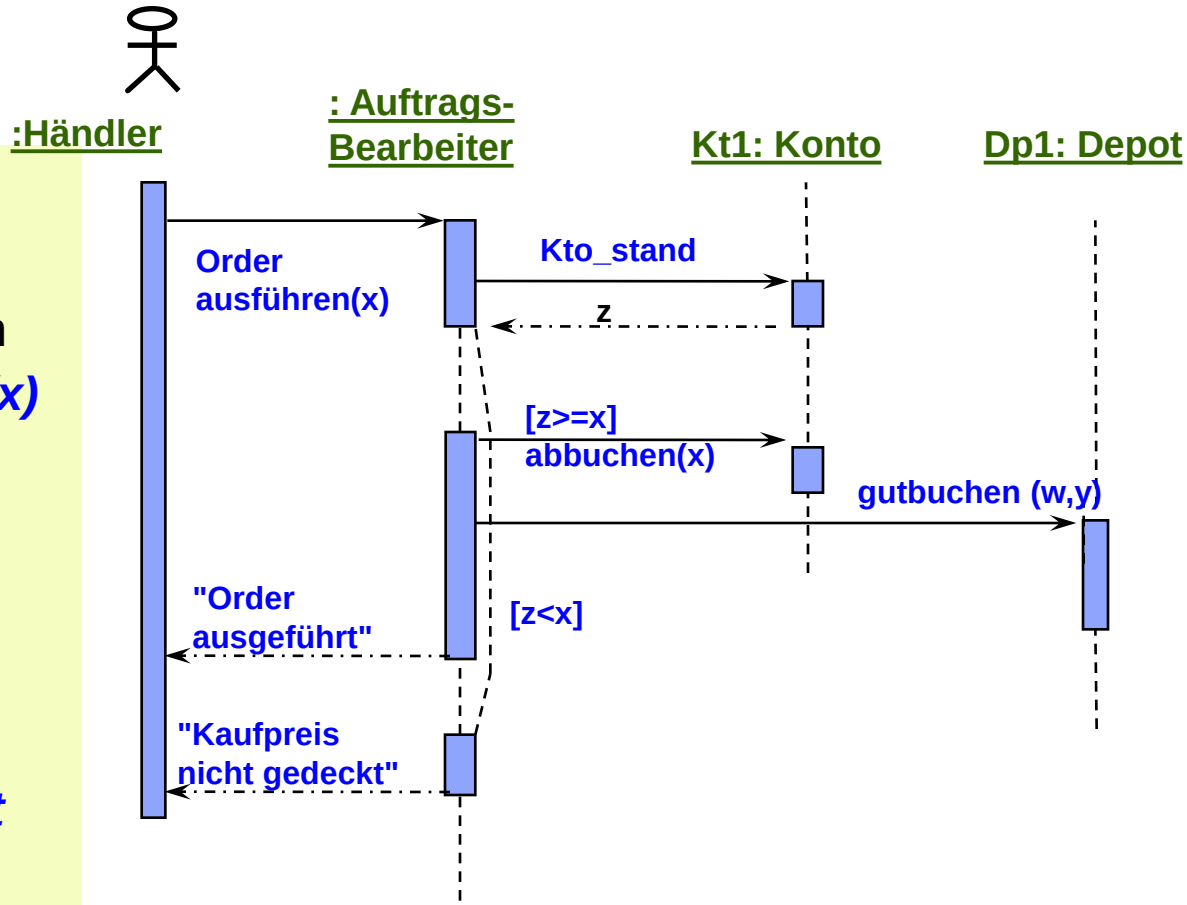
Vorlesung/Seminar: Modellierung dynamischer und adaptiver Systeme ,
Wintersemester 2011/12

Kap. 3: Modellierung und Simulation dynamischer Systeme

- **Ansätze für die Modellierung von Systemdynamik**
- **Naiver Ansatz: UML- Interaktionsdiagramme**
- **Mathematischer Ansatz: Arbeitsbeispiel Weltmodell**
- **Beispiele für weitere Ansätze (kausal, Automaten, ..): Zustands- Aktivitätsdiagramme**
- **Petri-Netze**

Naiver Ansatz: Beispiel: UML-Interaktionsdiagramm

- Ein Händler initiiert ***Order ausführen***:
- Wenn Kaufpreis x gedeckt durch Kontostand z, dann ***abbuchen (x)*** von Konto **Kt1**, ***gutbuchen (w,y)*** y Stück von Wertpapier w auf Depot **Dp1**,
- Bestätigung an Kunden *"**Order ausgeführt**"*
- sonst Meldung *"**Kaufpreis nicht gedeckt**"*



Mathematischer Ansatz: Arbeitsbeispiel Weltmodell – Schritt 1

1. Schritt: Wortmodell

".. Wir beobachten weltweit eine zunehmende Belastung der natürlichen Ressourcen und .. Umwelt. Sie ergibt sich vor allem aus der ständigen Zunahme der Bevölkerung und den .. damit verbundenen Verbräuchen verschiedenster Rohstoffe .. sowie Abgaben von Abfallstoffen an die Umwelt.

Eine wichtige Bestimmungsgröße dieser .. Belastung ist der Verbrauch an Rohstoffen und Energie. Dieser steigt tendenziell mit der wachsenden Umweltbelastung (z.B. durch mehr Aufwendungen für Umweltschutz und Abbau).

Mit wachsendem Konsum verbessern sich die Versorgungsmöglichkeiten – mit entsprechendem Einfluss auf die Bevölkerungsentwicklung. Die wachsende Umweltbelastung .. sowie die schwindende Ressourcenbasis haben Rückwirkungen auf die Gesundheit und Lebenserwartung der Bevölkerung.

Umweltbelastung und Ressourcenabbau führen zu wachsenden gesellschaftlichen Kosten, die wiederum zunehmendes gesellschaftliches Handeln erwarten lassen, um schädlichen Entwicklungen zu begegnen. .."

(n. H. Bossel, [Bos 04])

