Informatik II: Modellierung
Prof. Dr. Martin Glinz

Kapitel 7

# Verhaltensmodelle


Universität Zürich
Institut für Informatik

# Inhalt



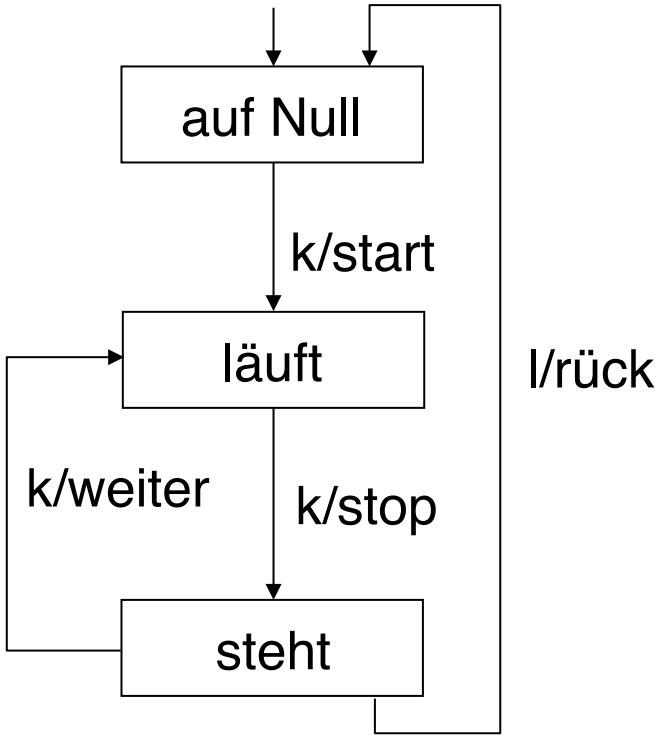
  

# 7.1 Grundlagen

- Modellierung des zeitlich/dynamischen Verhaltens eines Systems, insbesondere für
  - Dialoge
  - Parallele und verteilte Systeme
  - Echtzeit-Systeme
  - „Eingebettete“ Informationssysteme
- Interessierende Fragestellungen:
  - Wann geschieht etwas?
  - Wie lange geschieht etwas bzw. in welchem Zeitraum kann / muss etwas stattfinden?
  - Welche Funktionen stehen wann zur Verfügung?
  - Wie koordinieren parallele und verteilte Komponenten den zeitlichen Ablauf ihrer Tätigkeiten?

# 7.2 Zustandsautomaten

Eine einfache Stoppuhr mit einem Bedienknopf:




Ereignisse:

| | |
|---|---|
| k | Bedienknopf kurz gedrückt |
| l | Bedienknopf lang gedrückt |

Aktionen:

| | |
|---|---|
| start | Stoppuhr starten |
| stop | Stoppuhr anhalten |
| weiter | Stoppuhr weiterlaufen lassen |
| rück | Stoppuhr auf Null stellen |

# Zustandsautomaten: Definition, Grundlagen

- Ein Zustandsautomat (state automaton) besteht aus
  - einer endlichen Menge Z von Zuständen
  - einer Menge U von Zustandsübergängen zwischen Zuständen aus Z
  - einem oder mehreren Ereignissen zu jedem Zustandsübergang aus U, die den Zustandsübergang auslösen
  - Null bis n Aktionen zu jedem Zustandsübergang aus U, die durch den Zustandsübergang ausgelöst werden
  - der Kennzeichnung genau eines Zustands aus Z als Startzustand
  - optional der Kennzeichnung von Endzuständen

# Zustandsautomaten: Definitionen und Interpretation

- Zustand (state) – repräsentiert einen genau abgegrenzten *Zeitabschnitt*, in dem das System ein bestimmtes Verhalten zeigt und auf eine bestimmte Menge von Eingaben reagiert
- Zustandsübergang (state transition) – Zeitpunkt, an dem ein System einen gegebenen Zustand verlässt und in einen neuen Zustand übergeht
- Ereignis (event) – Zeitpunkt, an dem eine Handlung wirksam wird oder eine Veränderung festgestellt wird
- Aktion (action) – eine Operation, die bei einem Zustandsübergang angestoßen wird oder ein Ereignis, das bei einem Zustandsübergang erzeugt wird
- Diese Art von Automaten heißen in der Theorie Mealy-Automaten

# Definitionen und Interpretation – 2

- Ausgehend vom gegebenen Startzustand befindet sich ein Zustandsautomat immer in genau einem Zustand.
- Im Zustand Z reagiert der Automat auf alle Ereignisse, die Zustandsübergänge von Z in einen anderen Zustand auslösen. Alle übrigen Ereignisse werden ignoriert.
- Tritt ein Ereignis ein, das einen Zustandsübergang auslöst, werden die zugehörigen Aktionen angestoßen und der Automat geht in den durch den Zustandsübergang spezifizierten Folgezustand.
- Zustandsübergänge sind zeitfrei.

