LEHRSTUHL 14
SOFTWARE ENGINEERING

Vorlesung

Methodische Grundlagen des Software-Engineering

im Sommersemester 2013

Prof. Dr. Jan Jürjens

TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

Teil 2.1: Petrinetze

v. 09.05.2013

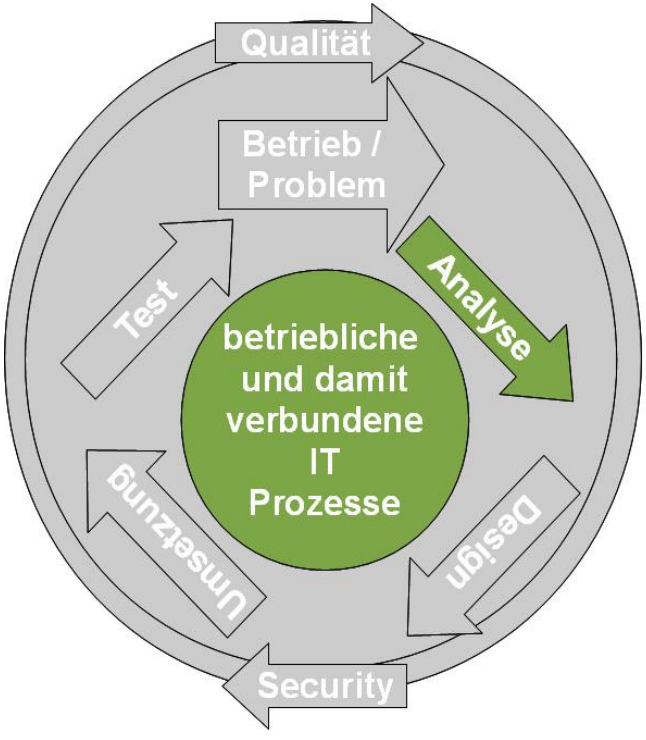
2.1 Petrinetze

[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: Wiederholung Petri-Netze

- Geschäftsprozessmodellierung
- Business Process Mining
  - Einführung: Process Mining
  - Petrinetze
  - Prozessmodellierung und Analyse
  - Data-Mining
  - Datenbeschaffung
  - Prozessextraktion
  - Fortgeschrittene Prozessextraktion
- Modellbasierte Softwareentwicklung
- Modellbasierte Entwicklung sicherer Software



Agenda

- Petrinetze – Beispiel
- Petrinetze – Grundlagen
- Stellen/Transitions-Netze
- Erreichbarkeit
- Grundsituationen in S/T-Netzen
- Analyse von Systemen
- Methodik

Motivation

Kap. 1: GP-Modellierungsnotationen BPMN 2 und EPK.

Informell einige Aspekte des intendierten Verhaltens der spezifizierten GP-Modelle diskutiert (vgl. z.B. Folien zu Kontrollfluss in EPK).

Informelle Diskussion nicht ausreichend für automatische Analyse der GP-Modelle (z.B. für GP-Optimierung oder Sicherheitsanalyse).

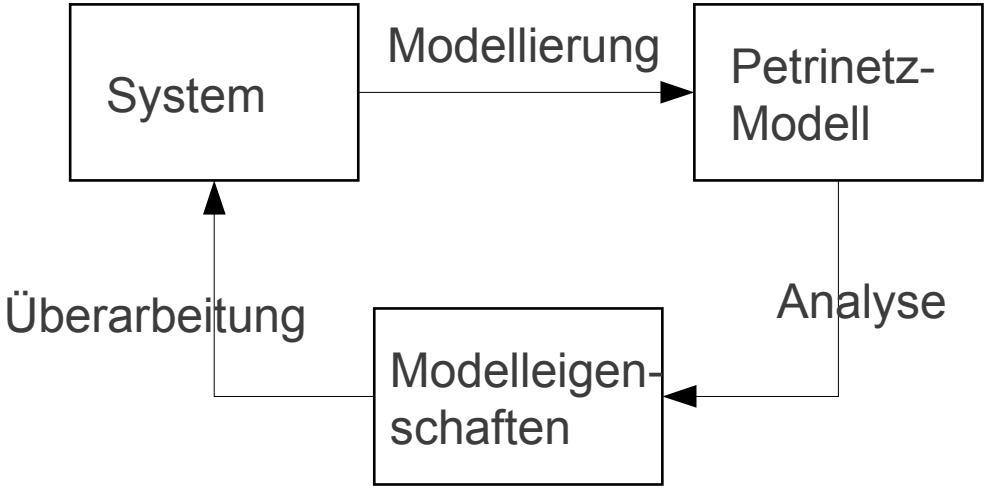
Für präzise und werkzeugseitig implementierbare Ausführungs-semantik von GP-Modellierungsnotationen: Petri-Netze.

Zum Beispiel: Ausführungssemantik von UML 2-Aktivitätsdiagrammen mit Petri-Netzen definiert.

Direkter Einsatz von Petri-Netze in der Praxis z.B. für Process-Mining (dieses Kapitel), insbes. Werkzeug ProM (cf. http://processmining.org).

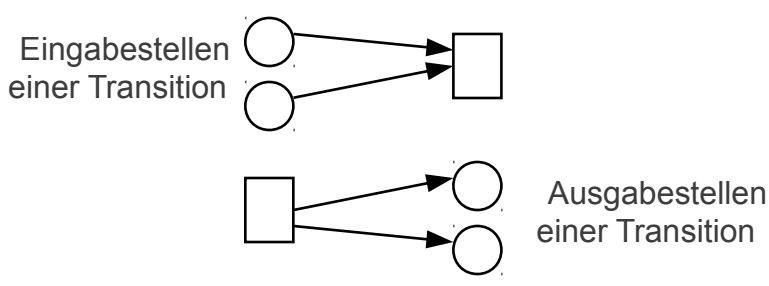
Petrinetze

- 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).



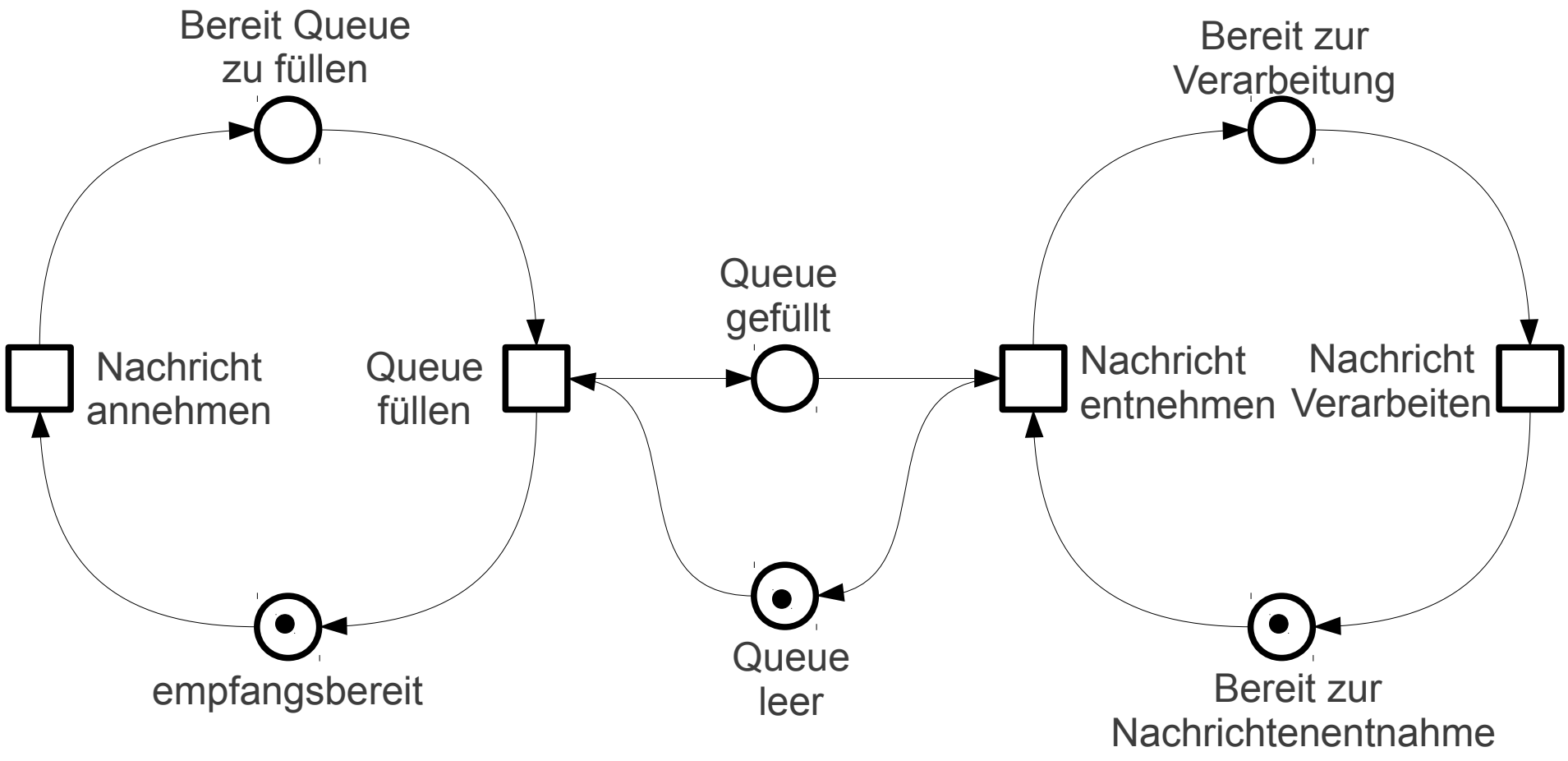
Petrinetz (informell)

- Gerichteter Graph, besteht aus zwei verschiedenen Sorten (Stellen, Transitionen) von Knoten (bipartiter Graph)
  - Stelle: Zwischenablage von Informationen
  - Transitionen: Verarbeitung von Informationen
- Kanten verbinden Stellen mit Transitionen, niemals Stellen mit Stellen oder Transitionen mit Transitionen
- Stellen werden mit Objekten (sog. Token oder Marken) belegt
- Durchlauf der Token durch Petri-Netz beschreibt dynamisches Verhalten des Systems



