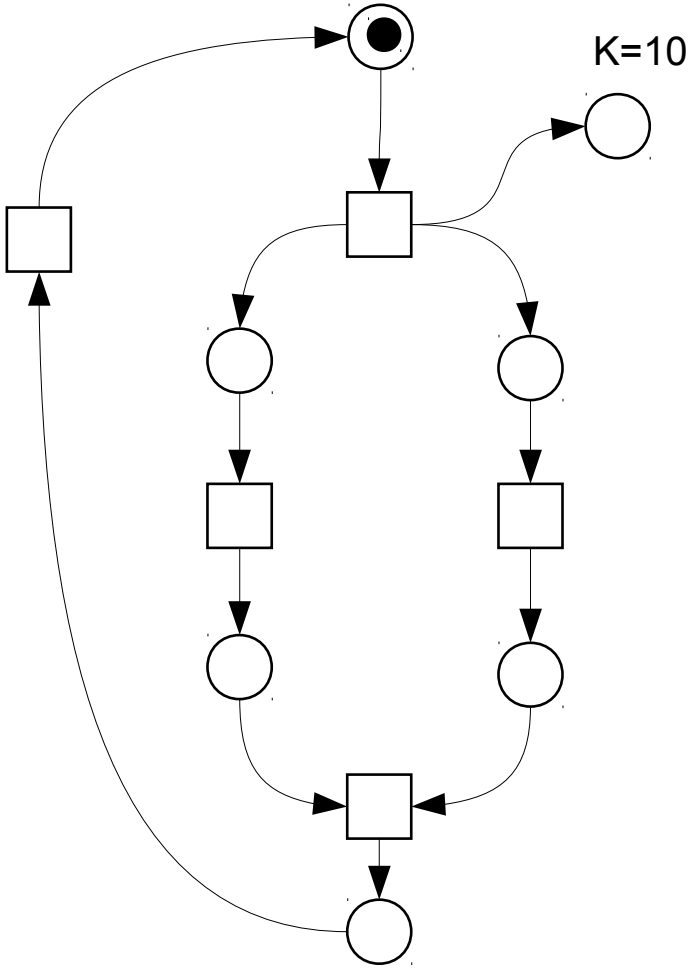
lebendig: alle Transitionen lebendig.

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

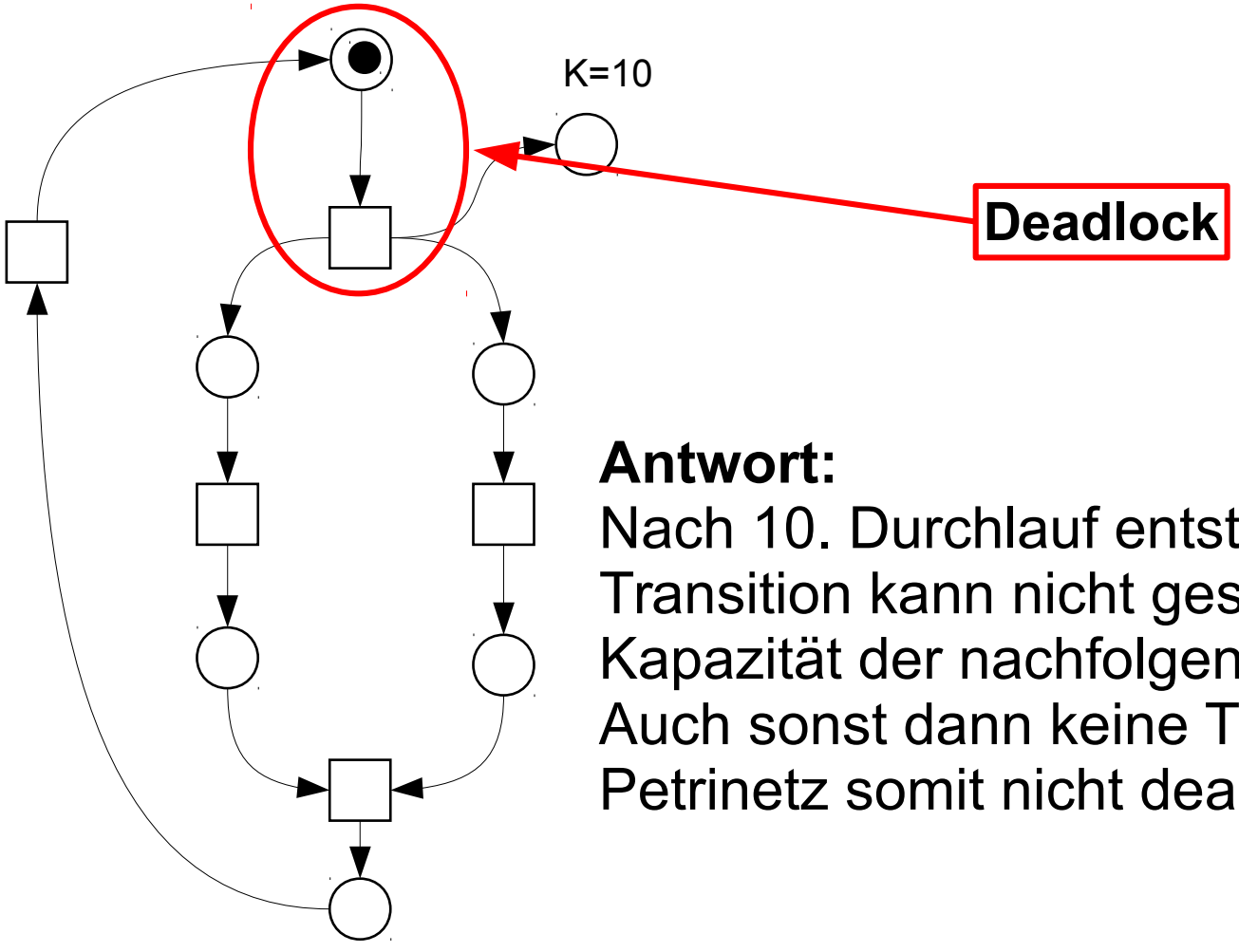
Lebendig ist nicht logische Negation von tot (sondern stärker).

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

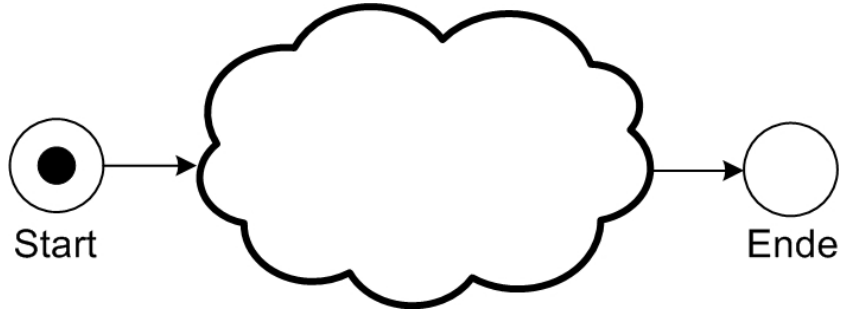
- Petrinetz-Syntax
- Ausführung
- Analyse von Systemen
- Workflow-Netze

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.

