Willkommen zur Vorlesung
Softwarekonstruktion
im Wintersemester 2011/2012

Prof. Dr. Jan Jürjens

TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

05. Petrinetze

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

Themen der Vorlesung

Kapitel 1: Intro mit Vorstellung der Professur und Gliederung der Vorlesung

Kapitel 2: Allgemeine Prinzipien des SW-Engineering

Kapitel 3: Spezifikation im Allgemeinen

Kapitel 4: Algebraische Spezifikation

Kapitel 5: Petrinetze

Kapitel 6: Modellgetriebene SW-Entwicklung

Kapitel 7: Object Constraint Language (OCL)

Kapitel 8: Testen im Allgemeinen, Kontrollflussorientierte Testverfahren, Datenflussorientierte Testverfahren

Agenda

Kap. 05: Petrinetze

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

Motivation

In Kapitel 04 beschäftigten wir uns mit der *algebraischen Spezifikation* als *Grundlage* für die *Spezifikation* des *Verhaltens einzelner Softwaremodule*.

Wir haben gesehen, dass algebraische Spezifikationen weniger gut geeignet sind, um interne Systemzustände und Interaktion zwischen verschiedenen Softwaremodulen explizit zu modellieren.

In diesem Kapitel beschäftigen wir uns daher mit Petri-Netzen als *Grundlage* für die Spezifikation von *internen Systemzustände* und *Interaktion zwischen verschiedenen Softwaremodulen*.

Petri-Netze bilden wiederum die Grundlage für Spezifikations-Ansätze in der Praxis, die wir uns im nächsten Kap. 06 ansehen werden. Zum Beispiel wird die Ausführungssemantik von Aktivitätsdiagrammen in UML 2.x auf Basis von Petri-Netzen definiert.

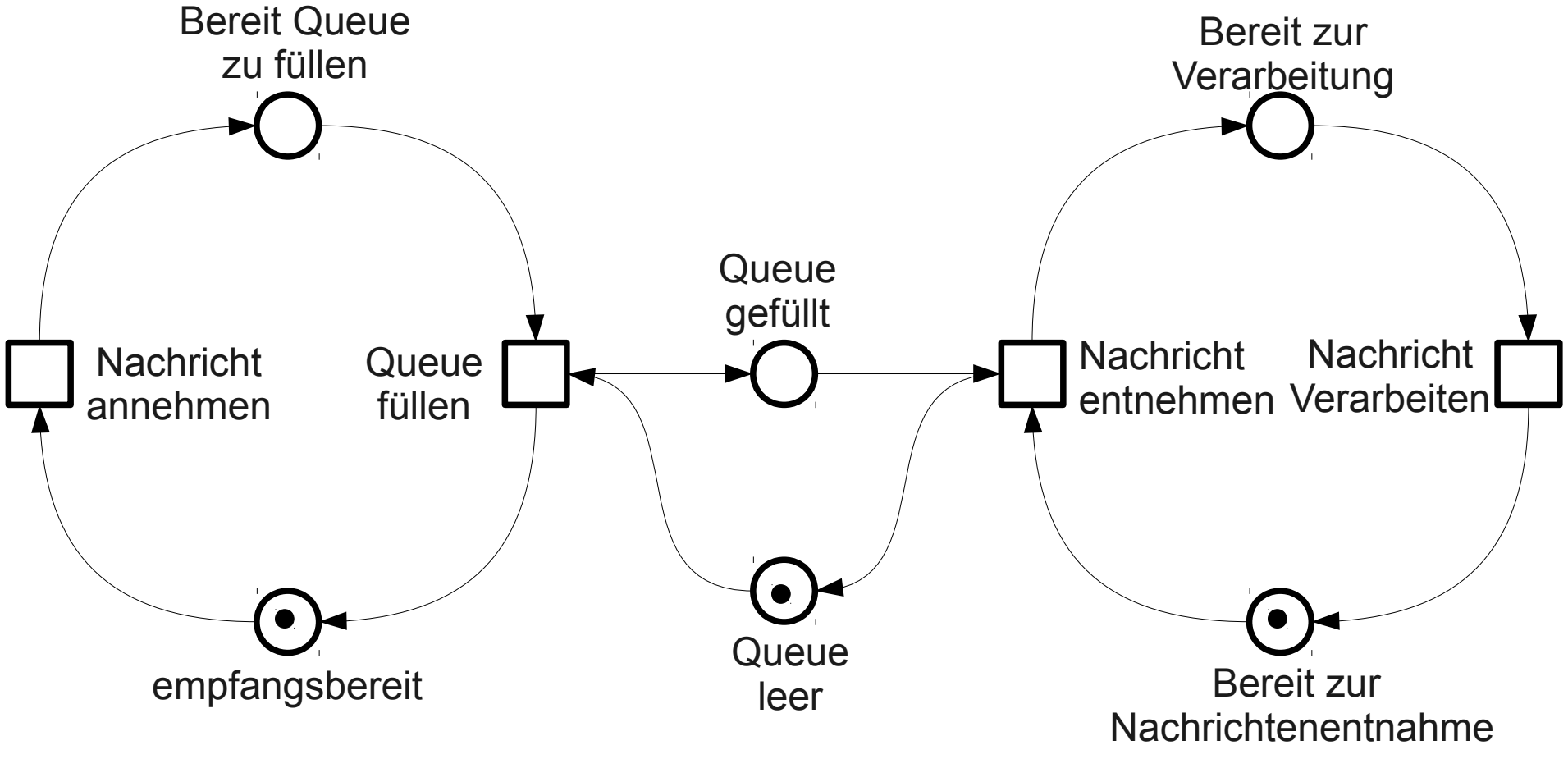
Ausserdem werden Petri-Netze zum Teil auch direkt in der Praxis eingesetzt, beispielsweise in dem Process-Mining-Werkzeug ProM (cf. http://processmining.org).

Petri-Netze

- Ursprung: Dissertationsschrift
  - "Kommunikation mit Automaten" von Carl Adam Petri (1962)
- Seither: mehr als 10.000 Arbeiten auf dem Gebiet.
- Bis 1985: hauptsächlich von Theoretikern benutzt.
- Seit Mitte der 80er Jahre: vermehrter Einsatz in praktischen Anwendungen.

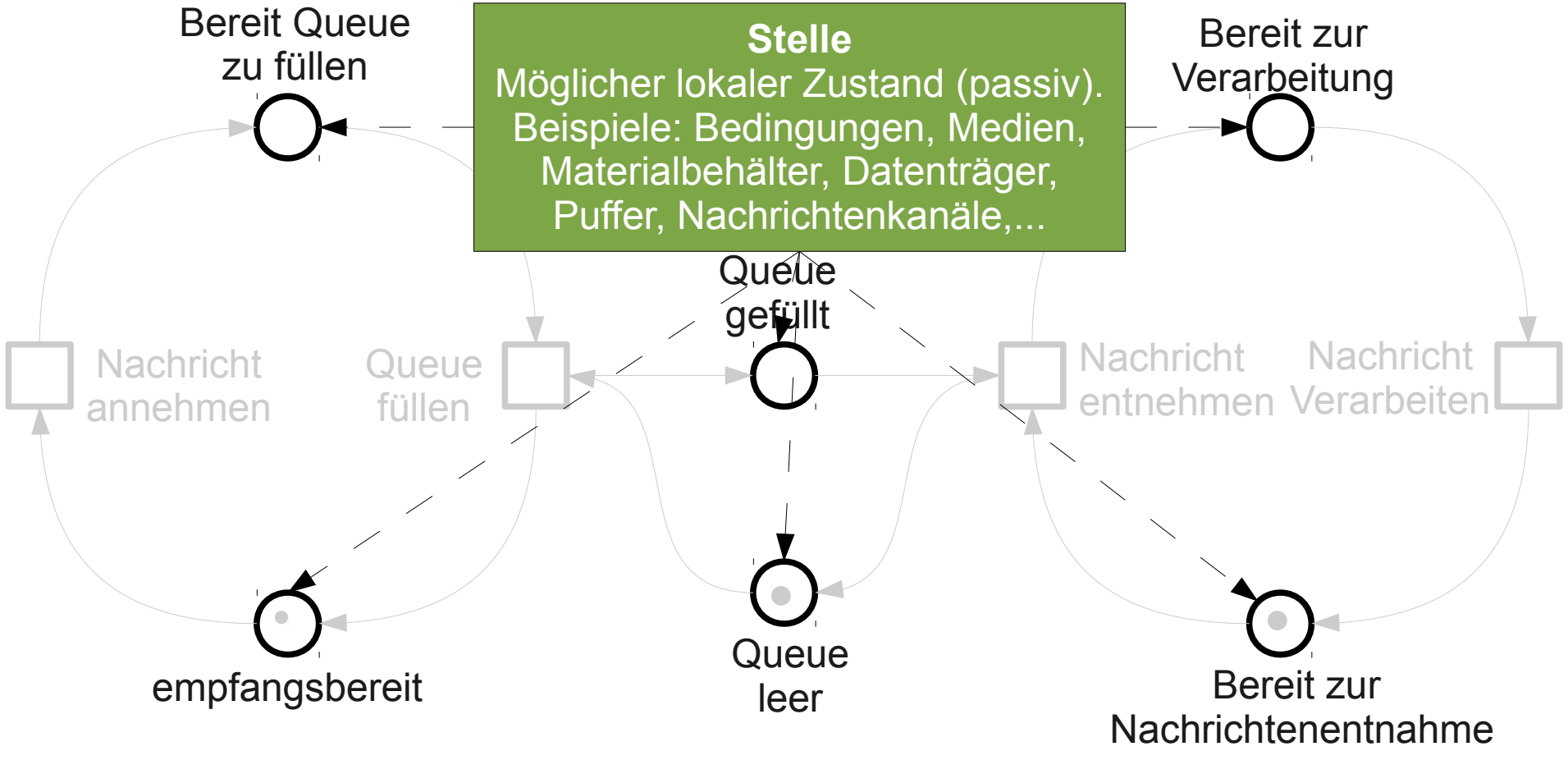
Petri-Netze: Beispiel

- Beispiel Nachrichten-Queue



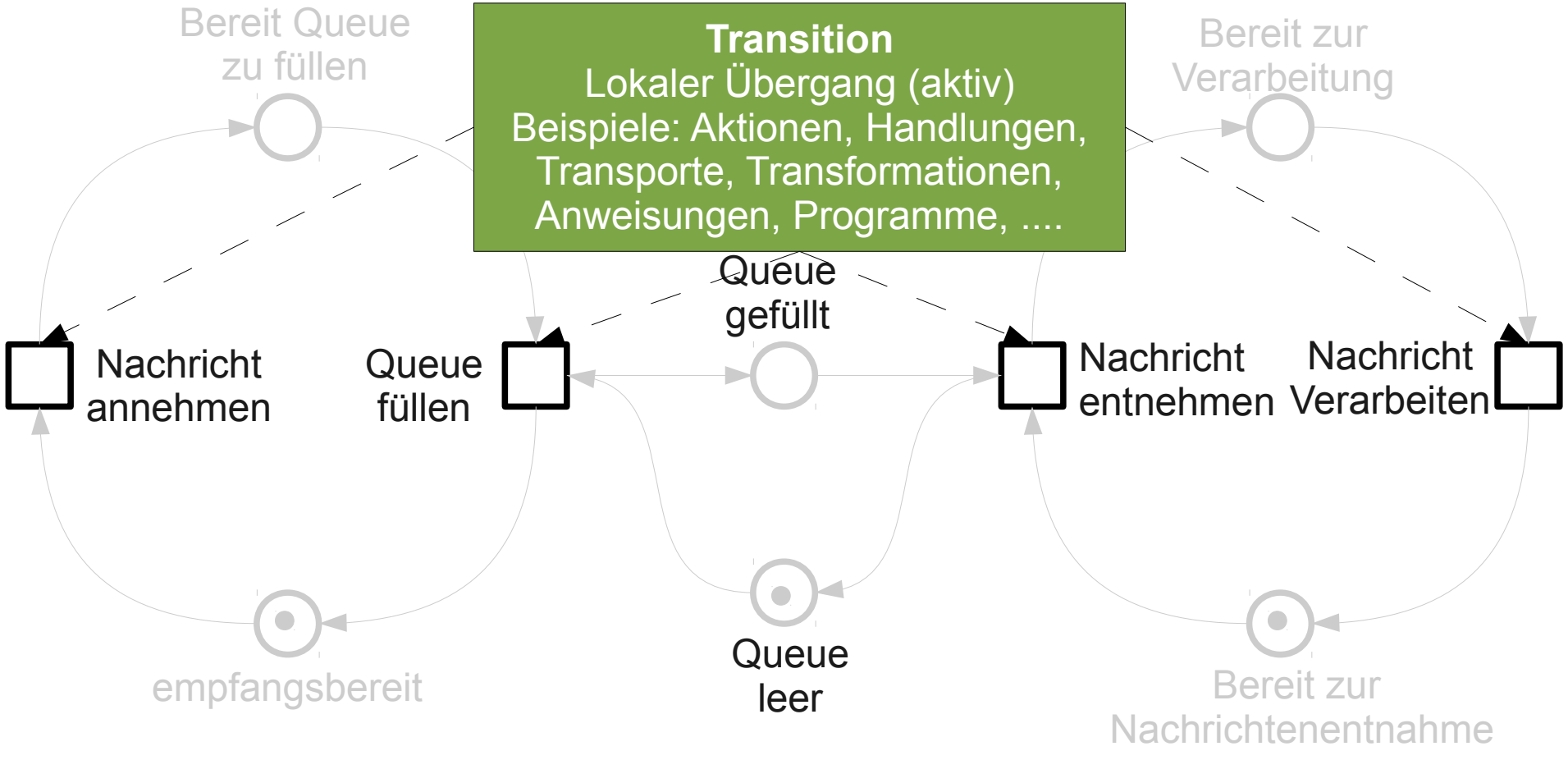
Petri-Netze: Stelle

- Beispiel Nachrichten-Queue



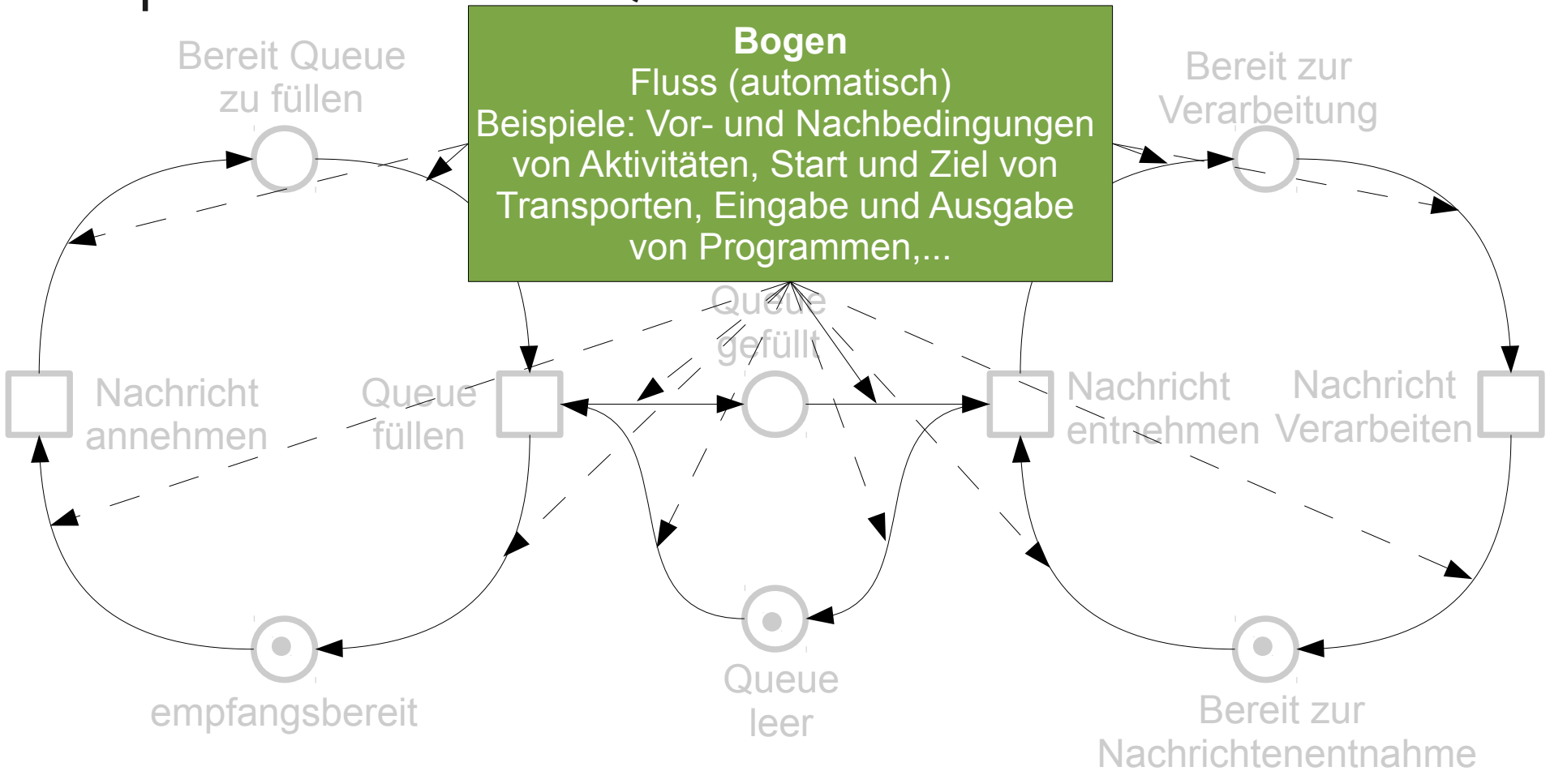
Petri-Netze: Transition

- Beispiel Nachrichten-Queue als Netz



Petri-Netze: Bogen

- Beispiel Nachrichten-Queue



Definition Netz

Definition Netz:

- Gegeben:
  - S - endliche Menge von Stellen
  - T - endliche Menge von Transitionen
  - F - Menge von Bögen
    - $F \subseteq (S \times T) \cup (T \times S)$ binäre Relation
- wenn: $S \cap T = \emptyset \; und \; S \cup T \neq \emptyset$
- dann: (S,T,F) ein Netz

Definition Netz: Beispiel

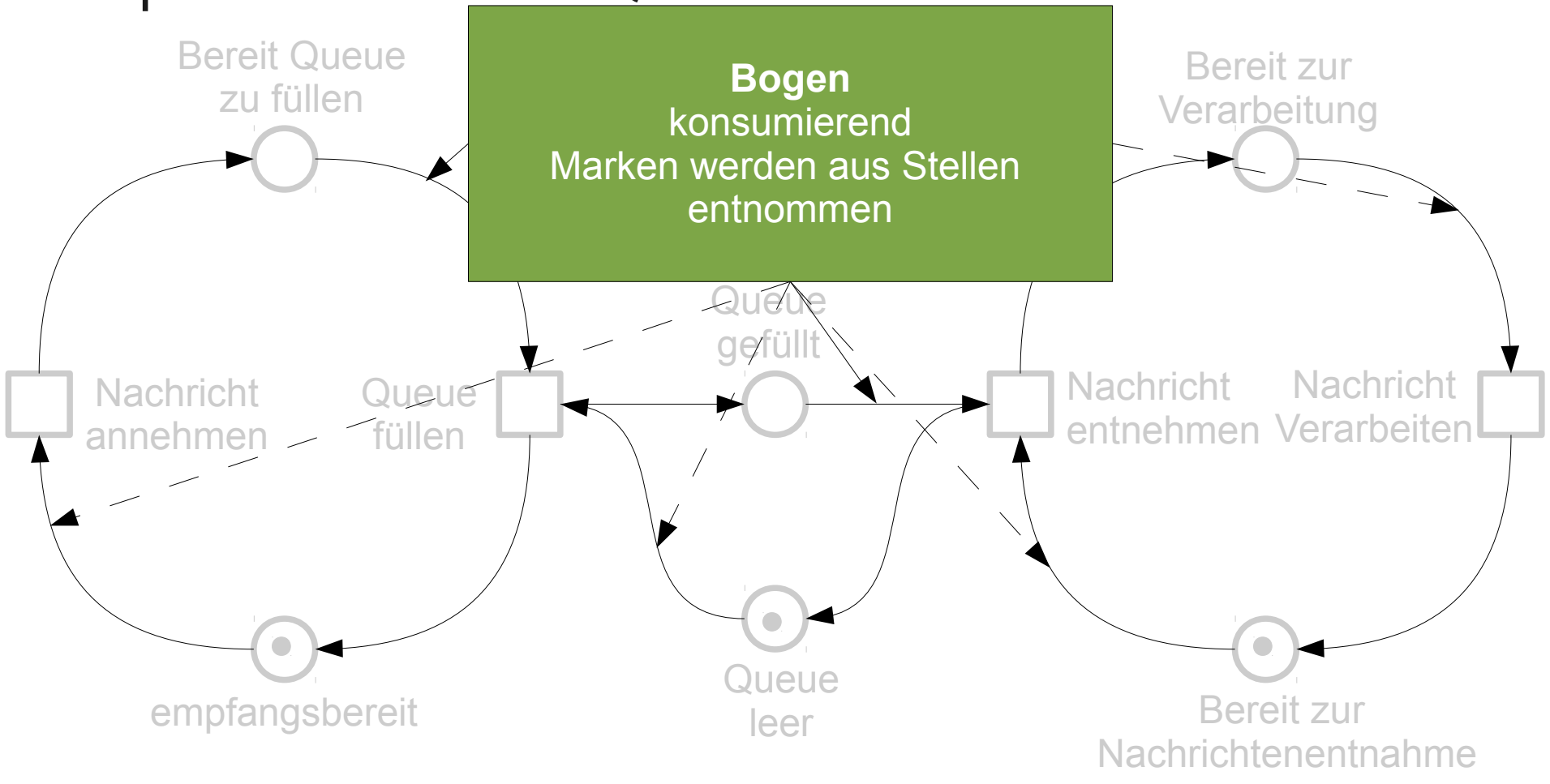
- Nachrichten-Queue: Netz (S,T,F) mit
  - S = {empfangsbereit, Bereit Queue zu füllen, Queue gefüllt, Queue leer, Bereit zur Verarbeitung, Bereit zur Nachrichtenentnahme}
  - T = {Nachricht annehmen, Queue füllen, Nachricht entnehmen, Nachricht verarbeiten}
  - F = {(empfangsbereit, Nachricht annehmen),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

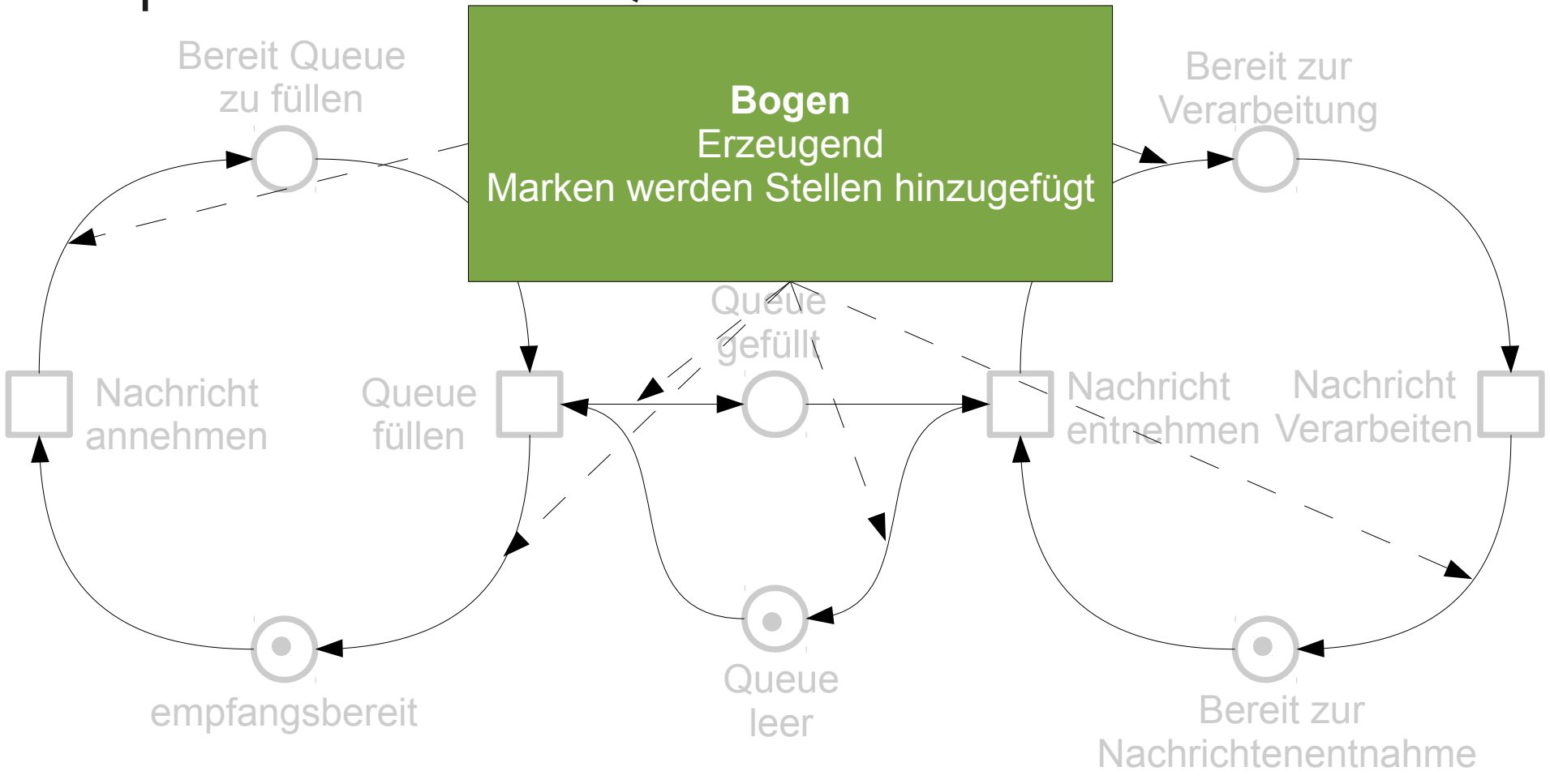
Bogen (konsumierend)

- Beispiel Nachrichten-Queue



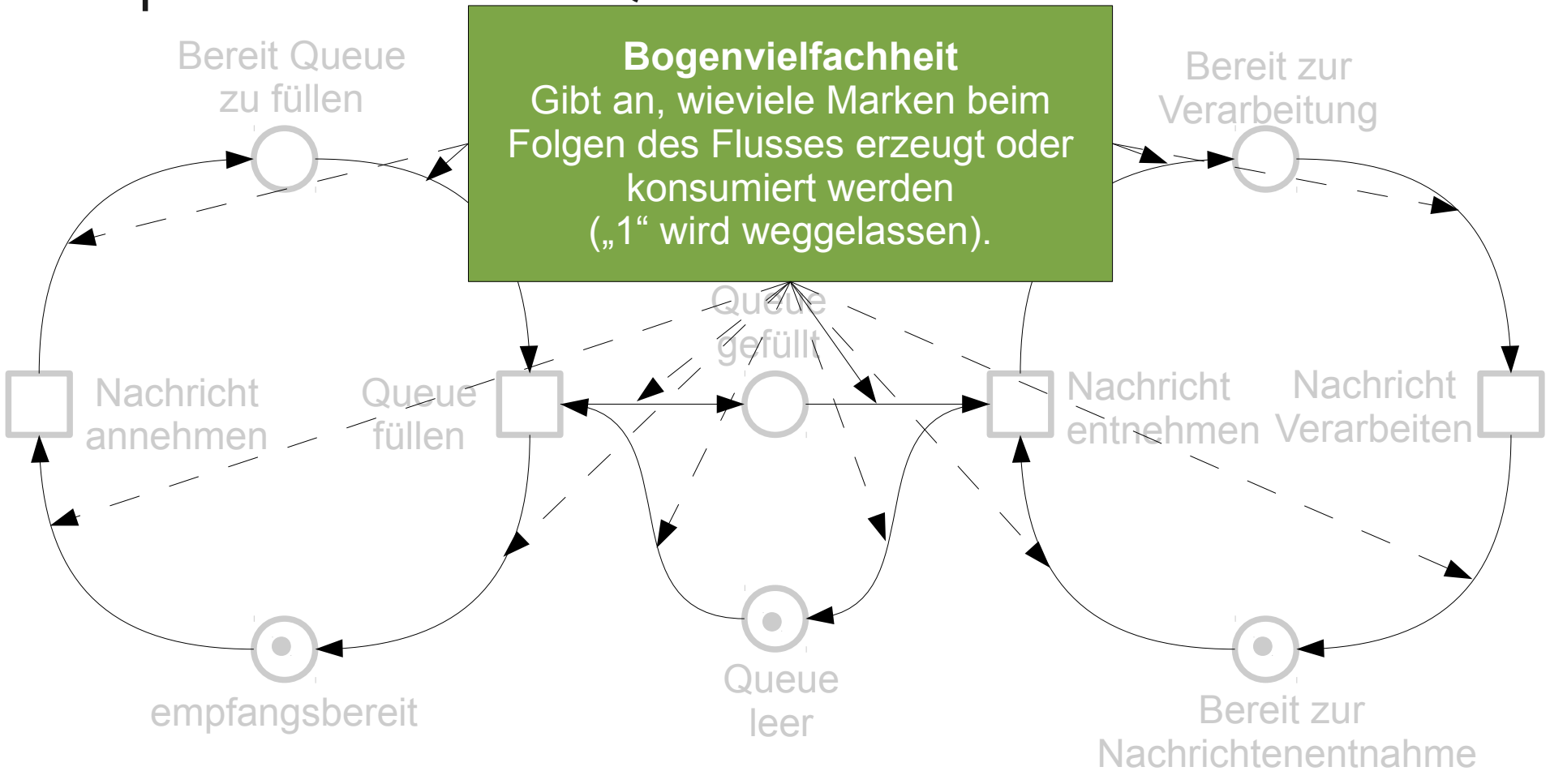
Bogen (erzeugend)