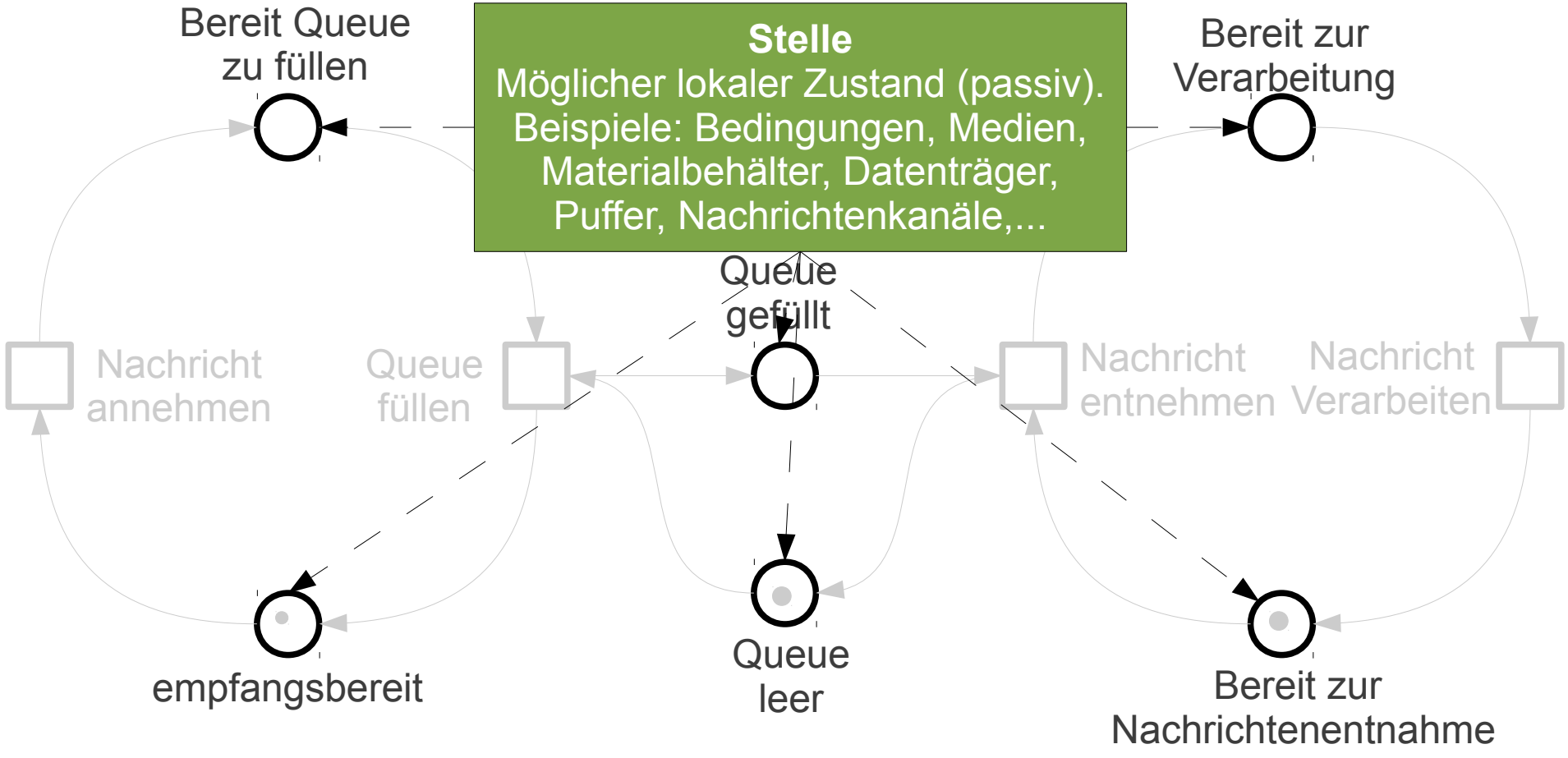
Petri-Netze: Beispiel

Beispiel Nachrichten-Queue



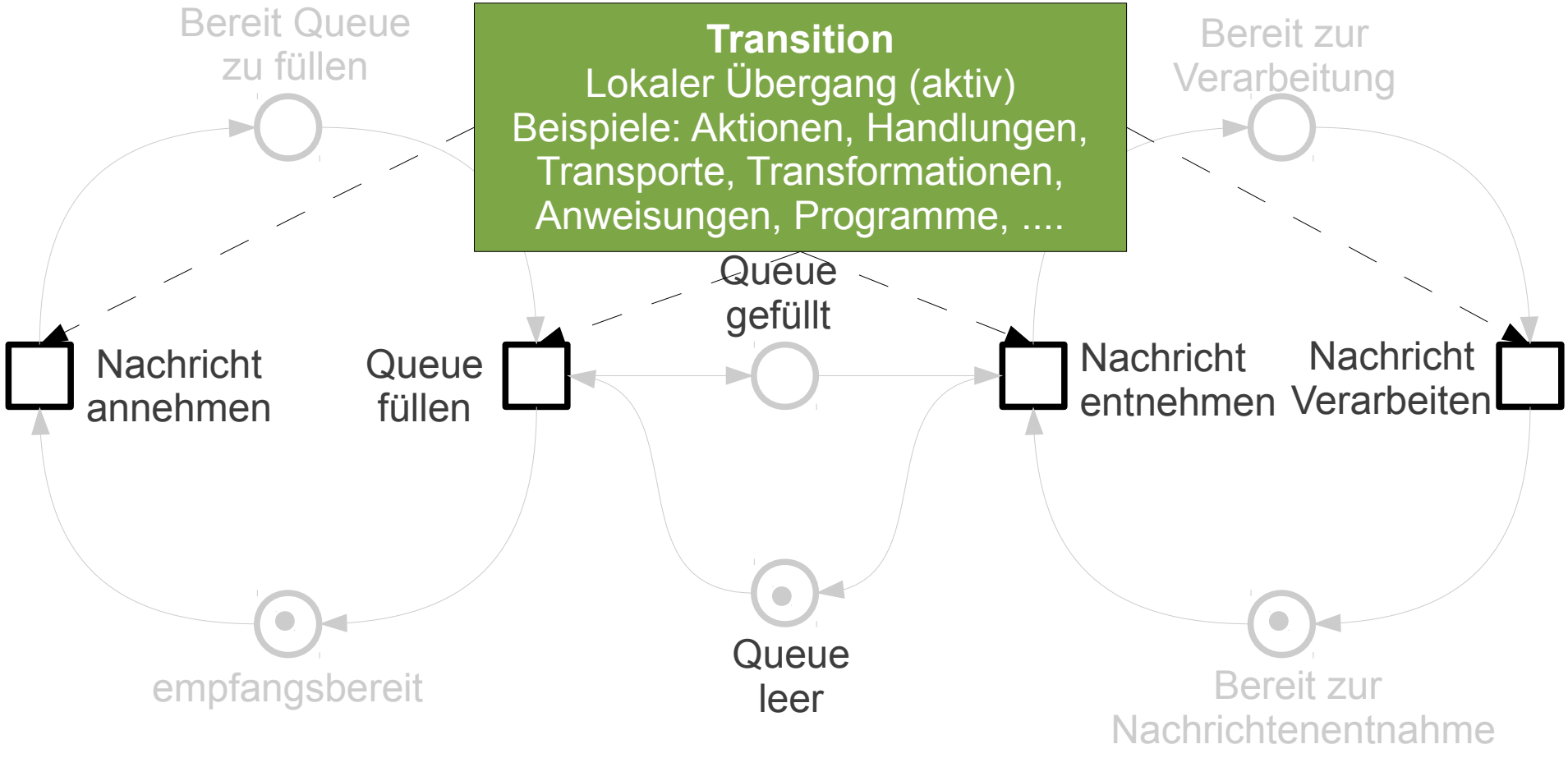
Petri-Netze: Stelle

Beispiel Nachrichten-Queue



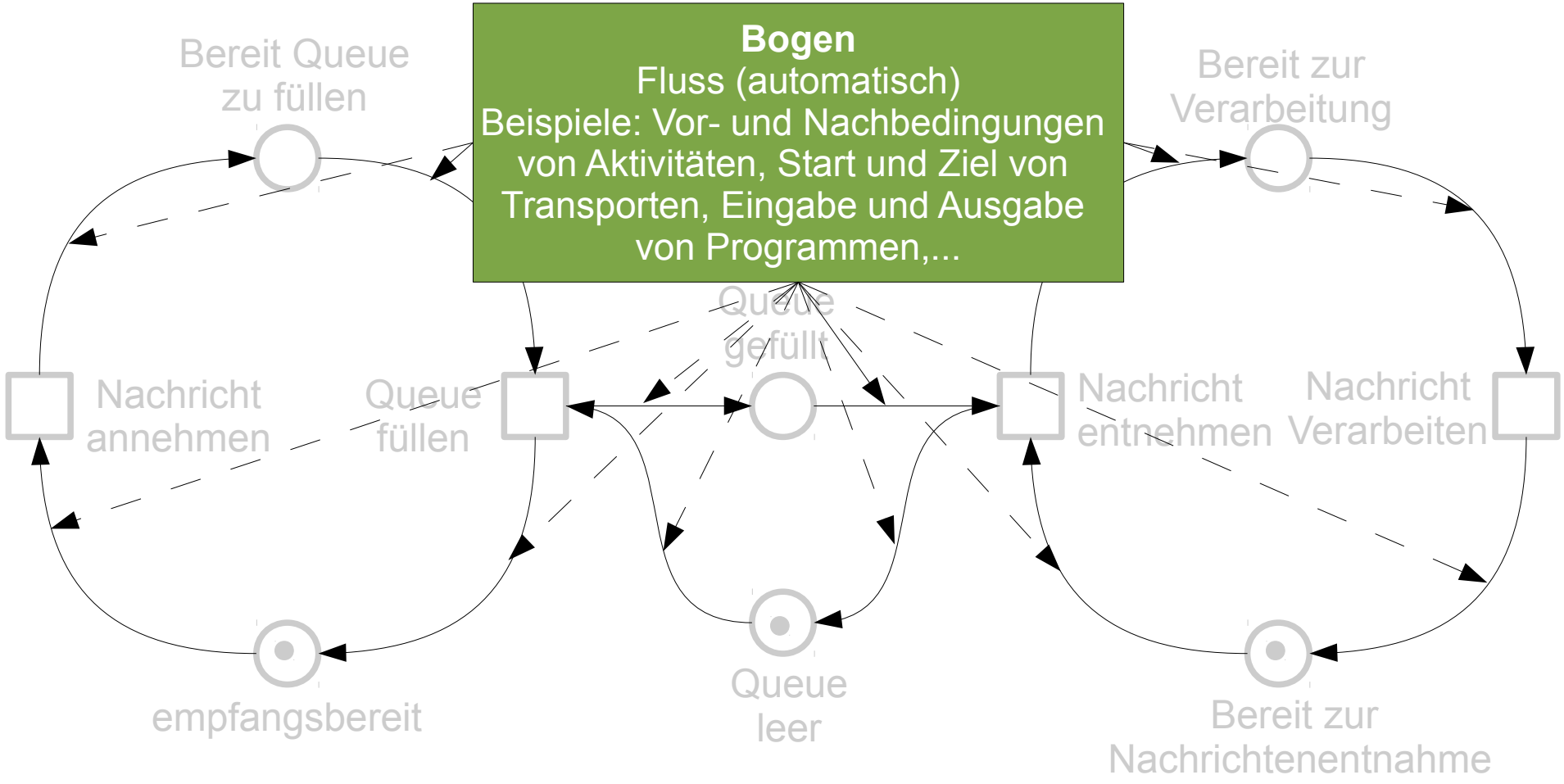
Petri-Netze: Transition

Beispiel Nachrichten-Queue als Netz



Petri-Netze: Bogen

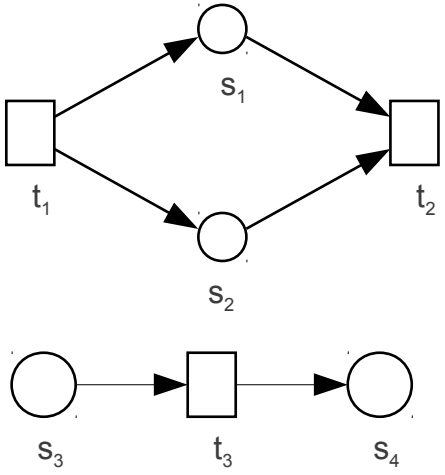
Beispiel Nachrichten-Queue



Definition Netz

Definition Netz:

- Gegeben:
  - S - endliche Menge von Stellen (auch genannt „S-Elemente“)
  - T - endliche Menge von Transitionen (auch genannt „T-Elemente“)
  - F - Menge von Bögen (auch genannt „Kanten“)
    - $F \subseteq (S \times T) \cup (T \times S)$ binäre Relation
- Wenn: $S \neq \emptyset$, $T \neq \emptyset$ und $S \cap T = \emptyset$
- dann: (S,T,F) ein Netz.



$S = \{s_1, s_2, s_3, s_4\}$

$T = \{t_1, t_2, t_3\}$

$F = \{ (t_1, s_1), (t_1, s_2), (s_1, t_2), (s_2, t_2), (s_3, t_3), (t_3, s_4) \}$

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)}

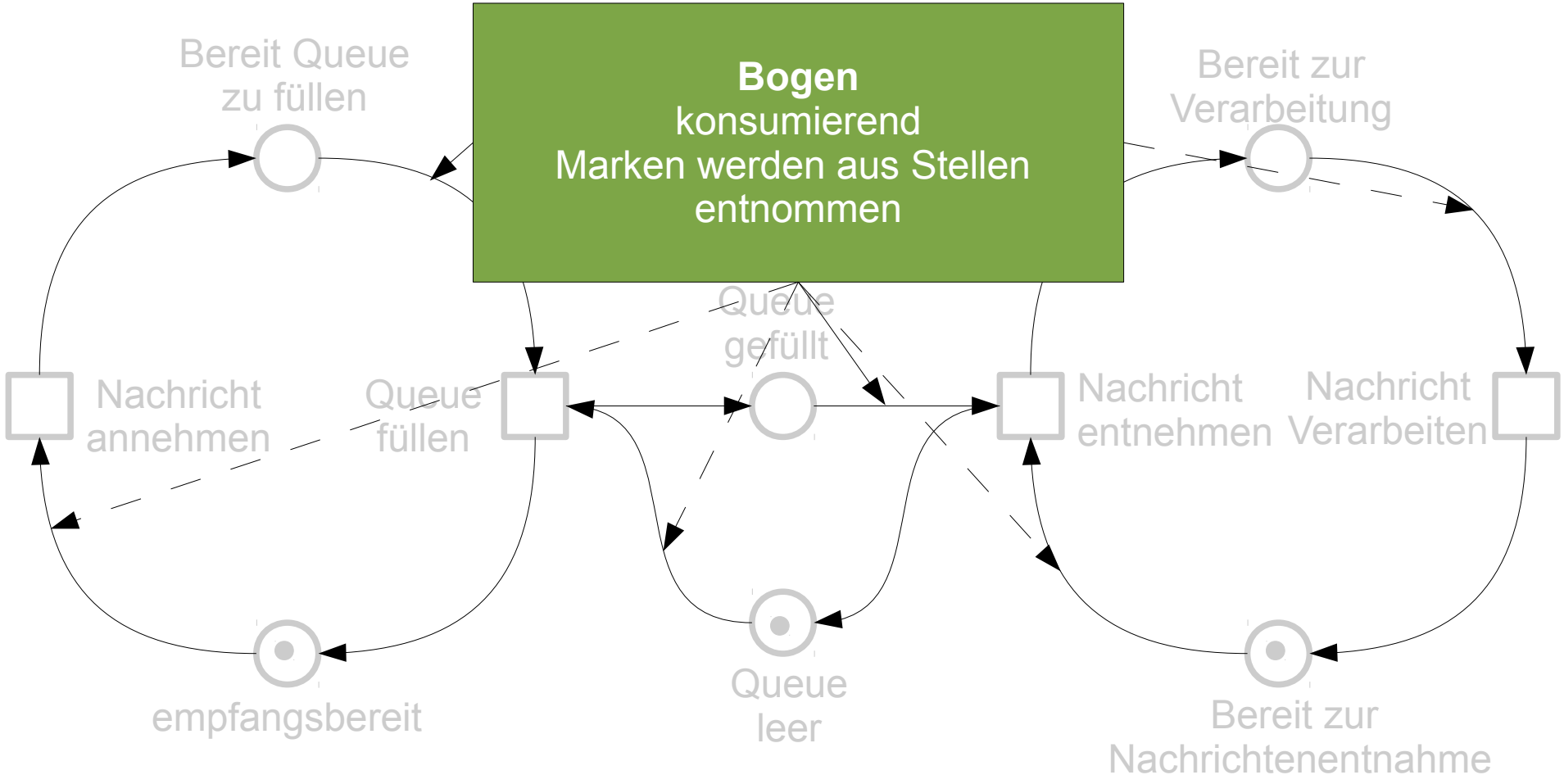
Petri-Netz: Marken

Beispiel Nachrichten-Queue

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

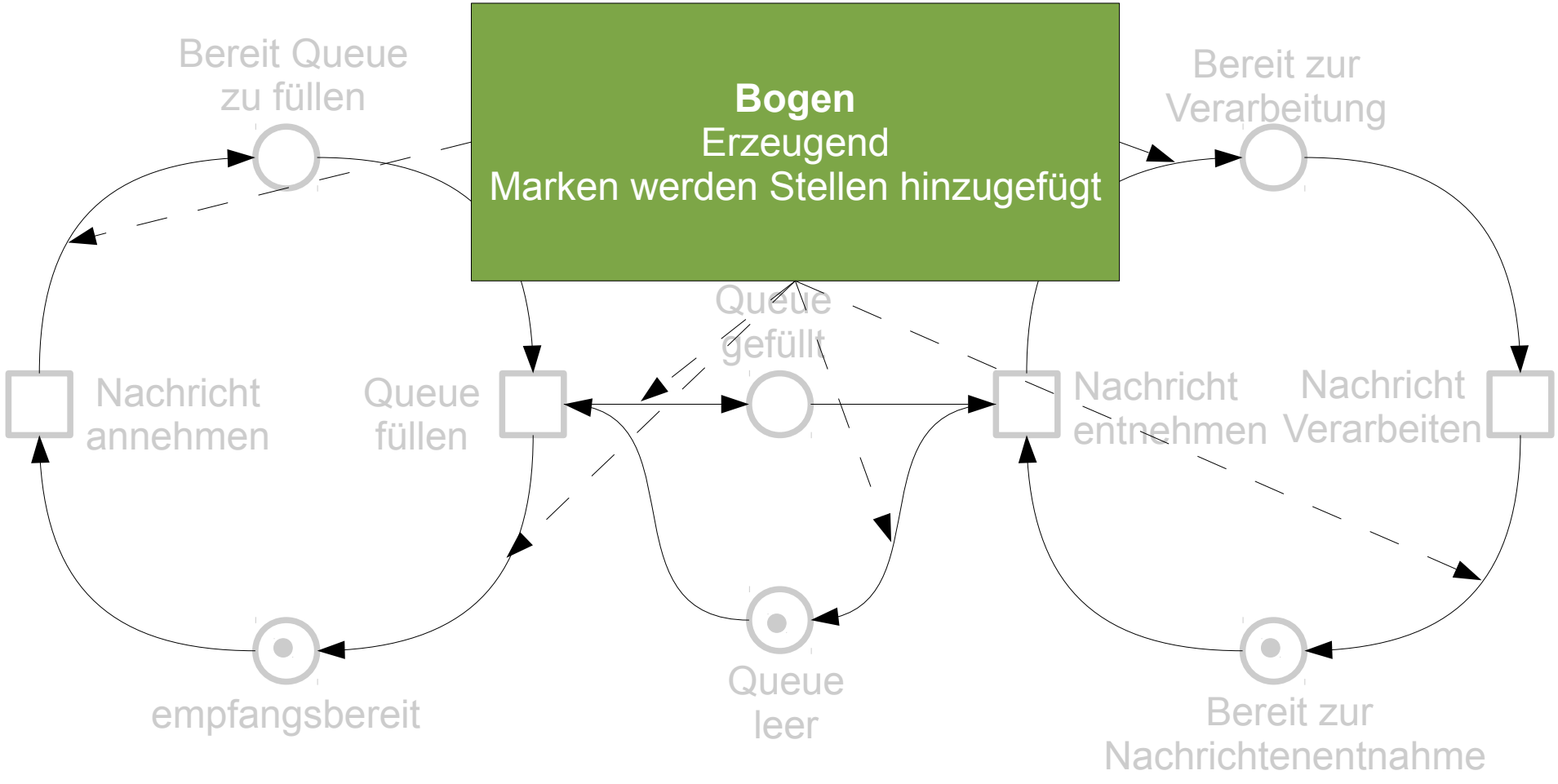
Bogen (konsumierend)

Beispiel Nachrichten-Queue



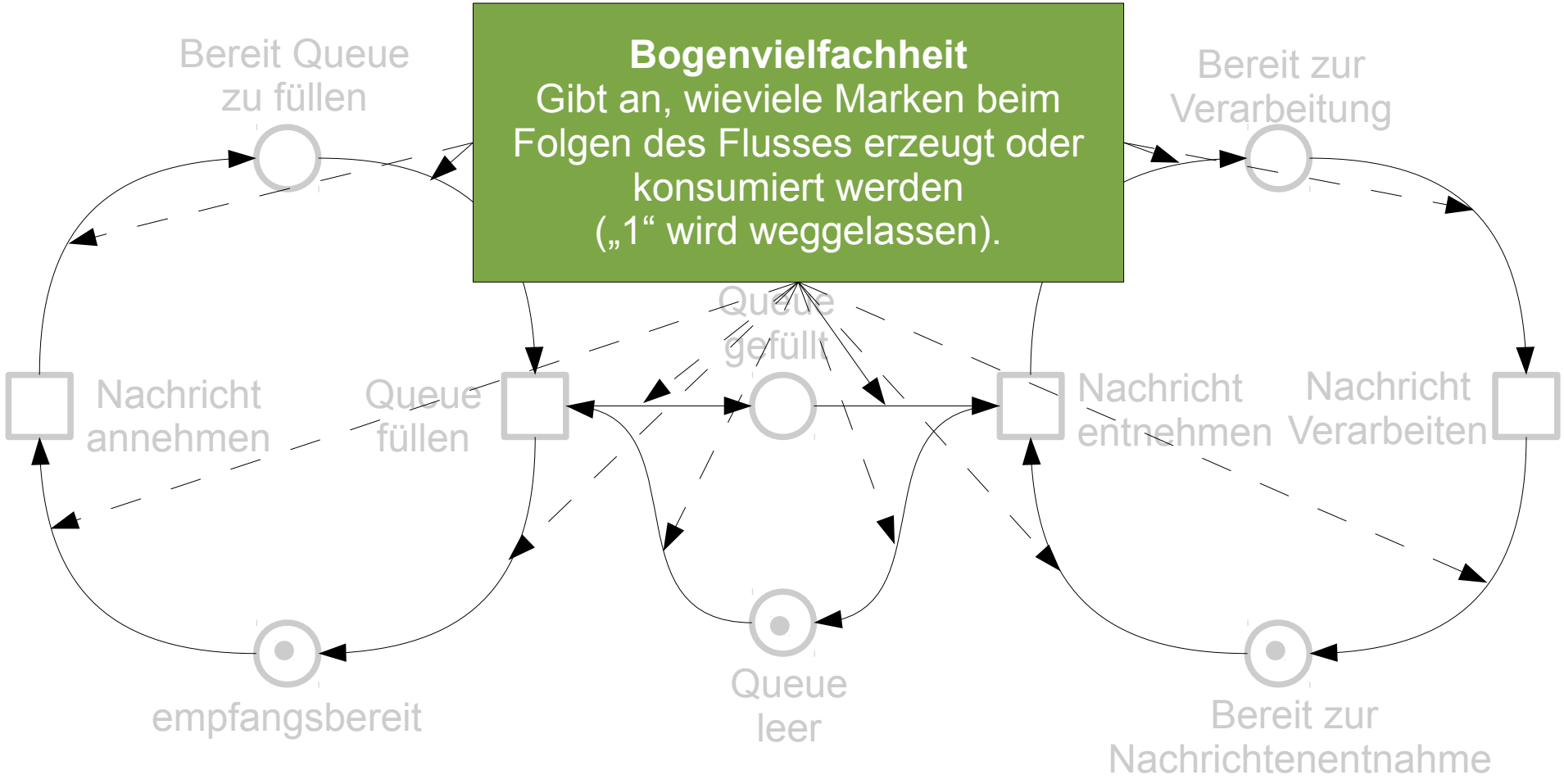
Bogen (erzeugend)