# Etwas Theorie: Endliche Automaten

- Ein Endlicher Automat ist ein 5-Tupel (X,Y,Z, $\lambda$, $\delta$), bestehend aus
  - einer Menge X = $\{x_1,...x_n\}$ von Eingabewerten (Eingabealphabet),
  - einer Menge Y = $\{y_1,...,y_m\}$ von Ausgabewerten (Ausgabealphabet),
  - einer Menge Z = $\{z_1,...,Z_s\}$ von Zuständen,
  - einer Zustandsübergangsfunktion $\lambda$: Z x X $\rightarrow$ Z,
  - einer Ausgabefunktion $\delta$
    - entweder $\delta$: Z x X $\rightarrow$ Y (Mealy-Automat)
    - oder $\delta$: Z $\rightarrow$ Y (Moore-Automat)
- Zur Modellierung des Verhaltens von Systemen werden meist Mealy-Automaten verwendet.
- In der Theorie formaler Sprachen dienen endliche Automaten zur Erkennung regulärer Sprachen. Das Ausgabealphabet solcher Automaten ist leer; die Ausgabefunktion entfällt.

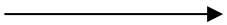
# Notation

- Zustandsautomaten können tabellarisch oder graphisch dargestellt werden durch
  - Zustandsmatrizen
  - Zustandsdiagramme
- In der Regel werden Zustandsdiagramme verwendet

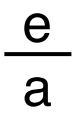
Zustandsdiagramm (state diagram) – Graphische Darstellung eines Zustandsautomaten.

# Notation – 2

Zustand (state)

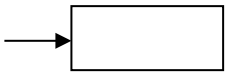


Zustandsübergang (state transition)



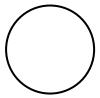

e/a

Auslösendes Ereignis / ausgelöste Aktion (event / action)

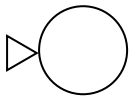


Startzustand (initial state)

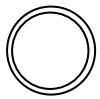
Alternative Notation (v.a. in der Automatentheorie verwendet)