Arbeitsbeispiel Weltmodell – Schritt 2

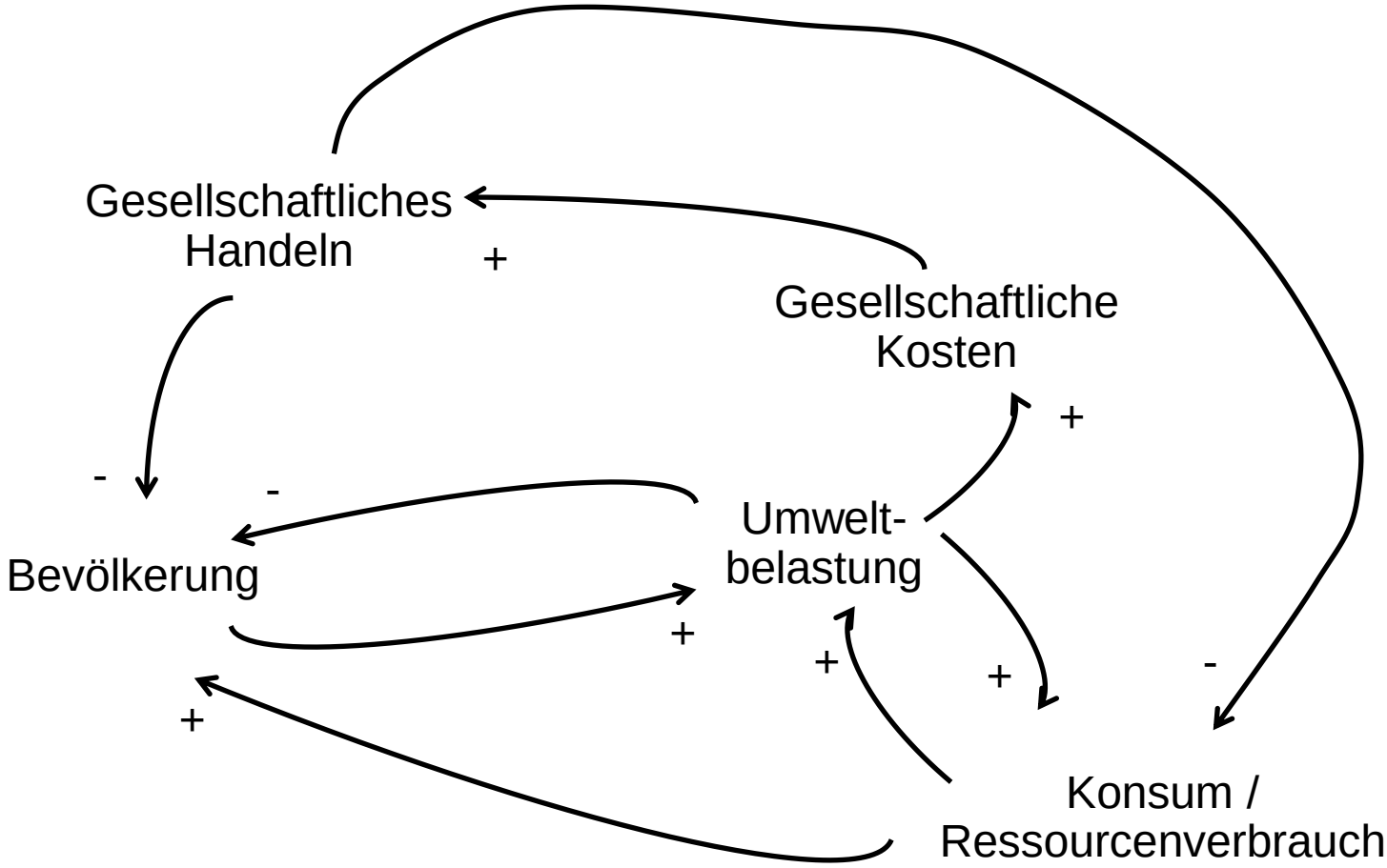
2. Schritt: Wirkungsbeziehungen

- Wenn die *Bevölkerung* wächst, so wächst auch die *Umwelt-* und *Ressourcenbelastung*.
- Wenn die *Umwelt-* und *Ressourcenbelastung* wächst, so wächst auch der *Ressourcen-verbrauch*.
- Wenn der *Ressourcenverbrauch* wächst, so wächst auch die *Umwelt-* und *Ressourcen-belastung*.
- Wenn sich der *Ressourcenverbrauch* erhöht (und sich damit die materiellen Bedingungen verbessern), so erhöht sich damit auch die *Bevölkerungszahl*.
- Wenn sich die *Umwelt-* und *Ressourcenbelastung* erhöht, so vermindert sich die *Bevölkerungszahl*.
- Wenn sich die *Umwelt-* und *Ressourcenbelastung* erhöht, so erhöhen sich damit auch die *gesellschaftlichen Kosten*.
- Wenn sich die *gesellschaftlichen Kosten* erhöhen, so ist mit entsprechend mehr *gesellschaftlichem Handeln* zu rechnen.
- *Gesellschaftliches Handeln* wird dafür sorgen, dass bei zu starkem *Bevölkerungswachs-tum* dieses reduziert wird.
- *Gesellschaftliches Handeln* wird dafür sorgen, dass bei zu hohem *Ressourcenverbrauch* dieser reduziert wird.

(n. H. Bossel, [Bos 04])

Arbeitsbeispiel Weltmodell – Schritt 3

3. Schritt: Wirkungsgraph



(n. H. Bossel, [Bos 04])

Arbeitsbeispiel Weltmodell – Schritt 4

4. Schritt: Wirkungsmatrix

Gebergrößen / Zielgrößen	Bevölkerung	Umweltbelastung	Konsum	Handeln
Bevölkerung	0	-0.1	0.3	-0.1
Umweltbelastung	1	0	1	0
Konsum	0	1.1	0	-1
Handeln	0	C	0	0

C = später zu spezifizierender Eingriffsparameter

In der Wirkungsmatrix wird die im Wirkungsgraphen enthaltene Information systematisch dargestellt und quantifiziert.

Beispiel:

Wenn die Umweltbelastung um 1 % steigt, dann nimmt die Bevölkerung um 0.1 % ab.

(n. H. Bossel, [Bos 04])

Arbeitsbeispiel Weltmodell – Schritt 5

5. Schritt: Mathematisches Modell und Simulation

Aus der Wirkungsmatrix werden – falls notwendig nach Aufteilung in Teilmodelle – *mathematische Formeln* und *(Differential-) Gleichungen* oder abgeleitet.

Beispiel:

Wenn die Umweltbelastung um x % steigt, dann nimmt die Bevölkerung um 0.x % ab

$\Rightarrow$ Bev' = Bev – Bev * [$\underbrace{(\text{UmBel}' - \text{UmBel})/ \text{UmBel}}_{x}$] /100.

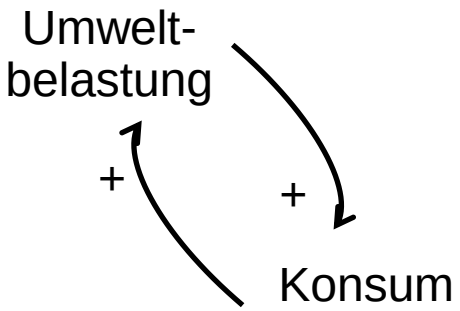
Aus den mathematischen Beziehungen (oder direkt aus der Wirkungsmatrix) wird ein *Simulationsprogramm* abgeleitet.

Beispiele für Simulationsprogramme:

- *Papier-Computer* , Spiel *"Ökolopoly"* von F. Vester [Ves 01]
- *Systemzoo* von H. Bossel [Bos 04a]

Spezialfall *Positive Rückkopplung*

Beispiel aus dem Wirkungsgraph:



Prinzip *Wachstum*:

$x' = x + p*x$ p = Wachstumsfaktor, z.B.

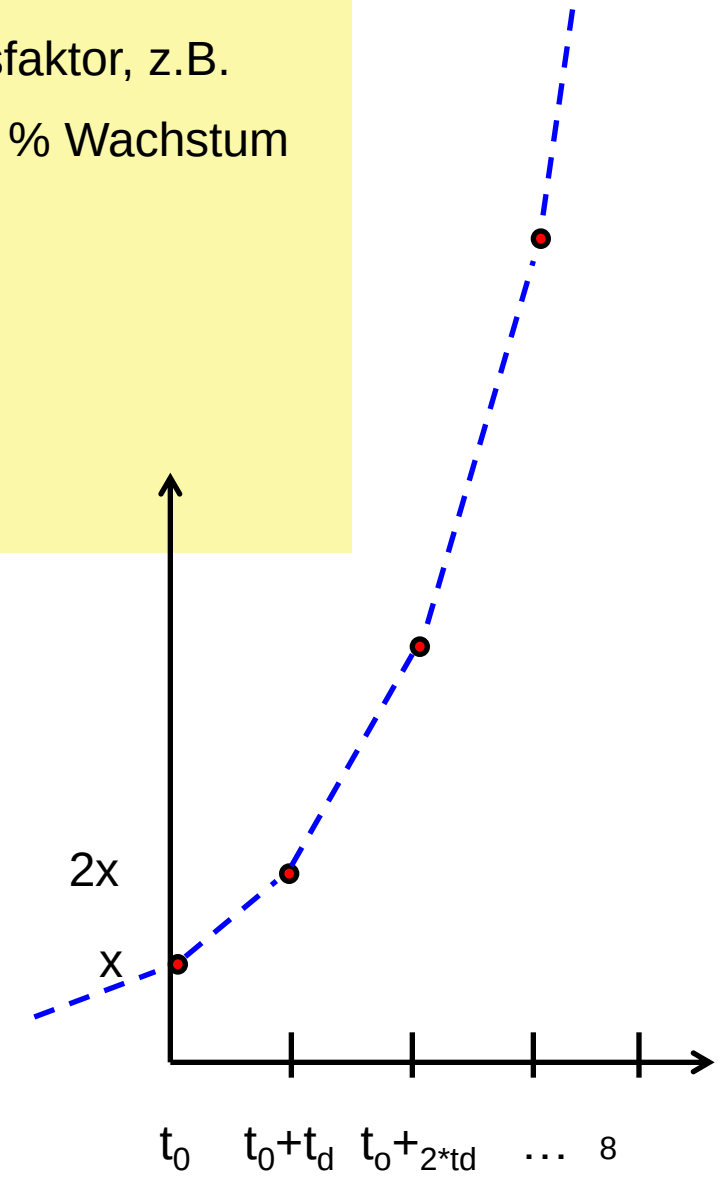
p = 0.05, d.h. 5 % Wachstum

$x' = x\,(1+p)$

$x'' = x'\,(1+p) = x\,(1+p)^2$

…

$x_n = x\,(1+p)^n$ d.h. x wächst *exponentiell*



Beispiele:

- 10% Wachstum: p = 0.10
  $\Rightarrow x_7 = 1.95$, d.h. Verdoppelung in ca. 7 Jahren
- 7 % Wachstum: p = 0.07

$\Rightarrow x_{10} = 1.97$, d.h. Verdoppelung in ca. 10 Jahren

Faustregel (die sog. 70-er-Regel)*:*

$p * t_{doppel} \cong 70$

Nachtrag zum "Weltmodell"

Weltmodelle wurden seit den 1970-er Jahren systematisch im Zuge der Diskussion um die *Grenzen des Wachstums* entwickelt (vgl. Meadows et al. [Mea 72], [Mea 92]).