Beispiel Nachrichten-Queue



Bogenvielfachheit

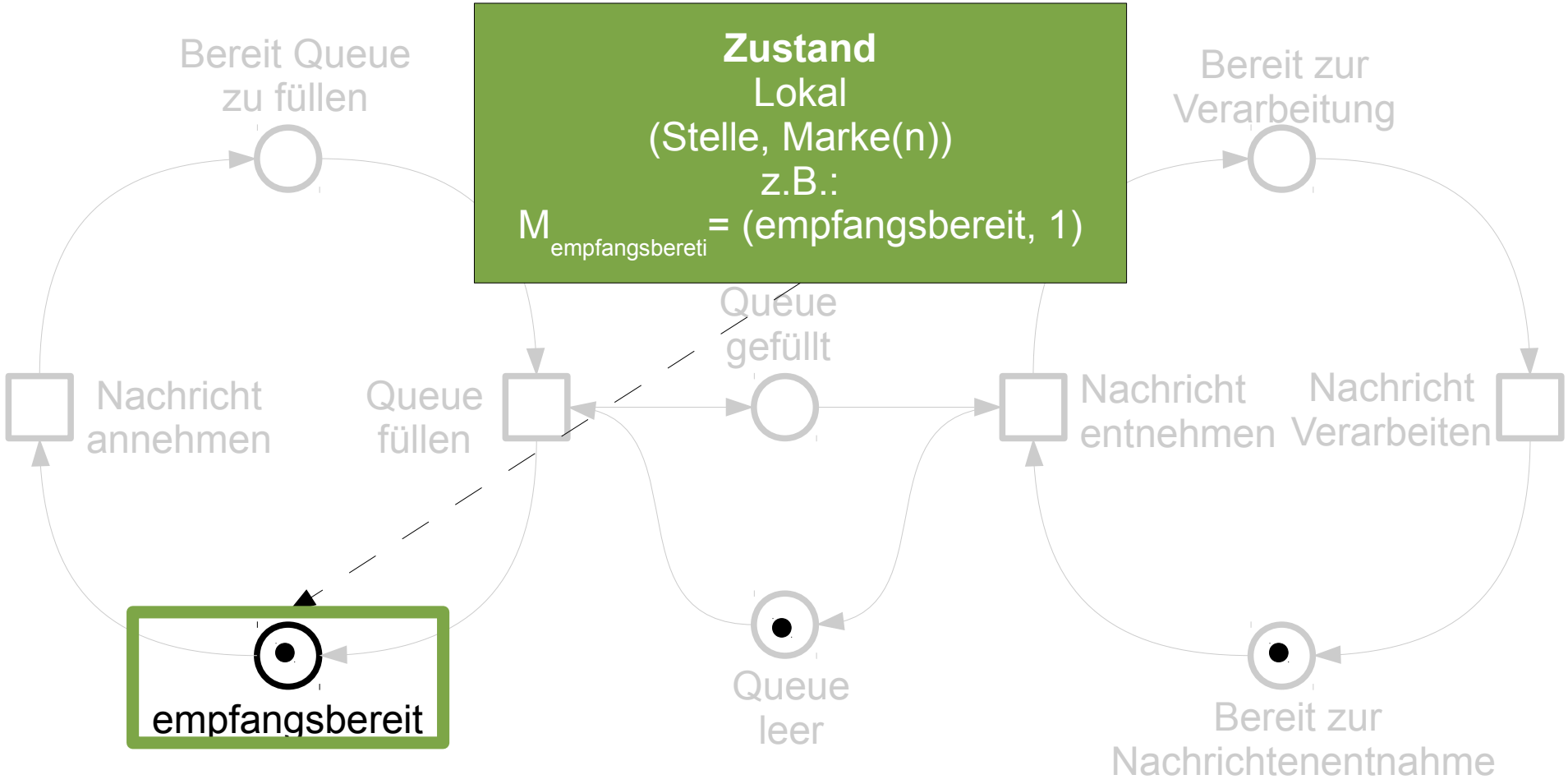
Beispiel Nachrichten-Queue



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

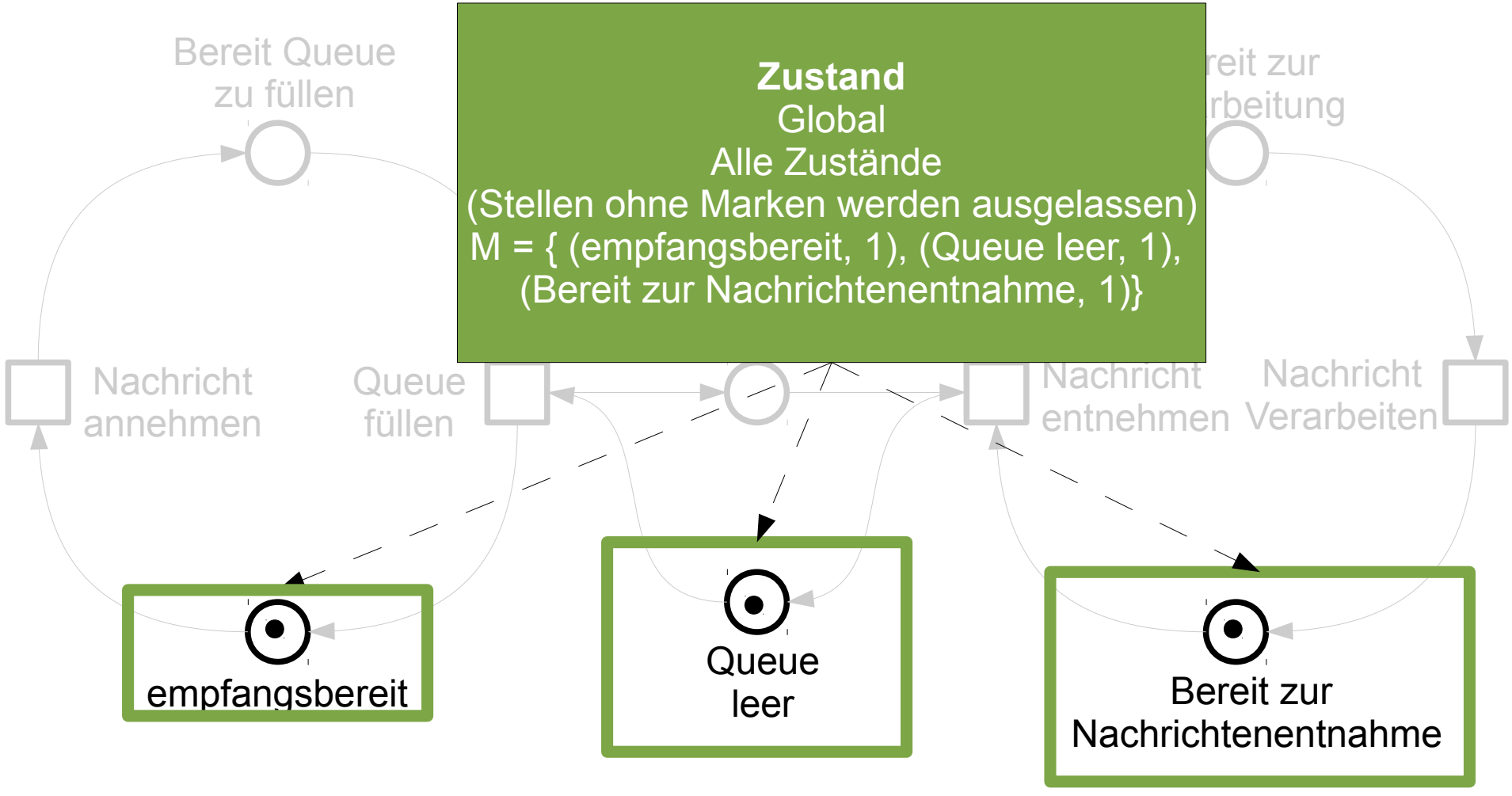
Zustand (lokal)

Beispiel Nachrichten-Queue



Zustand (global)

Beispiel Nachrichten-Queue



Definition Petri-Netz

Definition Petri-Netz:

Gegeben:

- Netz (S, T, F)
- W – Bogenvielfachheit (auch genannt „Gewicht“)
  - $W : F \rightarrow \mathbb{N} \setminus 0$
- M_0 – Globaler Startzustand (auch genannt „Anfangsmarkierung“)
  - $M_0 : S \rightarrow \mathbb{N}$

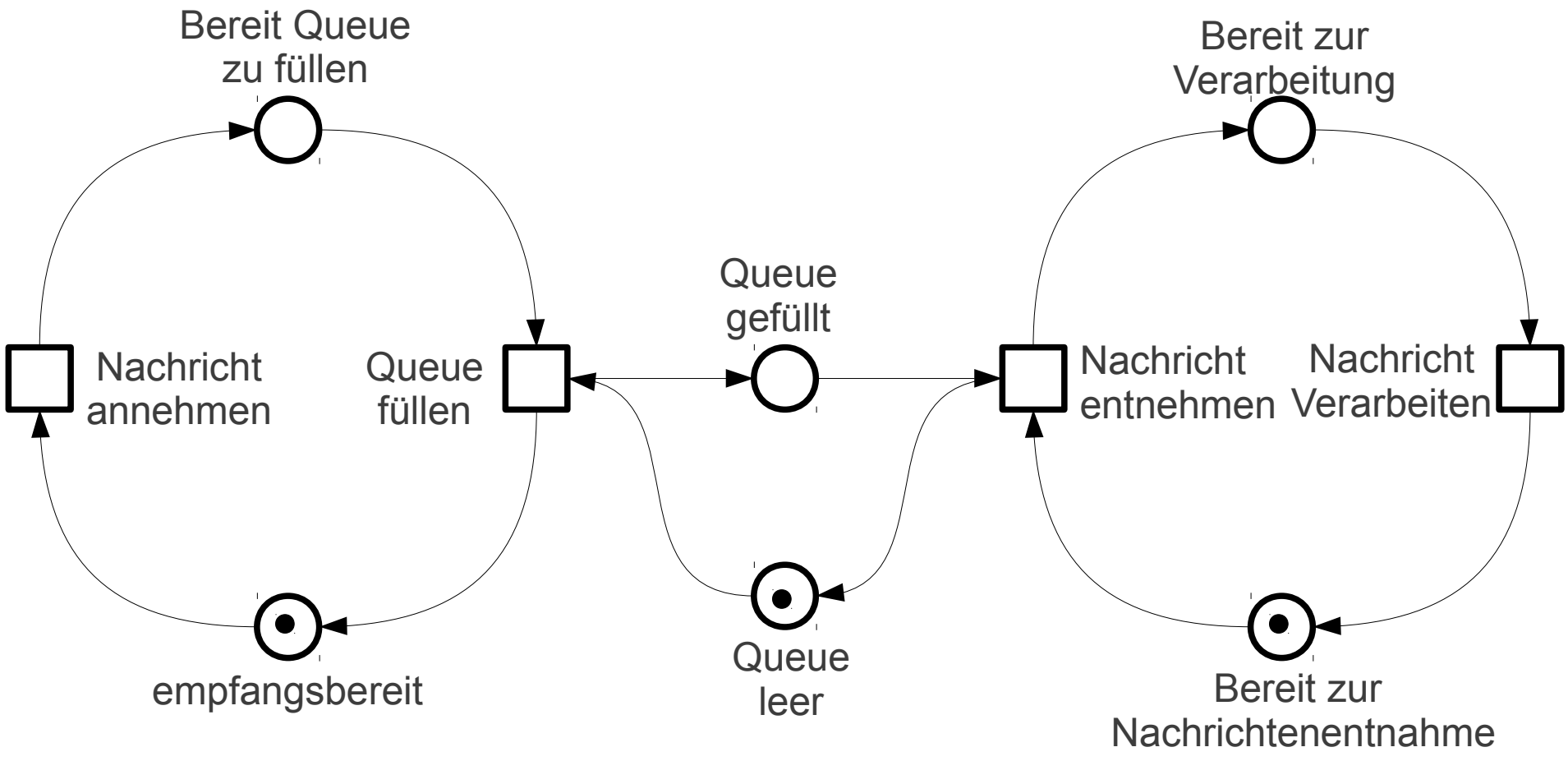
dann: (S,T,F,W,M_0) ein Petri-Netz.

Markierung: Verteilung der Marken auf die Stellen

Verhaltenssimulation: evolvierende Anzahl Marken pro Stelle betrachten (Zustandsbetrachtung - was kann im nächsten Zustand passieren ?).

Petri-Netz Ablauf: Beispiel

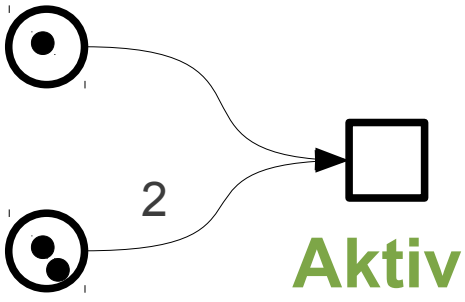
Beispiel Nachrichten-Queue

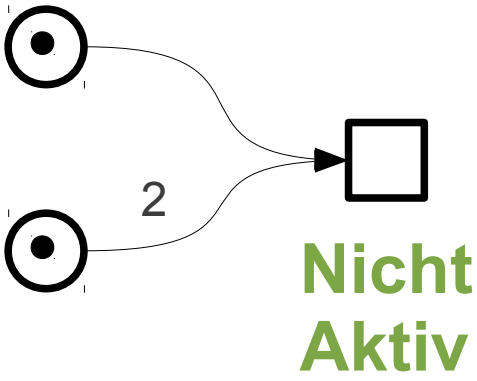


Ablauf: Aktivierung von Transition

Ablauf

- Transition t ist aktiviert, wenn:
  - $\forall\, p \in Vorgänger\ von\ t : W((p,t)) \leqslant m_p$