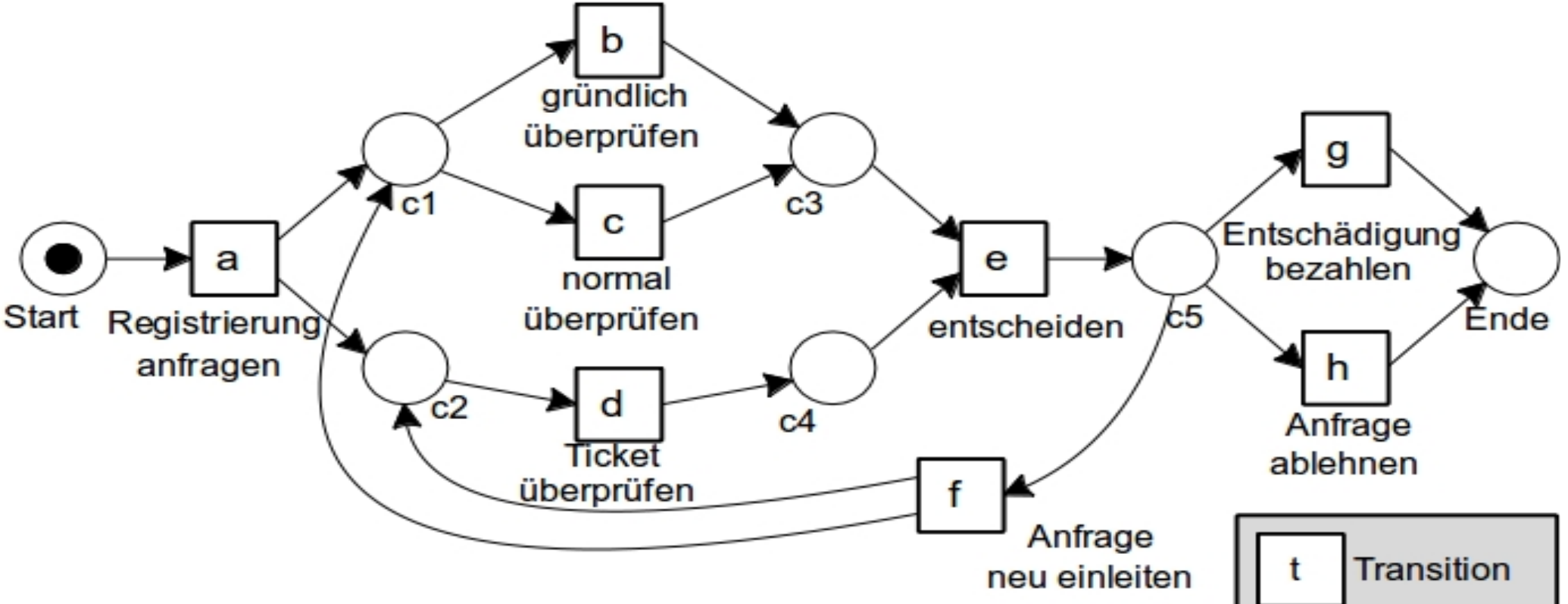
Initiale Markierung bei Ausführung Workflow-Netz:

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



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

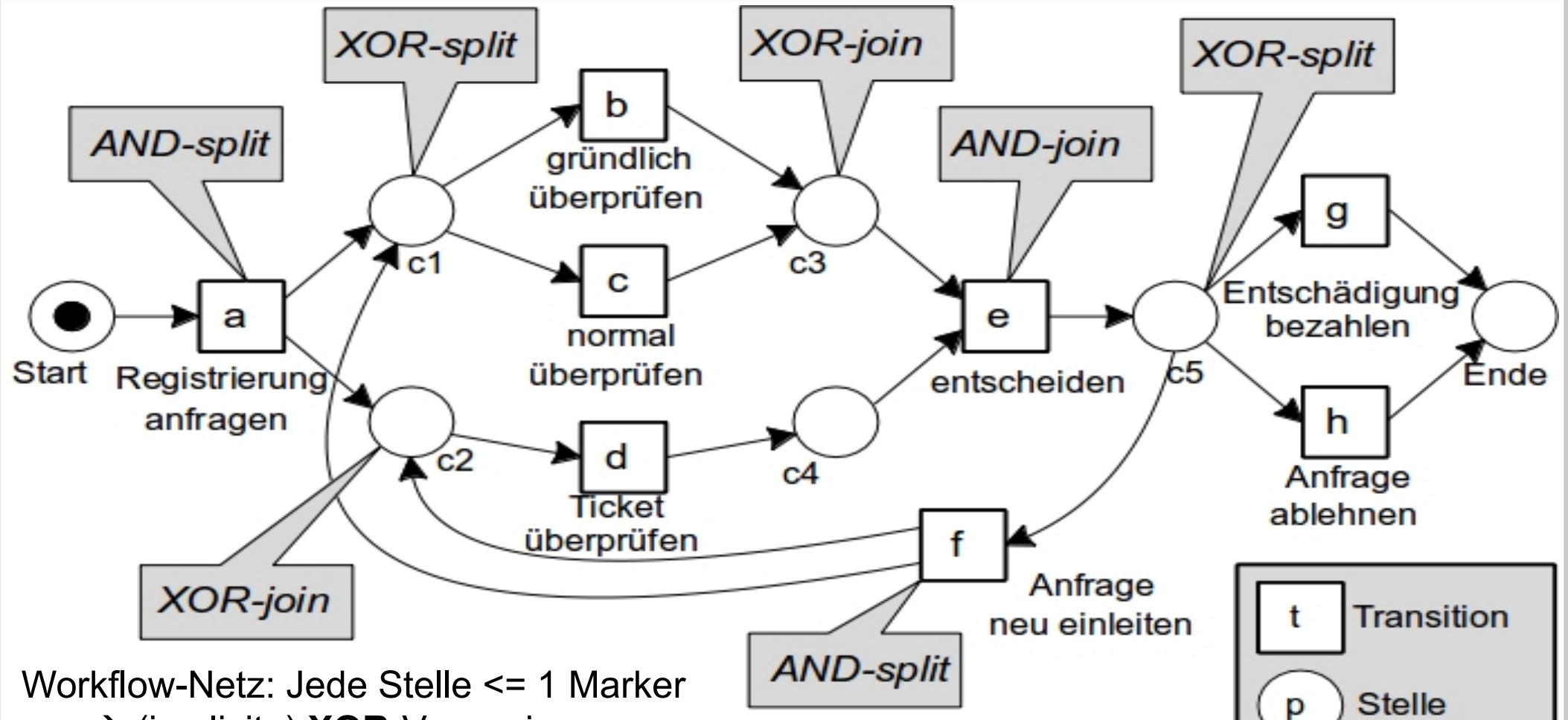
Verzweigungen an Stellen / Transitionen



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

Gilt dies auch in allgemeinen Petrinetzen ?