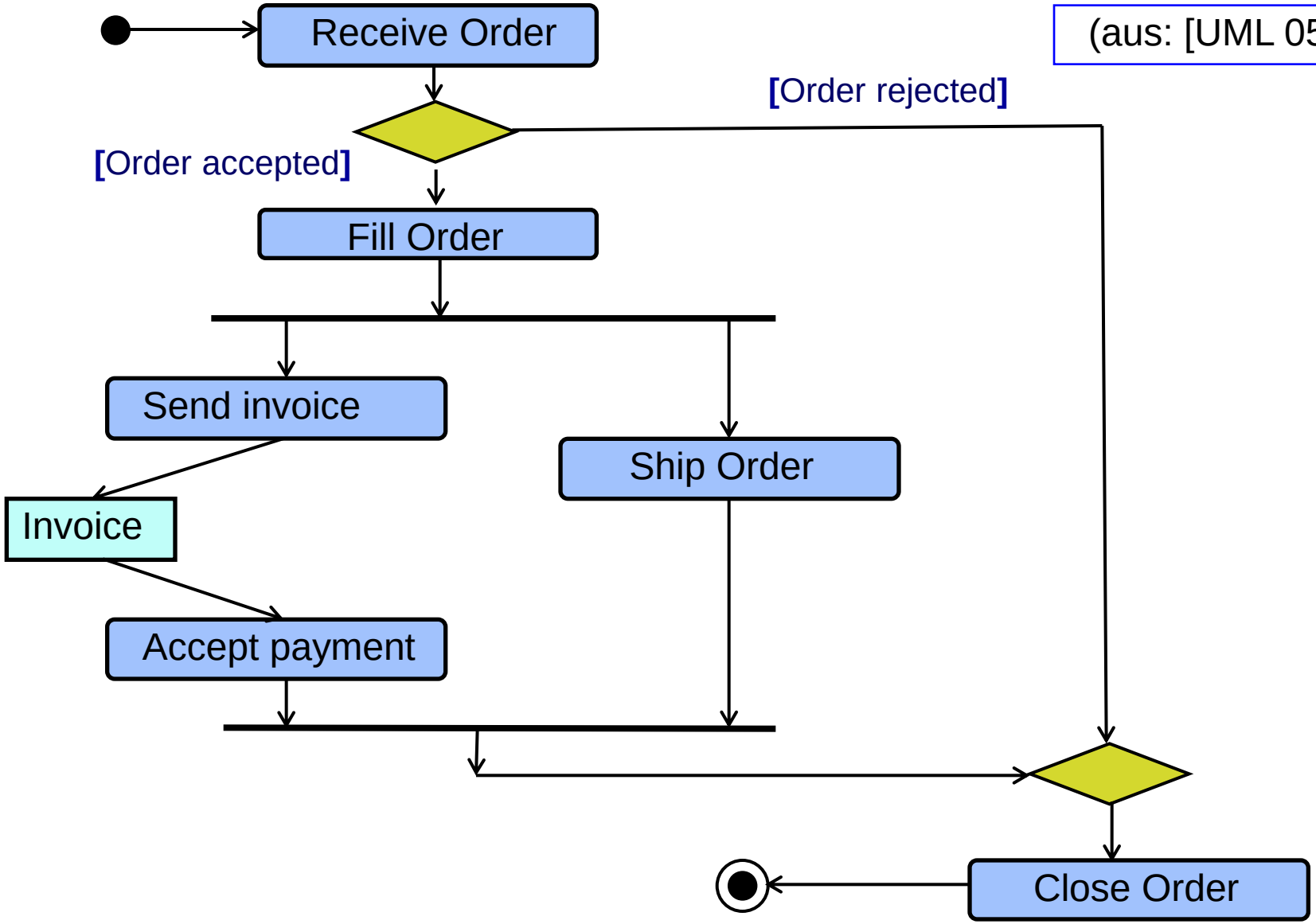
Das Modell *World3* von Meadows et al. ging auf das Weltmodell *World2* von Jay Forrester vom MIT (1970) zurück. Forrester war Entwickler der Methode *"System Dynamics"* , die großen Einfluss auf die Umwelt- und Klimaforschung hatte und auch dem Ansatz von H. Bossel zugrunde liegt.

World 3 ist ein sehr komplexes Modell mit 18 Zustandsgrößen, 60 Parametern, 52 Tabellenfunktionen und rund 200 Gleichungen für Zwischengrößen und Veränderungs-raten. Bossels "Miniwelt" ist einfacher und kleiner – weist aber qualitativ ähnliches Verhalten (und ähnliche Prognosen) auf.

Danach wird die Bevölkerung (und parallel dazu die Industrie- und Nahrungsmittel-produktion) zunächst weiter ansteigen, erhöhte Umweltbelastungen nach sich ziehen und schließlich zu einem "Kippen" der Bevölkerungsentwicklung führen.

(vgl. [Bos 04], S. 109ff.)

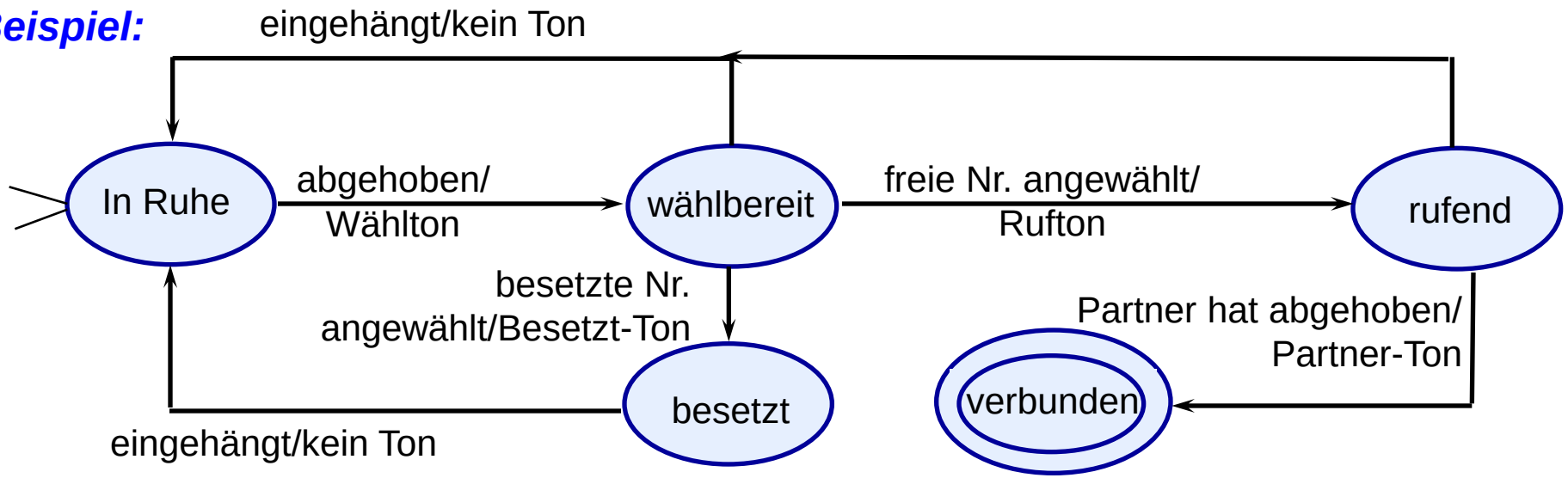
Kausaler Ansatz: Beispiel UML-Aktivitätsdiagramm



(aus: [UML 05], p. 325)

Automaten-Ansatz: Beispiel: Zustandsdiagramm

Beispiel:

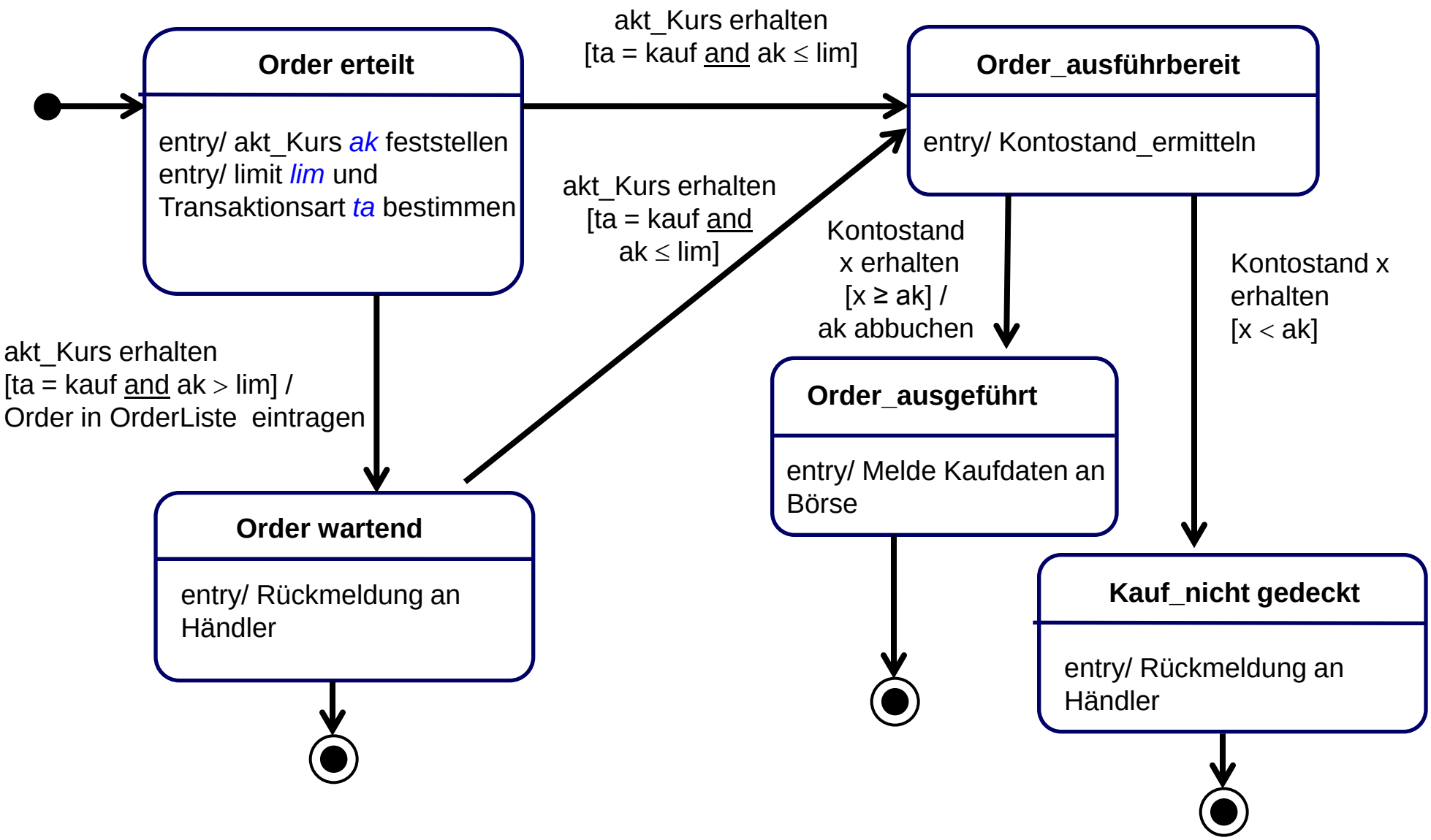


Automaten-Ansatz: Beispiel: Zustands-Matrix

Ereignis / Zustand	eingehängt	abgehoben	besetzte Nr. angew.	freie Nr. angew.	Pa. hat abgeh.	Rück-meldung
in Ruhe		wählbereit				Kein Ton
wählbereit	in Ruhe		besetzt	rufend		Wählton
besetzt	in Ruhe					Besetzt-Ton
rufend	in Ruhe				verbunden	Ruf-Ton
verbunden						Partn.-Ton

- ***Zeilen:*** markiert durch alle möglichen Zustände
- ***Spalten:*** markiert durch alle möglichen Ereignisse
- ***Matrixelemente (Zellen):*** markiert durch Folgezustände
- ***Zusätzliche Spalte:*** für System-Rückmeldungen

Automaten-Ansatz: Beispiel: UML-Zustandsdiagramm



Aktionen in Zustandsdiagrammen

- Zustandsdiagramme können durch die Definition von *Aktionen* verfeinert werden. *Aktionen* sind atomare (d.h. nicht zeitbehaftete, ununterbrechbare) Routinen, die während eines Ablaufs ausgeführt werden.
- Aktionen können auftreten:

  - bei *Zuständen*: Dort sind sie definiert als

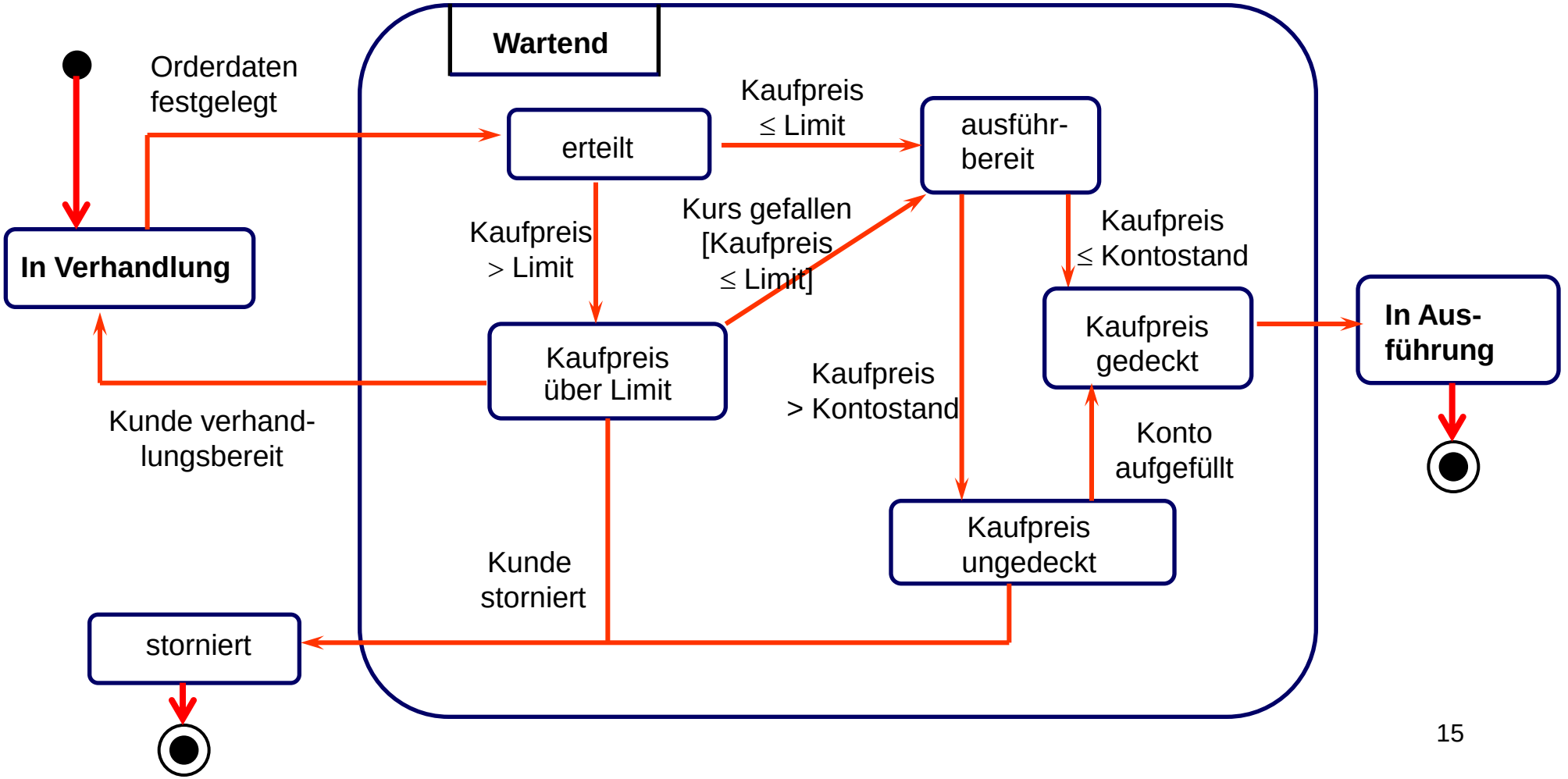
    `entry/<action>` (Eintrittsaktion – wird bei Erreichen des Zustands ausgeführt)
    `exit/<action>` (Austrittsaktion – wird bei Verlassen des Zustands ausgeführt)
    `do/<action>` (Aktivität – wird kontinuierlich bis zum Verlassen des Zustands ausgeführt) oder als
    `<event>/<action>` (interne Transition – wird im Zustands im Falle des Ereignisses **`<event>`** ausgeführt)

  - bei *Zustandsübergängen*: Dort haben sie die Form

    `<event>/<action>` (Übergangsaktion – wird beim Zustandsübergang im Falle des Ereignisses **`<event>`** ausgeführt)

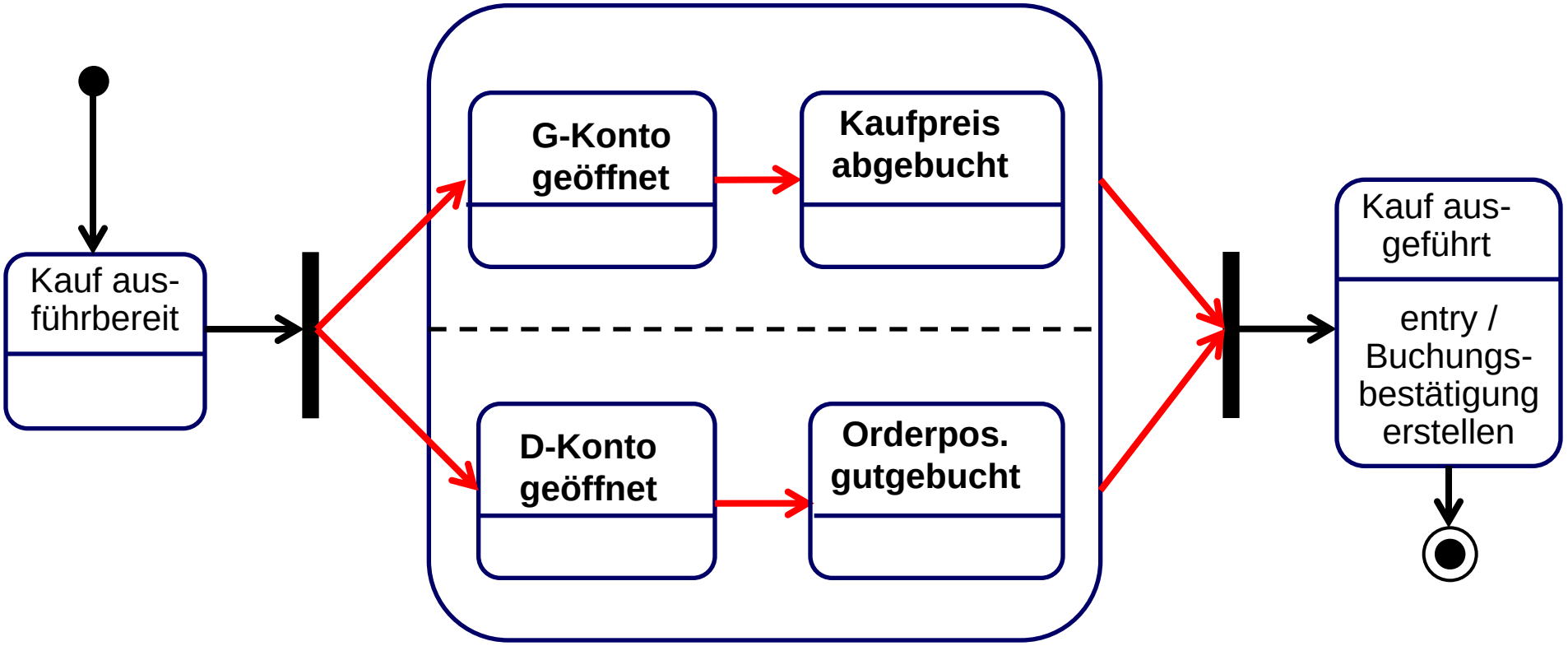
Zusammengesetzte Zustände

- Harel-Diagramme bieten die Möglichkeit, *zusammengesetzte Zustände* (mit einen eigenen Bezeichner) zu definieren. Diese bestehen aus *Unterzuständen* und werden durch ein umgreifendes Zustandssymbol gekennzeichnet.