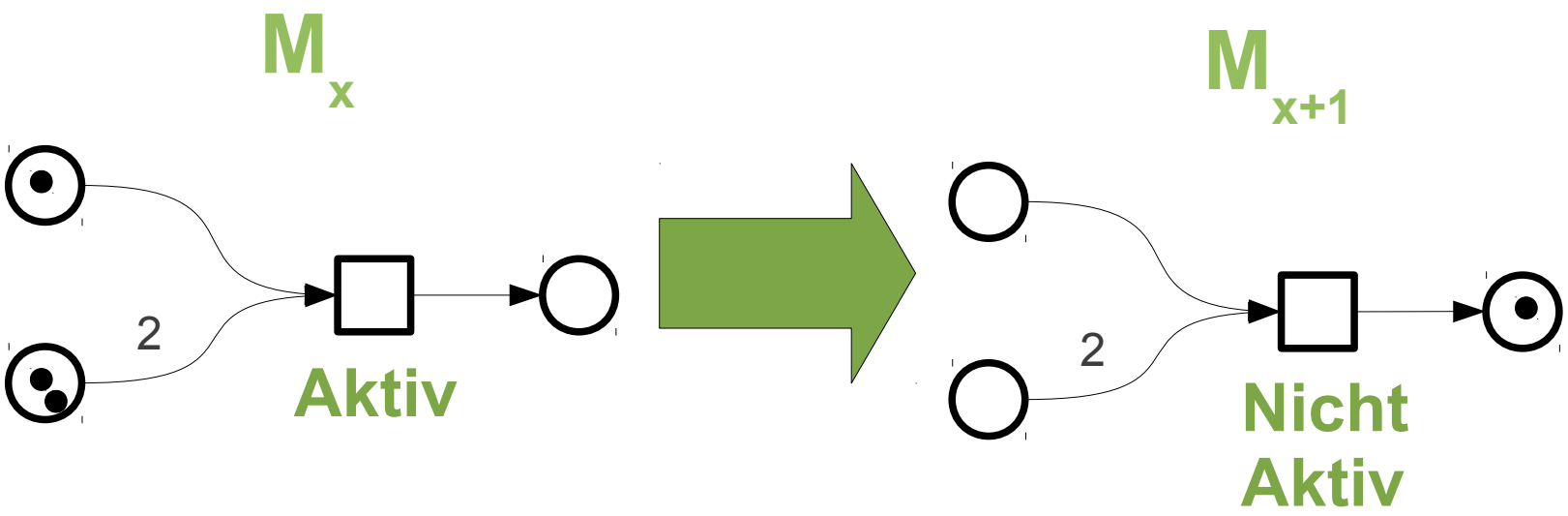




Ablauf: Schalten von Transition

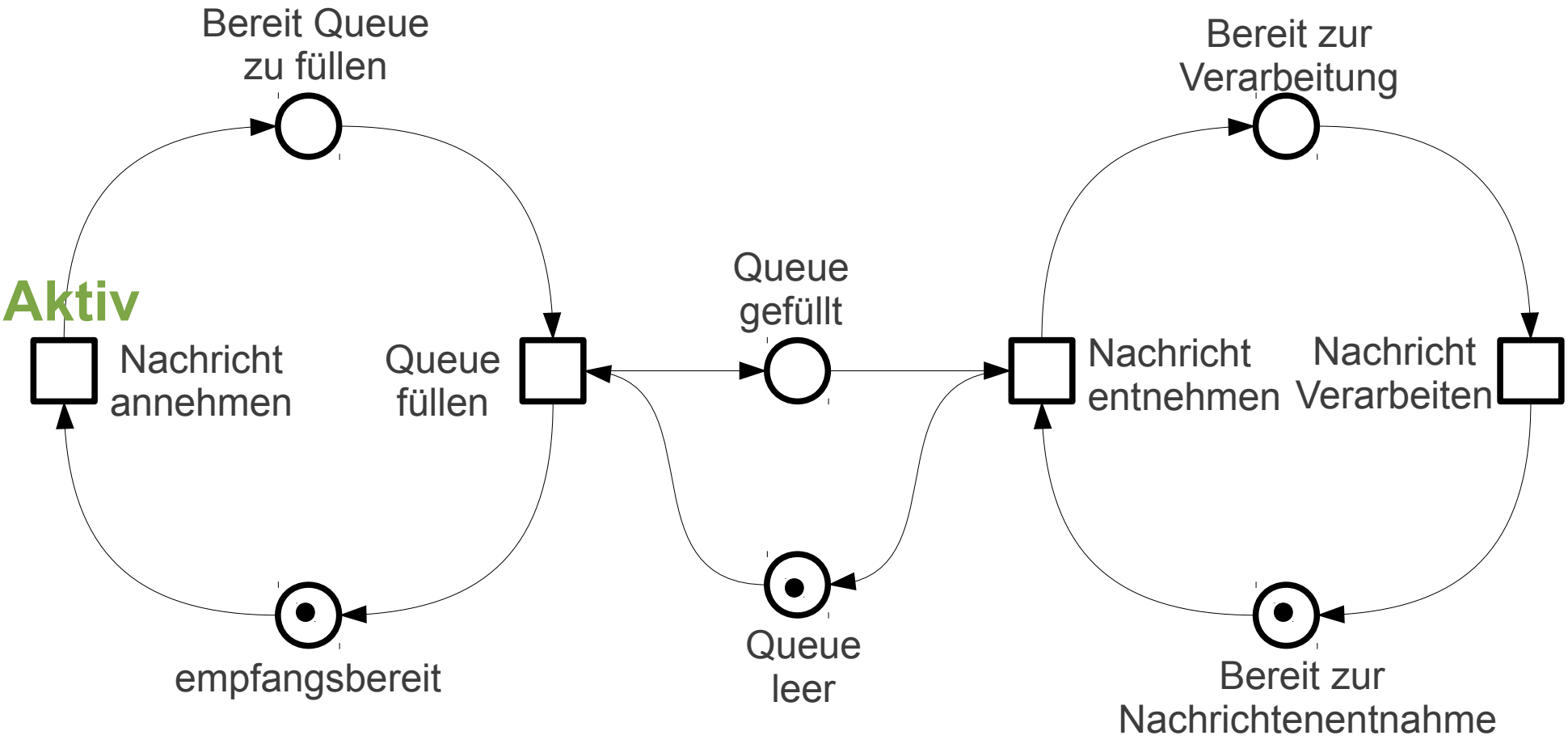
Ablauf

- Transition t ist aktiviert, wenn:
  - $\forall\, p \in Vorgänger\ von\ t : W((p,t)) \leqslant m_p$
- Eine der aktivierten Transitionen wird beim Übergang M_x nach M_{x+1} geschaltet (nicht-deterministische Auswahl).



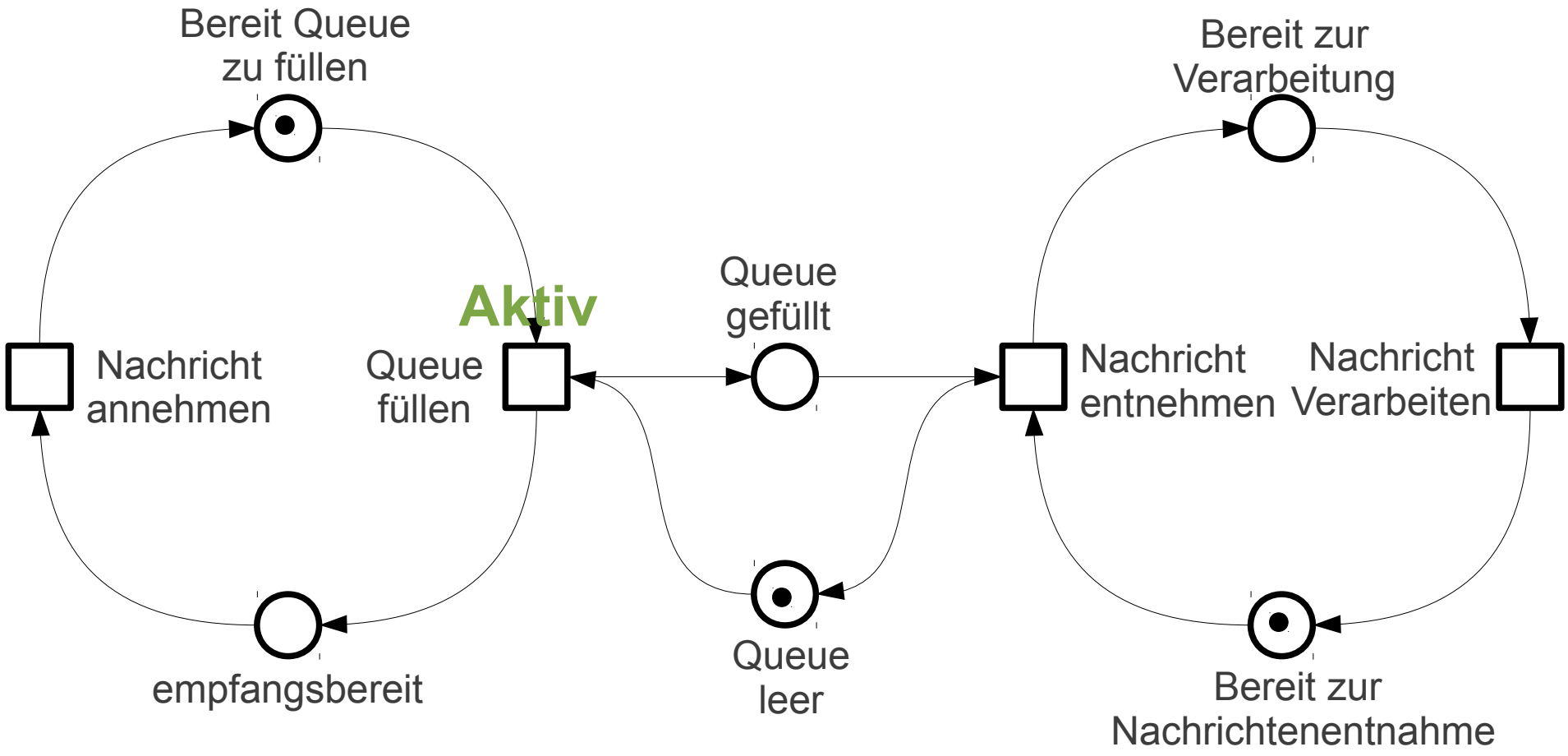
Ablauf: Beispiel (M_0)

Beispiel Nachrichten-Queue: M_0



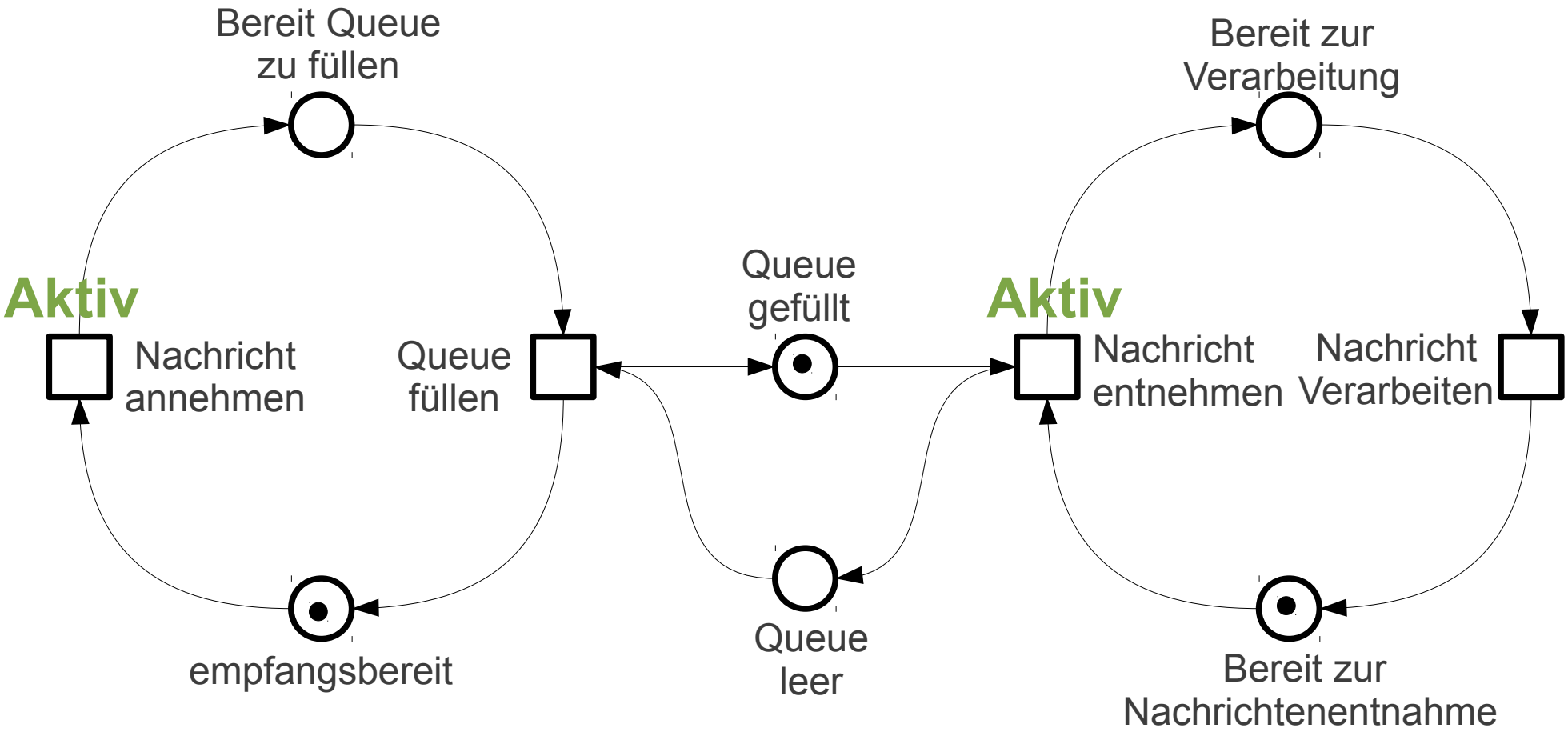
Ablauf: Beispiel (M_1)

Beispiel Nachrichten-Queue: M_1



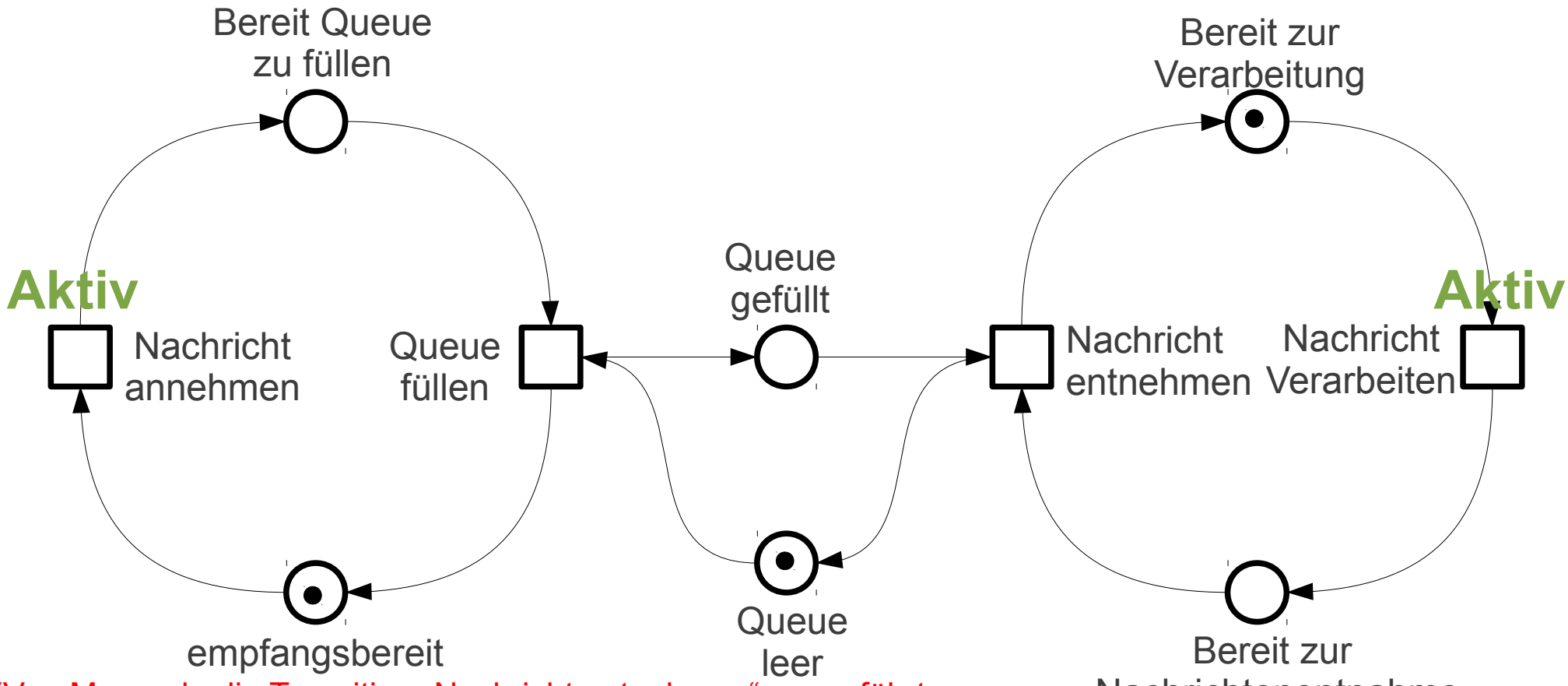
Ablauf: Beispiel (M_2)

Beispiel Nachrichten-Queue: M_2



Ablauf: Beispiel (M_3)

Beispiel Nachrichten-Queue: M_3



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

Frage

Was ist untypisch an der Nachrichten-Queue, wie sie bisher modelliert wurde?