Verzweigungen an Stellen / Transitionen

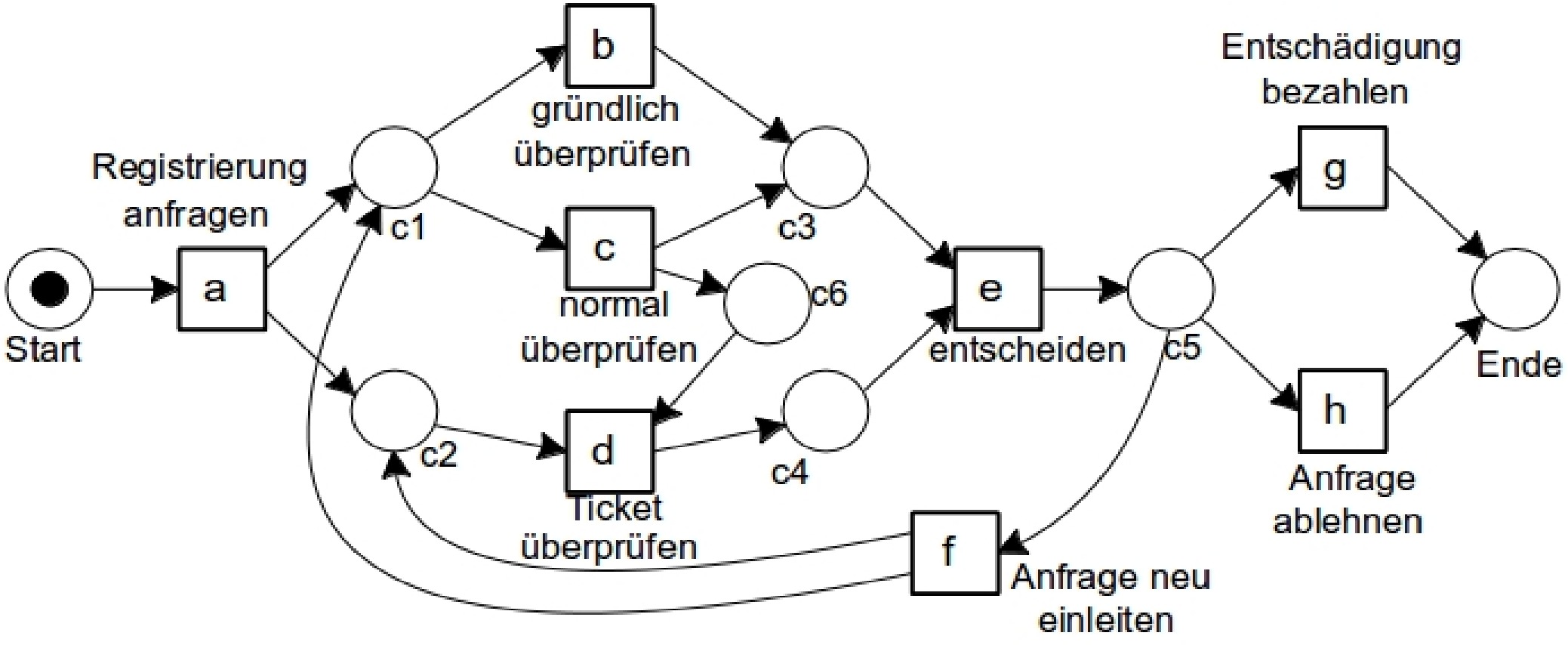


Workflow-Netz: Jede Stelle <= 1 Marker
→ (implizite) **XOR**-Verzweigung.

[Allgemeines Petrinetz: ggf. > 1 Marker, auch inklusives OR möglich.]

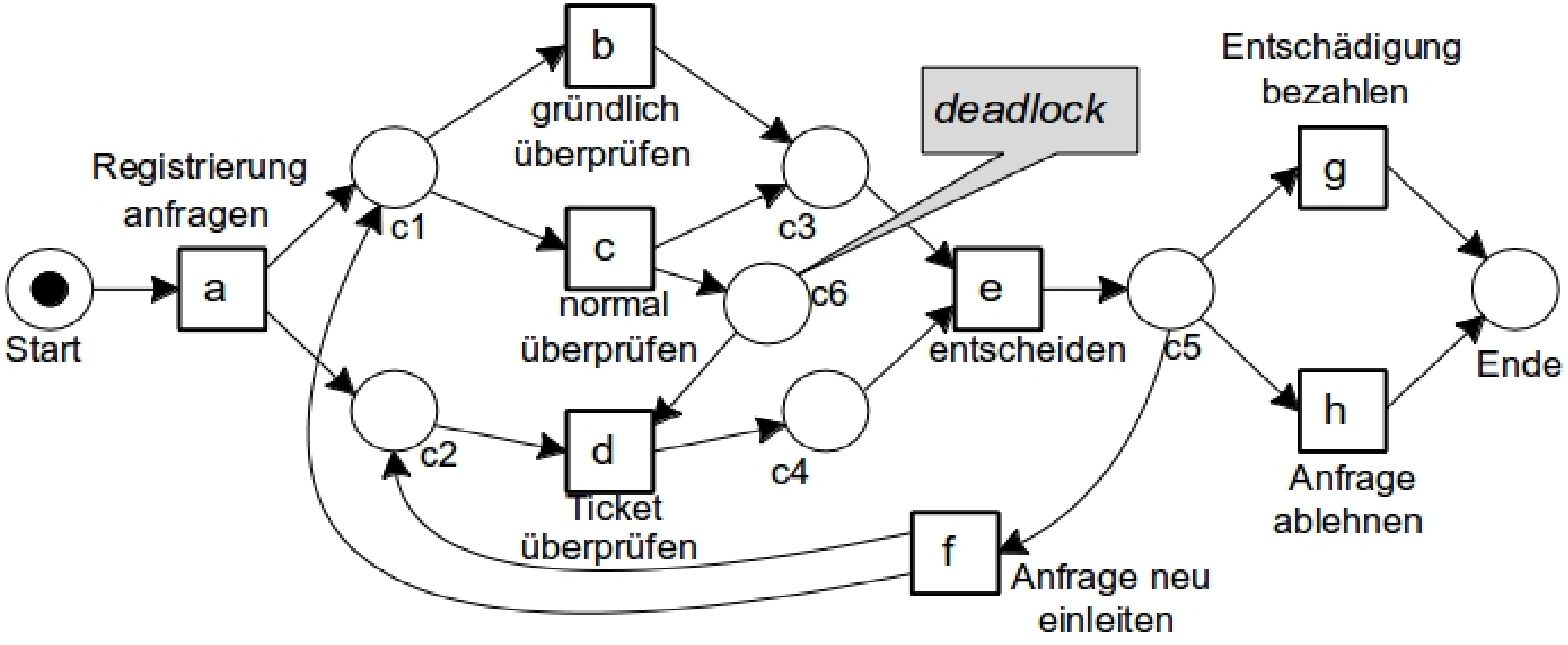
Transitionen: implizite AND-Verzweigung. [Auch allgemeine Petrinetze.]