Parallele Unterzustände

- *In einem zusammengesetzten Zustand* können die Unterzustände auch *parallel* angeordnet sein, d.h. die betreffenden (Unter-) Zustandsübergänge erfolgen *unabhängig* voneinander. .



Automaten-Ansatz: Beispiel Petri-Netze

Kurzbeschreibung:

Von C. A. Petri entwickelte Technik zur Darstellung von nebenläufigen Prozessen (vgl. [Pet 79], [Rei 90], [Sce 97]).

Petri-Netze sind ***bipartite Graphen***, bestehend aus zwei Arten von Knoten:

- den ***Stellen*** und
- den ***Transitionen*** sowie
- ***Kanten***, die jeweils von Stellen zu Transitionen oder umgekehrt führen.

Nebenläufige Prozesse lassen sich durch *Marken* modellieren, die sich nach bestimmten Regeln über die Transitionen von Stelle zu Stelle im Netz fortpflanzen.

Elemente eines Netzes N:

Eine Menge S von ***Stellen***, dargestellt durch Kreise

Eine (zu S disjunkte) Menge T von ***Transitionen***, dargestellt durch Rechtecke

Eine Relation $F \subset (S \times T) \cup (T \times S)$, genannt ***Flussrelation***, dargestellt durch Pfeile

$N = \langle S, T, F \rangle$ heißt ein *Netz* über *S* und *T*

Petri-Netz: Definition

PN = < S, T, F, V, m_0 > heißt ***Petri-Netz****, wenn*

- < S, T, F > ein Netz (im obigen Sinne) ist;
- K: S $\rightarrow$ $N^+ \cup \{\infty\}$ eine Abbildung ist, die jeder Stelle s $\in$ S eine natürliche Zahl (oder den Wert ∞) - genannt ***Kapazität*** von s – zugeordnet;
- V: F $\rightarrow$ N^+ eine Abbildung ist, die jedem Pfeil f $\in$ F eine natürliche Zahl - genannt ***Vielfachheit*** von f – zugeordnet;
- m_o: s $\rightarrow$ N eine ***Anfangsmarkierung*** ist, die jeder Stelle s $\in$ S eine natürliche Zahl m_o (s) $\leq$ K (s) zugeordnet.

Eine ***Markierung*** m: *S* $\rightarrow$ *N* von *PN* gibt die Markenzahl jeder Stelle *S* an, wobei immer $m(s) \leq K(s)$ sein muß.

Das Schalten von Transitionen

Die Dynamik in einem Petri-Netz wird durch das *Schalten von Transitionen* modelliert.

Eine Transition *t* heißt ***aktiviert*** (unter einer Markierung *m*), falls

(a) für jeden Pfeil f: $s \rightarrow t$ mit der Vielfachheit $V(f) = n$ gilt:

$$m(s) \geq n,$$

d. h. es sind genügend viele Marken auf *s* vorhanden, um davon *n* Stück abziehen zu können;

(b) für jeden Pfeil $f': t \rightarrow s'$ mit der Vielfachheit $V'(f') = n'$ gilt:

$$m(s') + n' \leq K(s'),$$

d. h. das Hinzufügen von n' Marken übersteigt die Kapazität von s' nicht.

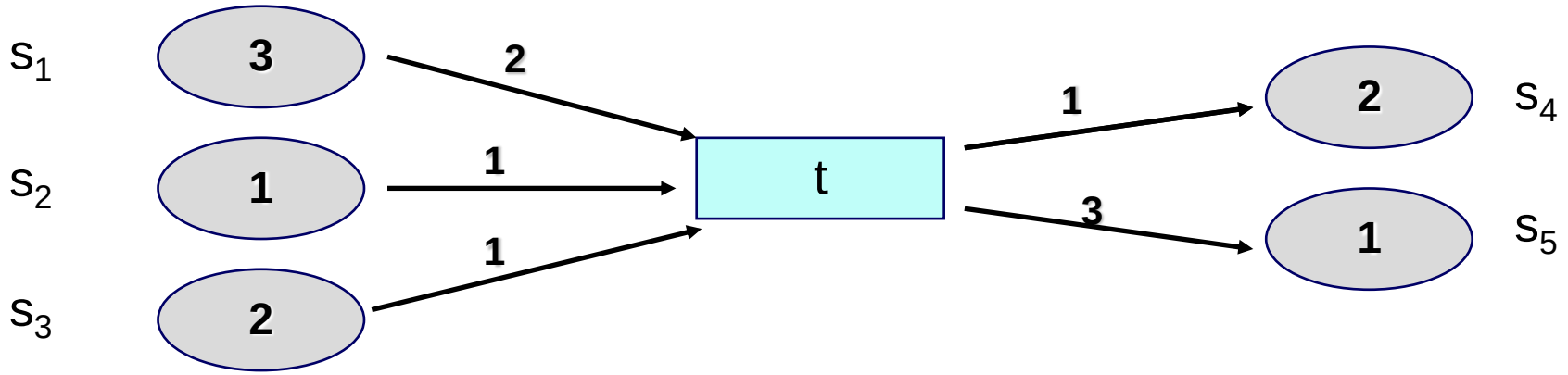
Ist eine Transistion *t* unter einer Markierung *m* aktiviert, so kann *t schalten* und es entsteht eine Folgemarkierung *m,* die gegeben ist durch

$$m'(s) = m(s) - V(s,t) + V(t,s)$$

für alle Knoten *s* und alle in *s* beginnenden Pfeile *(s,t)* bzw. endenden Pfeile *(t,s).*

Beispiel 1:

Es sei ein Netz mit folgender Markierung gegeben:

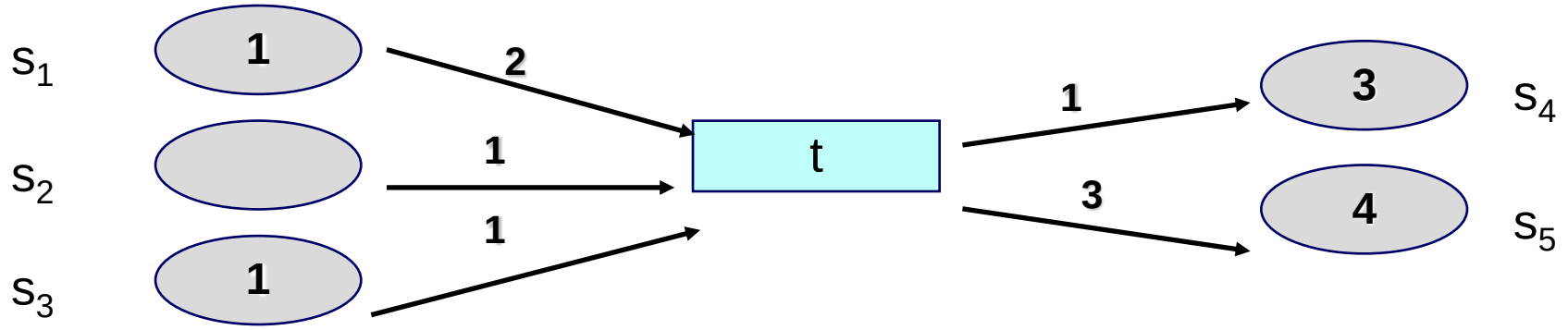


Es gelte: $k(s) = \infty$ für alle s

t ist aktiviert, da $m(s_1) \geq V(s_1, t)$, $m(s_2) \geq V(s_2, t)$, $m(s_3) \geq V(s_3, t)$ und

$m(s_4) + V(t, s_4), \leq k(s_4)$, $m(s_5) + V(t, s_5), \leq k(s_5)$

t kann schalten und führt zu folgender Markierung:



Beispiel 2: *Gegeben sei ein Netz PN mit:*

$S = \{s_1, ..., s_4\}$

$T = \{t_1, ..., t_4\}$

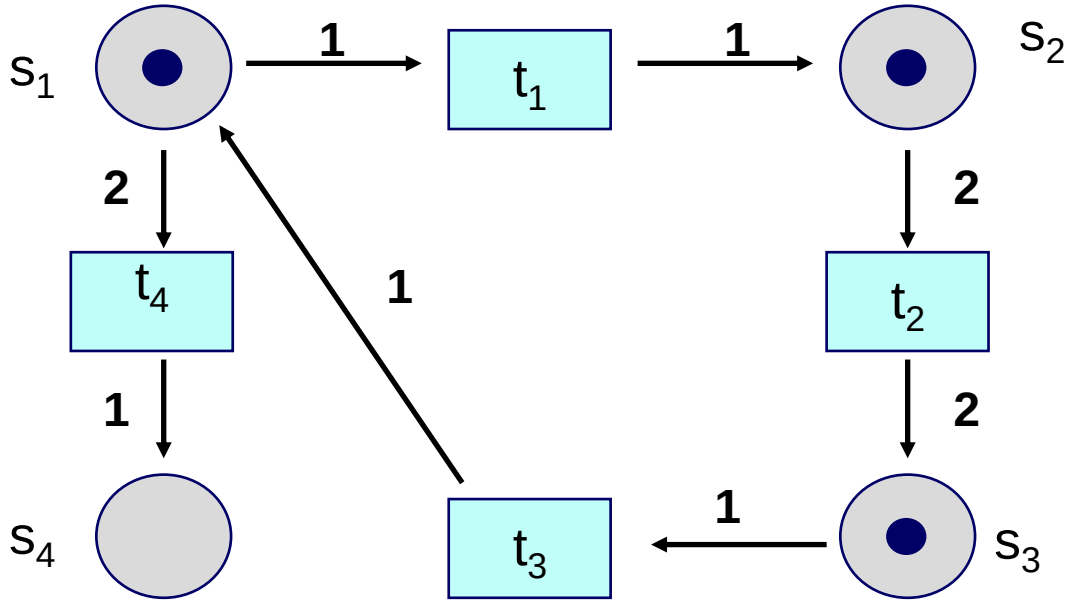
$F = \{(s_1, t_1), (s_1, t_4), (s_2, t_2), (s_3, t_3), (t_1, s_2), (t_2, s_3), (t_3, s_1), (t_4, s_4)\}$

$k(s) = \infty$ für alle s

$V(s_1, t_4) = V(t_2, s_3) = V(s_2, t_2) = 2$ und

$V(f) = 1$ für alle sonstigen Pfeile f

$m_o(s_1) = m_o(s_2) = m_o(s_3) = 1, m_o(s_4) = 0$



Durch Schalten der Transitionen ergeben sich folgende Markierungen:

$$m_o = \begin{bmatrix}1\\1\\1\\0\end{bmatrix} \rightarrow m_1 = \begin{bmatrix}1\\2\\0\\0\end{bmatrix} \rightarrow m_2 = \begin{bmatrix}0\\1\\2\\0\end{bmatrix} \rightarrow m_3 = \begin{bmatrix}1\\1\\1\\0\end{bmatrix}$$

Anwendungen

Mit Hilfe von Petri-Netzen ist es möglich,

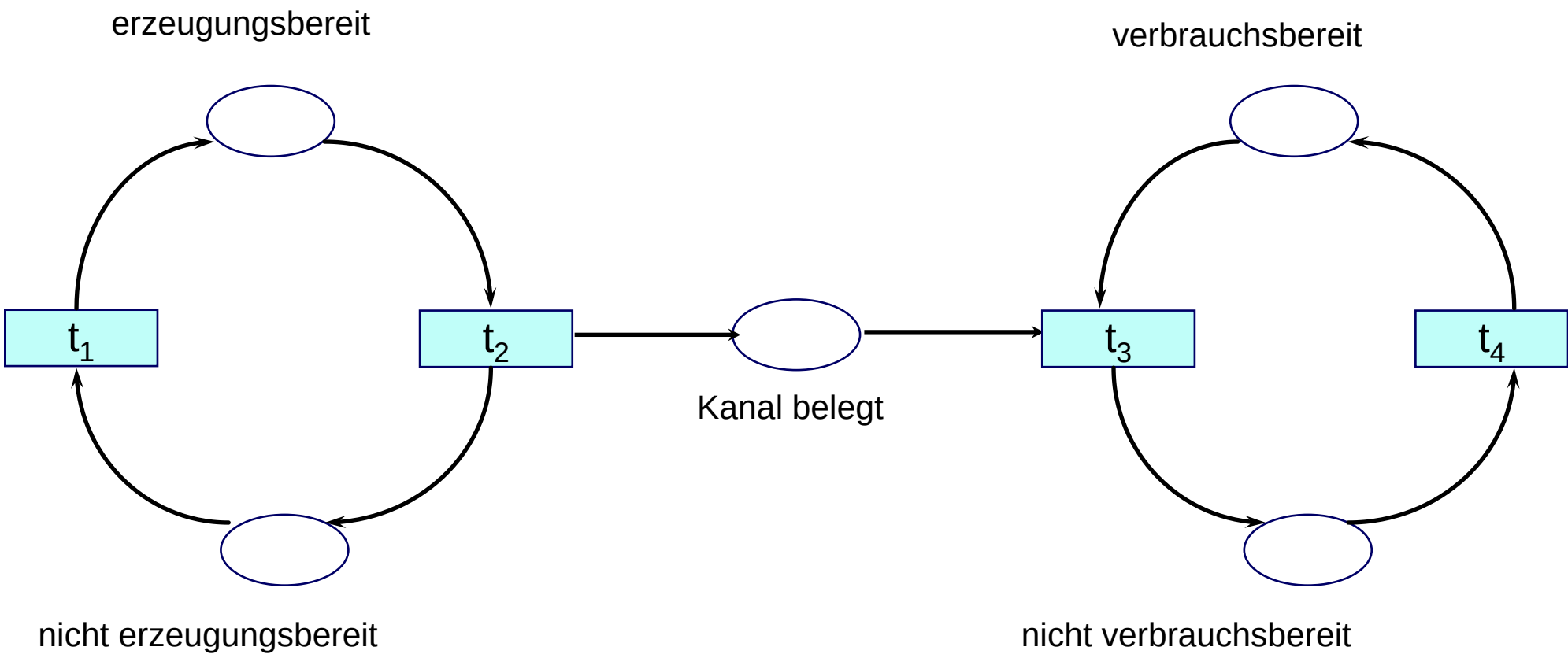
- ***Zustände*** *(=Markierungen),*
- ***Zustandsübergänge*** *(=Transitionen)* und
- Abläufe ***nebenläufiger Prozesse*** *(=Schaltfolgen von Transitionen)*

mathematisch exakt zu modellieren.

Dabei interessiert man sich für wichtige Eigenschaften nebenläufiger Systeme wie

- ***Erreichbarkeit von Zuständen*** (= Markierungen)
- ***Lebendigkeit:*** Sind noch Zustandsänderungen möglich?
- ***Fairness:*** Kommt jede Transition irgendwann einmal zum Schalten?

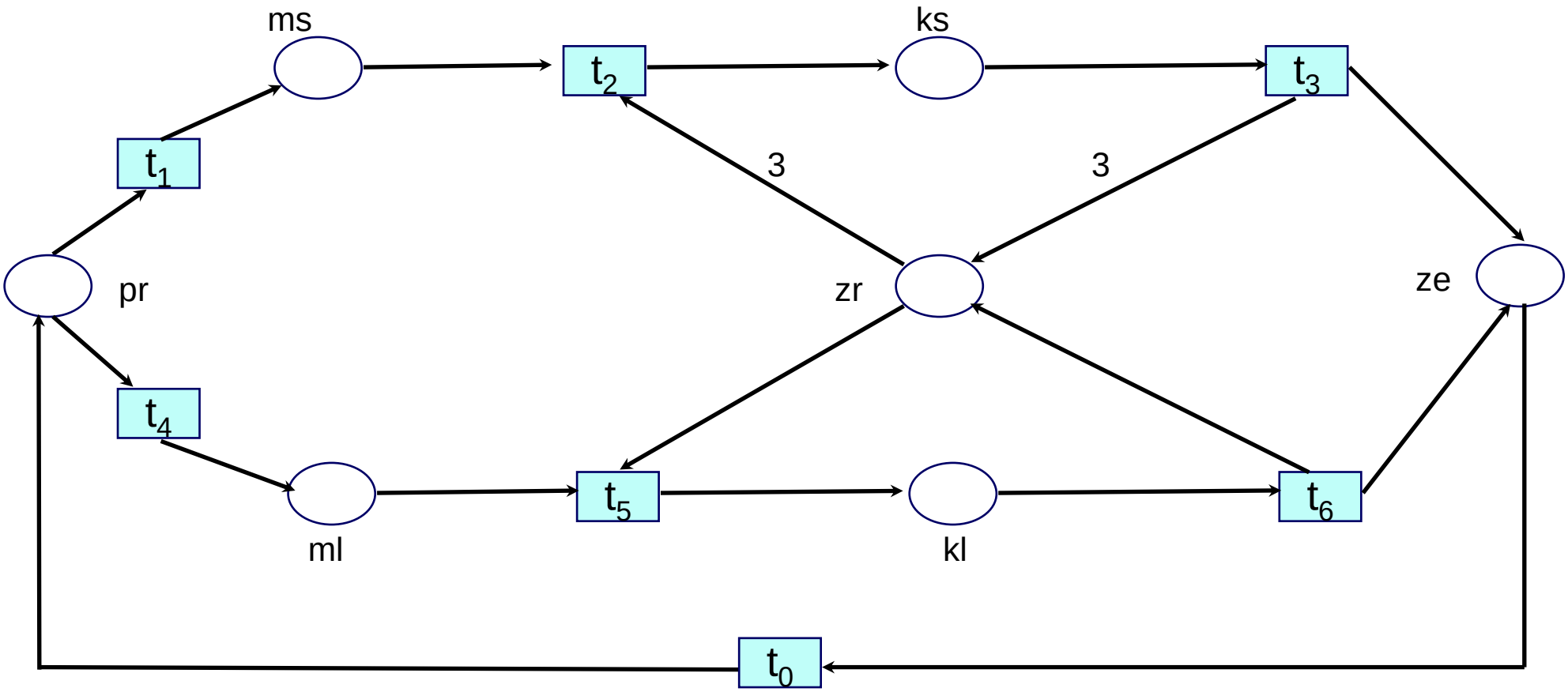
Anwendungsbeispiel 1: ***Producer/Consumer-Problem***