- Beispiel Nachrichten-Queue



Bogenvielfachheit

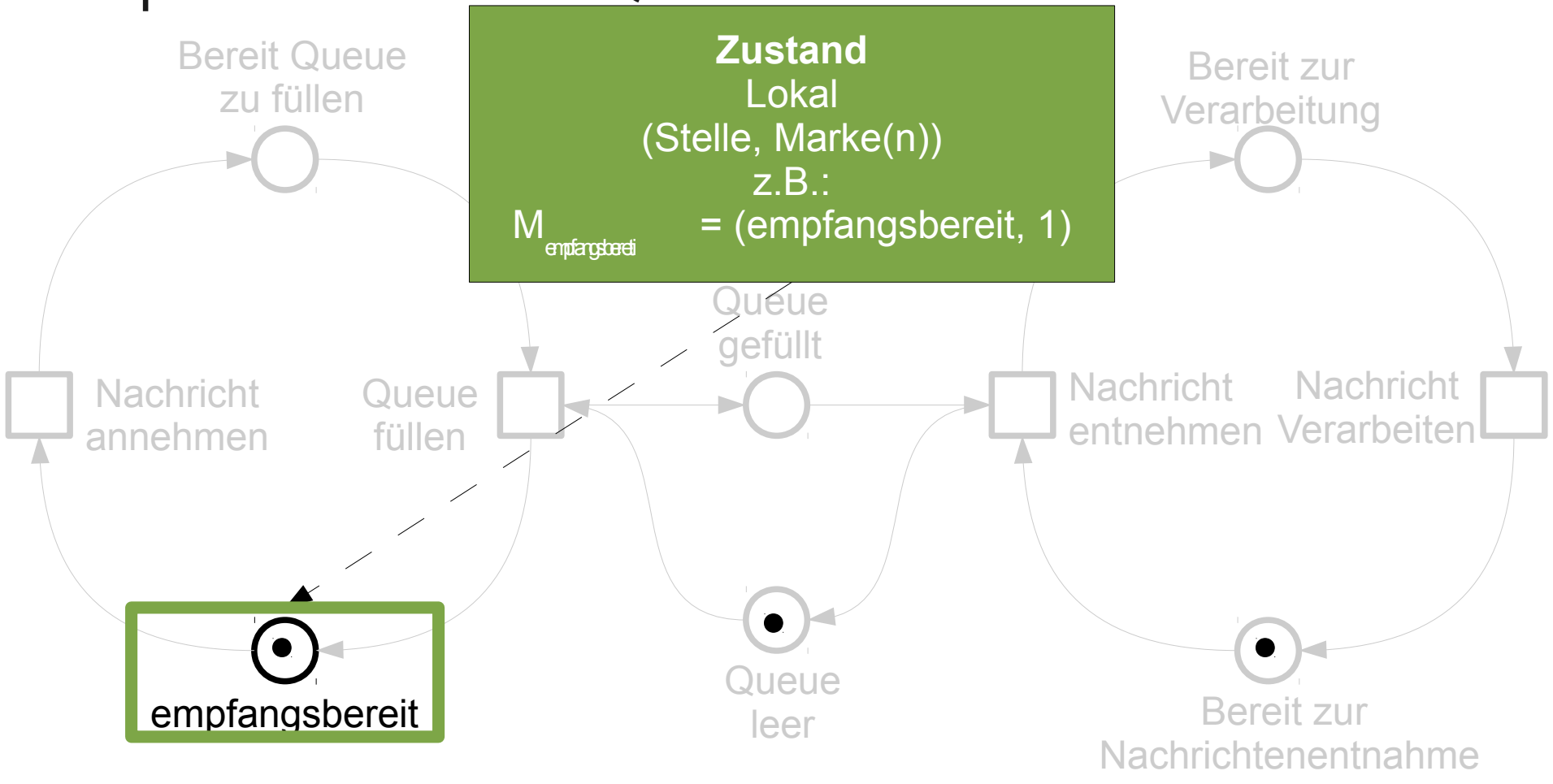
- Beispiel Nachrichten-Queue



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

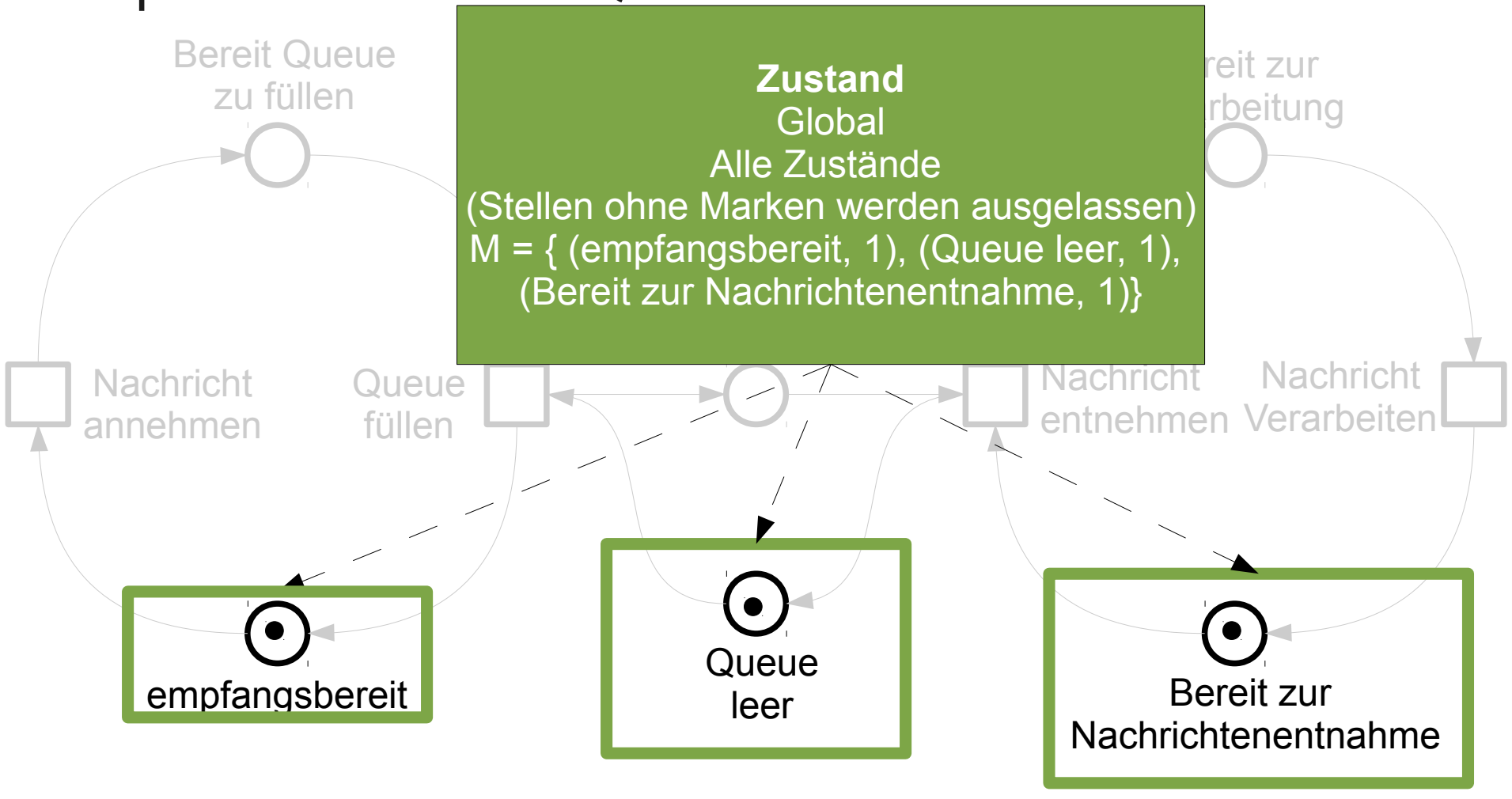
Zustand (lokal)

- Beispiel Nachrichten-Queue



Zustand (global)

- Beispiel Nachrichten-Queue



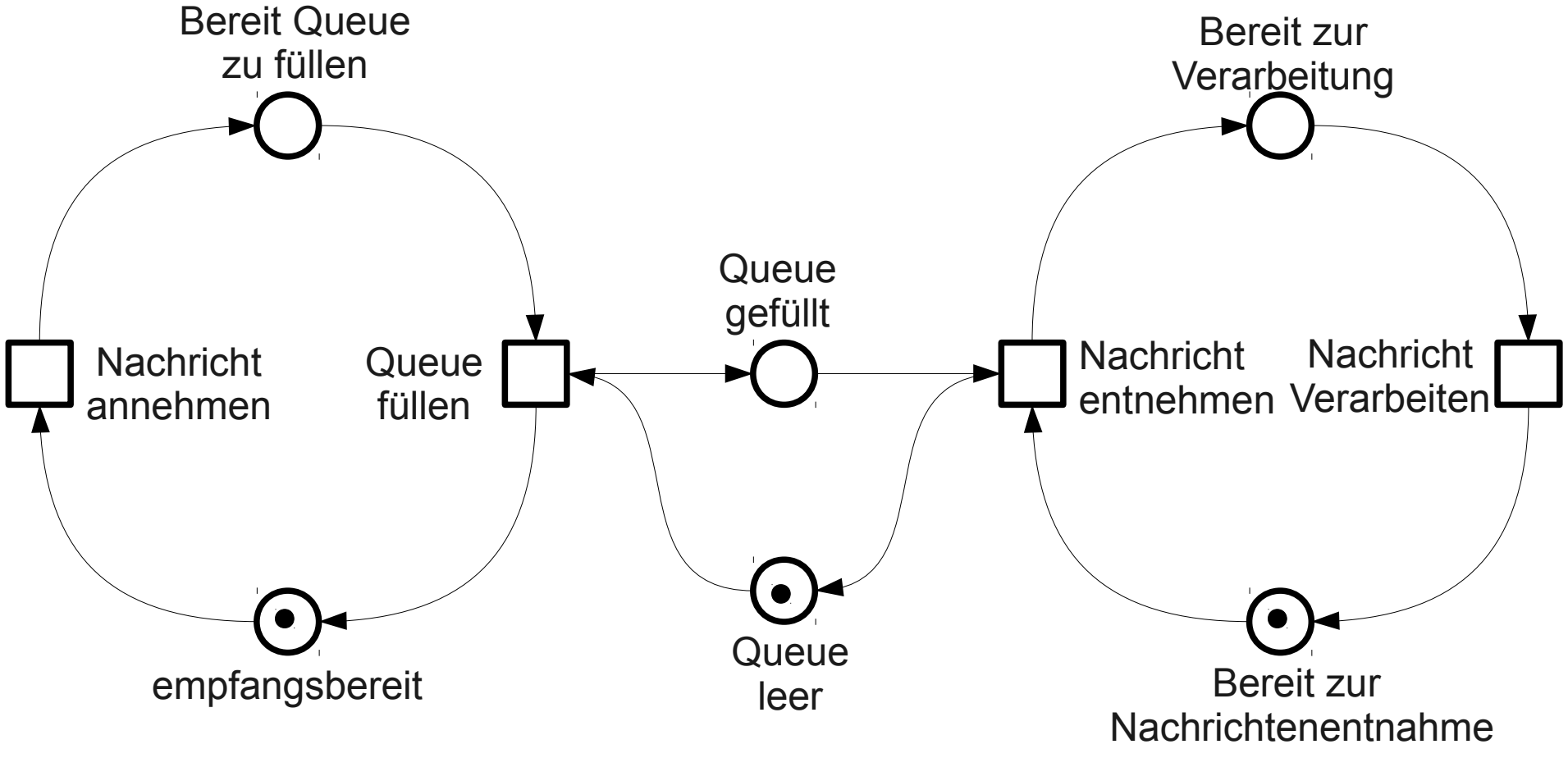
Definition Petri-Netz

Definition Petri-Netz:

- Gegeben:
  - Netz (S, T, F)
  - W – Bogenvielfachheit
    - $W : F \rightarrow \mathbb{N} \setminus 0$
  - M_0 – Globaler Startzustand
    - $M_0 : P \rightarrow \mathbb{N}$
- dann: (S,T,F,W,M_0) ein Petri Netz

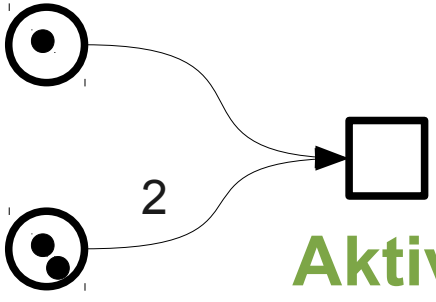
Petri-Netz Ablauf: Beispiel

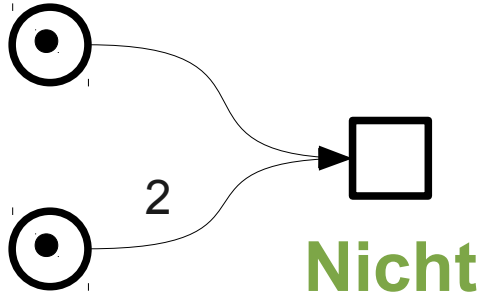
- Beispiel Nachrichten-Queue



Ablauf: Aktivierung von Transition

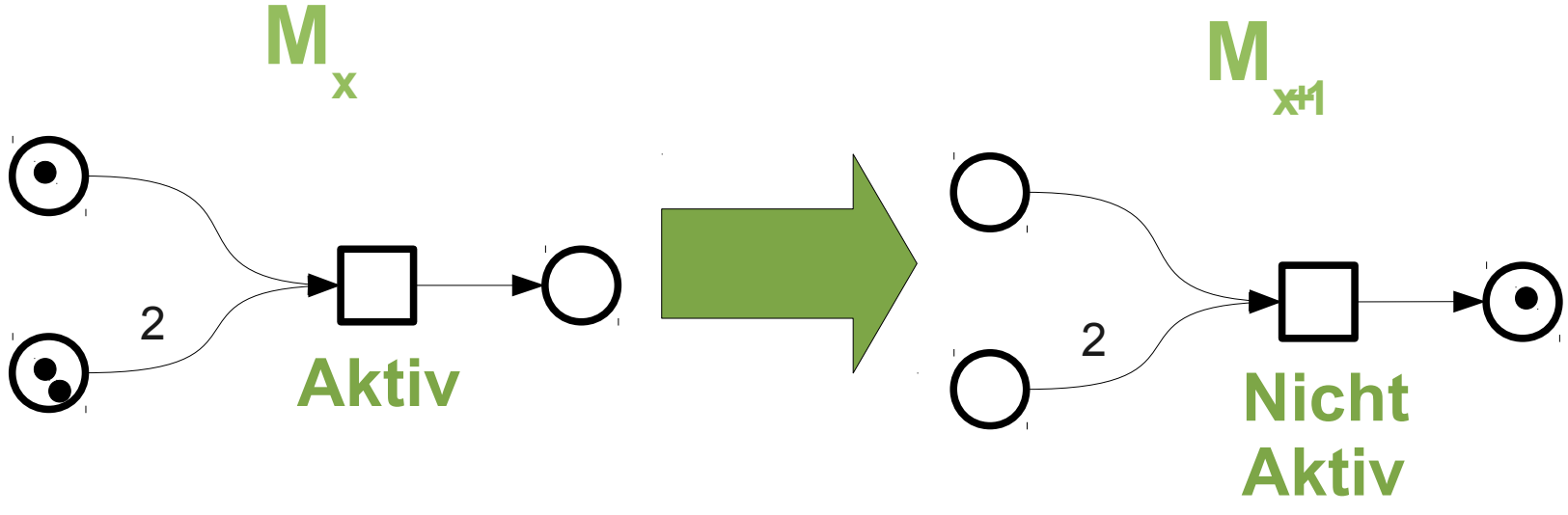
- Ablauf
  - Transition t ist aktiviert, wenn:
    - $\forall\ p \in Vorgänger\ von\ t : W((p,t)) \leqslant m_p$





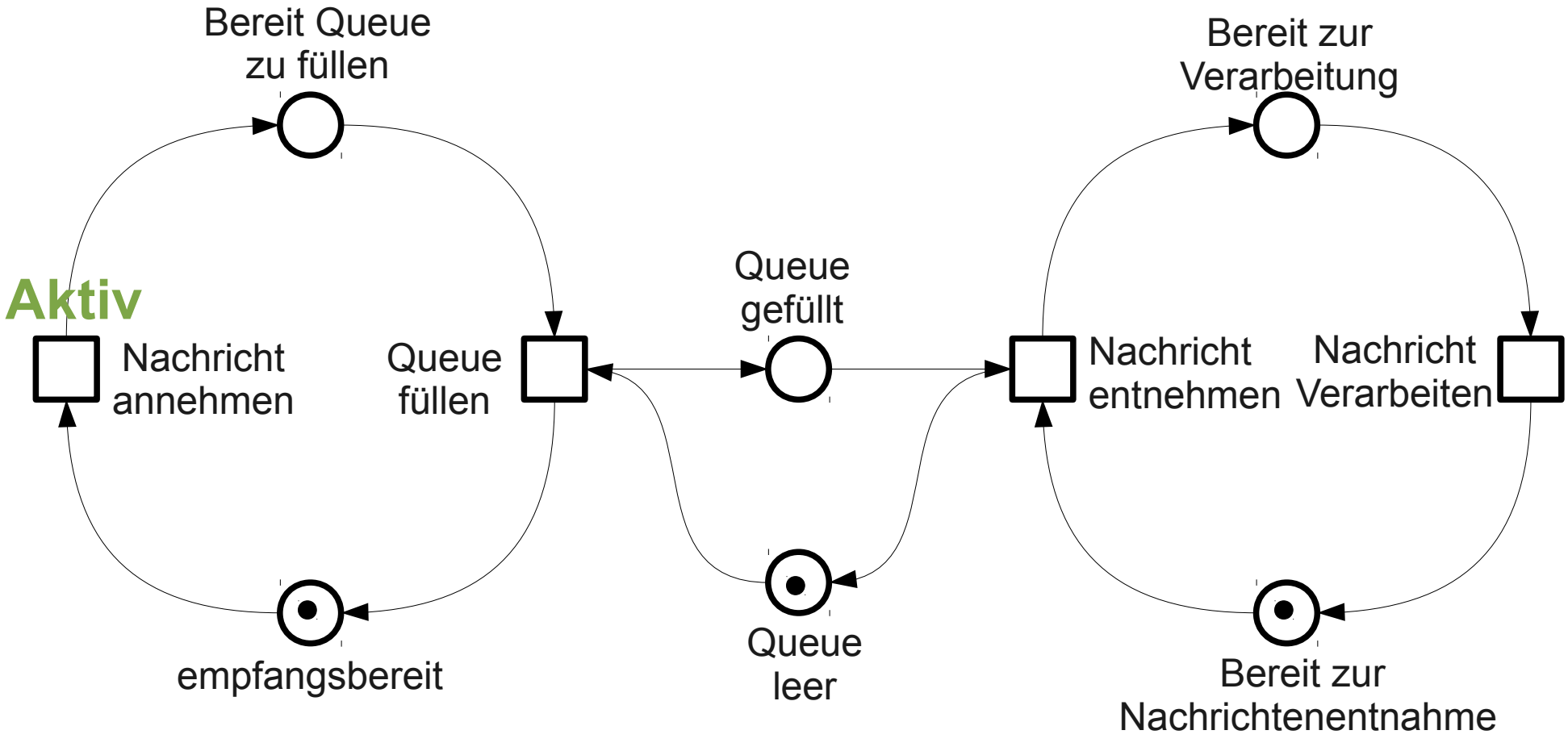
Ablauf: Schalten von Transition

- Ablauf
  - Transition t ist aktiviert, wenn:
    - $\forall\ p \in Vorgänger\ von\ t : W((p,t)) \leqslant m_p$
  - Alle aktivierten t werden beim Übergang M_x nach M_{x+1} geschaltet.