Beispiel Lebendigkeit in Workflow-Netzen



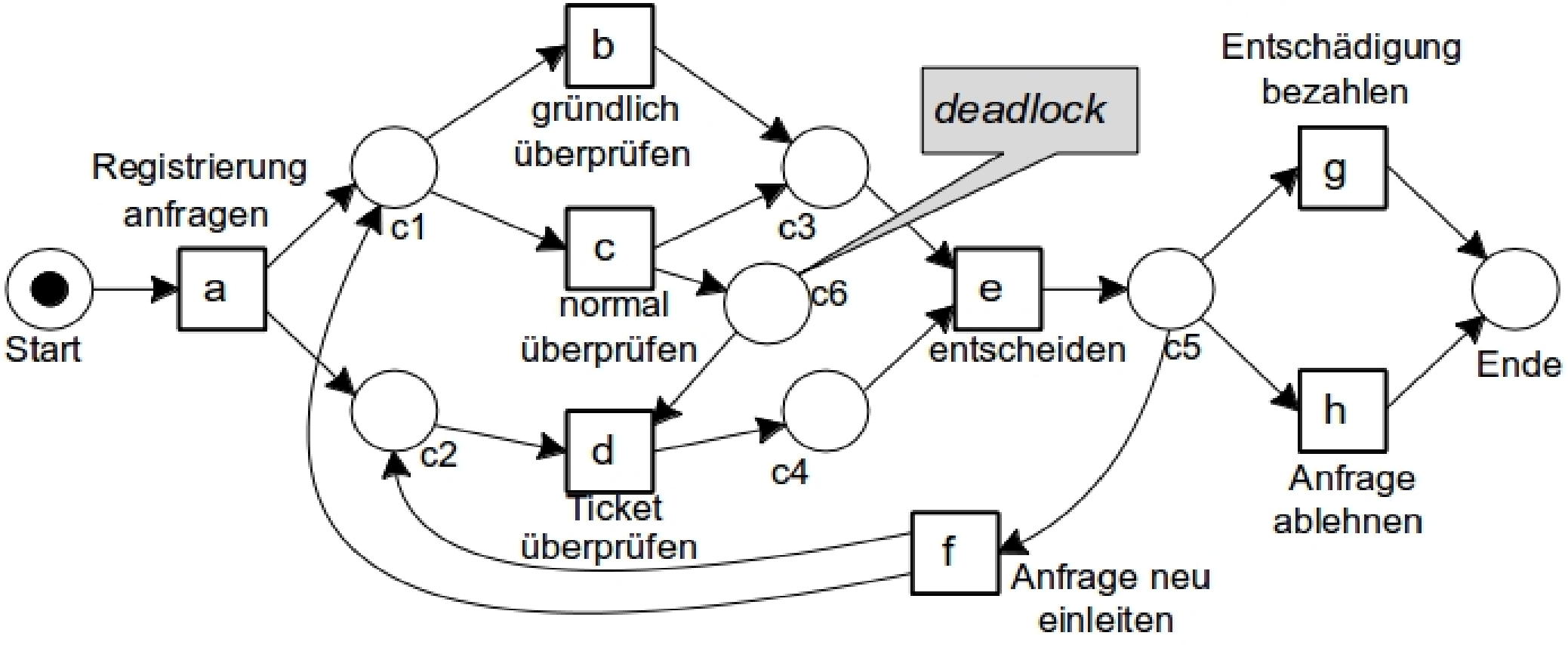
Wo ist das Problem ?

Beispiel Lebendigkeit in Workflow-Netzen



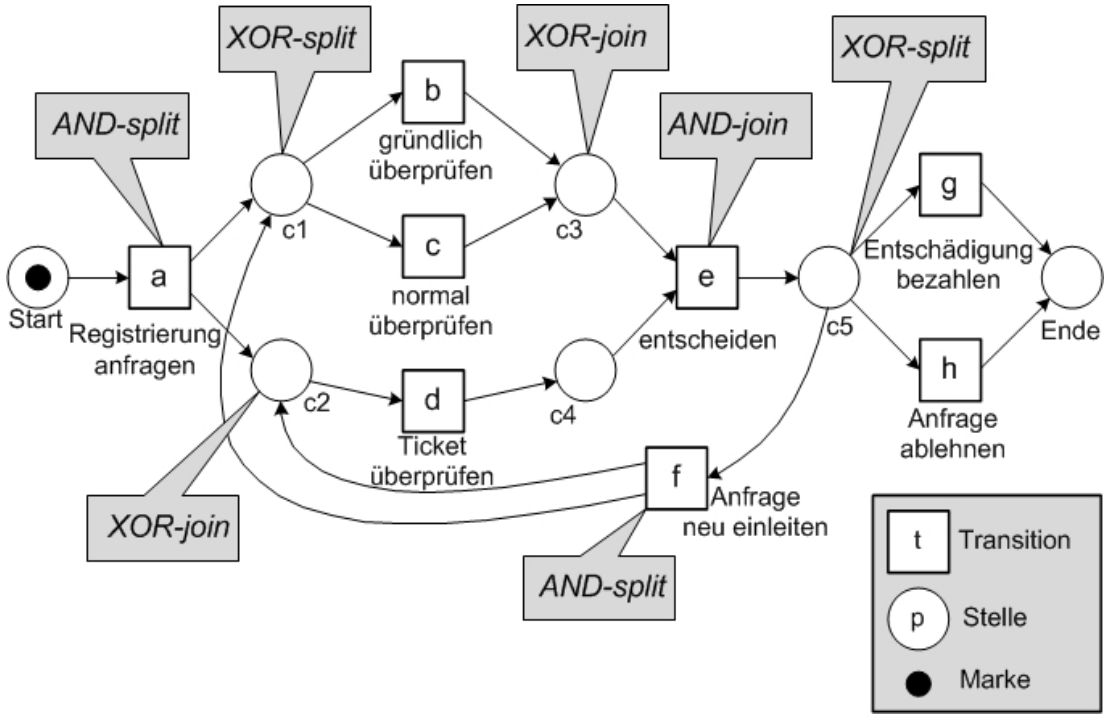
Warum ist hier ein Problem ?

Beispiel Lebendigkeit in Workflow-Netzen



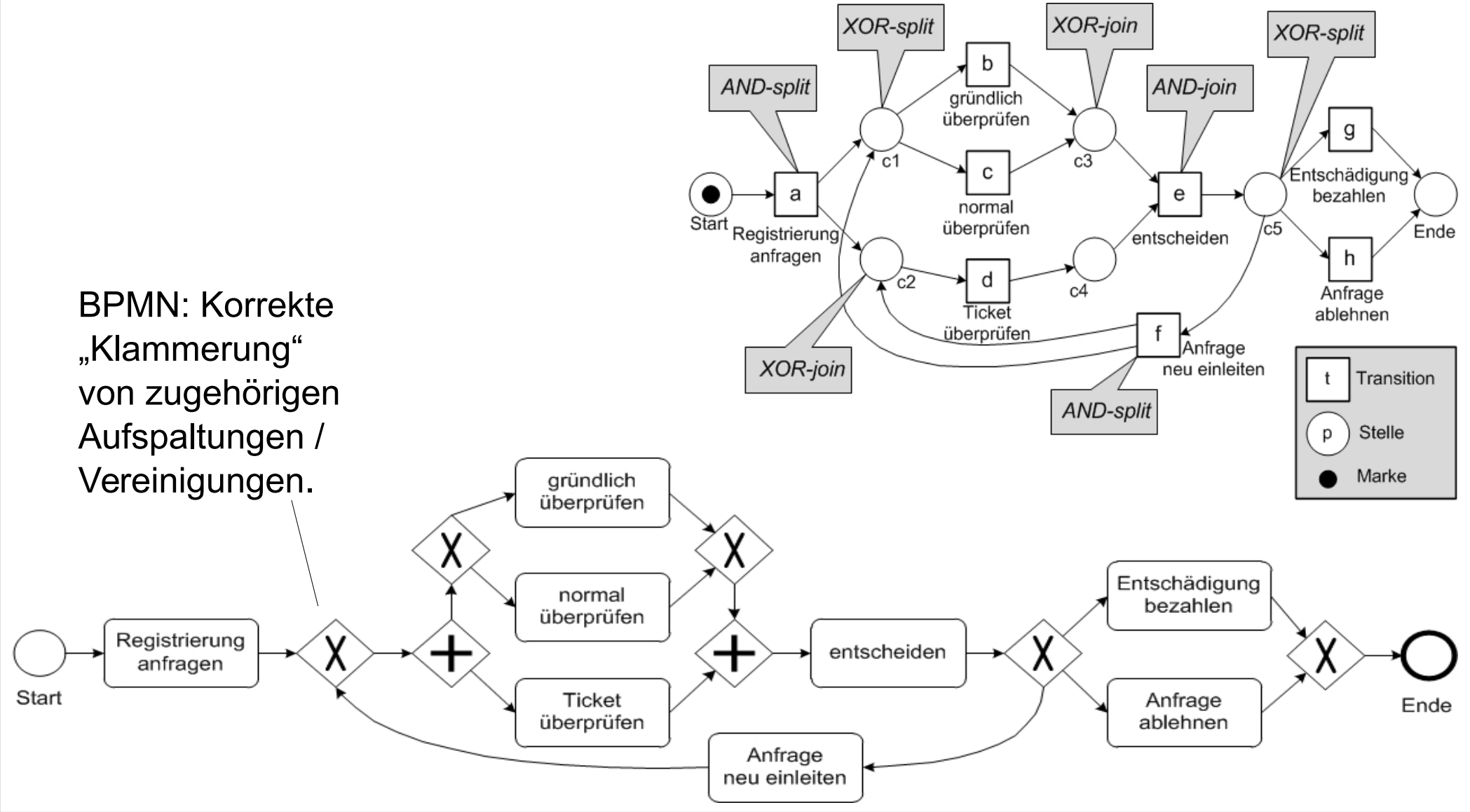
Falls **XOR**-Verzweigung c1 Zweig b wählt, kann d nie schalten, weil c6 kein Token erhält (auch da AND-Verknüpfung e nicht schalten kann).