Legende:

t_1 Erzeugungsbereitschaft herstellen	t_3 verbrauchen
t_2 erzeugen	t_4 Verbrauchsbereitschaft herstellen

Anwendungsbeispiel 2: ***Konkurrierende Lese-/ Schreibzugriffe***



Legende:

pr: privater (lokaler) Zugriff, **zr:** Zugriffsregelung (auf globale Daten),

ze: Zugriffsende

ml: möchte lesen, **ms:** möchte (global) schreiben

kl: kann lesen **ks:** kann (global) schreiben

Bedingungs-/Ereignis-Netze

Gilt in einem Petri-Netz *N* immer*:*

m(s) ≤ 1 für alle s ∈ S und

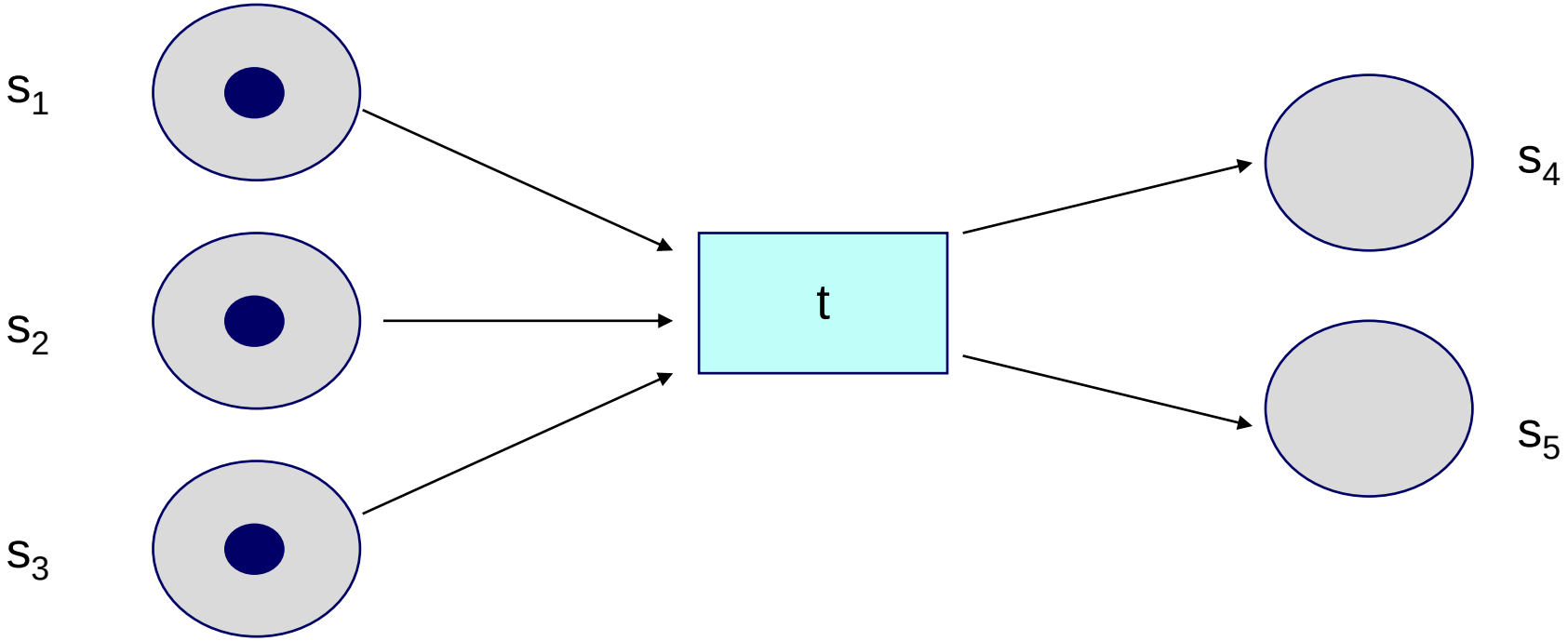
V(f) = 1 für jedes f ∈ F,

so sprechen wir von einem ***Bedingungs-/Ereignis-Netz*** (*B/E-Netz*).

Eine ***Bedingung*** ist dabei das Vorliegen einer bestimmten Markierung, wie z. B. der Markierung, die notwendig (und hinreichend) ist, um eine Transition zu aktivieren.

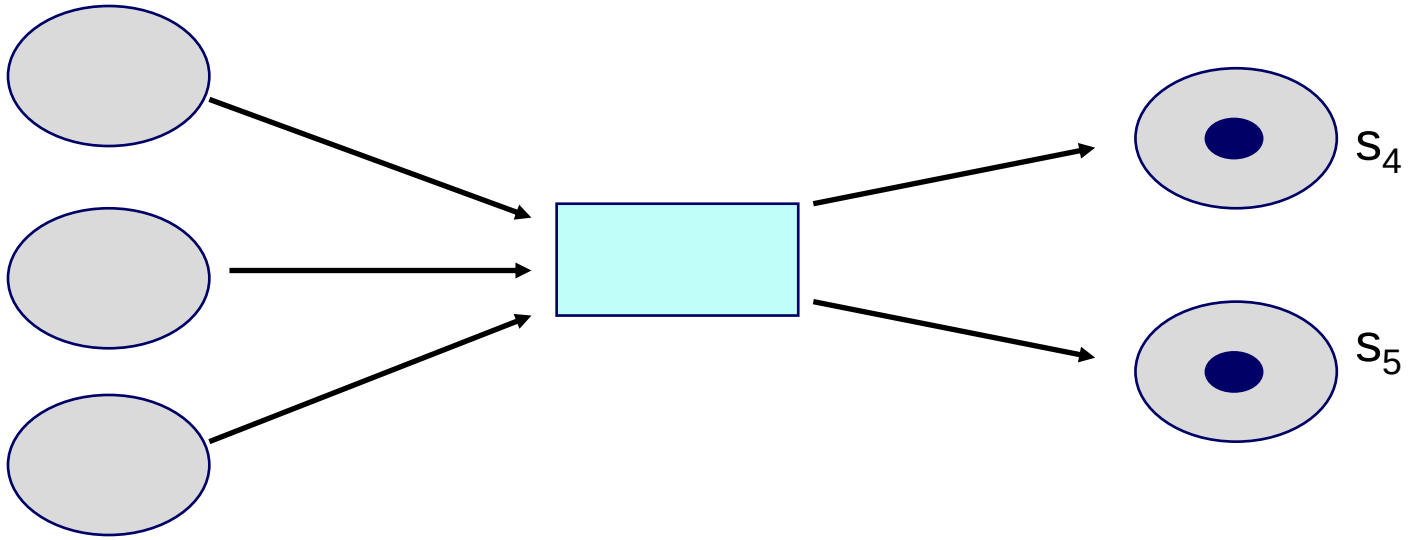
Ein ***Ereignis*** entspricht dem Schalten einer (oder mehrerer) Transition(en).

Beispiel: *Gegeben sei folgendes B/E-Netz:*



Bedingungs-/Ereignis-Netze: Fortsetzung des Beispiels

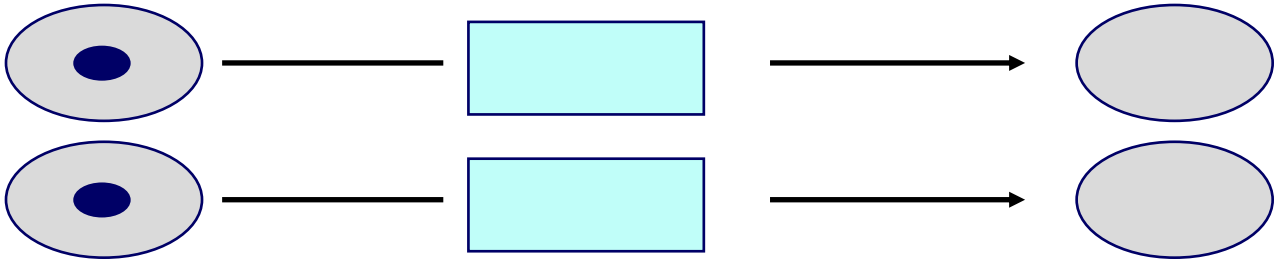
Die Vorbedingung für das Schalten von *t* ist gegeben (*t* ist aktiviert) und
das Schalten von *t* führt zu folgender Markierung (= Nachbedingung von *t*).