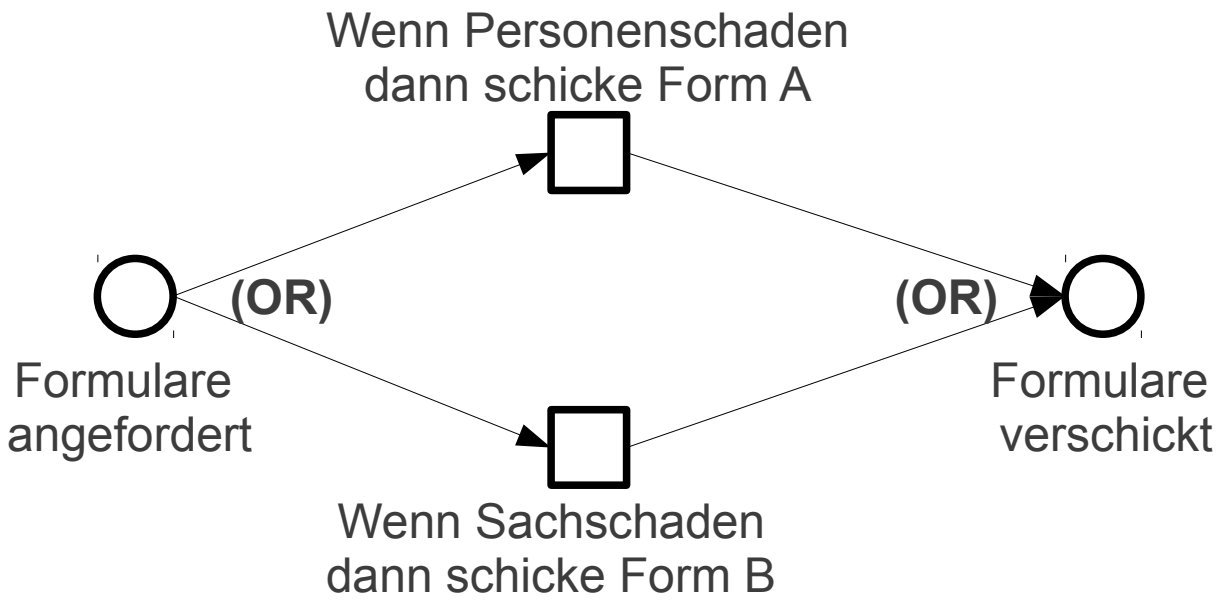
Frage

Was ist untypisch an der Nachrichten-Queue, wie sie bisher modelliert wurde?

Antwort: Es kann sich immer nur eine Nachricht im Buffer befinden. Die Transition „Queue füllen“ darf erst ausgeführt werden wenn die Stelle „Nachricht empfangen“ als auch die Stelle „Queue leer“ einen Marker besitzen.

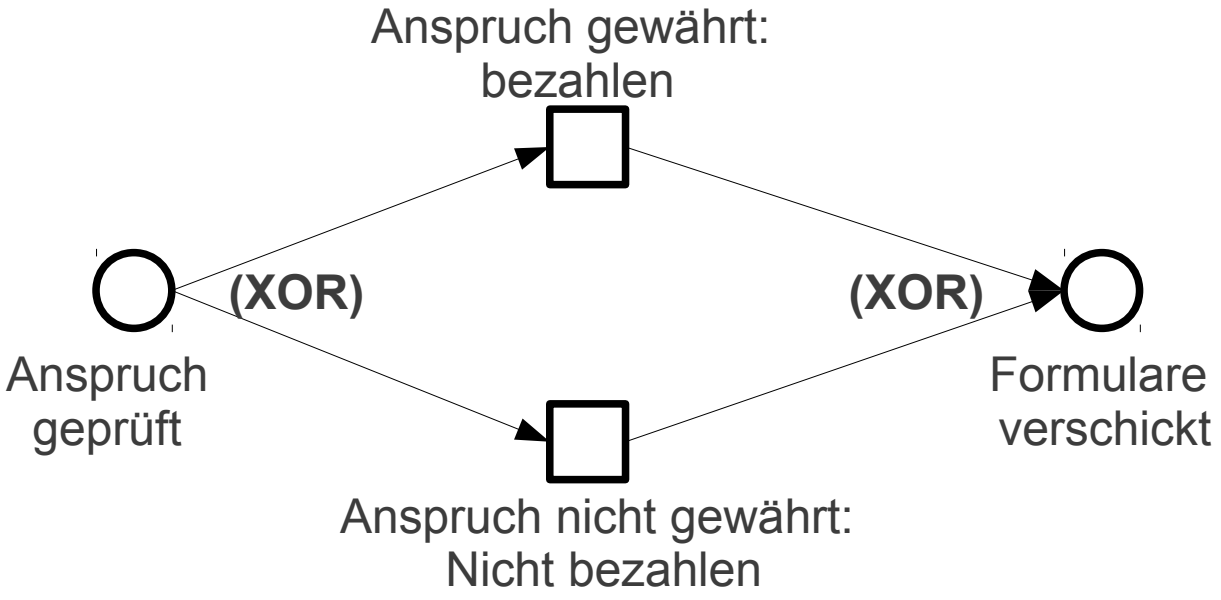
Verzweigungen an Stellen: OR

- Wenn mehrere Transitionen von einer Stelle ausgehend aktiviert sind, wird nicht-deterministisch entschieden, welche Transition ausgeführt wird (ggf. mehrere Transitionen (nacheinander), solange noch genügend Marker auf der Stelle vorhanden sind).
- Somit kann diese Situation implizit als „OR“ interpretiert werden.



Verzweigungen an Stellen: XOR, AND

- Wenn Kontext sicherstellt, dass die Stelle „Anspruch geprüft“ höchstens 1 Marker enthält, erhalten wir eine (implizite) XOR-Verzweigung. Kann mit sog. Kapazitäten erreicht werden (s. später).

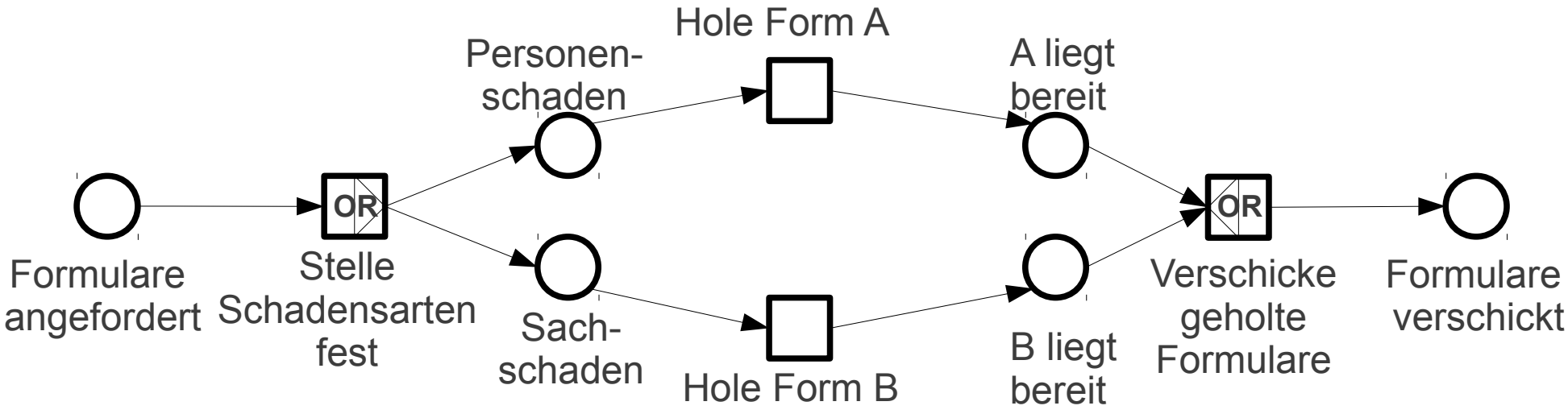


- Wenn an Stelle AND-Verzweigung intendiert ist, muss dies explizit annotiert werden (mit Auswirkungen auf die Petri-Netz-Semantik an dieser Stelle: beide Transitionen müssen ausgeführt werden).

Verzweigungen an Transitionen: AND, OR

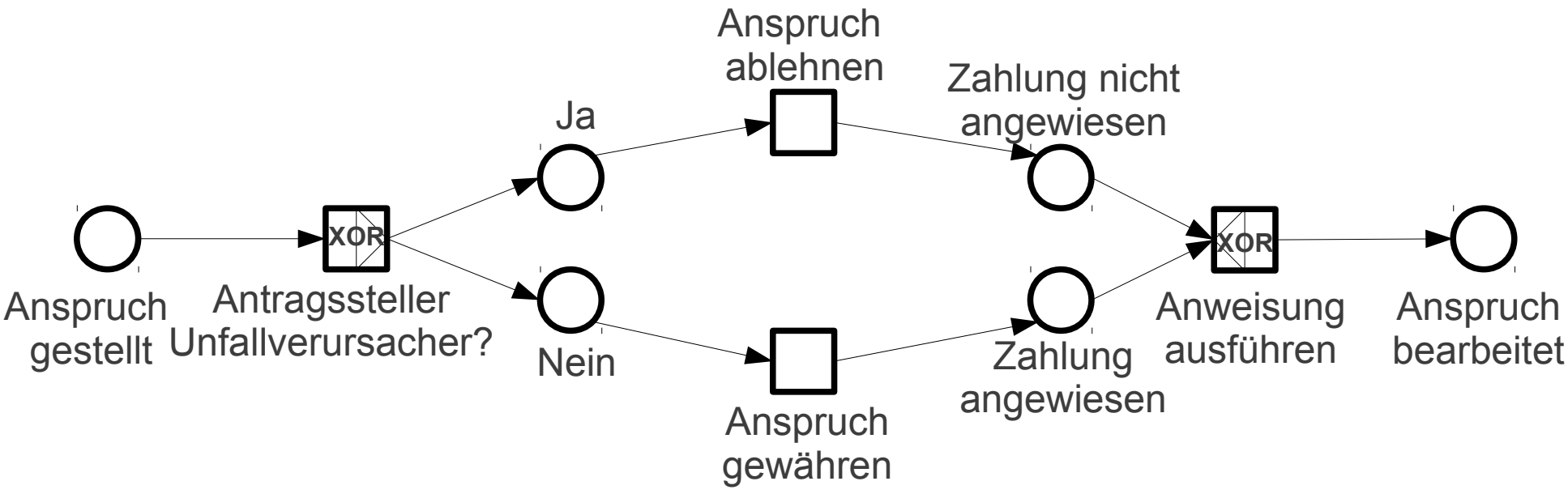
- Implizite Interpretation ist AND.
- Falls OR oder XOR intendiert sind, müssen diese explizit annotiert werden (mit der jeweiligen damit verbundenen Auswirkung auf die Petri-Netz-Semantik an dieser Transition).

Beispiel OR:



Verzweigungen an Transitionen: XOR

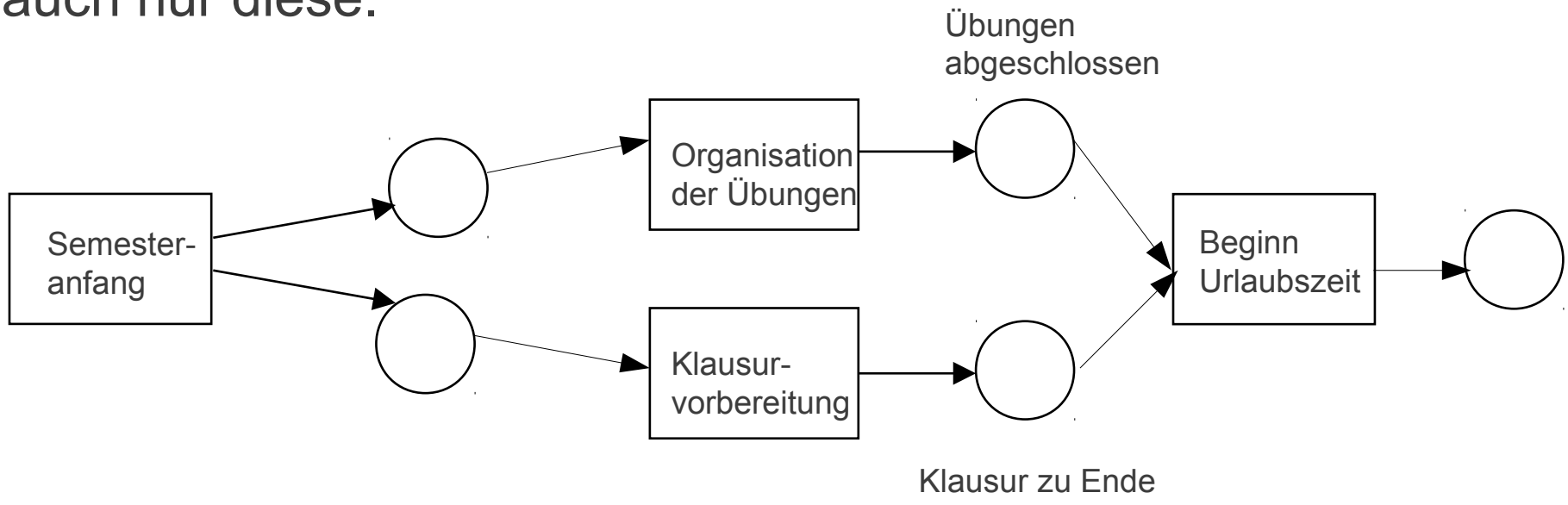
Beispiel XOR:



NB: Entscheidungs-Annotationen an Stellen bzw. Transitionen müssen zusammen passen (z.B.: ein Kontrollfluss, der mit AND verzweigt wurde, kann nicht durch OR wieder zusammengeführt werden).

Parallelität, Lokalität

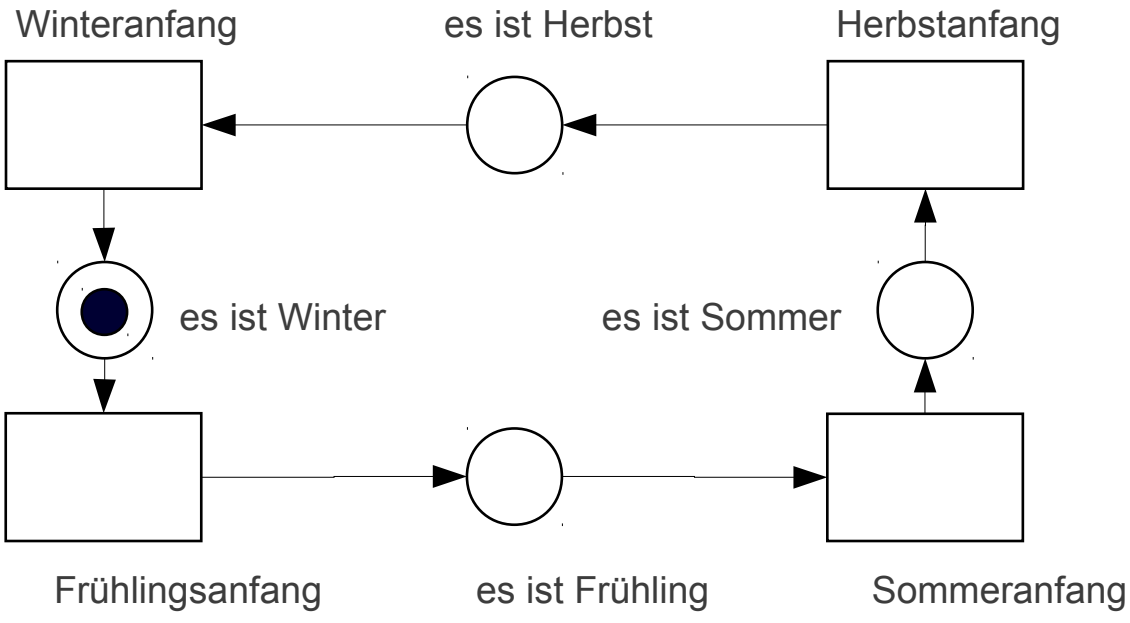
- Petrinetze erlauben einen natürlichen Begriff von Parallelität
- Lokalitätseigenschaft von Petrinetzen: Das Schalten einer Transition wird nur von ihrer direkten Umgebung beeinflusst und beeinflusst auch nur diese.



Ausführungs-Semantik

Formal definierte Semantik auf der Basis von Aussagenlogik

- Jede Stelle repräsentiert eine Aussage.
- s genau dann markiert, wenn durch s repräsentierte Aussage wahr.
- Das Eintreten eines Ereignisses t macht die Aussagen im Vorbereich von t falsch und im Nachbereich von t wahr.



Aussage:
- Es ist Winter.
- Es ist nicht Herbst.
- Es ist nicht Sommer.
- Es ist nicht Frühling.

Abgeleitete Aussagen:
- Es ist entweder Winter oder
- Frühling oder Sommer oder
- Herbst !