Petrinetz vs BPMN ?



Petrinetz vs BPMN

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



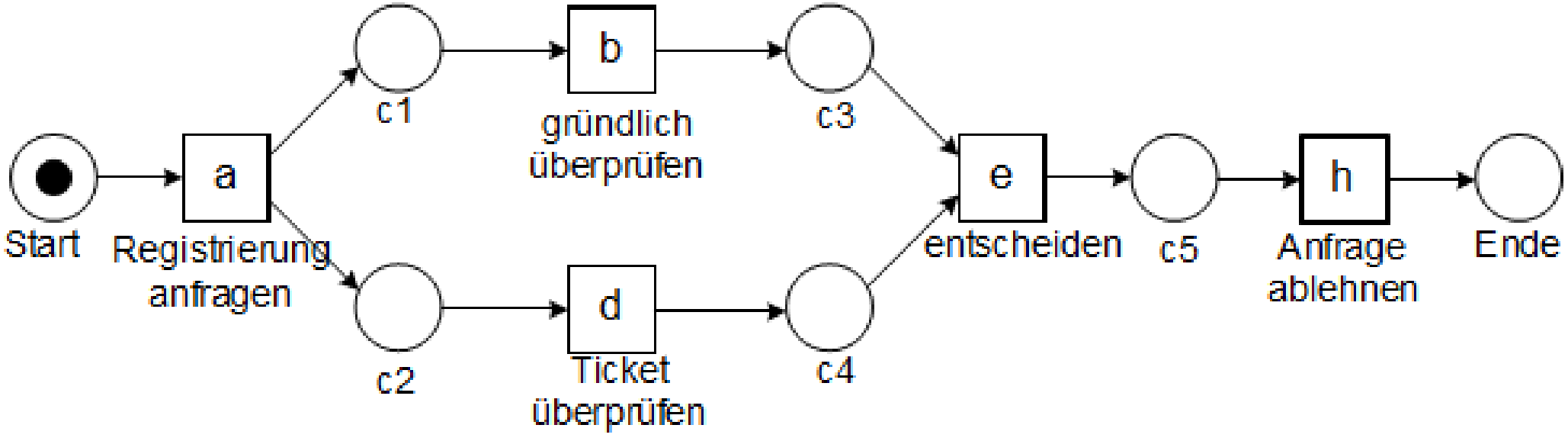
Prozess-Discovery: Von Event-Log zu Petrinetz

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

Zugehöriges Petrinetz ?

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

Zugehöriges Petrinetz ?

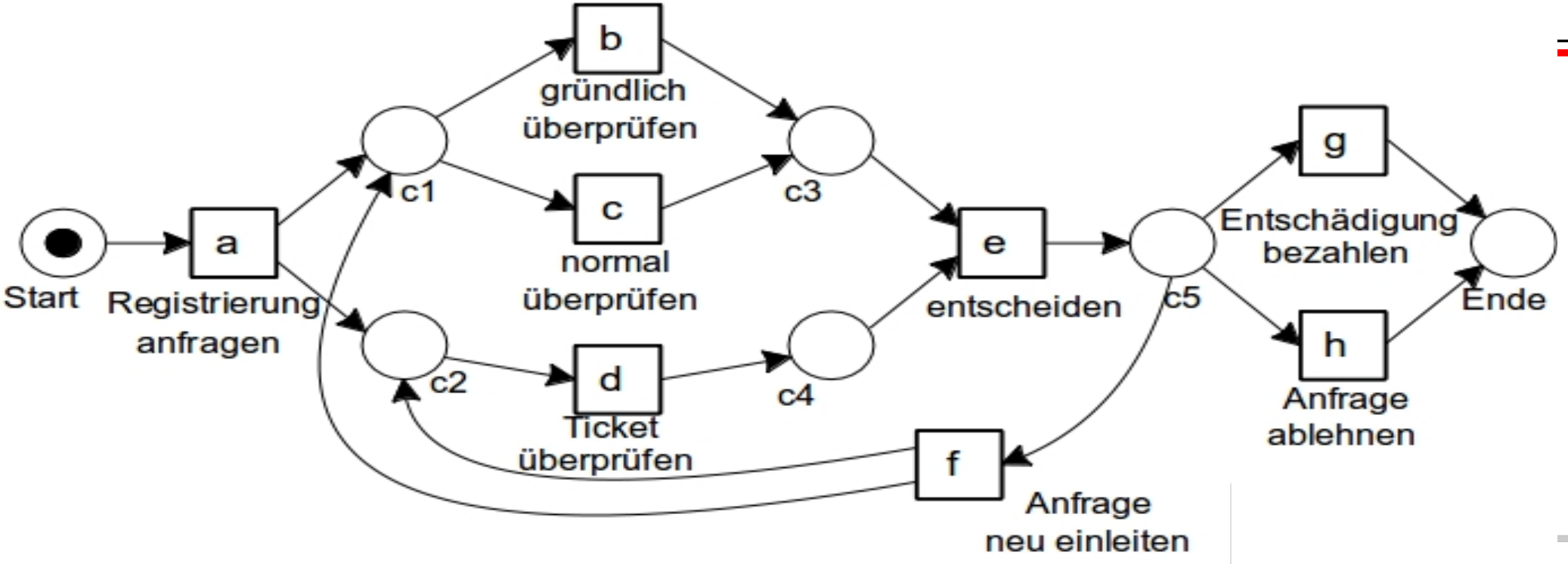


Prozess-Discovery: Weiteres Beispiel

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	$\langle a, c, d, e, g \rangle$
...	...