Zustand



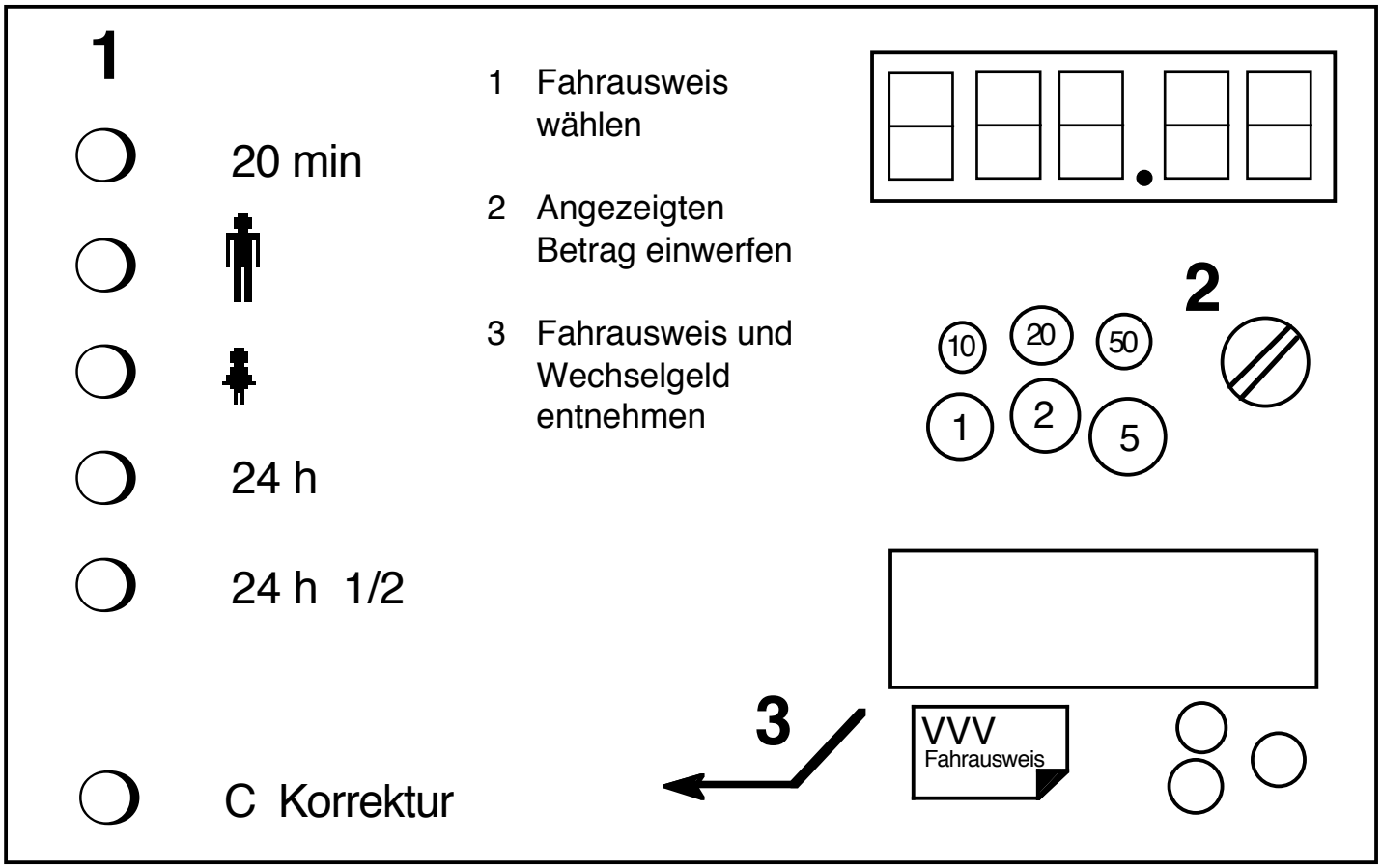
Startzustand



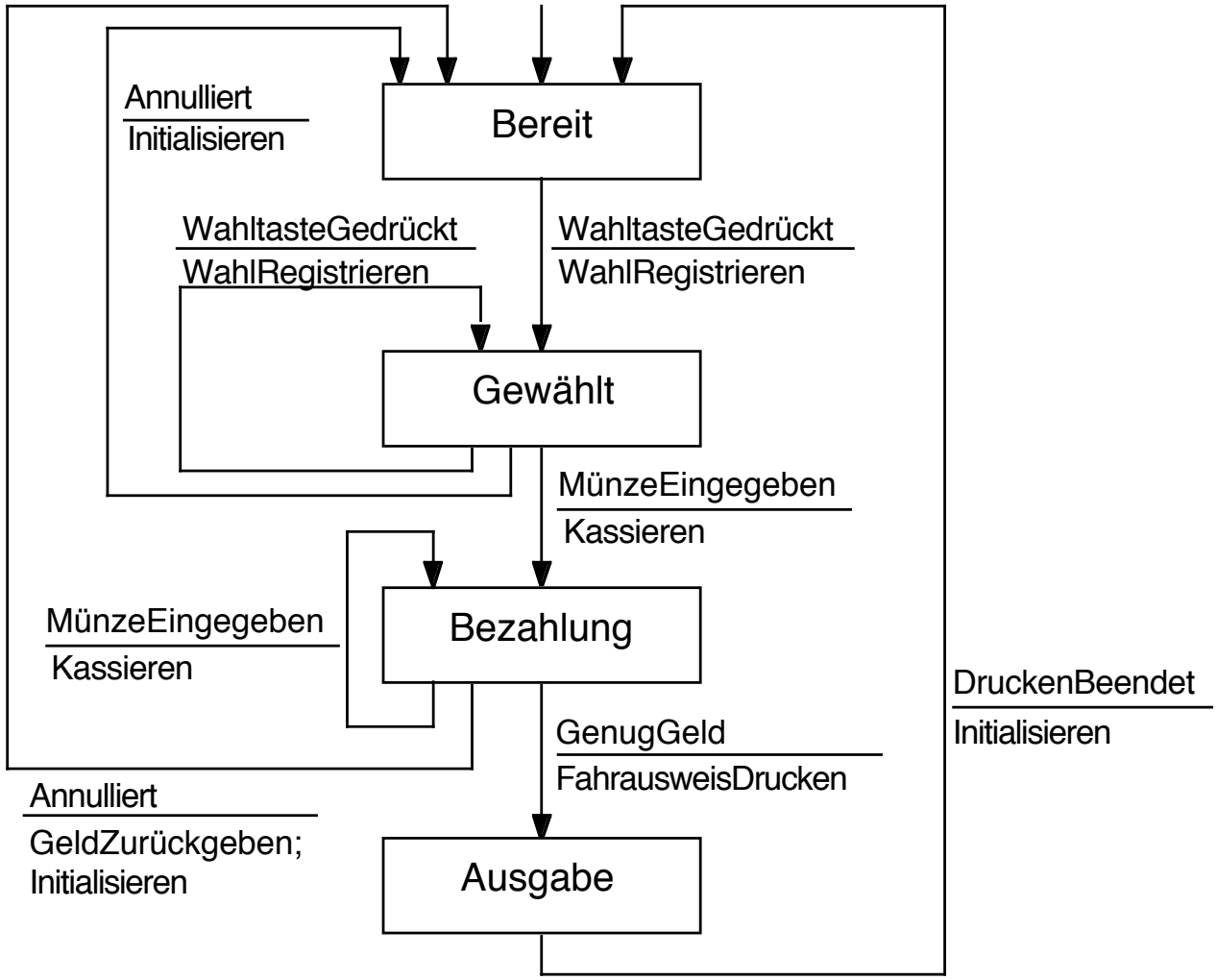
Endzustand

# Beispiel: Ein einfacher Fahrausweis-Automat

Bedienung:



# Beispiel – 2: Verhaltensmodell



# Aufgabe 7.1

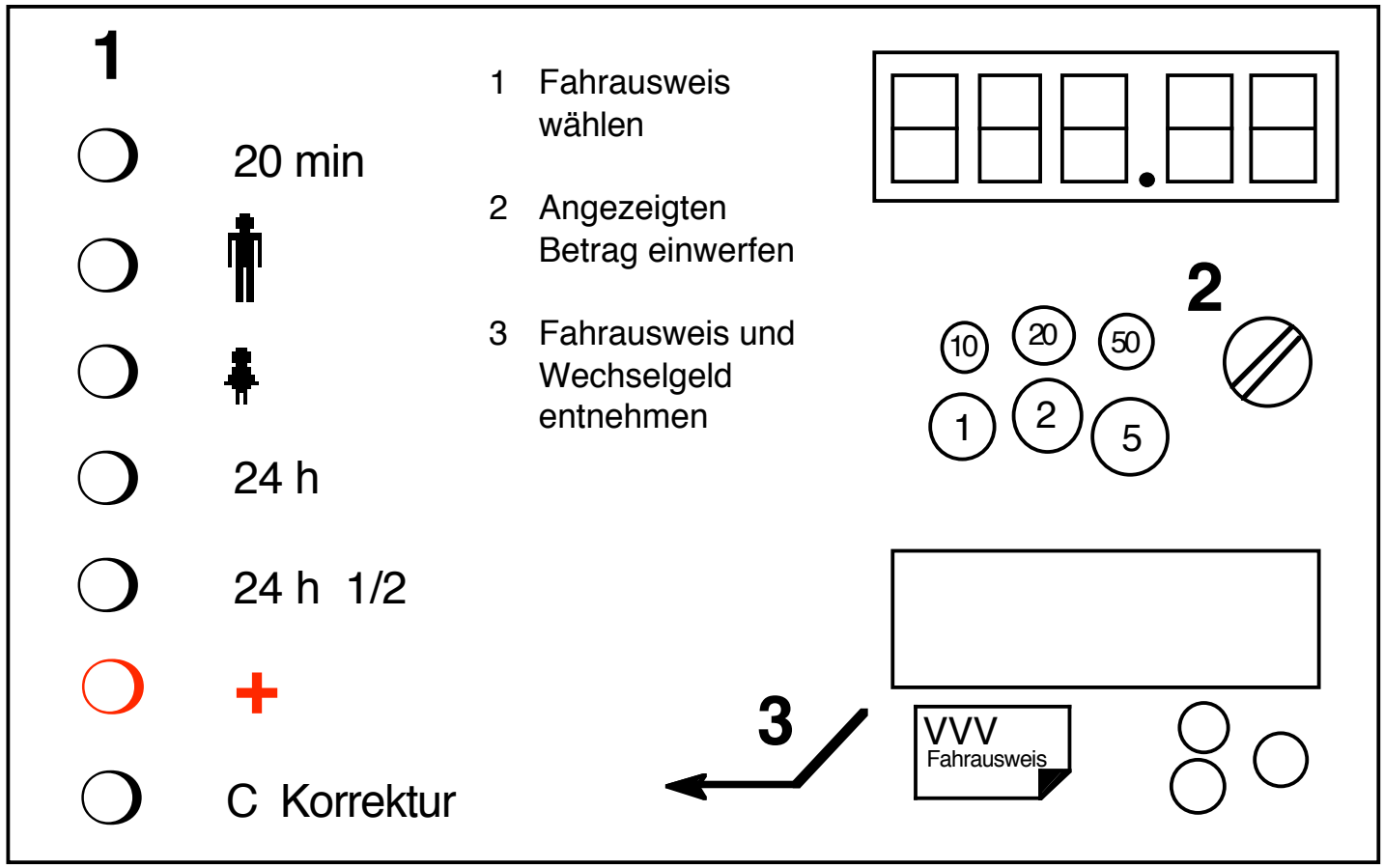
Der oben beschriebene Fahrausweis-Automat soll wie folgt erweitert werden:

Es gibt eine zusätzliche Taste, die PLUS-Taste, welche es ermöglicht, mehrere Auswahlen hintereinander zu treffen und gesammelt zu bezahlen. Nach dem Drücken einer Wahltaste kann also entweder bezahlt oder die PLUS-Taste gedrückt werden. Im letzteren Fall muss anschließend eine weitere Wahltaste gedrückt werden.

Wie muss das oben gegebene Zustandsdiagramm ergänzt/geändert werden, wenn diese Erweiterung modelliert werden soll?

Wo stoßen Sie beim Modellieren auf Unklarheiten in der Aufgaben-stellung?

# Aufgabe 7.1: Neues Bedienfeld



# Vor- und Nachteile von Zustandsautomaten

- Anschaulich
- Symbolisch ausführbar

- Ignoriert Daten und Funktionalität
- Zustandsexplosion bei Modellen mit vielen parallelen Abläufen
- keine Mittel zur Dekomposition großer Modelle

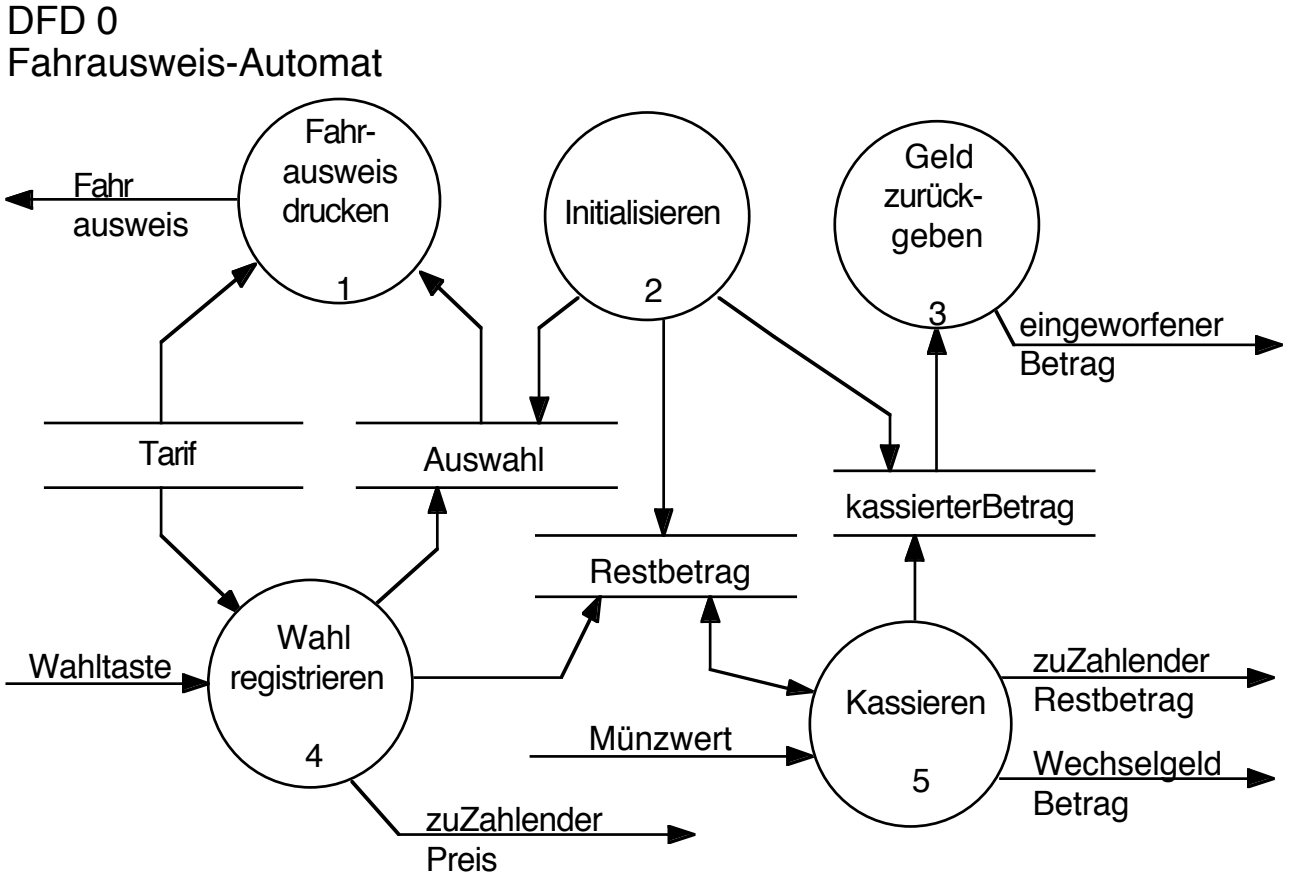
- Die Probleme von Zustandsexplosion und fehlender Dekomposition werden durch Statecharts (siehe nächster Abschnitt) gelöst