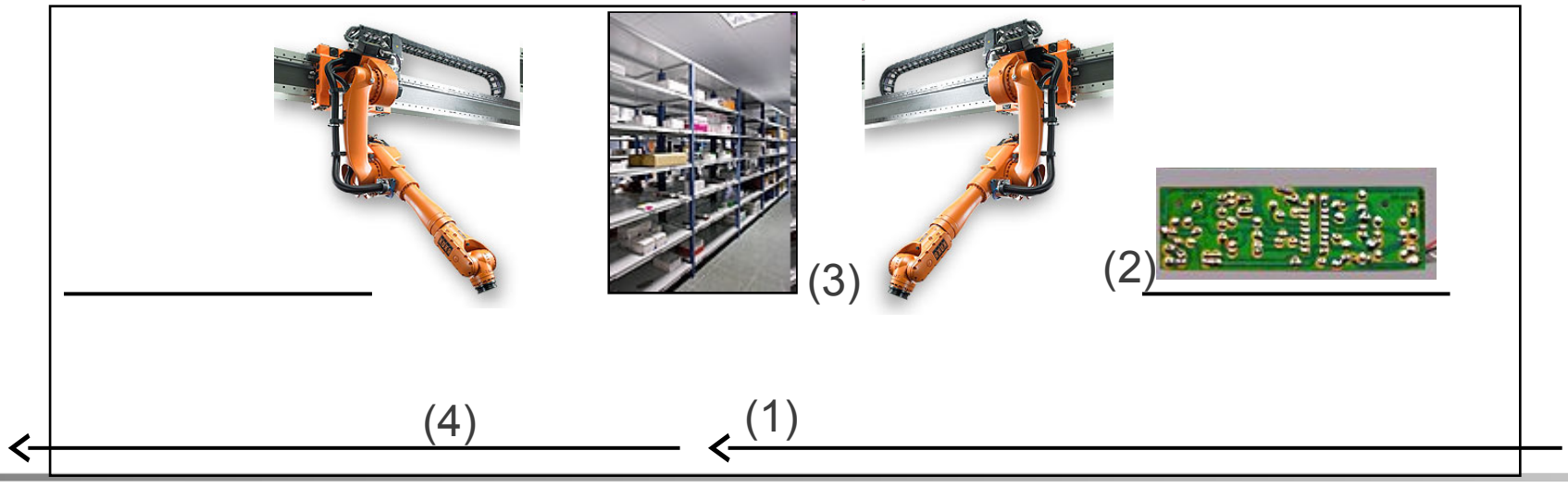
Kapazitäten

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

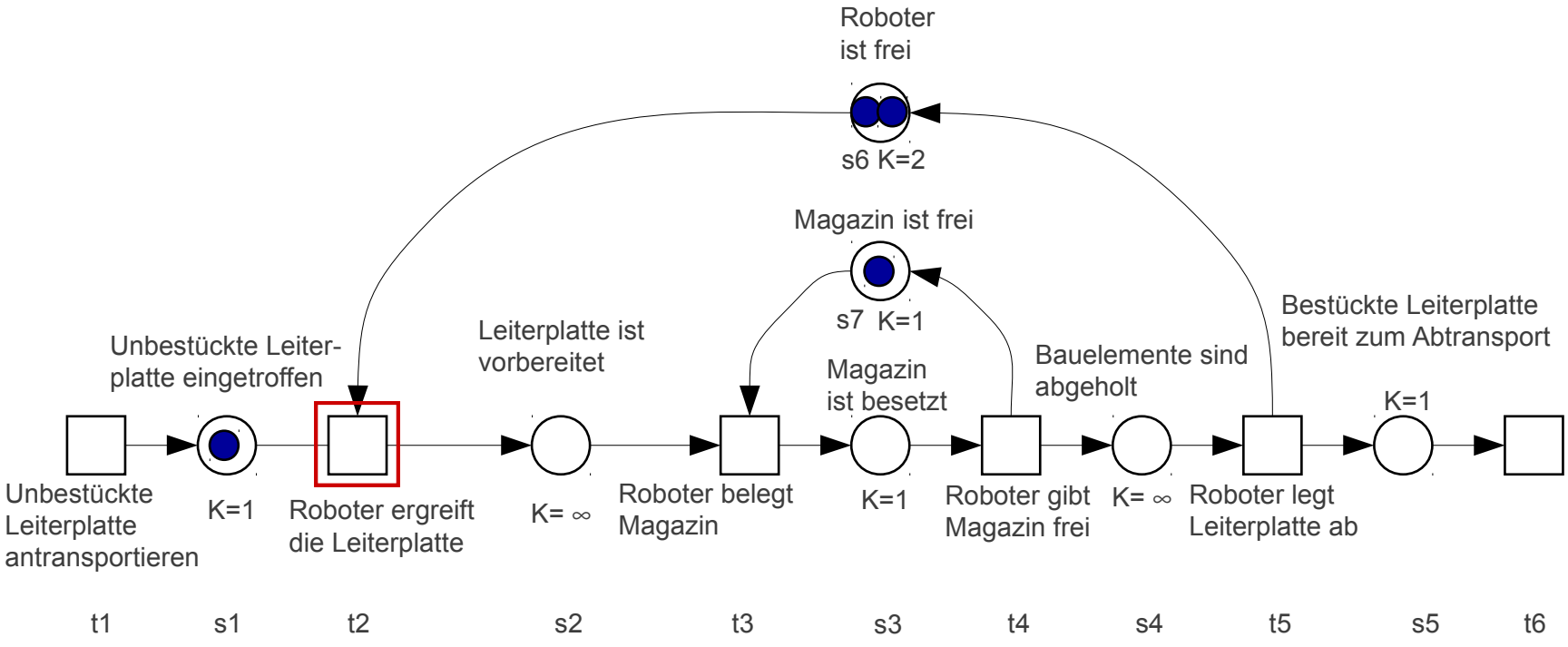
Beispiel: S/T-Netz Bestückungsroboter

Zwei Roboter bestücken Leiterplatten mit elektronischen Bauelementen:

- Leiterplatten werden auf Fließband antransportiert (1).
- Ist einer der Roboter frei, nimmt er die Leiterplatte vom Fließband (2).
- Sind beide Roboter frei, wird nichtdeterministisch entschieden, wer die Leiterplatte nimmt.
- Nur jeweils ein Roboter darf auf Bauelemente-Magazin zugreifen (3), um die Leiterplatte mit Bauelementen bestücken zu können.
- Nur jeweils eine Leiterplatte wird zu einem Zeitpunkt abtransportiert (4).

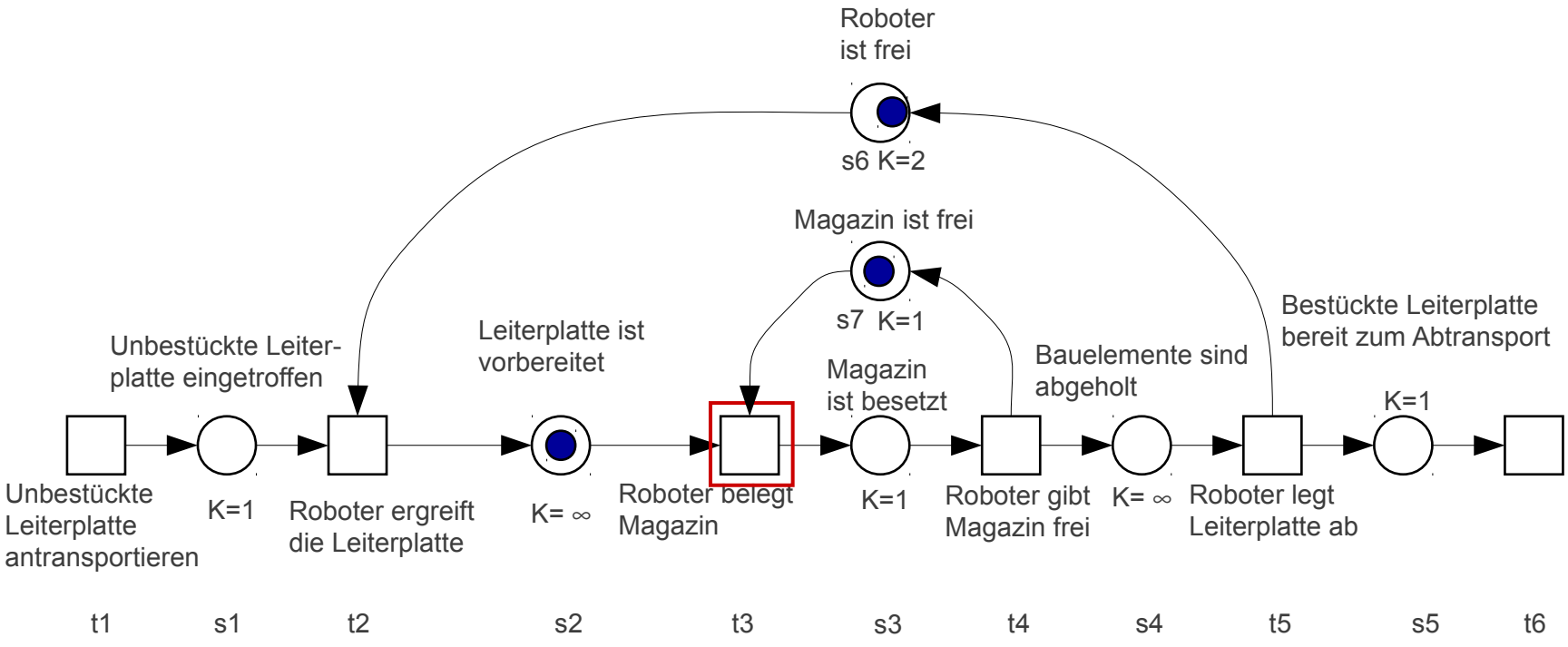


Beispiel: S/T-Netz Bestückungsroboter

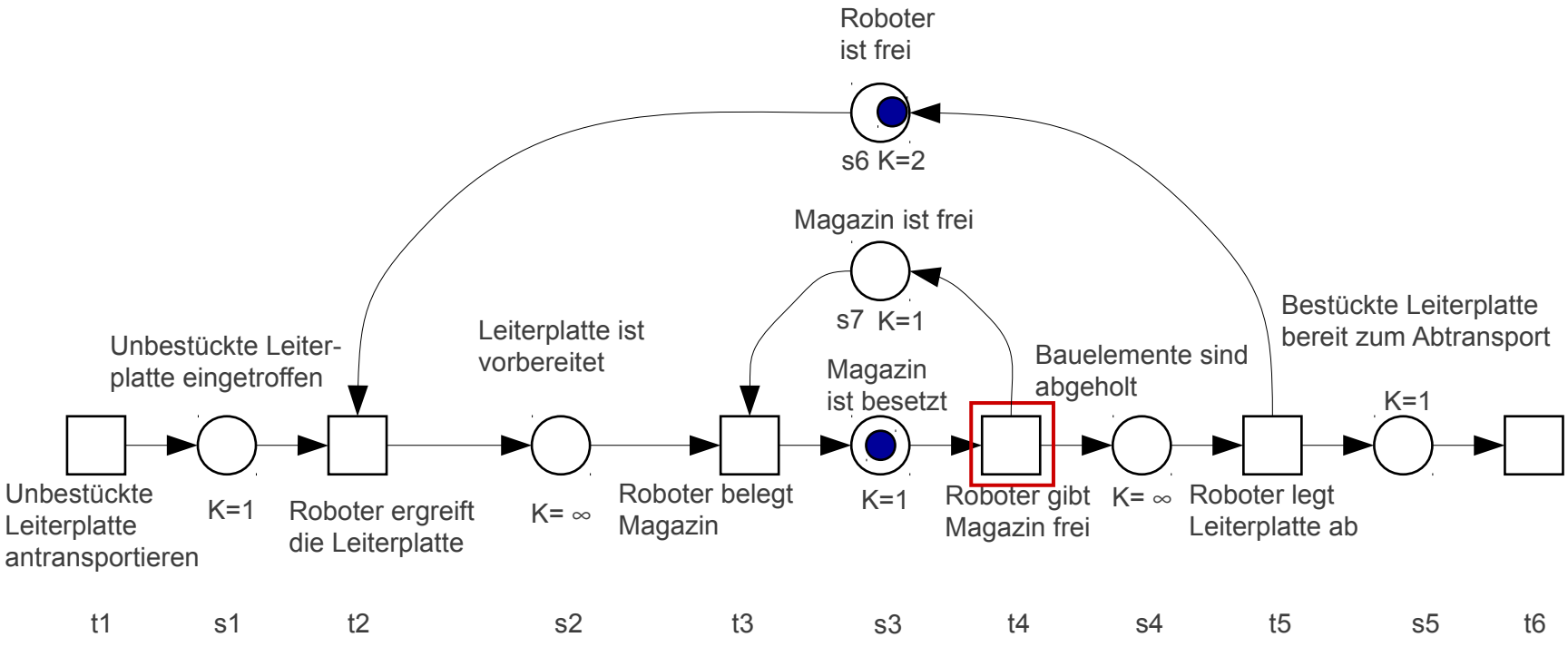


bezeichnet die aktivierte Transition.