Prozess-Discovery: Weiteres Beispiel

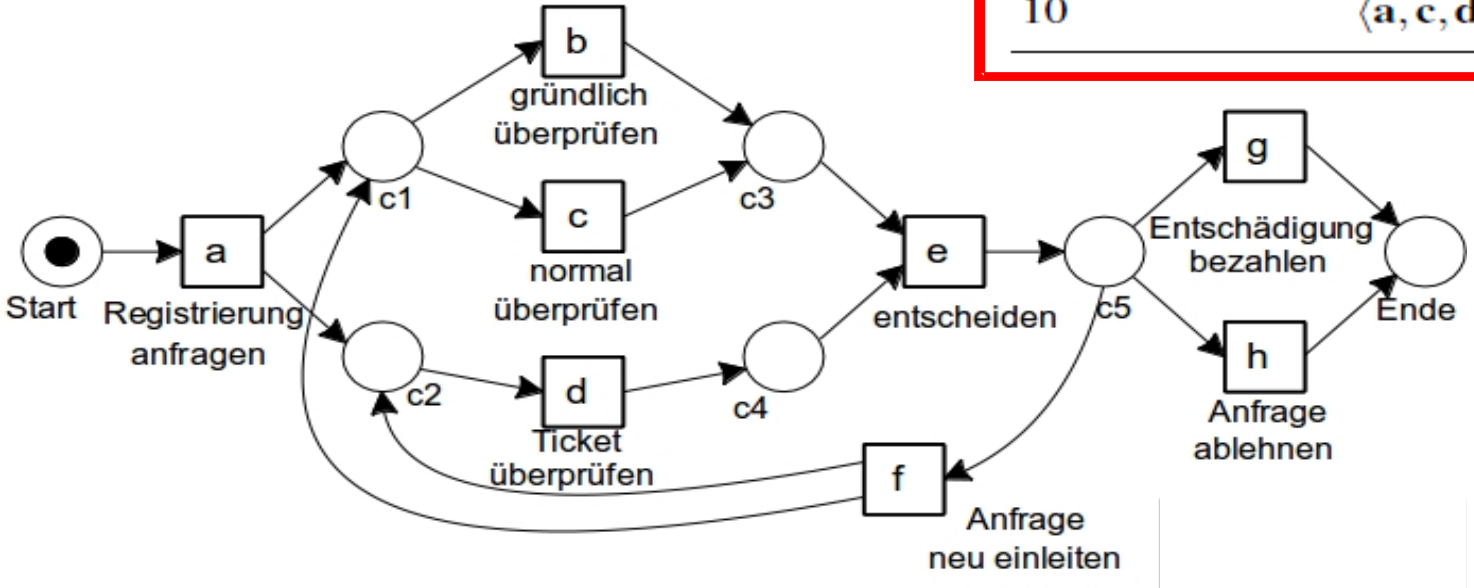
case id	trace
1	$\langle a, b, d, e, h \rangle$
2	$\langle a, d, c, e, g \rangle$
3	$\langle a, c, d, e, f, b, d, e, g \rangle$
4	$\langle a, d, b, e, h \rangle$
5	$\langle a, c, d, e, f, d, c, e, f, c, d, e, h \rangle$
6	$\langle a, c, d, e, g \rangle$



Prozess-Discovery: Konformanz

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	$\langle \mathbf{a},\mathbf{c},\mathbf{d},\mathbf{e},\mathbf{f},\mathbf{b},\mathbf{d},\mathbf{g} \rangle$

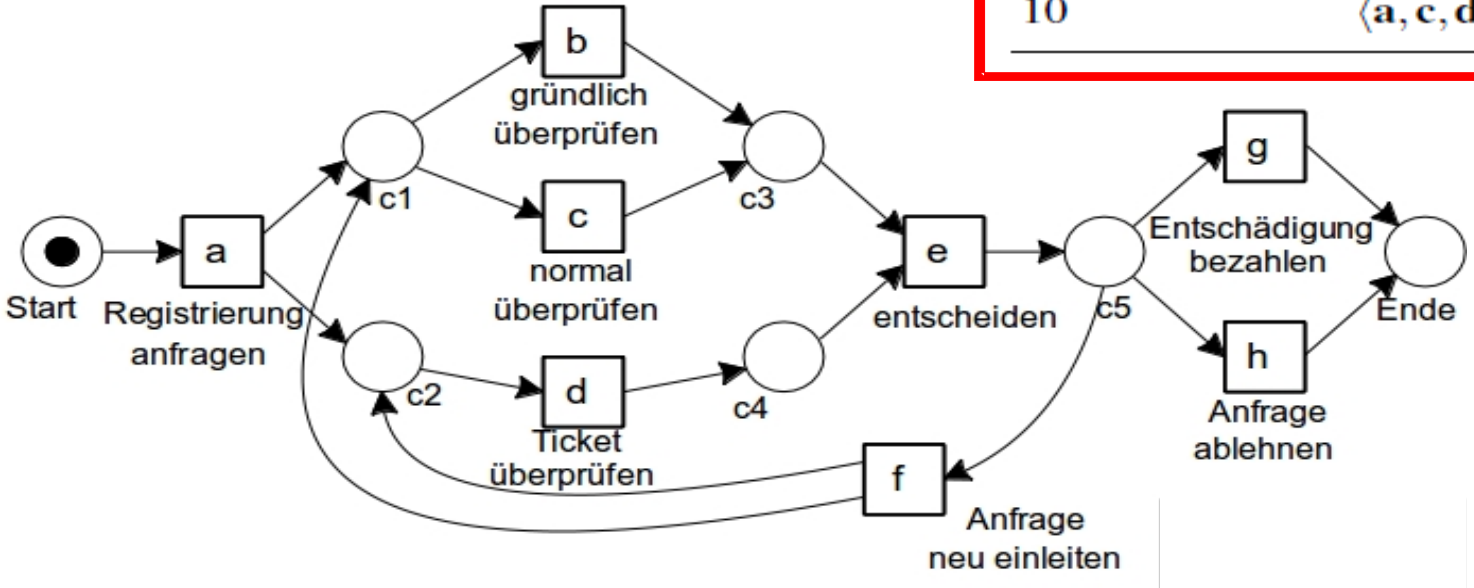


Prozess-Discovery: Konformanz

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).

Anhang (zum selbständigen Nacharbeiten)

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

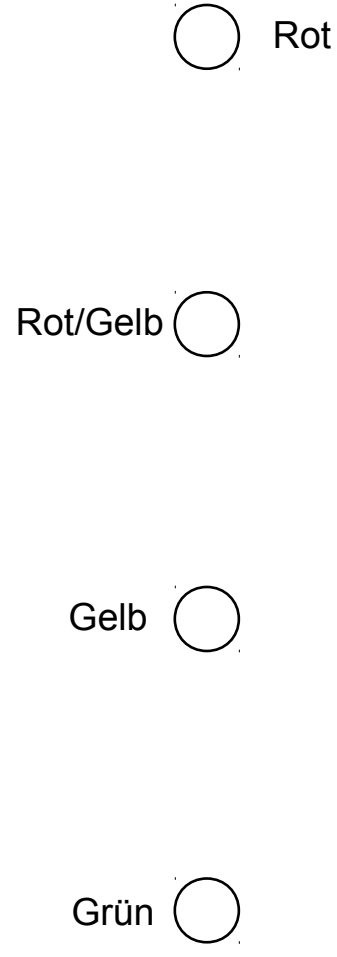
Wir wollen eine Ampelschaltung modellieren.

- 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.

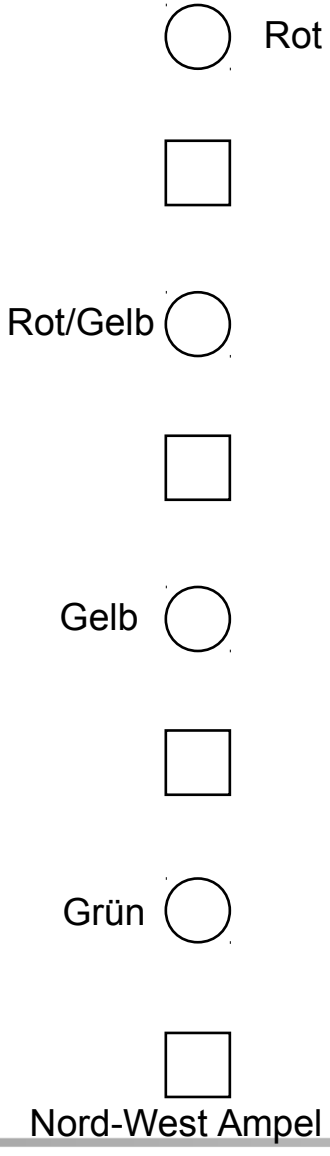
- Zwei Ampeln an einer Kreuzung
- Nord-Süd-Ampel und Ost-West-Ampel
- Abbieger-Ampel vernachlässigen wir
- Folgende Stellen gibt es:
  - Rot
  - Rot/Gelb
  - Gelb
  - Grün



Beispiel: Ampelschaltung

Wir wollen eine Ampelschaltung modellieren.