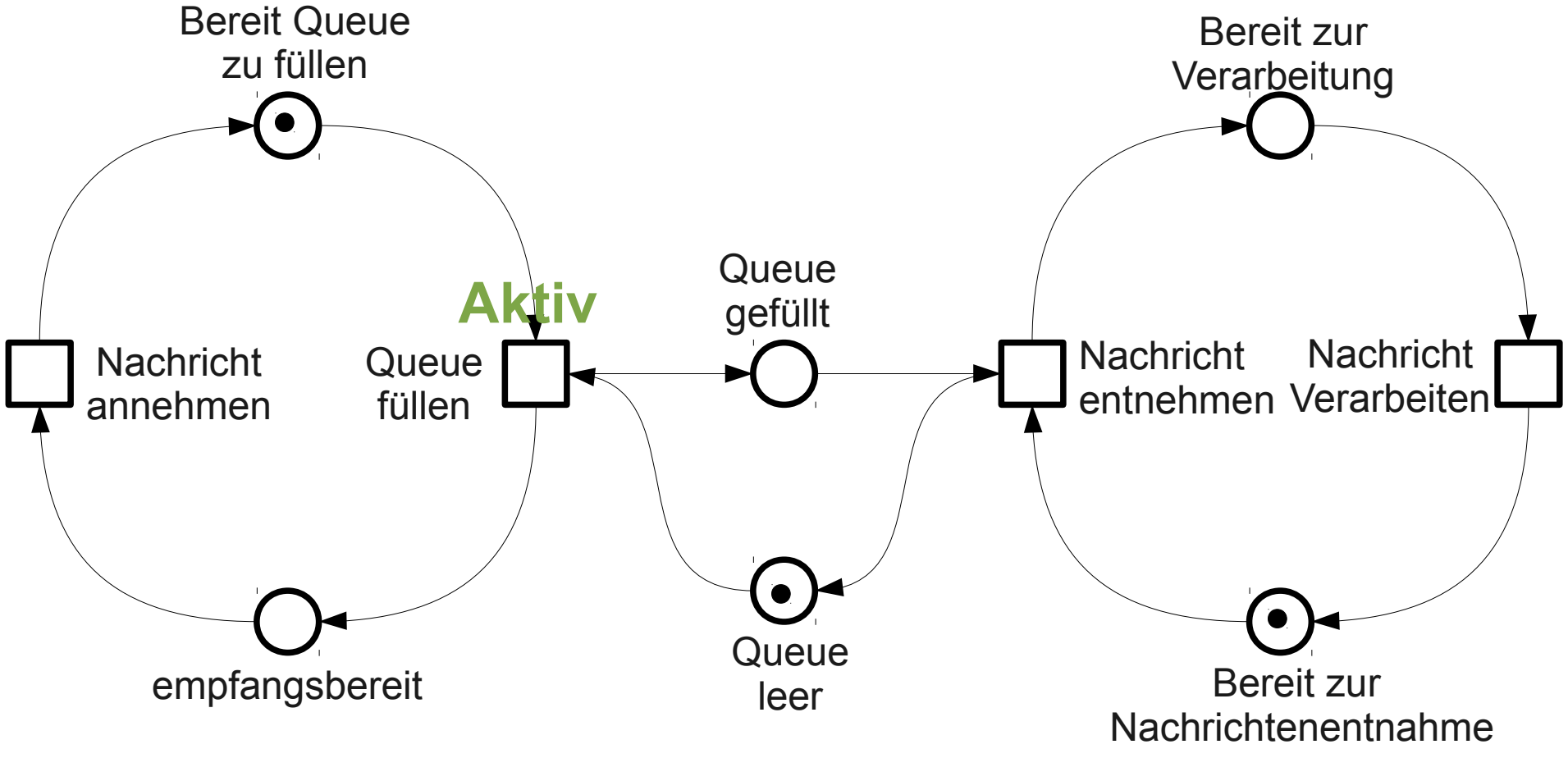
Ablauf: Beispiel (M_0)

- Beispiel Nachrichten-Queue: M_0



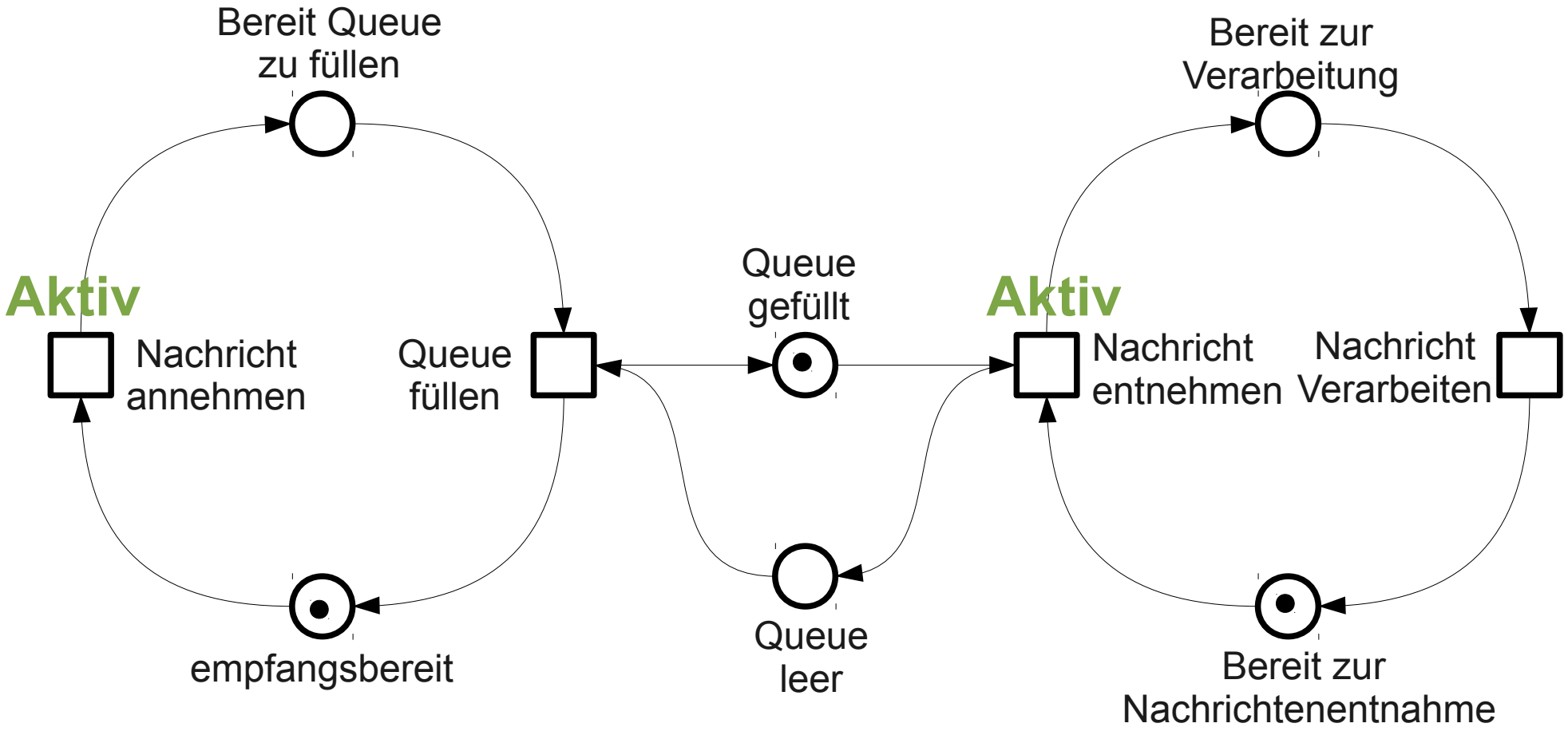
Ablauf: Beispiel (M_1)

- Beispiel Nachrichten-Queue: M_1



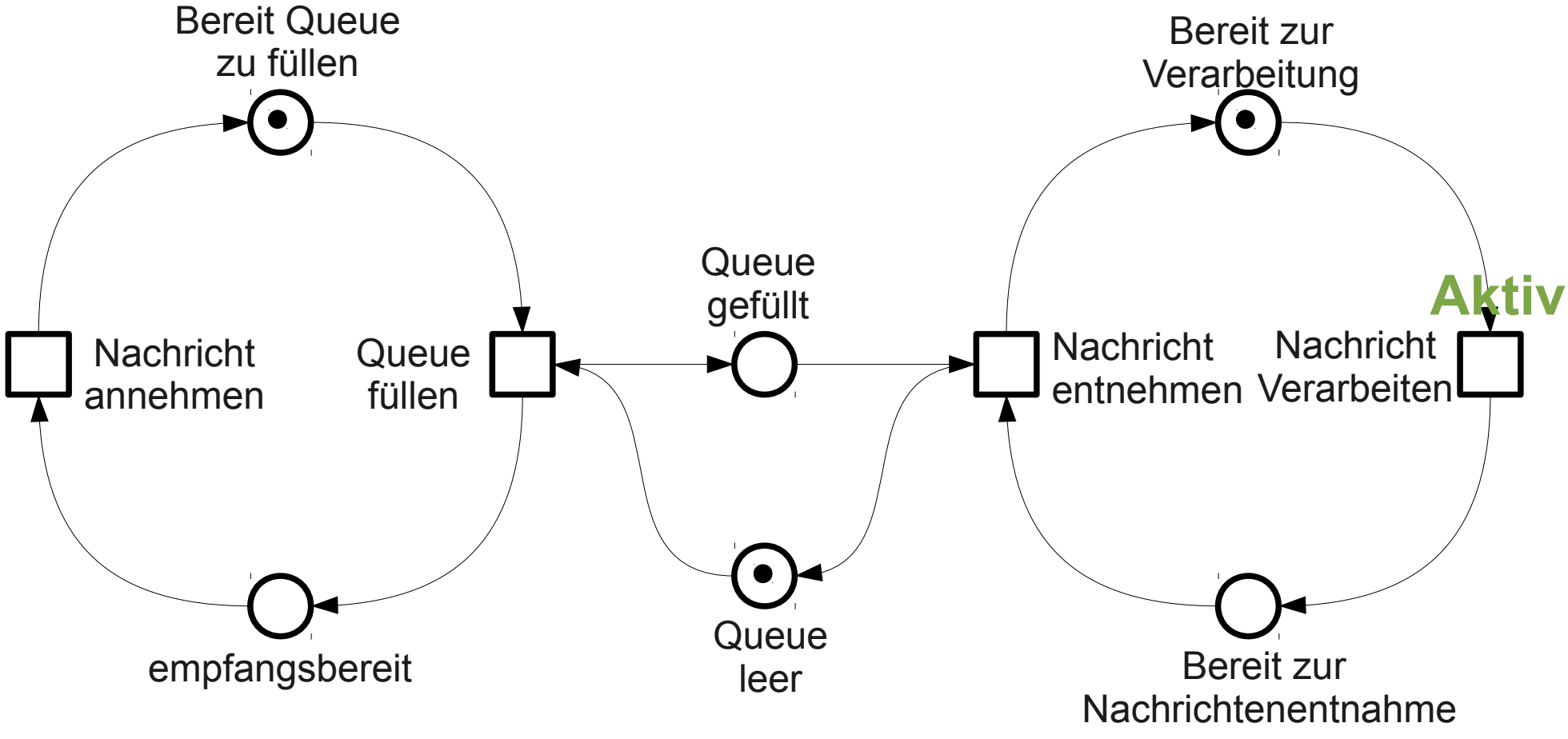
Ablauf: Beispiel (M_2)

- Beispiel Nachrichten-Queue: M_2



Ablauf: Beispiel (M_3)

- Beispiel Nachrichten-Queue: M_3



Frage

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

Frage

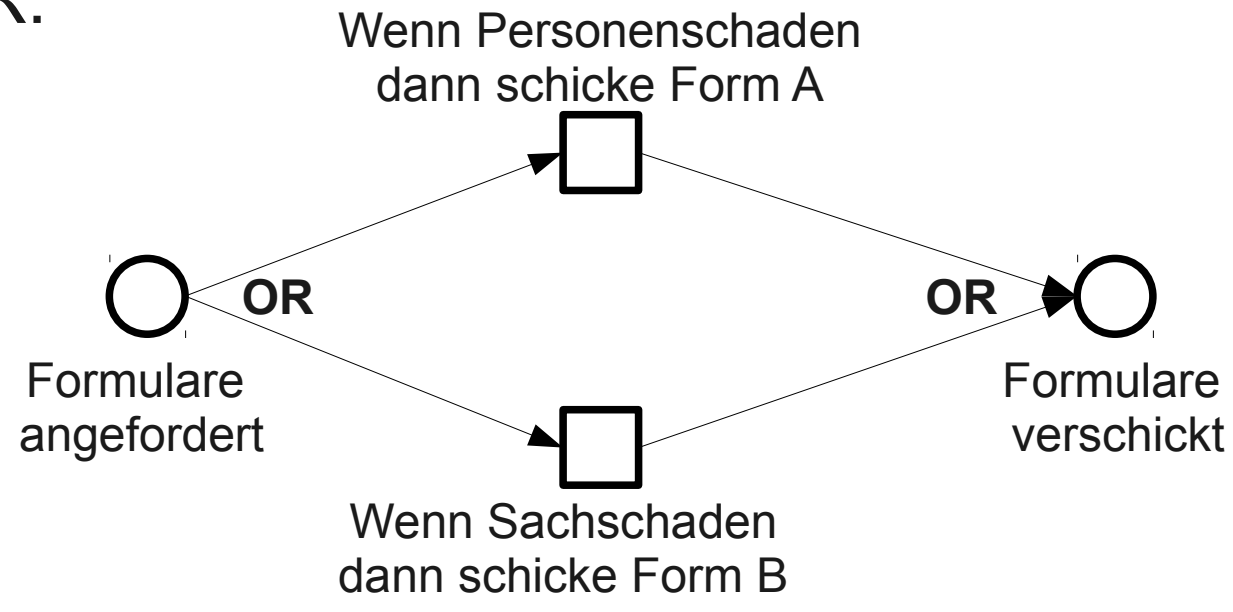
- Was ist untypisch an der Nachrichten-Queue, wie sie bisher modelliert wurde?

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

Implizite Verzweigungen

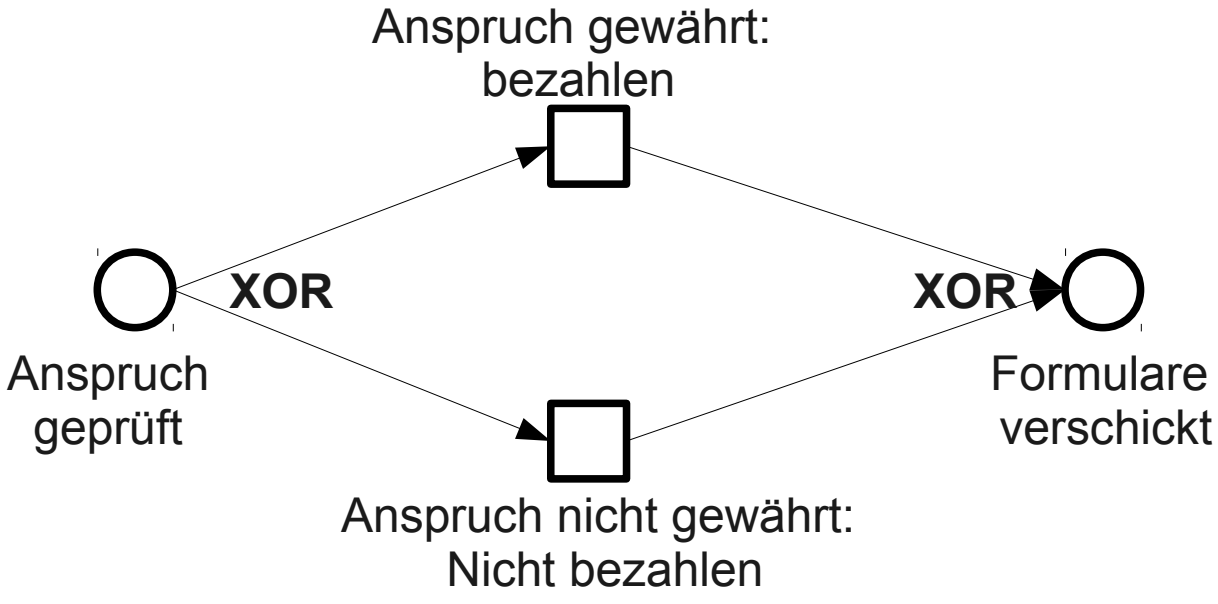
- Gewählter Bogen hängt von Zieltransitionen ab
- Annotation mit AND/XOR/OR kann weggelassen werden, da Folgetransitionen den Fluss vorgeben und interpretiert werden müssen.