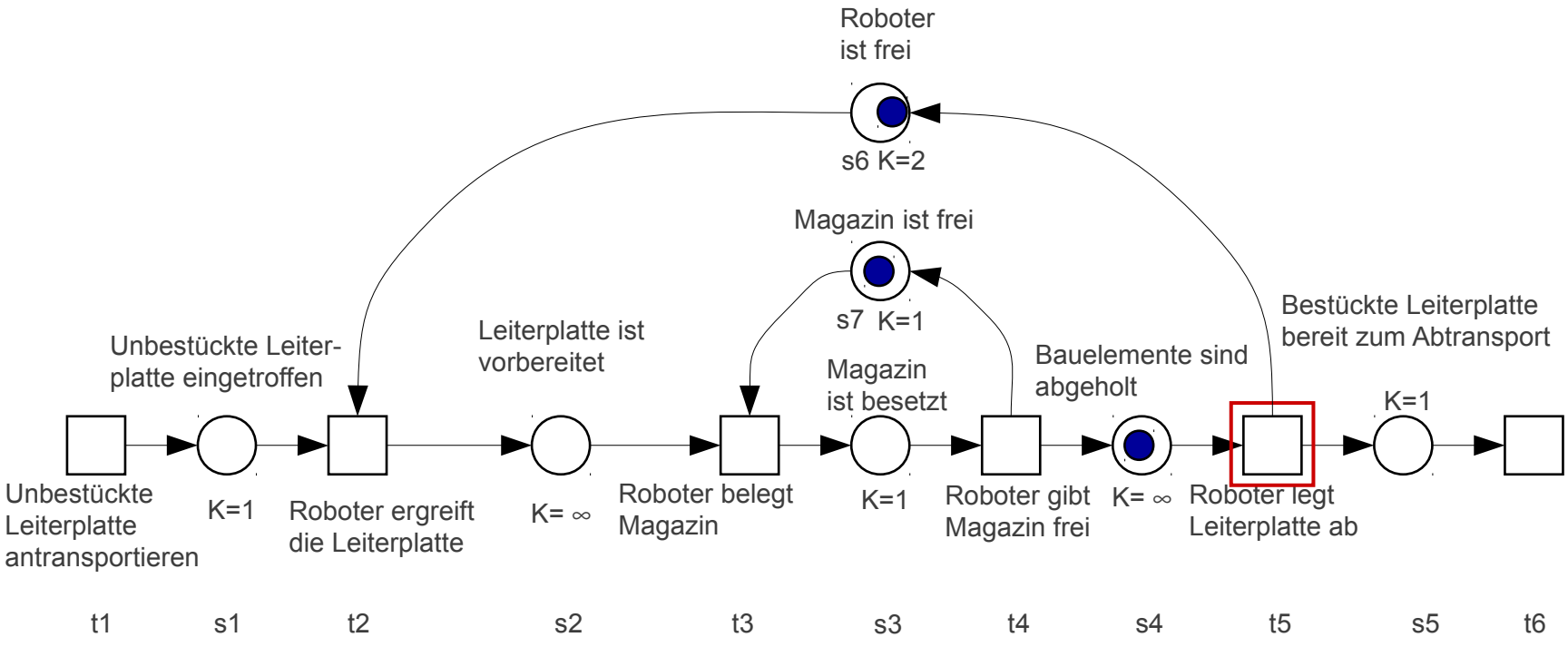
Beispiel: S/T-Netz Bestückungsroboter



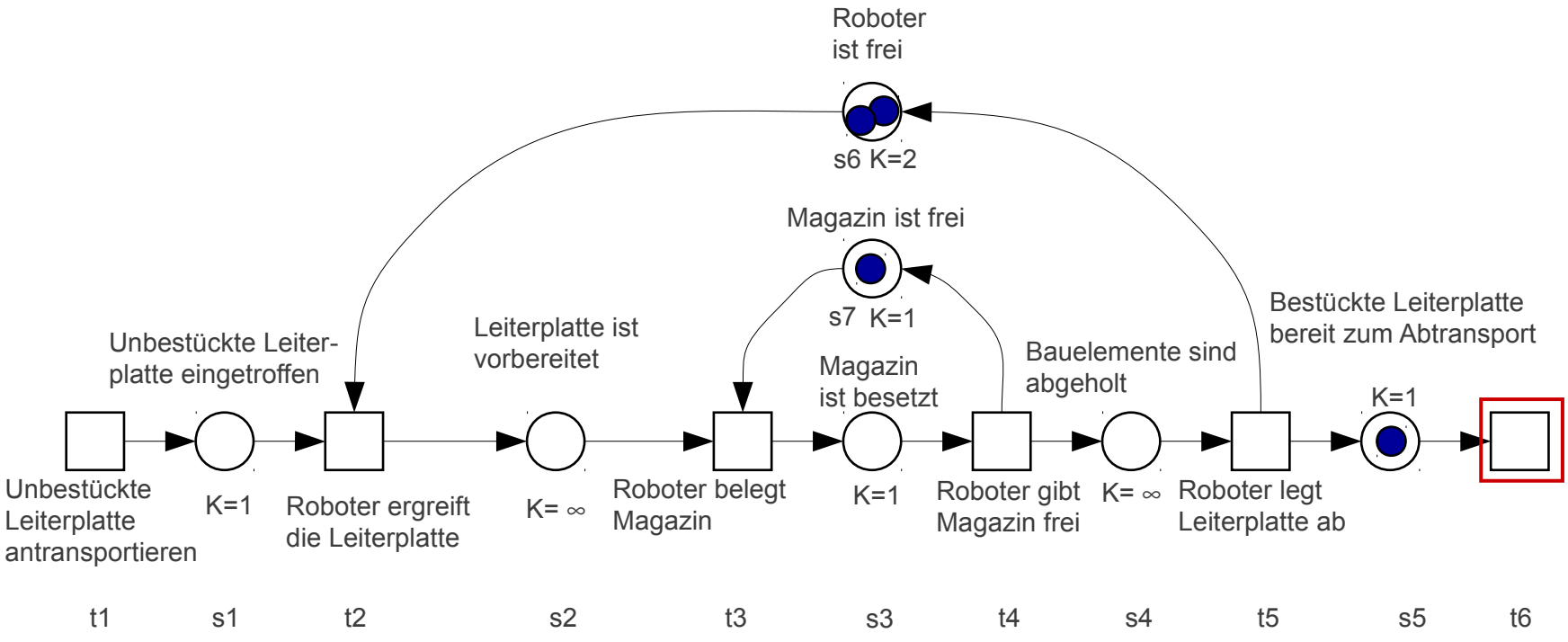
Beispiel: S/T-Netz Bestückungsroboter



Beispiel: S/T-Netz Bestückungsroboter



Beispiel: S/T-Netz Bestückungsroboter



Petri-Netz Ablauf (formal)

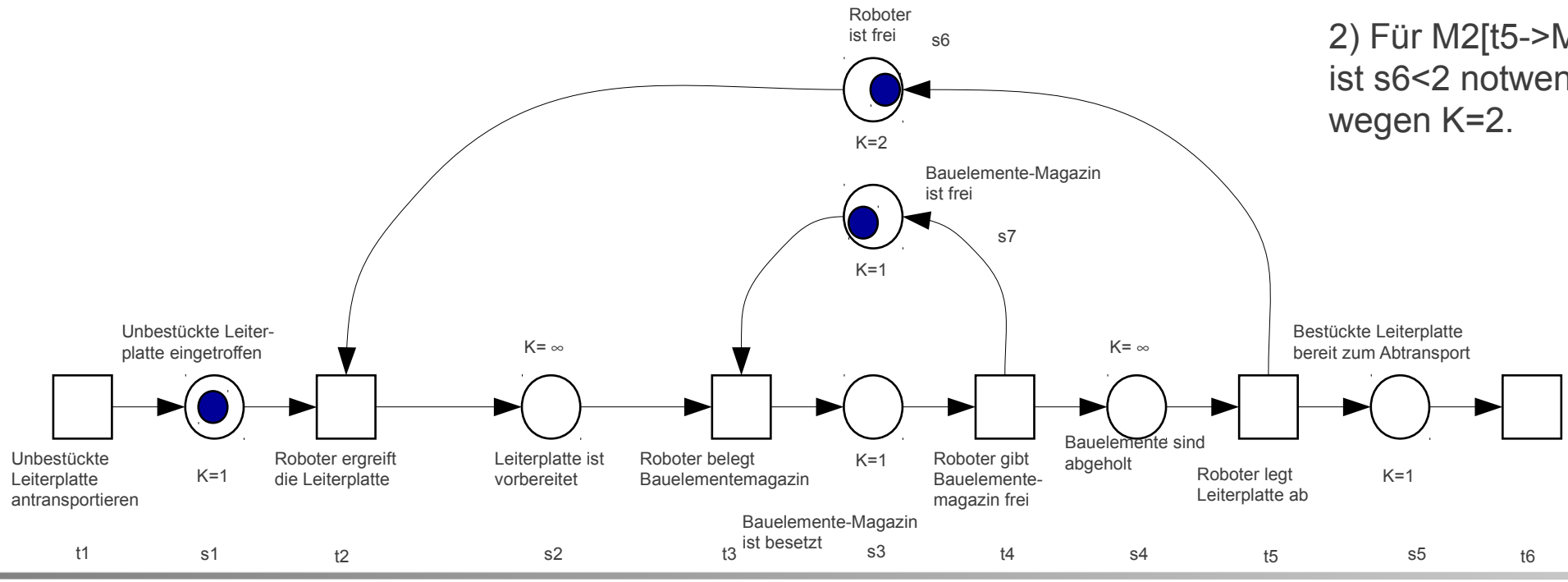
- Die initiale Markierung ist Bestandteil des Anfangszustandes eines Netzes.
  - Basierend auf der initialen Markierung werden aktivierte Transitionen ermittelt, deren Schalten dann zu Folgemarkierungen führen.
    Falls bei einer Markierung M die Transition t aktiviert ist, schreiben wir: **M [t>**

    Die Folgemarkierung M' zur Markierung M nach Schaltung der Transition t schreiben wir als: **M [t> M'** (die Klammern [> symbolisieren einen Pfeil).

    Bei mehreren aktivierten Transitionen wird nicht-deterministisch entschieden, welche als nächstes geschaltet wird. **Jede Folgemarkierung (= Folgezustand) ergibt sich aus dem Schalten jeweils genau einer Transition.**
  - Unter den Folgemarkierungen sind dann (möglicherweise) wieder andere Transitionen aktiviert. Dies setzt sich fort, bis kein Transitionen mehr aktiviert ist. Eine Liste von Transitionen [t1,t2,...,tn] heißt "nebenläufig aktiviert" unter einer Markierung M, wenn alle Permutationen als Schaltfolgen aktiviert sind. Schreibweise: **M [{t1, t2, ..., tn}>**
- Eine Markierung, unter der kein Transitionen aktiviert ist, heißt tote Markierung.
- Token, Marke = Element, das durch eine Markierung einer Stelle zugeordnet wird.
- **$[M_0>$** := $\{M \mid \exists\, w \in T^* \text{mit } M_0[w>M\}$ heißt **Erreichbarkeitsmenge** des Systems.

Erreichbarkeitsmenge: 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->M4 t5->M5



NB:

1) Wir wählen hier bewusst ein anderes M0 als auf F.39, um nicht-deterministische Verzweigungen betrachten zu können.

2) Für M2[t5->M5 ist s6<2 notwendig wegen K=2.

Erreichbarkeit

Erreichbarkeitsalgorithmus (breadth-first)

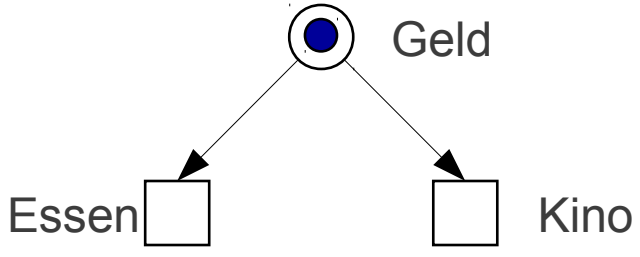
1. Trage in ein Schema mit den Spalten „Markierungsnummer“, „Markierung“ und „Schaltungen“ die Anfangsmarkierung M_0 ein.
2. In der aktuellen Markierung M_i wird jede Transition t untersucht, ob sie aktiviert[1] ist:
   - Falls t aktiviert ist: Berechne Folgemarkierung.
     - Stelle fest, ob diese Folgemarkierung bereits als eine Markierung M_j benannt wurde.
     - Wenn nicht: Benenne Folgemarkierung M_j (für ein neues j>i) und lege eine 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. Sind alle Transitionen überprüft, gilt M_i als erledigt.
4. Prüfen, ob alle eingetragenen Markierungen erledigt sind
   - Ja: Erreichbarkeitsanalyse abgeschlossen
   - Nein: Überprüfe die nächste Markierung und fahre bei 2 fort.

[1] Aktivierte Transitionen müssen insbesondere Kapazitäten respektieren (F. 37).

Grundsituationen in S/T-Netzen: Konflikt

Konflikt:

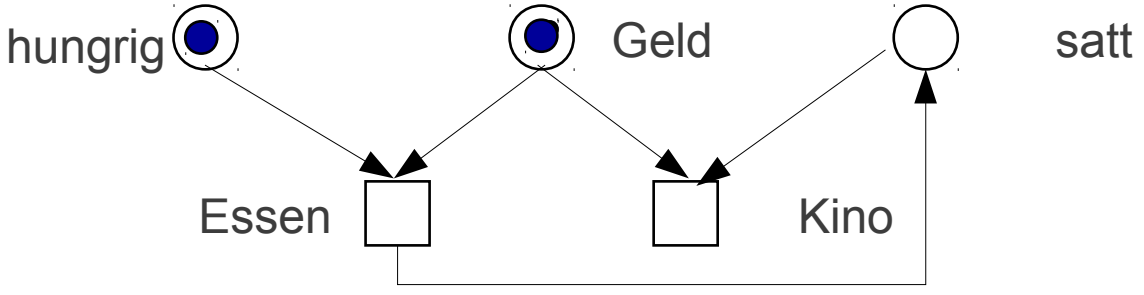
- Nicht-nebenläufige gleichzeitige Aktiviertheit von Transitionen.
- Konfliktierende Transitionen nehmen sich gegenseitig Input-Marken weg.
- Def.: Zwei Transitionen t_1 und t_2 eines S/T-Systems mit Höchstbelegung $K=\infty$ $\forall s \in S$ sind im *Konflikt*, wenn t_1 und t_2 aktiviert sind (geschrieben: $M[t_1>$ und $M[t_2>$), aber $\{t_1, t_2\}$ nicht nebenläufig aktiviert sind (d.h. es gilt nicht $M[\{t_1, t_2\}>$).
- Die Menge kft(t1,M) ist die Menge der mit t1 konfliktierenden Transitionen.



Grundsituationen in S/T-Netzen: Konfliktlösung

Konfliktlösende Einbettung

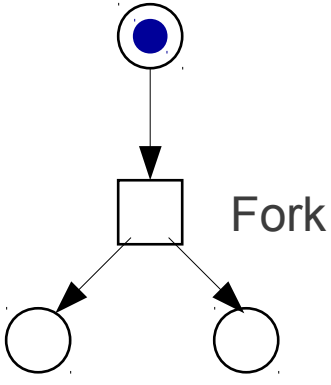
- Ist durch Einführung zusätzlicher Eingabestellen möglich.
- Hierdurch wird Information über zu wählende Alternative ergänzt

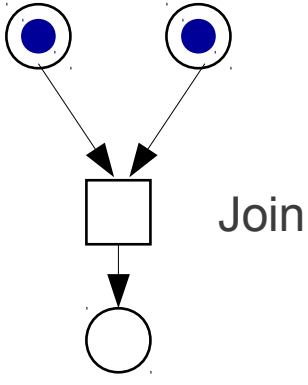


Grundsituationen in S/T-Netzen: Synchronisation

Synchronisation

- Einführen von Abhängigkeiten zwischen Transitionsfolgen, d.h. Wegnahme von Nebenläufigkeit
- Beispiel: Fork-Join-Synchronisation zur Aufspaltung und Zusammenführung eines Ereignisstroms





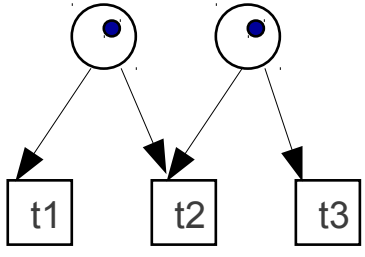
Grundsituationen in S/T-Netzen: Konfusion

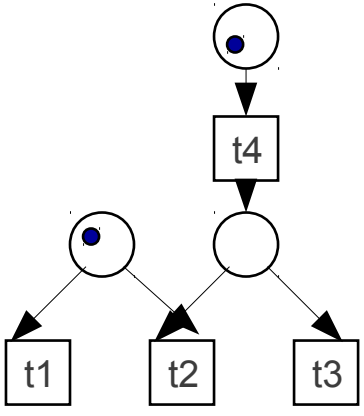
Konfusion:

Ein Tripel (M, t1 ,t3) nennt man eine Konfusion, wenn t1 ≠ t3 und

- kft(t1,M) ≠ kft(t1,M[t3>)
  (d.h. Ausführung von t3 ändert die Konfliktmenge von t1)

  Dabei gibt es zwei mögliche Fälle:
- Konfliktvergrößerung:
  kft(t1,M) ⊆ kft(t1,M[t3>)
- Konfliktverminderung:
  kft(t1,M[t3>) ⊆ kft(t1,M)

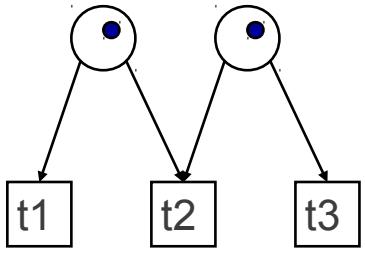




Grundsituationen in S/T-Netzen: Konfusion

Symmetrische Konfusion:

$kft(t1,M) \setminus kft(t1,M[t3>) \neq \emptyset$



Konfliktbeseitigung

kft(t1,M)=t2

kft(t2,M)={t1,t3}

kft(t3,M)=t2

kft(t1,M)=t2
$\neq$kft(t1,M[t3>)=$\emptyset$

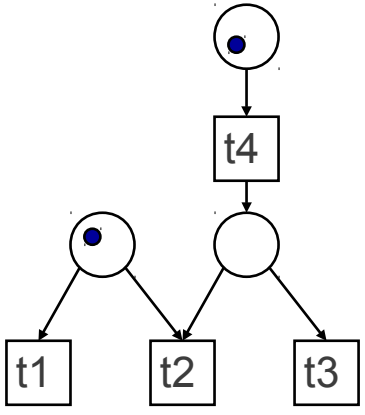
kft(t1,M) $\supseteq$ kft(t1,M[t3>)

$\rightarrow$ Konfliktverminderung

$\rightarrow$ symmetrische Konfusion

Asymmetrische Konfusion

$kft(t1, M[t3>) \setminus kft(t1,M) \neq \emptyset$



Konflikt-
einführung

kft(t1,M)= $\emptyset$

kft(t2,M)= $\emptyset$

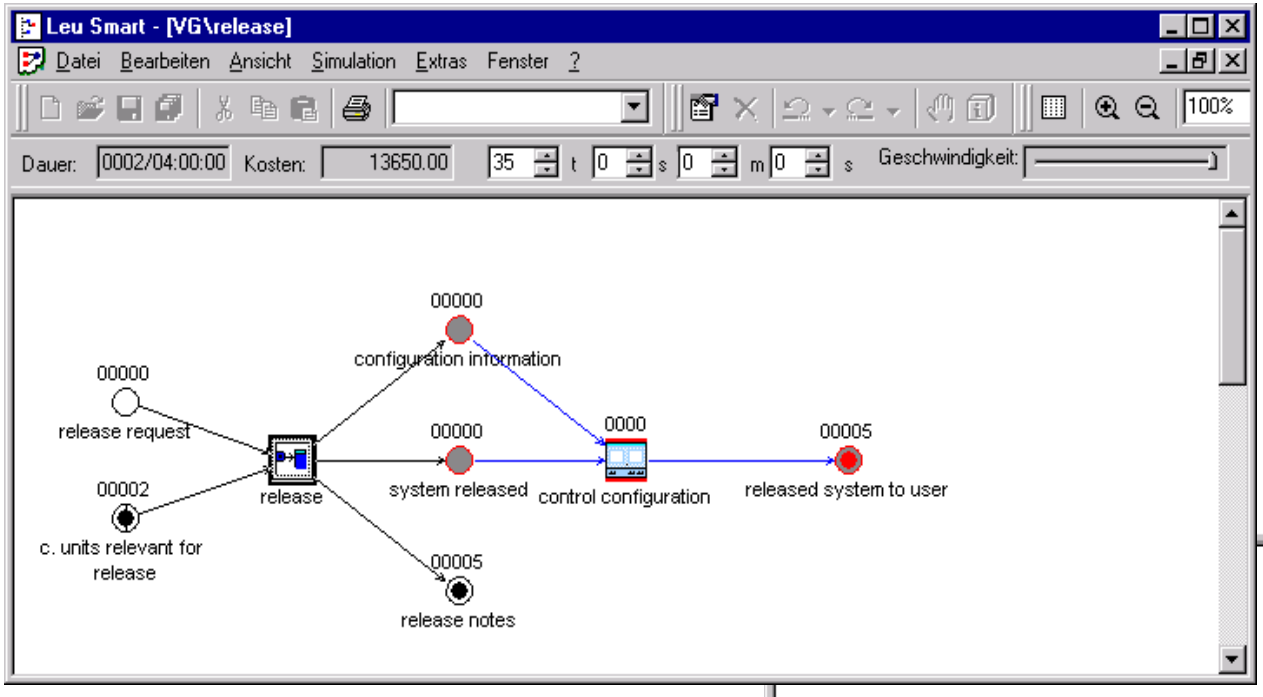
kft(t1,M)= $\emptyset \neq$
kft(t1,M[t4>)=t2