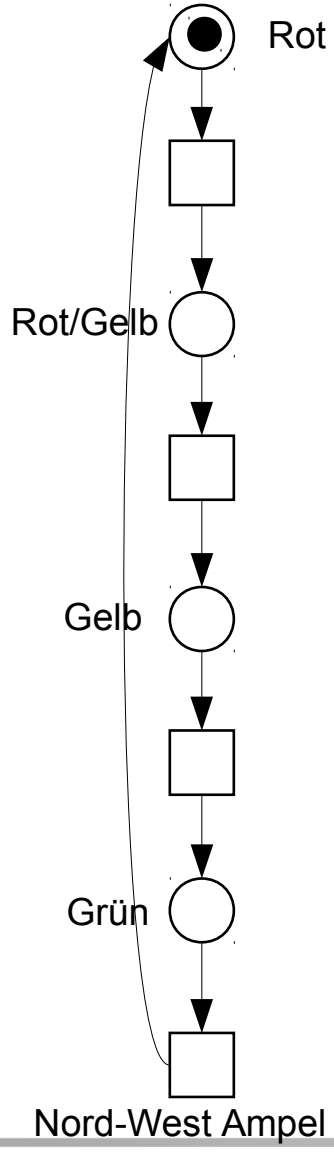
- Zwei Ampeln an einer Kreuzung
- Nord-Süd-Ampel und Ost-West-Ampel
- Abbieger-Ampel vernachlässigen wir
- Folgende Stellen gibt es:
  - Rot
  - Rot/Gelb
  - Gelb
  - Grün



Beispiel: Ampelschaltung

Wir wollen eine Ampelschaltung modellieren.

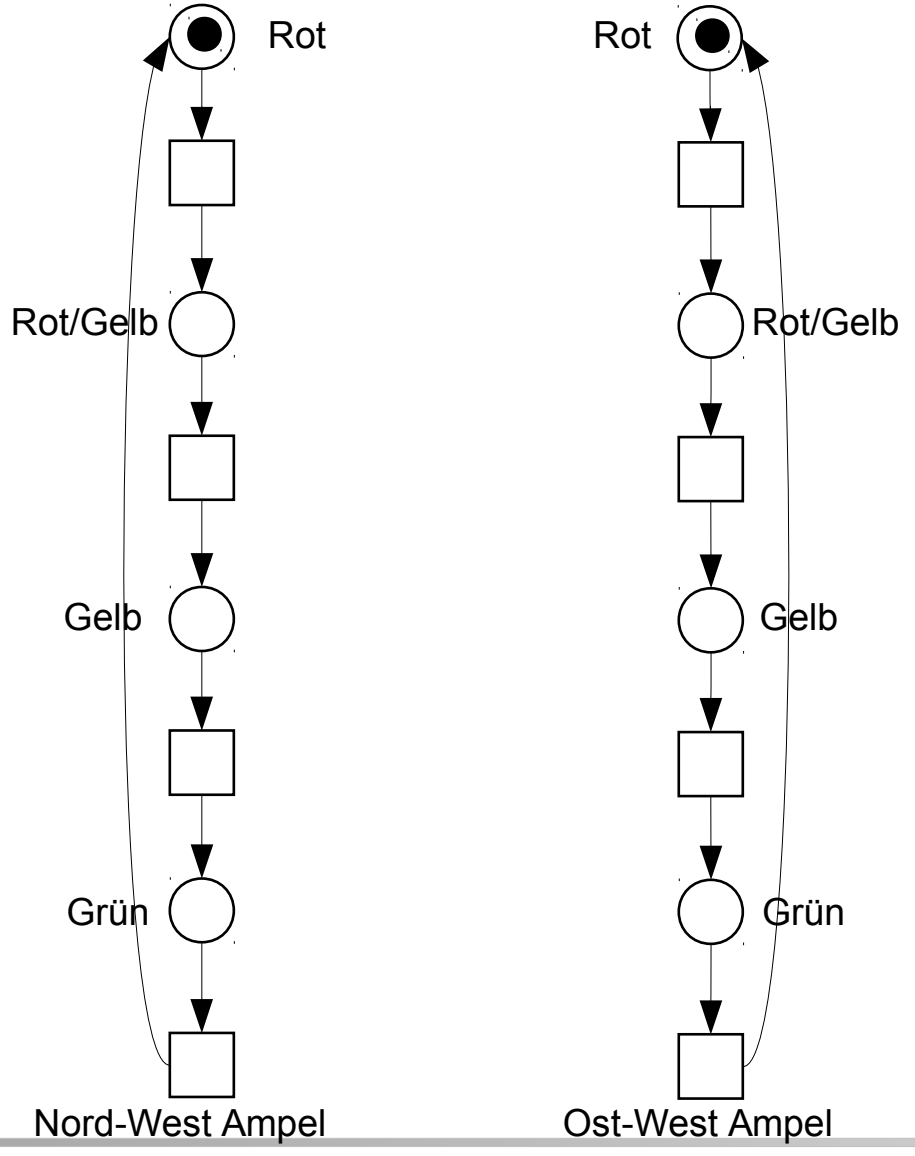
- Zwei Ampeln an einer Kreuzung
- Nord-Süd-Ampel und Ost-West-Ampel
- Abbieger-Ampel vernachlässigen wir
- Folgende Stellen gibt es:
  - Rot
  - Rot/Gelb
  - Gelb
  - Grün



Beispiel: Ampelschaltung

Wir wollen eine Ampelschaltung modellieren.