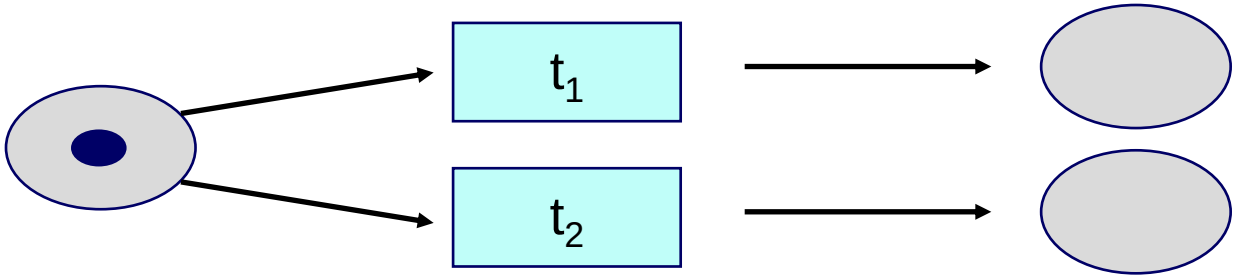
Bedingungs-/Ereignis-Netze: Situationen

Mit B/E-Netzen lassen sich z. B. folgende Situationen modellieren:

(a) **Nebenläufigkeit**

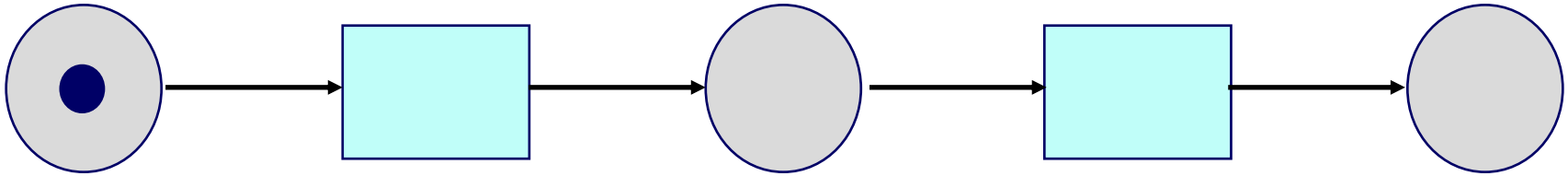


(b) **Auswahl** (= Konflikt) zwischen Transition t_1 oder t_2

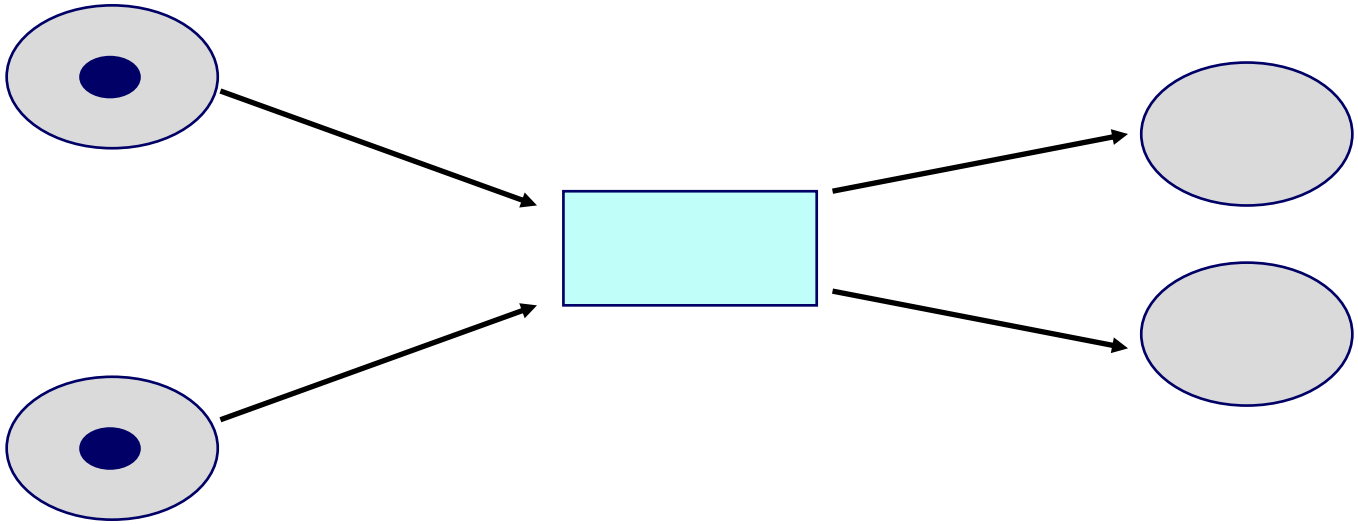


Bedingungs-/Ereignis-Netze: Situationen (2)

(c) **Sequentialität**



(d) **Synchronisation** zweier nebenläufiger Prozesse



Petri-Netze: Fazit, Werkzeug-Unterstützung

- Petri-Netze (PN) sind eine formal fundierte, anerkannte und bewährte Methode zur Modellierung nebenläufiger Prozesse.
- Es gibt viele Erweiterungen und Varianten von PN, deren sinnvollere Einsatz jeweils vom speziellen Anwendungsfeld abhängt.
- Die Anwendung von PN für praktische Modellierungsaufgaben ist i.a. aufwändig und erfordert eine gute Vorbildung bei Modell-Erstellern und –Nutzern.

Werkzeugunterstützung:

- CPN tool, University of Aarhus, *http://wiki.daimi.au.dk/cpntools/cpntools.wiki*
- CPN-AMI, a computer-aided software engineering environment that with Petri Net Specifications*. http://en.wikipedia.org/wiki/CPN-AMI (free for universities and non profit organizations)*
- *Poseidon (s. Kap. SA 14)*

Literatur: [Jen 97], [Sto 01]

Literatur

[Bal 96] H. Balzert: Lehrbuch der Software-Technik, Kap. I. 2; Spektrum Akadem. Verlag 1996

[Bal 99] Heide Balzert: Lehrbuch der Objektmodellierung, Spektrum Akadem. Verlag 1999

[Bos 04] Bossel, Hartmut (2004): Systeme, Dynamik, Simulation: Modellbildung, Analyse und Simulation komplexer Systeme. Books on Demand, Norderstedt/Germany, 2004

[Bos 04a] H. Bossel: Systemzoo 1-3 Books on Demand, Norderstedt/Germany, 2004

[B-D 04] B. Bruegge, A.H. Dutoit: Object-oriented Software Engineering 2nd Ed., Prentice Hall 2004. Dt. Ausgabe: Objektorientierte Softwaretechnik, Pearson Studium 2004

[Fow 04] M. Fowler: UML konzentriert. Addison-Wesley eBook 2004

[Har 88] D. Harel: On visual formalisms. Comm. of the ACM, Vol. 31.5 (1988)

[H-G 96] D. Harel, E. Gery: Executable object modelling with statecharts. Proc. 18th Int'l. Conf. on Software Eng., Berlin, pp. 246-257, IEEE 1996

[H-K 99] M. Hitz, G. Kappel: UML@Work. dpunkt.verlag 1999

[Mea 72] D.H. Meadows, D.L. Meadows, J. Randers, W.W. Behrens: The limits to growth. Potomac Ass., Washington D.C. 1972

[Mea 92] D.H. Meadows, D.L. Meadows, J. Randers: Beyond the limits. Chelsea Green, Post Mills VT 1992

[Neu 02] H.A. Neumann: Analyse und Entwurf von Softwaresystemen mit der UML. 2. Aufl., Hanser-Verlag 2002

Literatur (Forts.)

[Pet 79] C. A. Petri: Introduction to General Net Theory; in: Net Theory and Applications, Proc. Advanced Course on General Net Theory of Processes and Systems LNCS84, Springer 1979

[Rei 90] W. Reisig: Petrinetze - Eine Einführung, Berlin: Springer, 1990

[Sce 95] G. Scheschonk: Petri-Systeme (mit math. Anhang); Vorlesungsscript, SS 1995, Technische Fachhochschule Berlin 1995

[SWG 00] J. Seemann, J. Wolff v. Gudenberg: Software-Entwurf mit UML - Objektorientierte Modellierung mit Beispielen in Java. Springer 2000

[Sto 01] H. Störrle: Models of Software Architecture - Design and Analysis with UML and Petri-Nets. Books on Demand. www.bod.de 2001

[UML 01] OMG Unified Modelling Language Specification. Version 1.5 (2001). http://www.rational.com/uml/resources/documentation.

[UML 05] OMG Unified Modelling Language: Superstructure, Version 2.0. Object Management Group 2005

[Ves 01] F. Vester: Die Kunst vernetzt zu denken, dtv 2001

[ZGK 03] W. Zuser, Th. Grechenig, M. Köhle: Software Engineering mit UML und dem Unified Process, 2. überarb. Aufl., Pearson 2003