Beispiel OR:



Implizite Verzweigungen

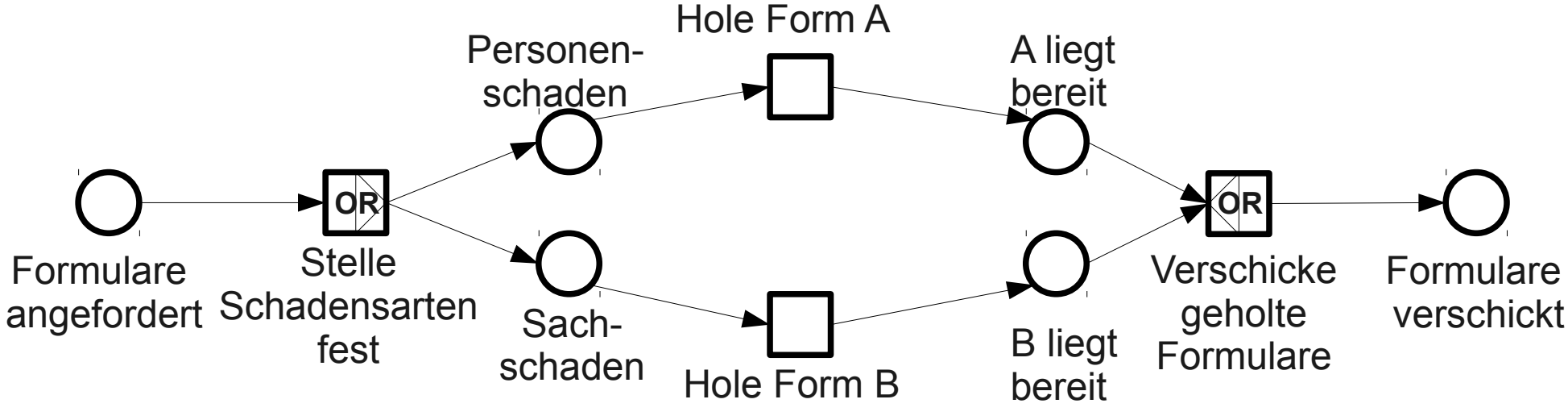
Beispiel XOR:



Explizite Verzweigung vs. Entscheidungstransition: Beispiel 1

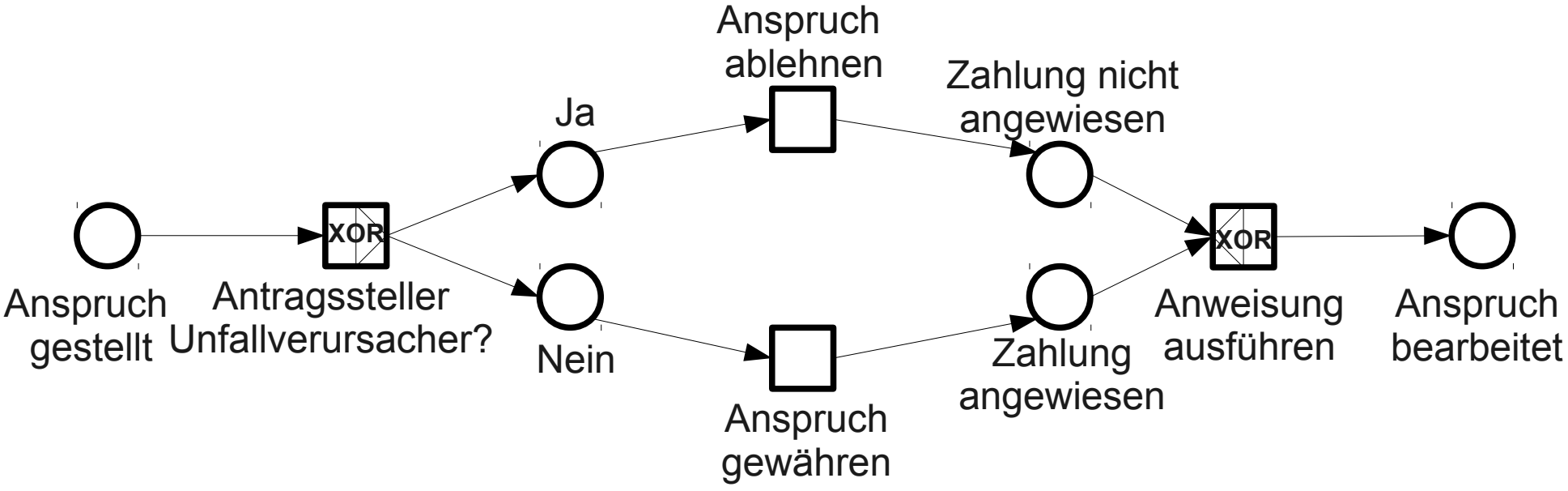
- Gewählter Bogen hängt von Entscheidungstransition ab
- Transition, in der die Entscheidung fällt, die Art der Entscheidung (AND/OR/XOR), sowie alle möglichen Entscheidungen als Nachfolgestellen müssen modelliert werden.

Beispiel OR:



Explizite Verzweigung vs. Entscheidungstransition: Beispiel 2

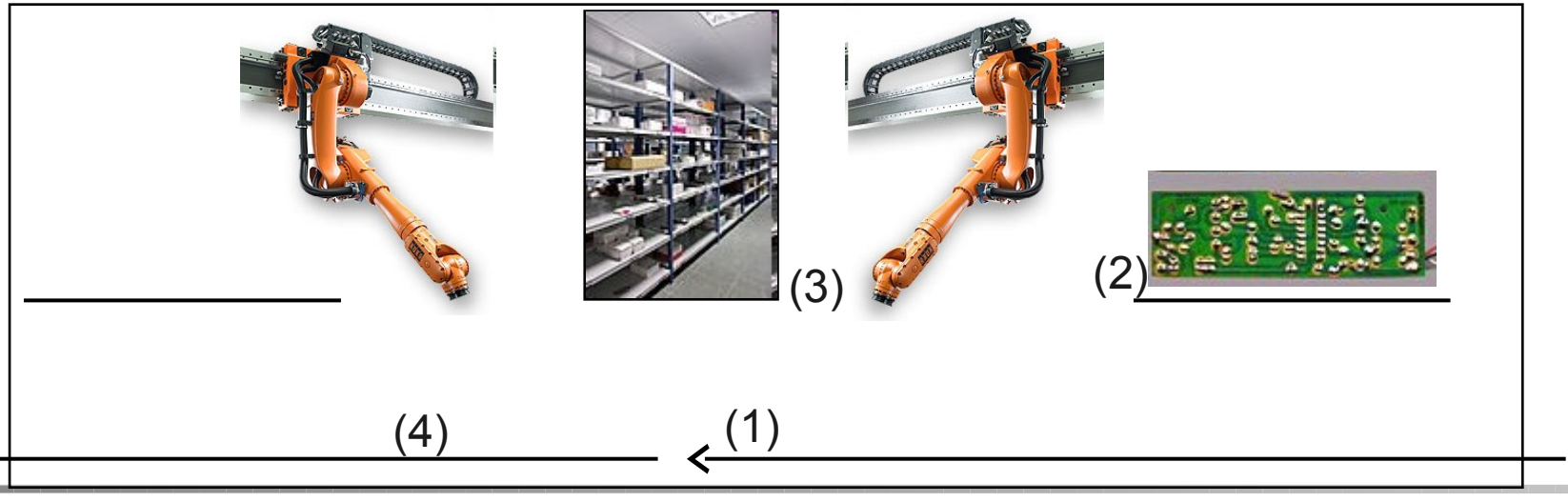
Beispiel XOR:



Beispiel

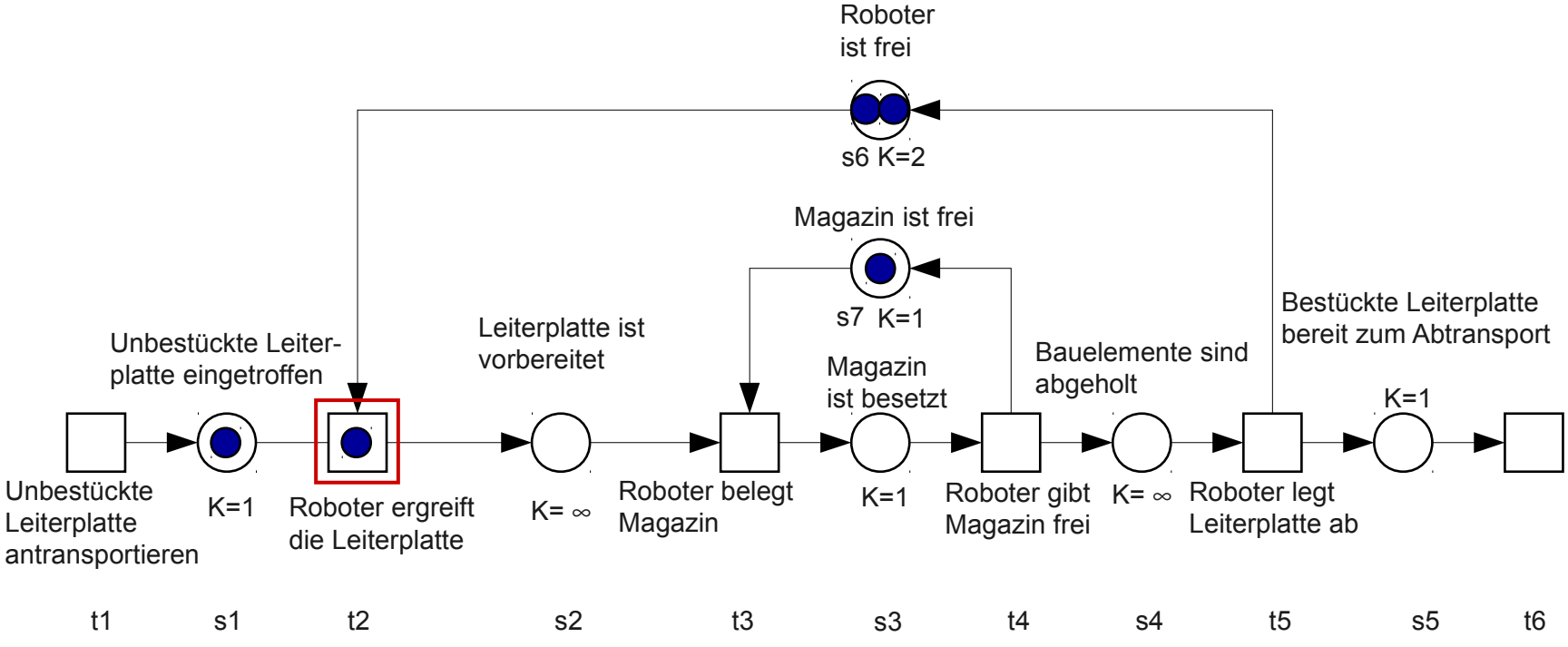
S/T-Netz Bestückungsroboter

- 2 Roboter bestücken Leiterplatten mit elektronischen Bauelementen
- Bauelemente 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)
- Nur jeweils eine Leiterplatte wird zu einem Zeitpunkt abtransportiert (4)



Beispiel

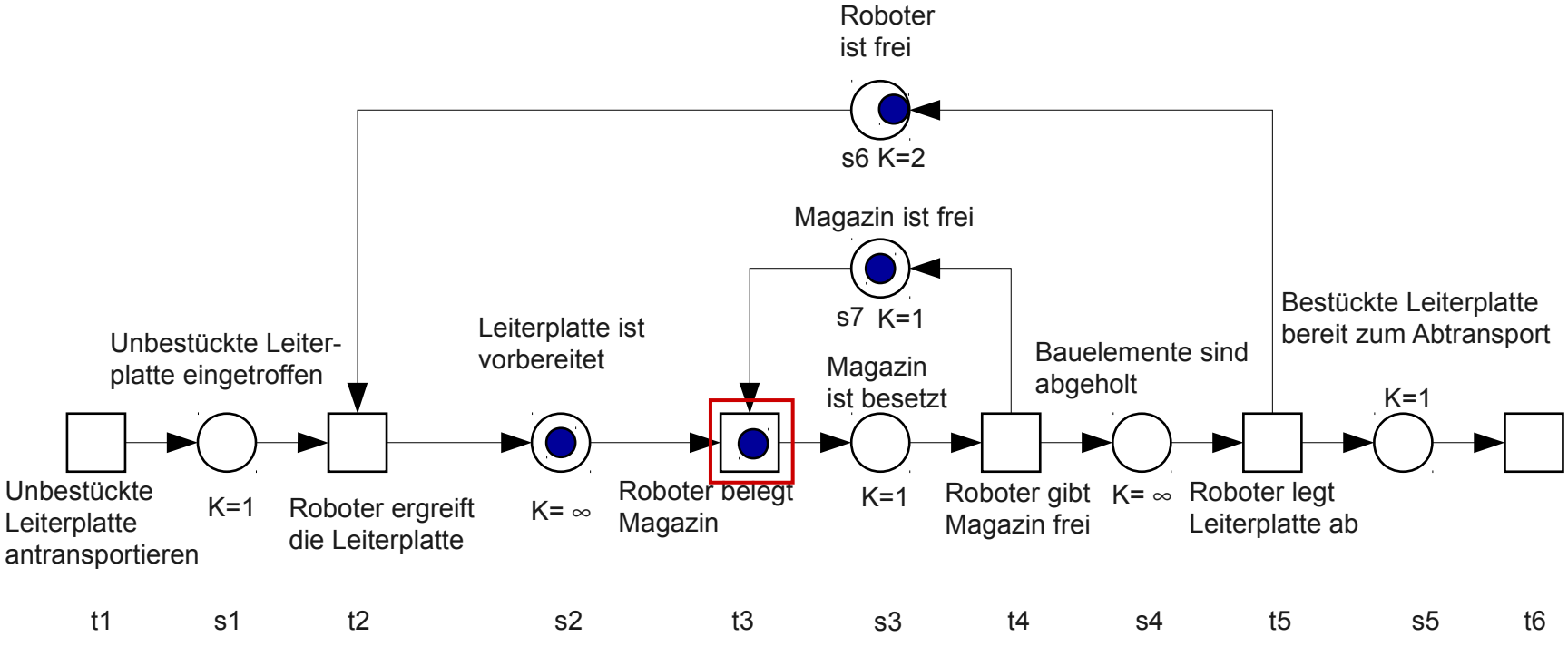
S/T-Netz Bestückungsroboter



NB: Die Marke auf der Transition ist hier nur eingezeichnet, um den Kontrollfluss besser zu veranschaulichen; formal können nur Stellen (und nicht Transitionen) Marken tragen.