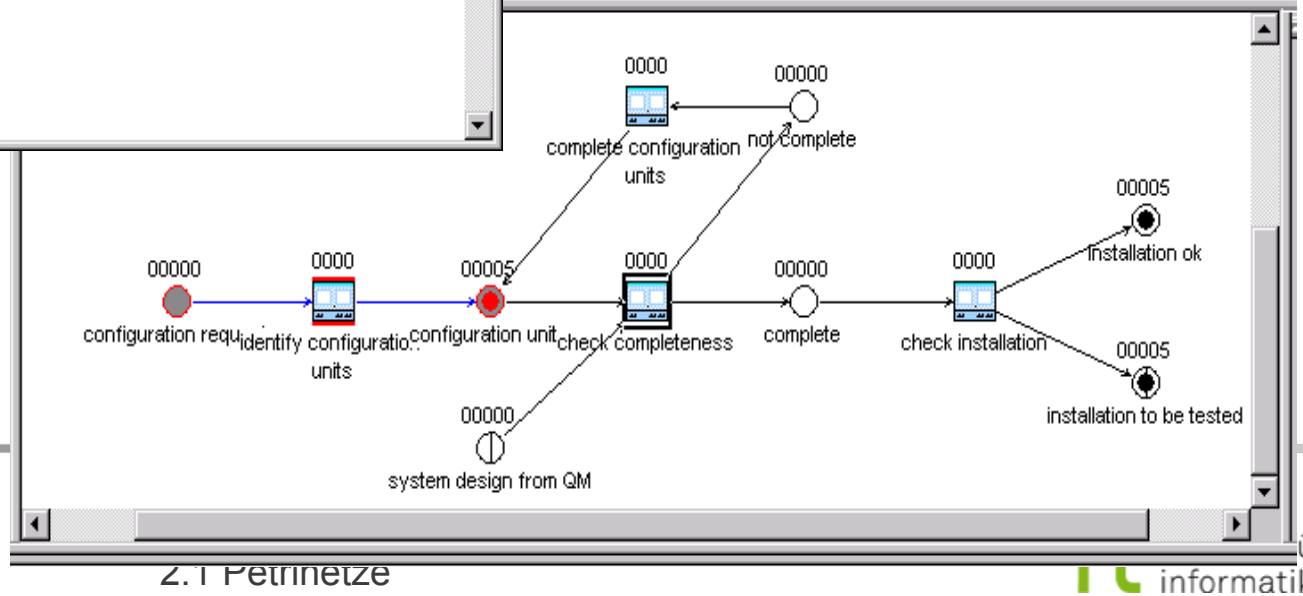
kft(t1,M) $\subseteq$ kft(t1,M[t2>)

$\rightarrow$ Konfliktvergrößerung

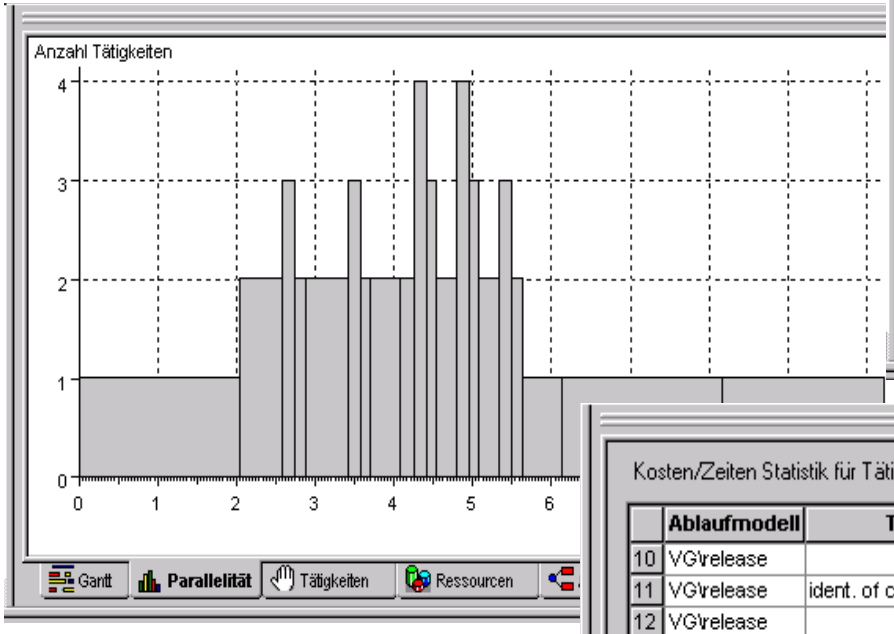
$\rightarrow$ asymmetrische Konfusion

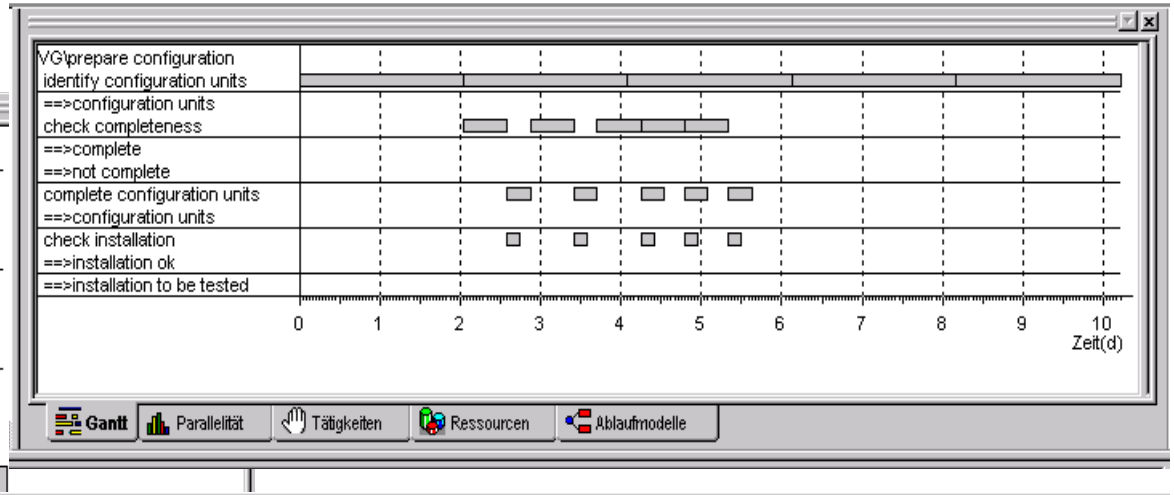
Simulation von Petrinetzen: Animation





Simulation von Petrinetzen: Analyse von Simulationsläufen





Kosten/Zeiten Statistik für Tätigkeiten

	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	

Exportieren... Text-Datei

Gantt | Parallelität | Tätigkeiten | Ressourcen | Ablaufmodelle

Analyse von Systemen: Simulation

- Simulation kann nicht zeigen, dass bestimmte Situationen nicht auftreten, da Simulationen immer nur einen Ausschnitt aus der Menge aller möglichen Verhalten darstellen.
- Simulation kann zeigen, dass bestimmte Situationen auftreten können, sagt aber nichts über deren Eintrittswahrscheinlichkeit.

Beweis von Eigenschaften

- Statische Eigenschaften: solche, die unabhängig von Markierungen sind und nur von der Netztopologie abhängen.
  - Verklemmungen / Deadlocks
  - Traps
- Dynamische Eigenschaften: solche, die von der Menge der erreichbaren Markierungen abhängen.
  - Standardhilfsmittel: Erreichbarkeitsgraphen (s.o.)

Analyse von Systemen: (S/T-)Systemeigenschaften

- Analyse von (S/T-)Systemeigenschaften
  - Annahmen zur Vereinfachung: Betrachtete S/T-Systeme sind endlich, schlicht und schwach zusammenhängend
- Untersuchte Eigenschaften
  - Sicherheit
  - Lebendigkeit

Analyse von Systemen: Sicherheit

Sicherheit:

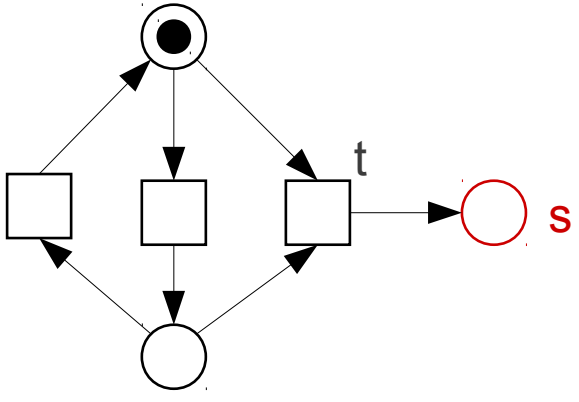
- Seien $Y=(S,T,F,K,M_0)$ ein S/T-System und $B: S \rightarrow \mathbb{N}_0 \cup \{\infty\}$ eine Abbildung, die jeder Stelle eine „kritische Markenzahl“ zuordnet. Y heißt **B-sicher** bzw. **B-beschränkt**, wenn für alle erreichbaren Markierungen die Anzahl der Markierungen pro Stelle durch B begrenzt ist, d.h.:

  $$\forall M \in [M_0>, s \in S: M(s) \leq B(s).$$

- Man spricht von 1-sicher, 2-sicher usw., wenn B=1, B=2 usw. Y heißt beschränkt, wenn es eine natürliche Zahl b gibt, für die Y b-sicher ist.
- Eine Stelle s heißt b-sicher, wenn Y B-sicher ist 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)

Analyse von Systemen: Sicherheit

- Beispiel: Verkehrsplaner modelliert Ampelsystem. An bestimmter Stelle s darf sich nur ein Auto aufhalten. Modelliert er K(s)=1, kann die Analyse über die Korrektheit der Modellierung nichts aussagen. Durch Modellierung K(s)=∞ kann in der Analyse geprüft werden, 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.



Analyse von Systemen: Lebendigkeit

- Eine Transition t eines S/T-Systems Y=(N,M0) heißt **aktivierbar**, wenn sie mindestens unter einer Folgemarkierung aktiviert ist:

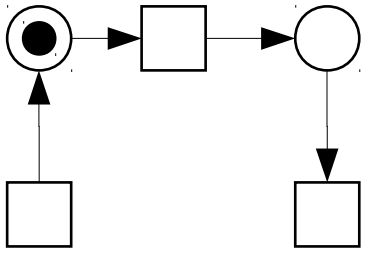
  $\exists M_1 \in [M_0>: M_1[t>$

- Sie heißt **lebendig**, wenn sie unter allen Folgemarkierungen aktivierbar ist:

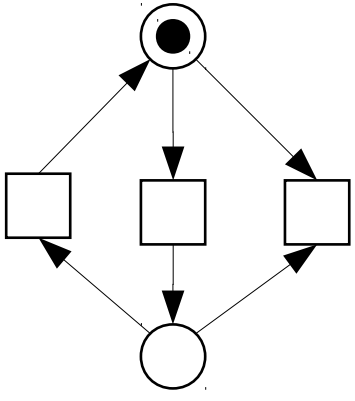
  $\forall M_1 \in [M_0>: \exists M_2 \in [M_1>: M_2[t>$

- Sie heißt **tot**, wenn sie unter keiner erreichbaren Markierung aktiviert ist:

  $\forall M \in [M_0>: \neg M[t>$



Lebendige Transition



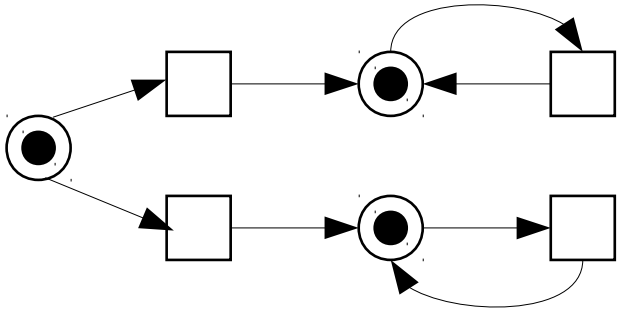
Tote Transition

Analyse von Systemen: Lebendigkeit

- Ein S/T-System $Y=(S,T,F,K,W,M_0)$ heißt **deadlockfrei** oder **schwach lebendig**, wenn unter jeder erreichbaren Markierung mindestens eine Transition aktivierbar ist:

  $\forall M_1 \in [M_0>: \ \exists t \in T: \ M_1[t>$

Schwach lebendiges System



Analyse von Systemen: Lebendigkeit

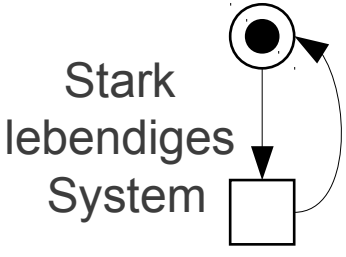
- Ein S/T-System heißt **lebendig** oder **stark lebendig**, wenn aus jeder erreichbaren Markierung jede Transition aktivierbar ist:

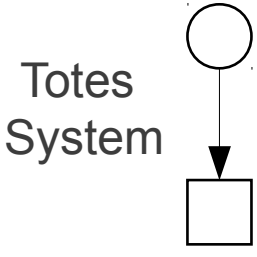
  $\forall M_1 \in [M_0>: \forall t \in T: \exists M_2 \in [M_1>: M_2[t>$

  - Eigenschaft permanent aktiver Systeme, die nie auch nur teilweise ausfallen
  - Berücksichtigung partieller Ausfälle („graceful degradation“) führt zur schwachen Lebendigkeit, BSP
- Es heißt **tot**, wenn keine Transition aktiviert ist:

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

  - Bedeutet häufig einen Deadlock





Analyse von Systemen: Lebendigkeit

Aussagen zur Lebendigkeit

- Ein S/T-System ist genau dann tot, wenn keine Transition aktivierbar ist, d.h. alle Transitionen tot sind.
- Ein S/T-System ist genau dann schwach bzw. stark lebendig, wenn keine erreichbare Markierung tot ist bzw. alle seine Transitionen lebendig sind.
- Ein S/T-System ist genau dann stark lebendig, wenn sein Erreichbarkeitsgraph von jedem Knoten ausgehend für jedes t aus T eine Pfad besitzt, in dem t als Etikett (d.h. als Label) vorkommt.
- Eine erreichbare Markierung ist genau dann tot, wenn von ihr im Erreichbarkeitsgraph keine Kante ausgeht.

Methodik

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

Fazit: Petri-Netze

+ Einfache und wenige Elemente
+ Graphisch gut darstellbar
+ Durch Marken übersichtliche Visualisierung des Systemzustands
+ Syntax und Semantik formal definiert
+ Werkzeuge zur Erstellung, Analyse, Simulation,Code-Generierung vorhanden (insbesondere Process Mining !)
+ Petri-Netze gut geeignet zur Modellierung von Systemen mit kooperierenden Prozessen
+ Besitzen größere Mächtigkeit als Zustandsautomaten, da zu einem Zeitpunkt mehrere Zustände darstellbar
- Höhere Petri-Netze schwer zu erstellen und zu analysieren
- Von anderen Modellierungskonzepten isoliert
- Statische Struktur
- Keine Methode zur Erstellung von Petri-Netzen vorhanden

Zusammenfassung

In diesem Kapitel haben wir betrachtet:

- Petrinetze – Beispiel
- Petrinetze – Grundlagen
- Stellen/Transitions-Netze
- Erreichbarkeit
- Grundsituationen in S/T-Netzen
- Analyse von Systemen
- Methodik

Petri-Netzen als Grundlage für Geschäftsprozessmodellierung.

Weiter in diesem Kapitel: Petri-Netze als Grundlage für Process-Mining.