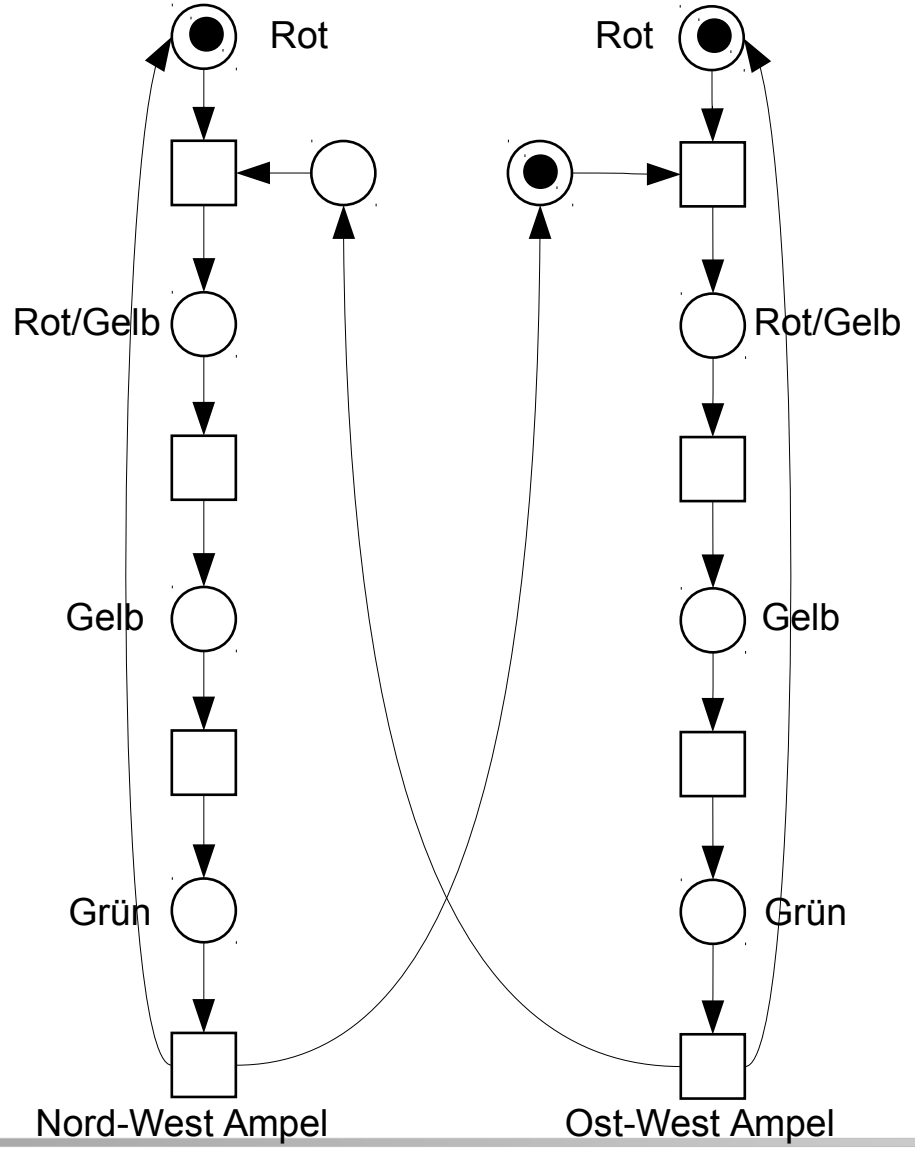
- Zwei Ampeln an einer Kreuzung
- Nord-Süd-Ampel und Ost-West-Ampel
- Abbieger-Ampel vernachlässigen wir
- Folgende Stellen gibt es:
  - Rot
  - Rot/Gelb
  - Gelb
  - Grün



Beispiel: Ampelschaltung

Wir wollen eine Ampelschaltung modellieren.

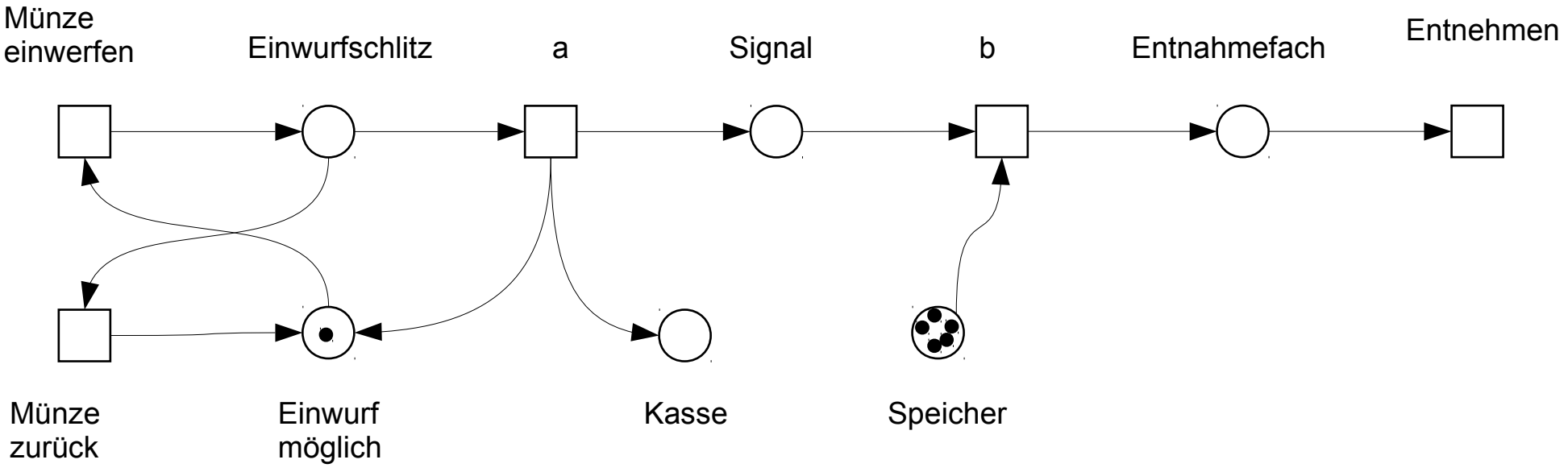
- Zwei Ampeln an einer Kreuzung
- Nord-Süd-Ampel und Ost-West-Ampel
- Abbieger-Ampel vernachlässigen wir
- Folgende Stellen gibt es:
  - Rot
  - Rot/Gelb
  - Gelb
  - Grün



Zweites Beispiel

Was könnte folgendes Petri-Netz modellieren?

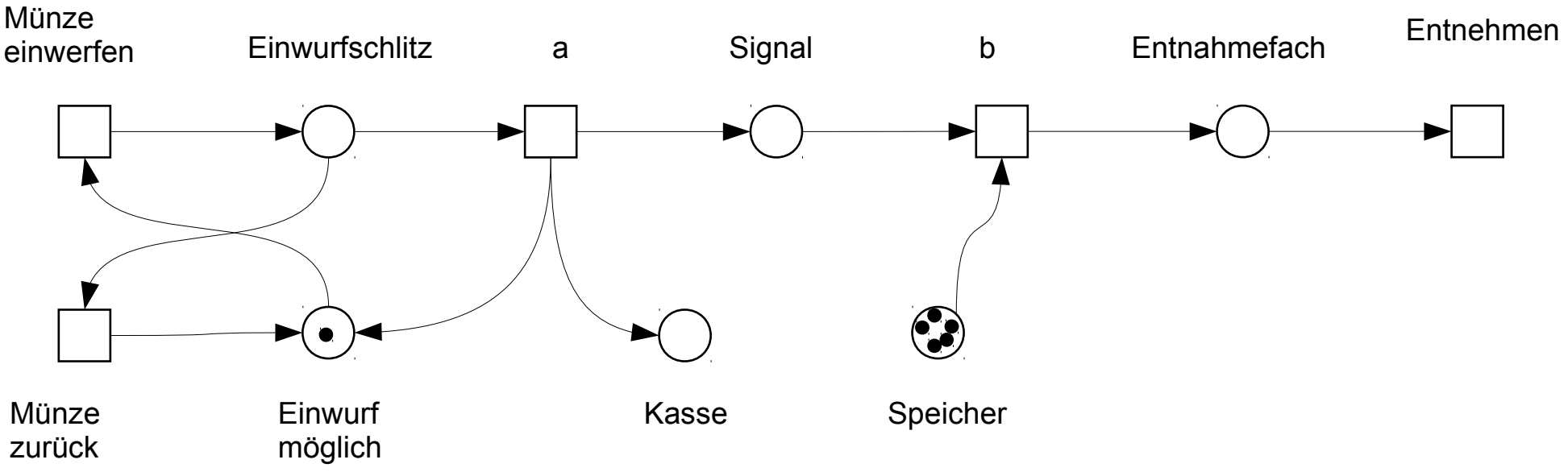
Was könnten die einzelnen Token repräsentieren?



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.