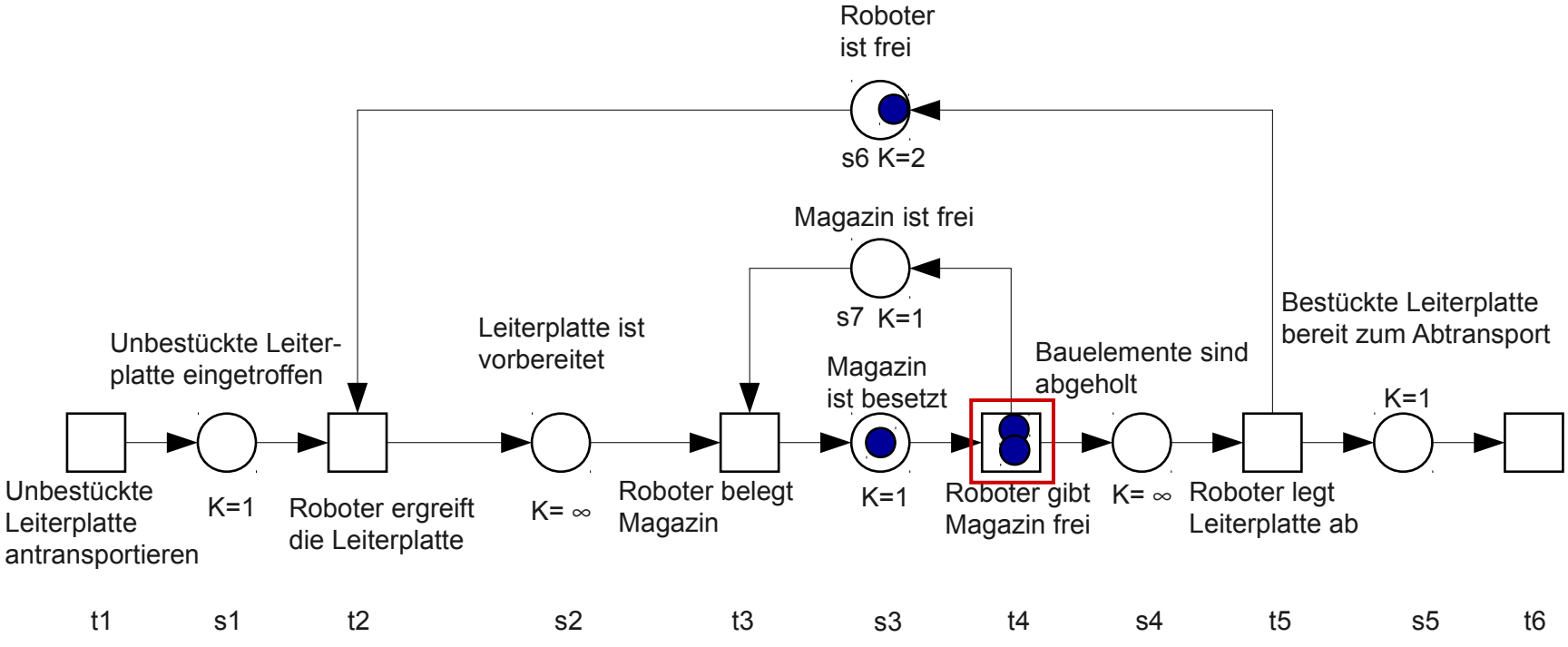
Beispiel

S/T-Netz Bestückungsroboter



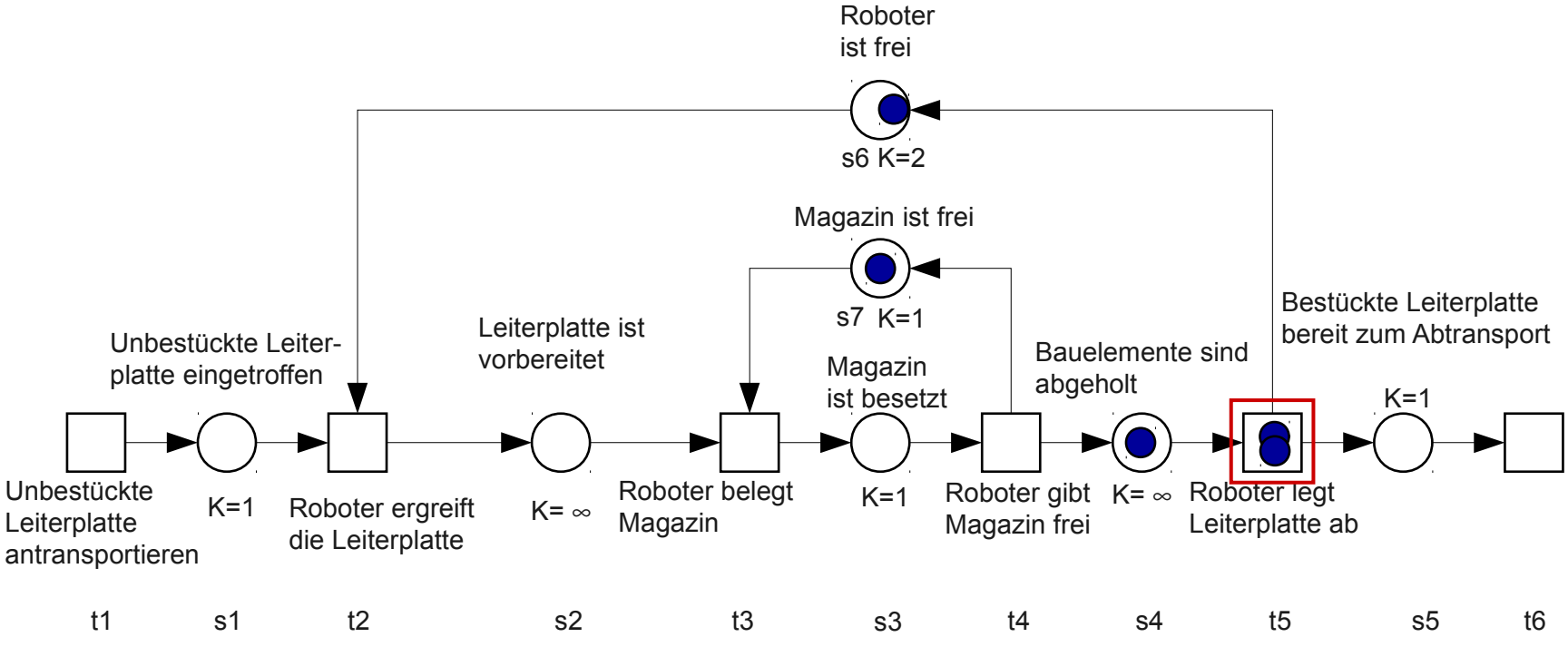
Beispiel

S/T-Netz Bestückungsroboter



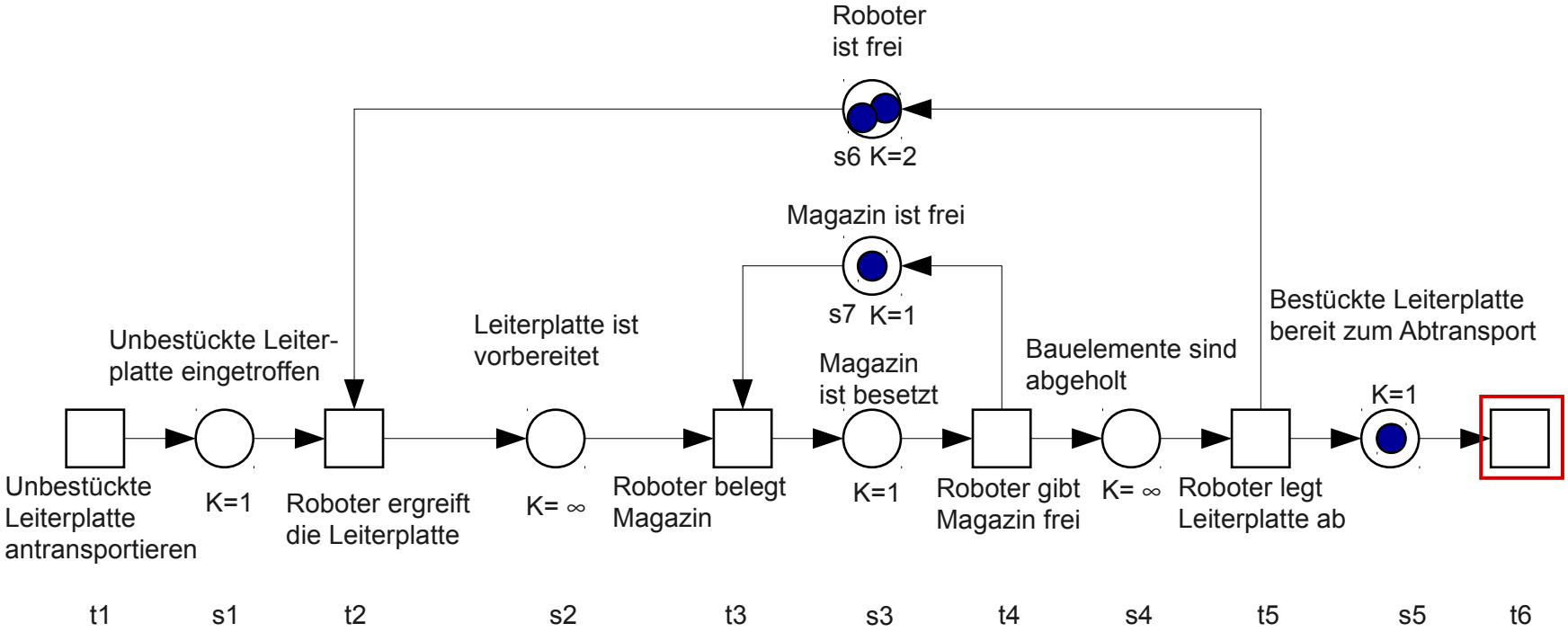
Beispiel

S/T-Netz Bestückungsroboter



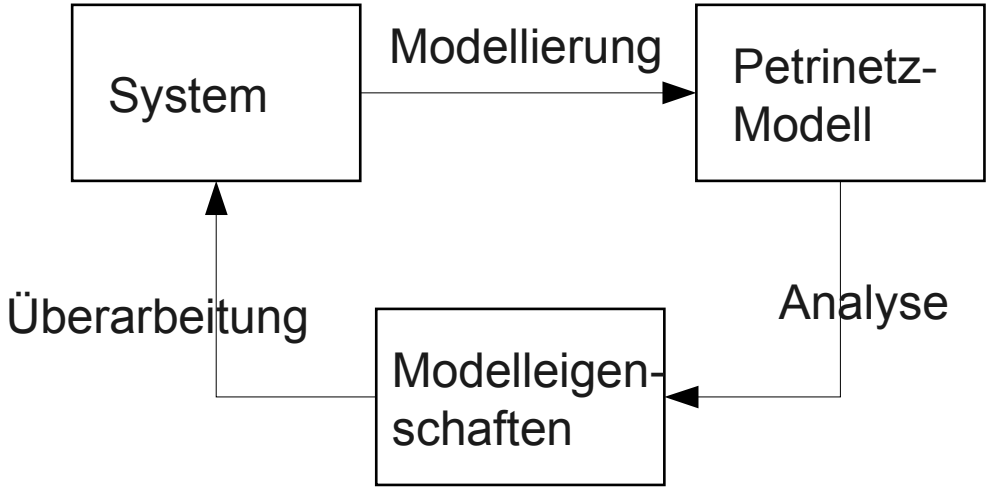
Beispiel

S/T-Netz Bestückungsroboter



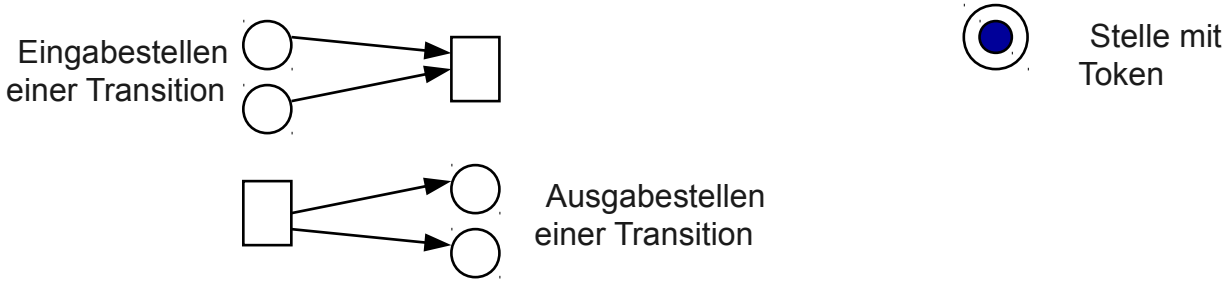
Petrinetze

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

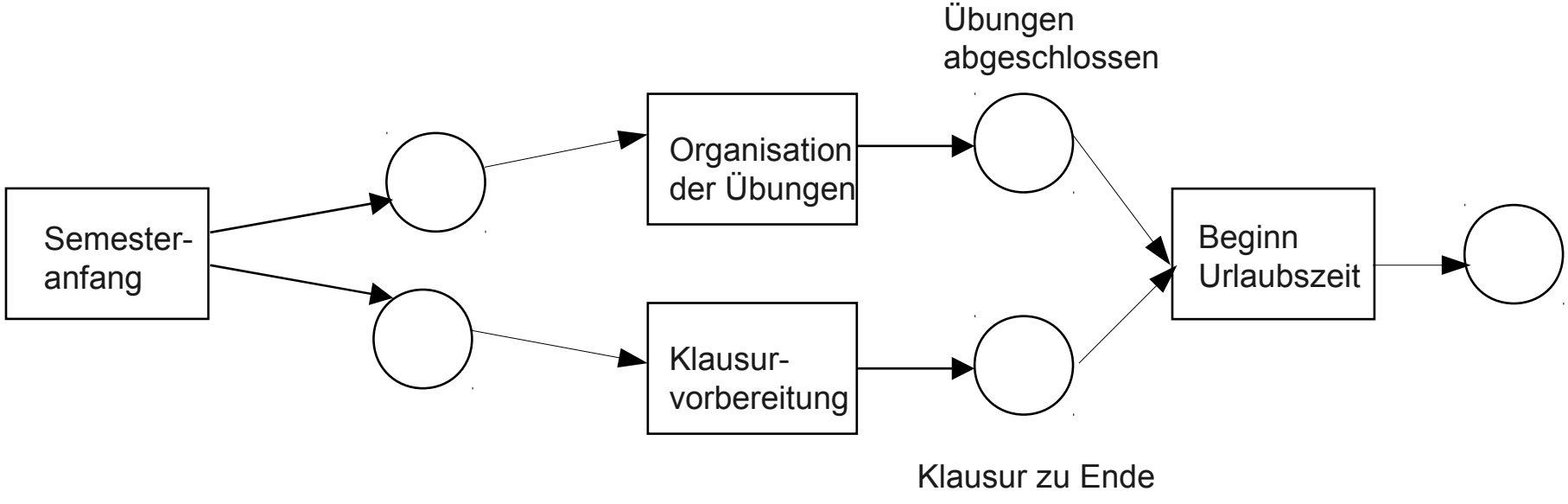


Petrinetz (informell)

- 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

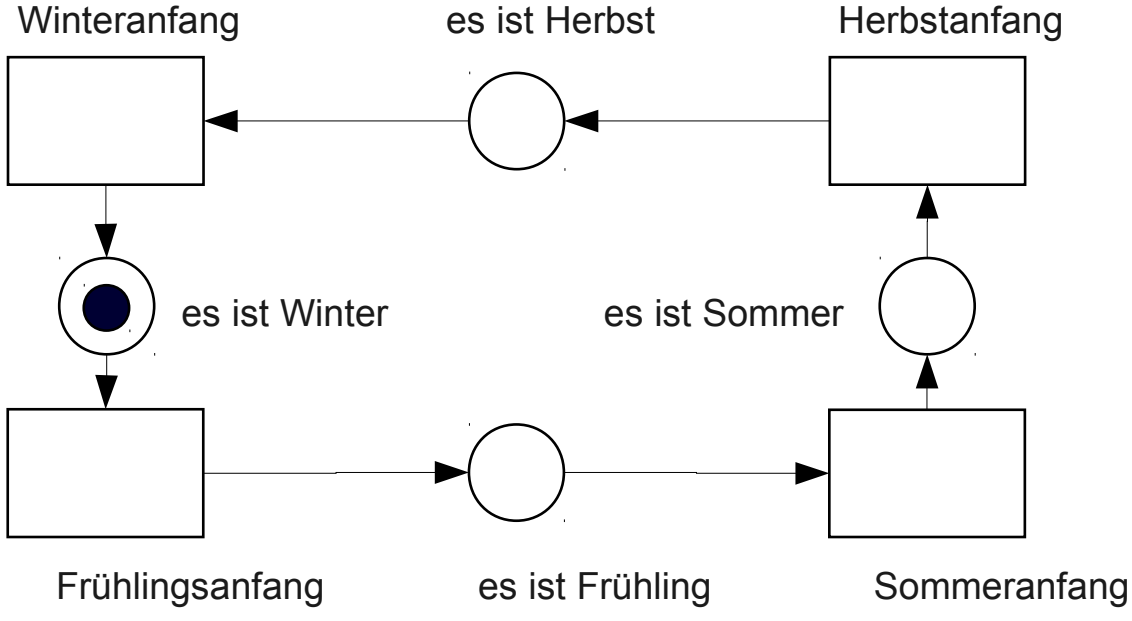


- Petrinetze erlauben einen natürlichen Begriff von Parallelität
- Lokalitätseigenschaft von Petrinetzen: Das Schalten einer Transition wird nur von ihrer direkten Umgebung beeinflusst und beeinflusst auch nur diese.



Formal definierte Semantik auf der Basis von Aussagenlogik

- Jede Stelle repräsentiert eine Aussage.
- s ist genau dann markiert, wenn die durch s repräsentierte Aussage wahr ist.
- 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!

Ein Beispiel

Definition

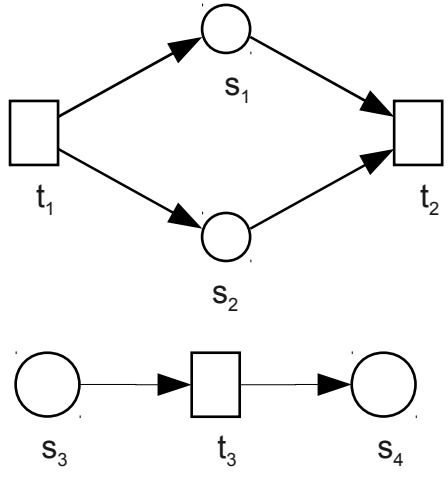
- Ein (Petri)-Netz N = (S,T,F) ist folgendermaßen definiert

  (i) $S \neq \emptyset$, $T \neq \emptyset$

  (ii) $S \cap T = \emptyset$

  (iii) $F \subseteq S \times T \cup T \times S$

- „Petri“ in Klammern, weil zu Petrinetz auch eine Festlegung des dynamischen Verhaltens gehört.
- $s \in S$ heißen S-Elemente oder **Stellen**,
  $t \in T$ heißen T-Elemente oder **Transitionen**,
  $f \in F$ heißen **Kanten**.
- K: $S \rightarrow \mathbb{N} \cup \{\infty\}$ erklärt eine (möglicherweise unbeschränkte) **Kapazität** für jede Stelle.
- W: $F \rightarrow \mathbb{N}$ bestimmt zu jedem Pfeil des Netzes ein **Gewicht**.
- M_0: $S \rightarrow \mathbb{N}_0$ ist eine **Anfangsmarkierung**, die die Kapazitäten respektiert, d.h. für jede Stelle $s \in S$ gilt: $M_0(s) \leq K(s)$. Eine Stelle s kann mit einer Höchstbelegung K annotiert werden.



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) }

Stellen/Transitions-Netze

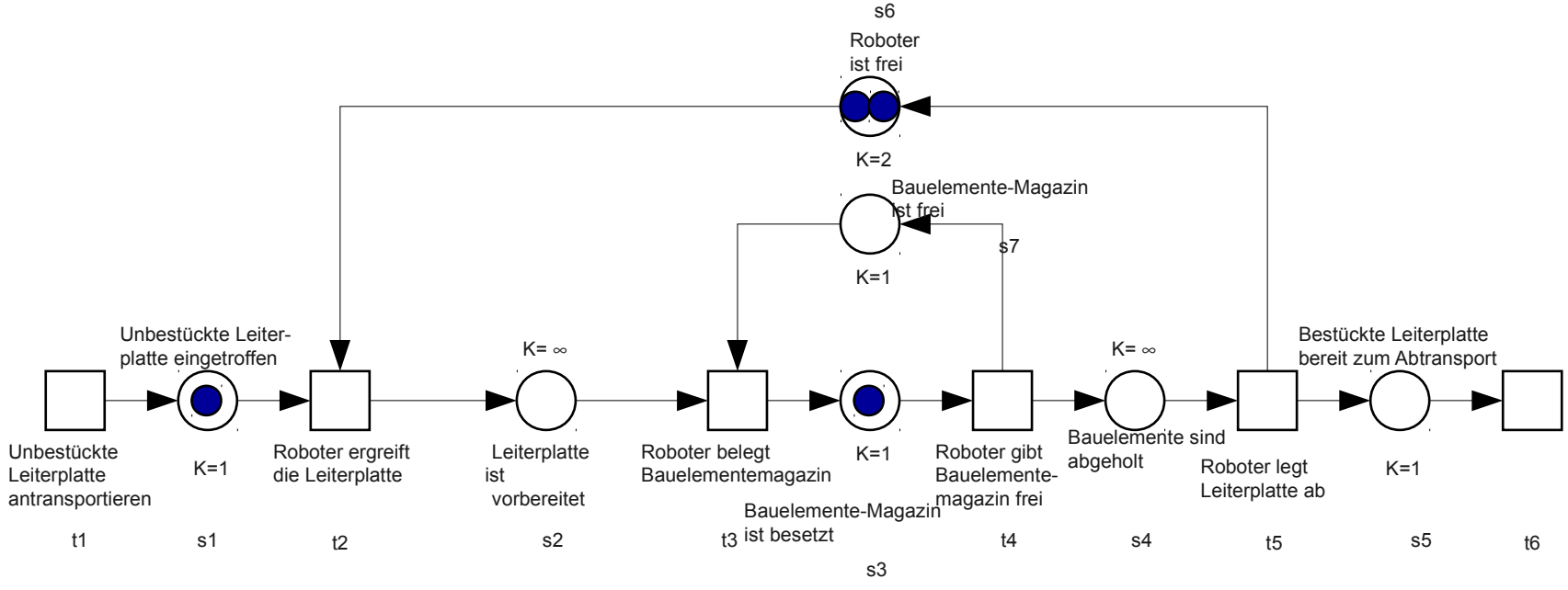
- Anzahl der Marken pro Stelle wird betrachtet
  - Zustandsbetrachtung
  - Was kann im nächsten Zustand passieren?

- Verschiedene Arten von Petrinetzen mit anonymen Marken
  - Stellen/Transitions-Netz (S/T-Netz)
  - [Bedingungs/Ereignis-Netze (B/E-Netz), nicht Teil der Vorlesung]

Stellen/Transitions-Netze

- Markierung allgemein: „Verteilung der Marken auf die Stellen“
- 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 | ∃ w ∈ T*mit M_0[w>M} heißt Erreichbarkeitsmenge des Systems

S/T-Netz Bestückungsroboter

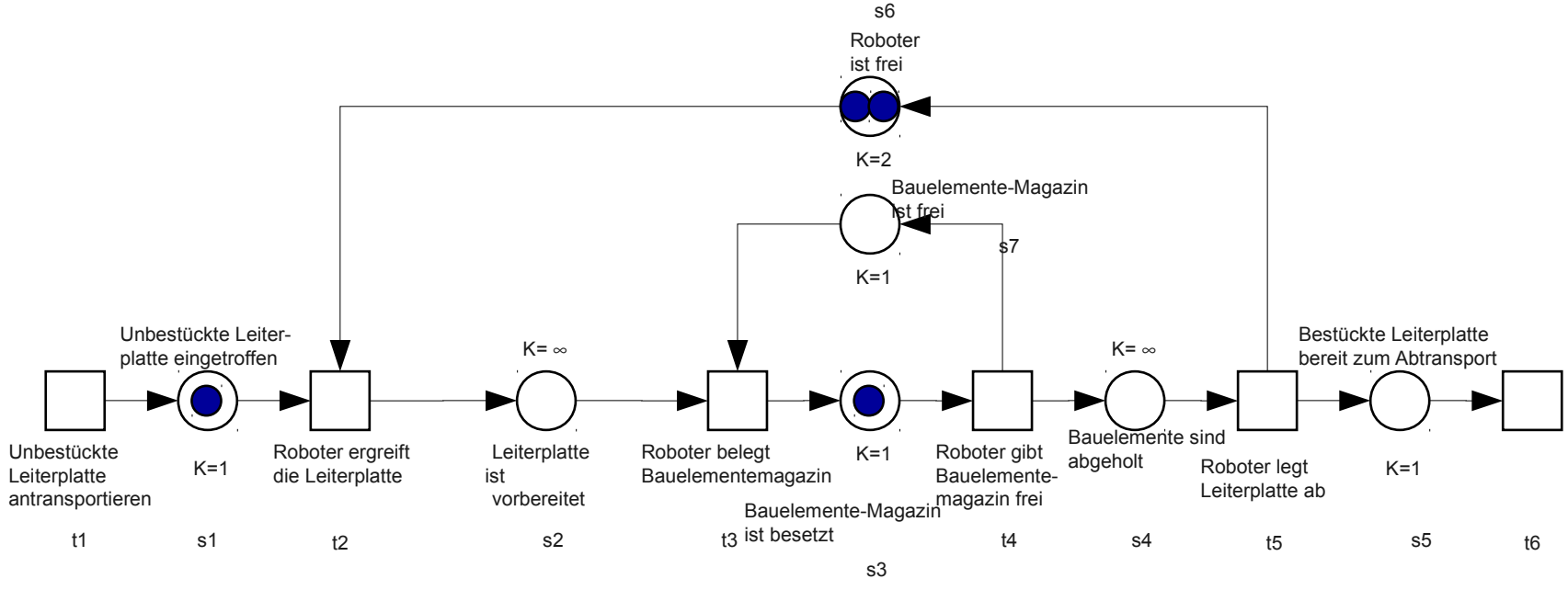


Erreichbarkeit

Erreichbarkeitsalgorithmus (breadth-first)

1. Trage in ein Schema mit den Spalten „Markierungsnummer“, „Markierung“ und „Schaltungen“ die Anfangsmarkierung M_i ein
2. In der aktuellen Markierung M_i wird jede Transition t untersucht, ob sie aktiviert ist
   - t nicht aktiviert: fertig
   - t aktiviert: Berechne Folgemarkierung M_{i+1} (d.h. $M_i [t>M_{i+1}$)
     - Neue Markierung: Trage neue Zeile mit Markierungsnummer M_i ein
     - Existierende Markierung: Fertig
1. Alle Transitionen überprüft, gilt M_i als erledigt
2. Prüfen, ob alle eingetragenen Markierungen erledigt sind
   - Ja: Erreichbarkeitsanalyse abgeschlossen
   - Nein: Überprüfe die nächste Markierung und fahre bei 2 fort.

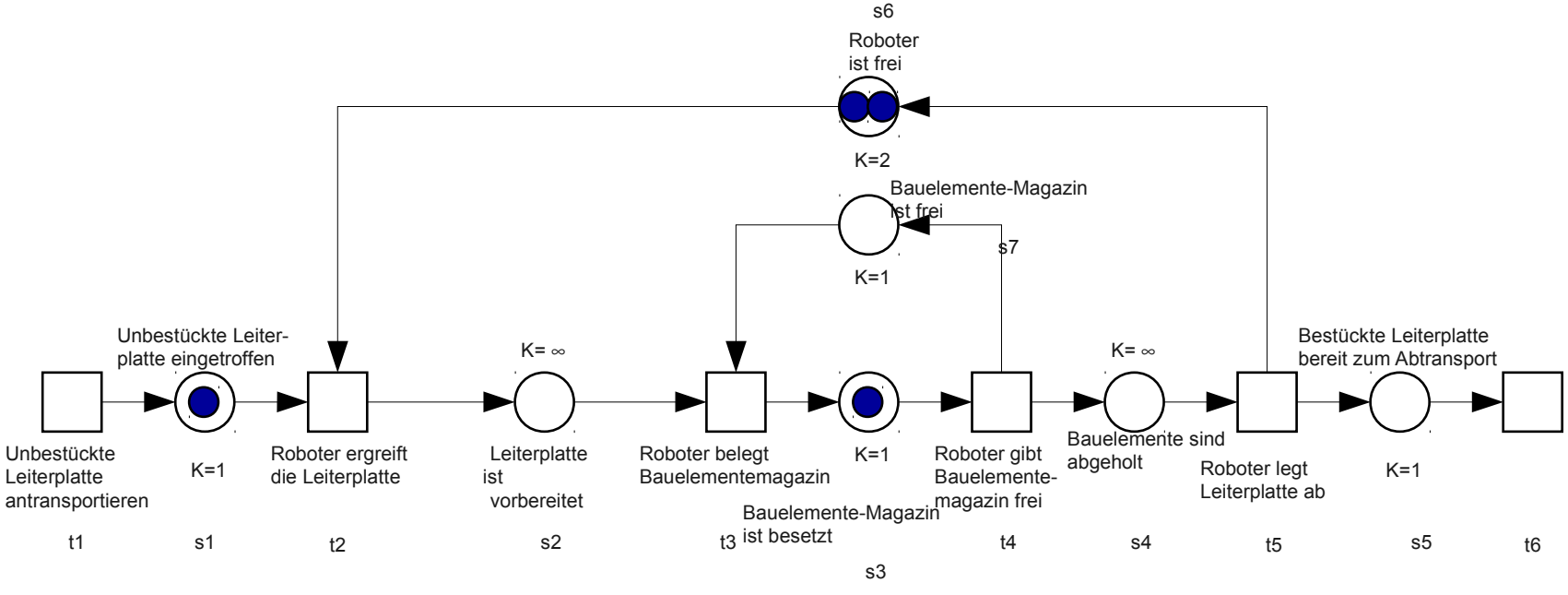
Zur Erinnerung: S/T-Netz Bestückungsroboter



Erreichbarkeit

- **Erreichbarkeitsmenge** im Beispiel:

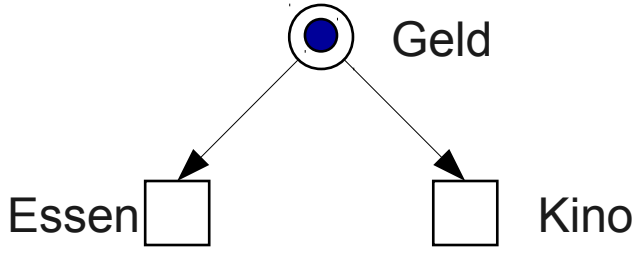
Nr.	s1	s2	s3	s4	s5	s6	s7	Schaltungen
M0	1	0	1	0	0	2	0	t2->M1 t4->M2
M1	0	1	1	0	0	1	0	t4->M3
M2	1	0	0	1	0	2	1	t2->M4 t5->M5



Grundsituationen in S/T-Netzen

Konflikt

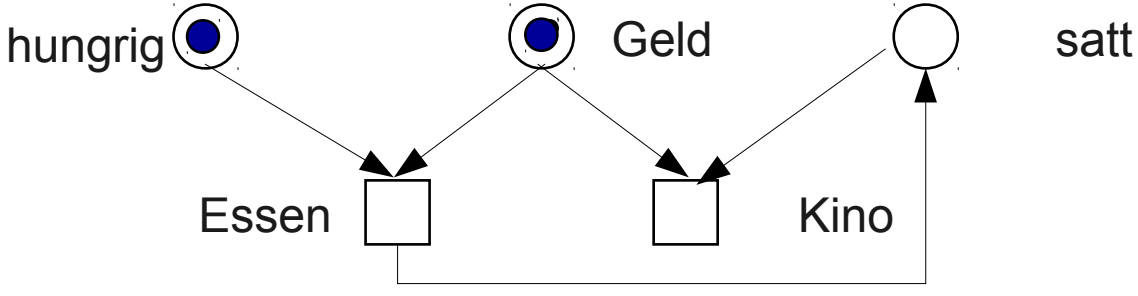
- Nicht-nebenläufige gleichzeitige Aktiviertheit von Transitionen
- Konfliktierende Transitionen nehmen sich gegenseitig Input-Marken weg
- Def.: Zwei Transitionen t_1 und t_2 eines S/T-Systems mit Höchstbelegung $K=\infty$ $\forall s \in S$ sind im *Konflikt*, wenn t_1 und t_2 aktiviert sind (geschrieben: $M[t_1>$ 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

Konflikt

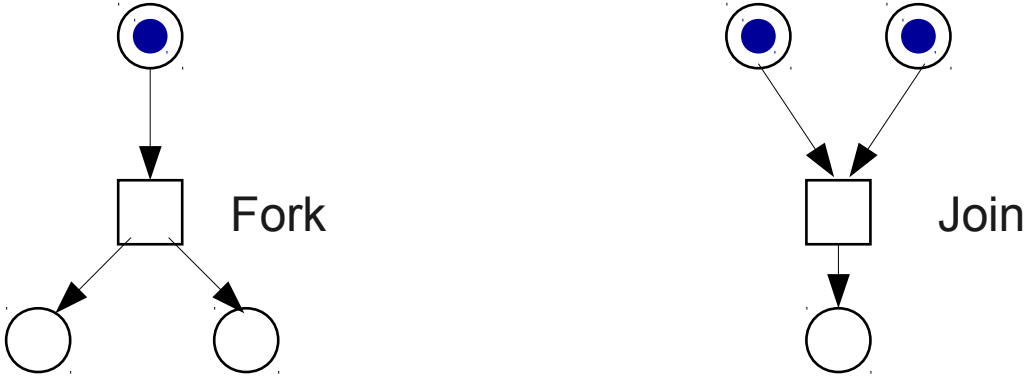
- Konfliktlösende Einbettung
  - Ist durch Einführung zusätzlicher Eingabestellen möglich.
  - Hierdurch wird Information über zu wählende Alternative ergänzt



Grundsituationen in S/T-Netzen

Synchronisation

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

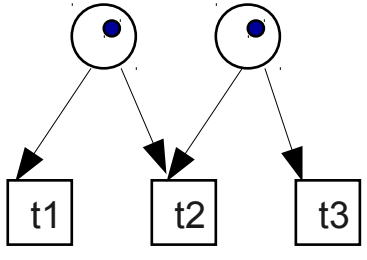


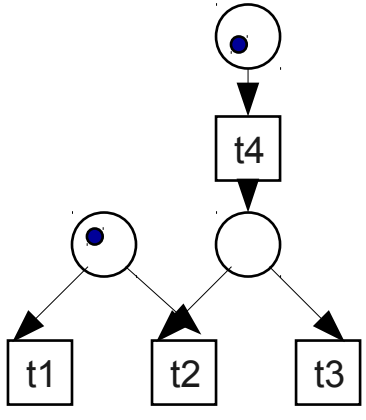
Grundsituationen in S/T-Netzen

Konfusion ff.

- 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)



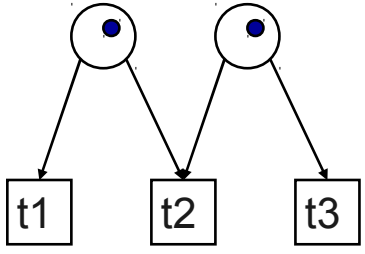


Grundsituationen in S/T-Netzen

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 ≠kft(t1,M[t3>)=∅

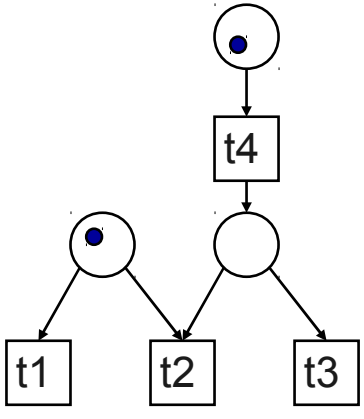
kft(t1,M) ⊇ kft(t1,M[t3>)

->Konfliktverminderung

->symmetrische Konfusion

Asymmetrische Konfusion

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



Konflikteinführung

kft(t1,M)= ∅

kft(t2,M)= ∅

kft(t1,M)= ∅ ≠ kft(t1,M[t4>)=t2

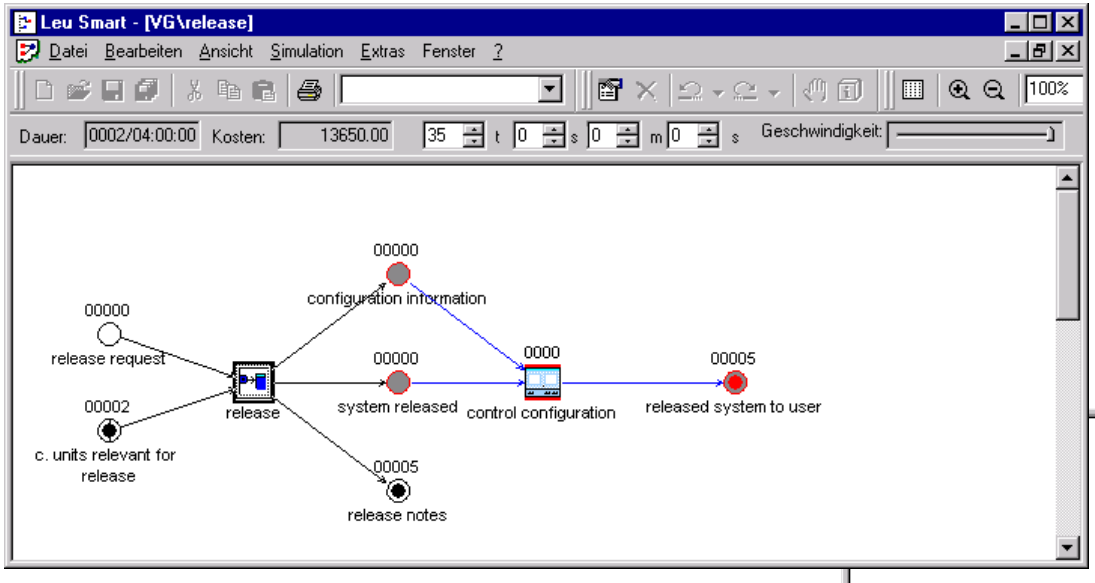
kft(t1,M) ⊆ kft(t1,M[t2>)

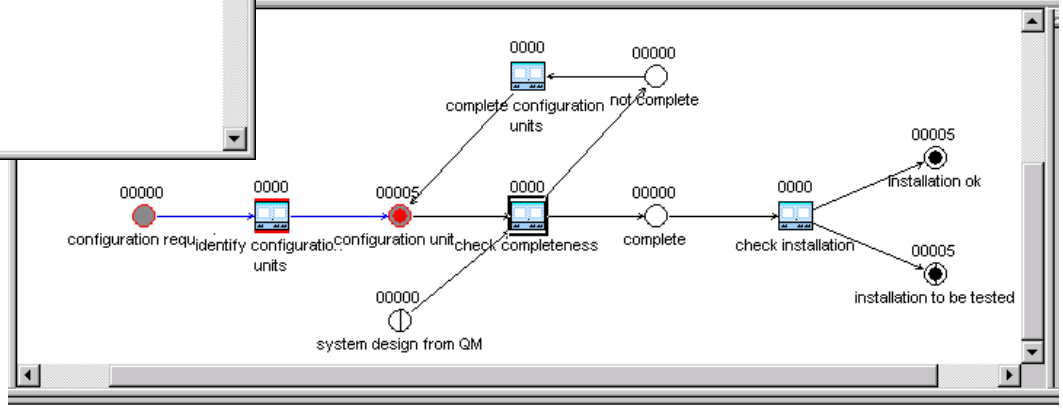
->Konfliktvergrößerung

->asymmetrische Konfusion

Simulation von Petrinetzen

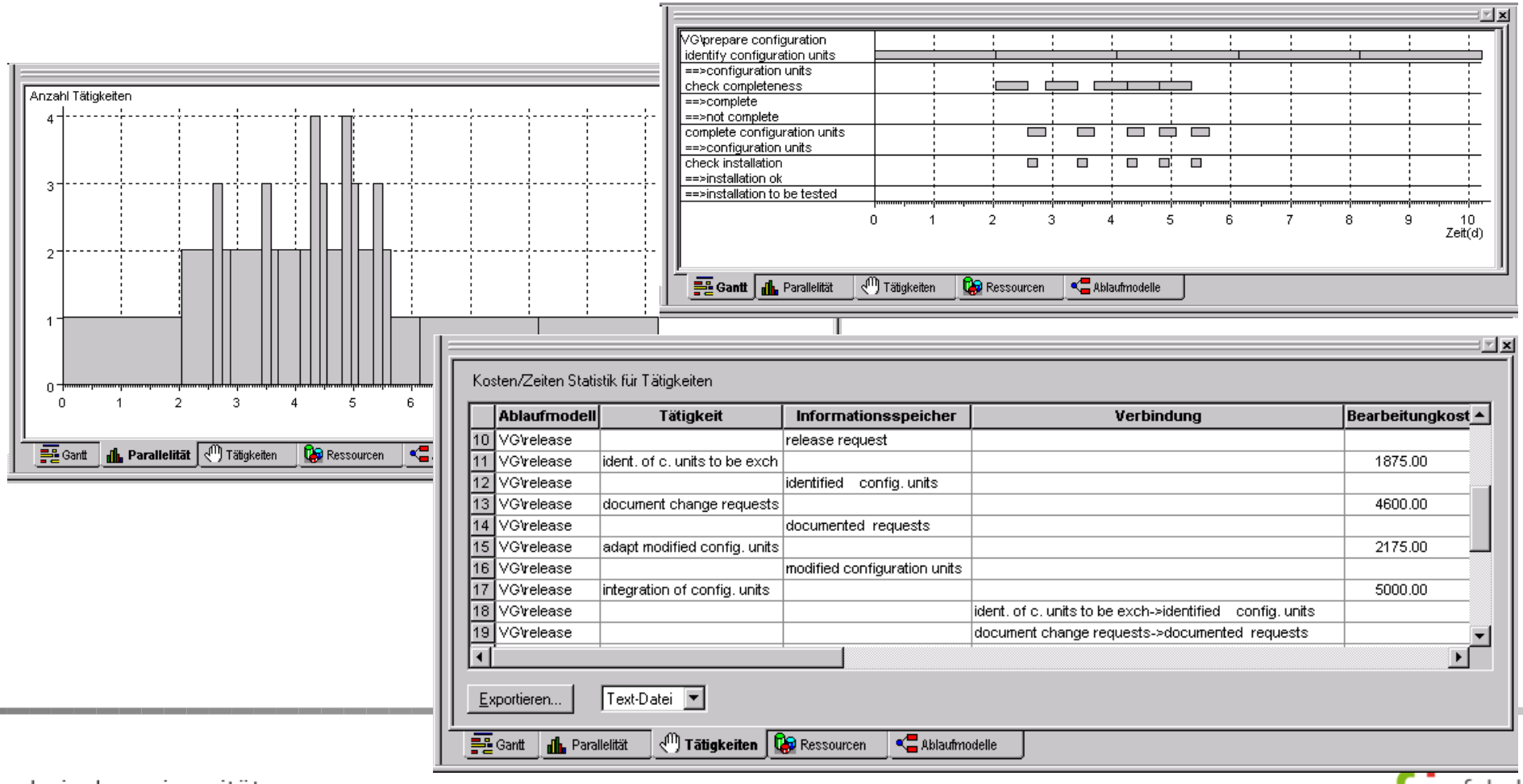
- Animation





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	

Analyse von Systemen

- 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. Vorlesung vorherige Woche)

Analyse von Systemen

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

Sicherheit

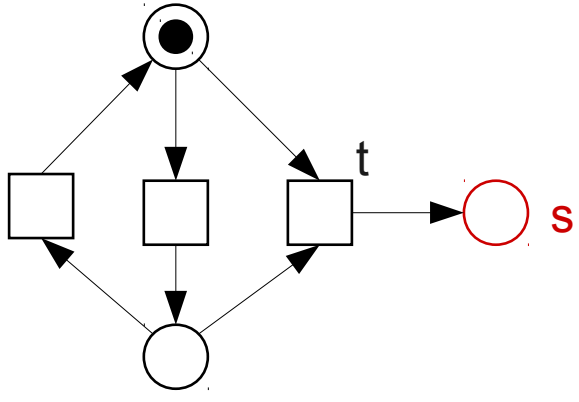
- Seien Y=(S,T,F,K,M_0) ein S/T-System und B: S $\rightarrow \mathbb{N}_0 \cup \{\infty\}$ eine Abbildung, die jeder Stelle eine „kritische Markenzahl“ zuordnet. Y heißt **B-sicher** bzw. **B-beschränkt**, wenn für alle erreichbaren Markierungen die Anzahl der Markierungen pro Stelle durch B begrenzt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

Sicherheit

- Beispiel: Verkehrsplaner modelliert Ampelsystem. An einer bestimmten 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.



Lebendigkeit

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

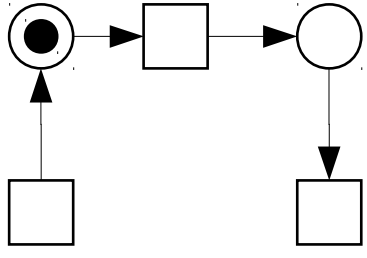
  $\exists M_1 \in [M_0>: M_1[t>$

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

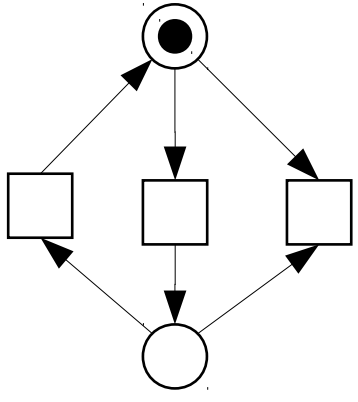
  $\forall M_1 \in [M_0>: \exists M_2 \in [M_1>: M_2[t>$

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

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



Lebendige Transition



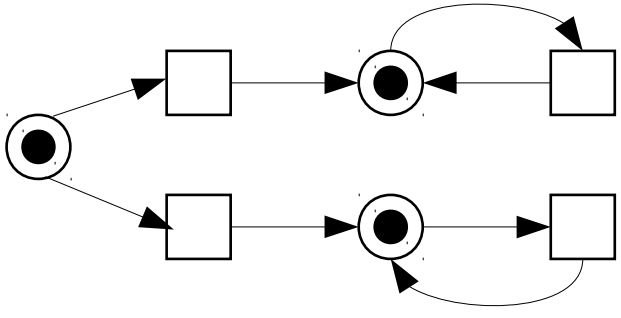
Tote Transition

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



Lebendigkeit ff.

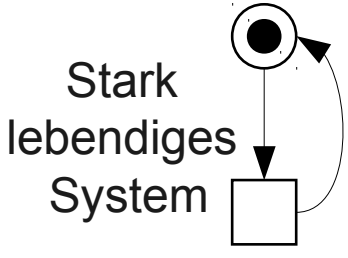
- Ein S/T-System heißt **lebendig** oder **stark lebendig**, wenn aus jeder erreichbaren Markierung jede Transition aktivierbar ist:

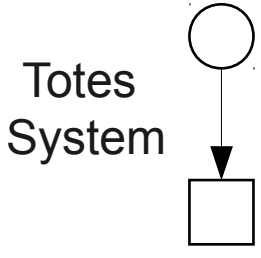
  $\forall M_1 \in [M_0>: \forall t \in T: \exists M_2 \in [M_1>: M_2[t>$

  - 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 Deadlocks





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.

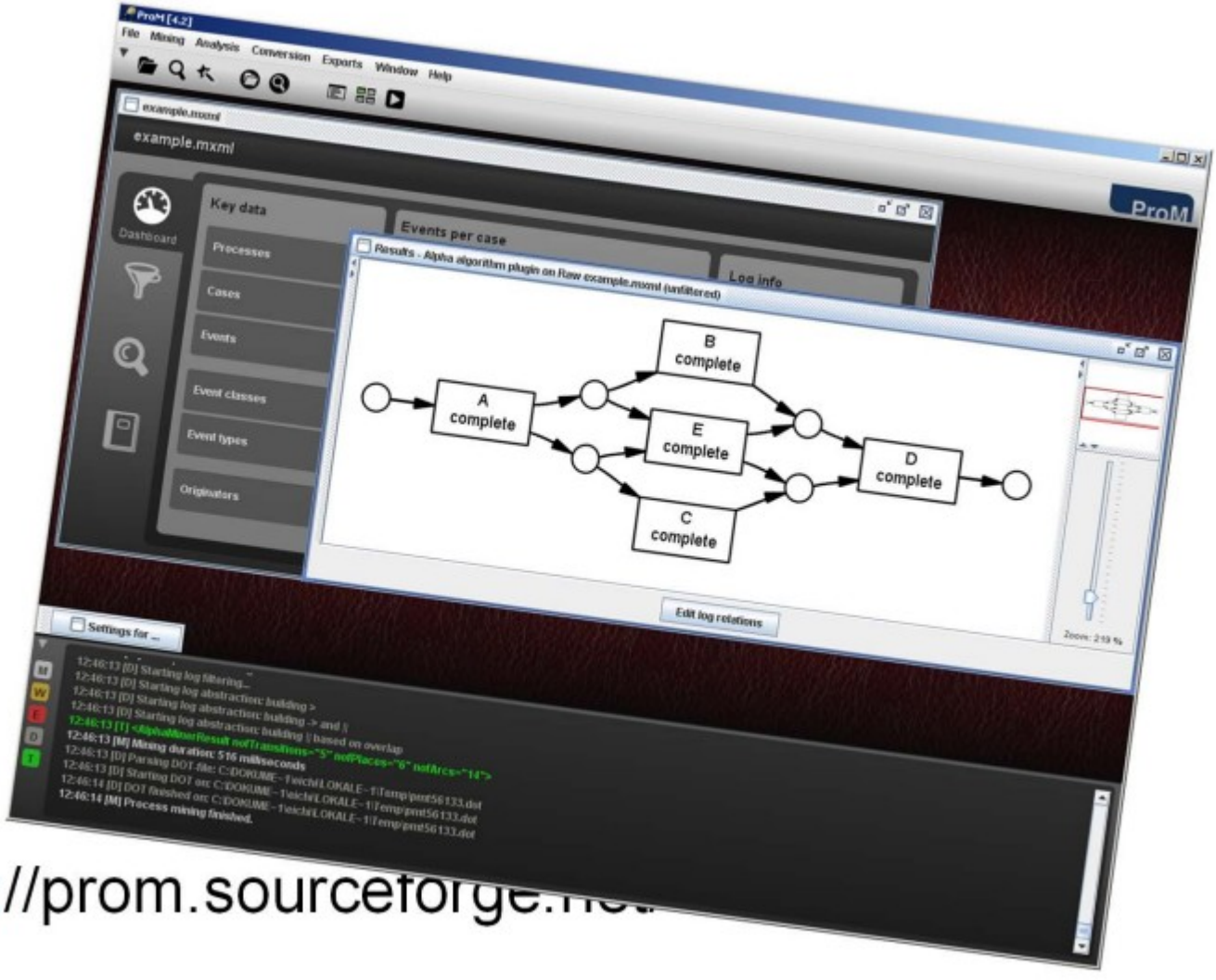
Methodik

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

Beispielanwendung: Process Mining mit ProM



http://prom.sourceforge.net

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

Wir beschäftigten uns mit Petri-Netzen als Grundlage für die Spezifikation von internen Systemzustände und Interaktion zwischen verschiedenen Softwaremodulen.

Petri-Netze bilden die Grundlage für Spezifikations-Ansätze in der Praxis, die wir uns im nächsten Kap. 06 ansehen werden (vgl. Ausführungssemantik von Aktivitätsdiagrammen in UML 2.x auf Basis von Petri-Netzen).

Ausserdem werden Petri-Netze zum Teil auch direkt in der Praxis eingesetzt, beispielsweise in dem Process-Mining-Werkzeug ProM.

Literatur

- Baumgarten, B.: Petri-Netze, Spektrum, 1996
- Kiencke, U.: Ereignisdiskrete Systeme, Oldenbourg, München, 1997
- Oberweis, A.: Modellierung und Ausführung von Workflows mit Petri-Netzen, Teubner, Stuttgart, 1996
- Peterson, J.L.: Petri Net Theory and the Modelling of Systems, Prentice Hall, London, 1981
- Reisig, M.: Petri-Nets, Springer, Berlin, 1985
- Schöning, U.: Ideen der Informatik, Oldenbourg, München, 2002
- Starke, P.H.: Analyse von Petri-Netz Modellen, Teubner, Stuttgart, 1990