# Modellierung zusätzlicher Aspekte

- Zustandsautomaten ignorieren Daten und Funktionalität
- Kombination von Verhaltensmodellen mit anderen Modellen, zum Beispiel
  - Klassenmodelle (siehe Kapitel über Klassenmodellierung)
  - Datenflussdiagramme

# Beispiel: Funktionalität des Fahrausweis-Automaten

Ergänzung des Verhaltensmodells des Fahrausweis-Automaten durch ein Datenflussmodell:



# Methodik der Erstellung von Zustandsautomaten

0. Problemstellung analysieren.
1. Startzustand $Z_0$ identifizieren und benennen.
2. Sei $Z_a$ der aktuelle Zustand (zu Beginn ist dies der Startzustand). Alle Arten von Ereignissen, auf die im Zustand $Z_a$ reagiert werden soll, identifizieren und die zugehörigen auszuführenden Aktionen bestimmen.
3. Für jede Ereignisart wird ein Zustandsübergang $ZÜ_i$ von $Z_a$ in einen potenziellen Folgezustand $Z_i$ mit den zugehörigen Ereignissen bzw. Aktionen modelliert.
4. Für jeden potenziellen Folgezustand $Z_i$:
   4a. Alle Arten von Ereignissen, auf die in $Z_i$ reagiert werden soll, identifizieren und die zugehörigen auszuführenden Aktionen bestimmen.

# Methodik der Erstellung von Zustandsautomaten – 2

4b. Sind diese Ereignisarten und Aktionen für zwei potenzielle Folgezustände $Z_i$ und $Z_k$ gleich, so führen die Zustandsübergänge $ZÜ_i$ und $ZÜ_k$ in einen gemeinsamen Folgezustand.

4c. Sind die Ereignisarten und Aktionen, auf die in $Z_i$ reagiert wird, gleich wie diejenigen, auf die in einem bereits modellierten Zustand $Z_e$ reagiert wird, so ist $Z_i = Z_e$ und $ZÜ_i$ ist ein Zustandsübergang vom aktuellen Zustand $Z_a$ auf $Z_e$. (Dabei ist auch $Z_e = Z_a$ möglich, wobei ein Zustandsübergang von $Z_a$ auf sich selbst entsteht.)

4d. Die auf diese Weise verbleibenden, voneinander verschiedenen Folgezustände benennen.

5. Schritte 2, 3 und 4 rekursiv für jeden neu modellierten Zustand wiederholen.

# Aufgabe 7.2

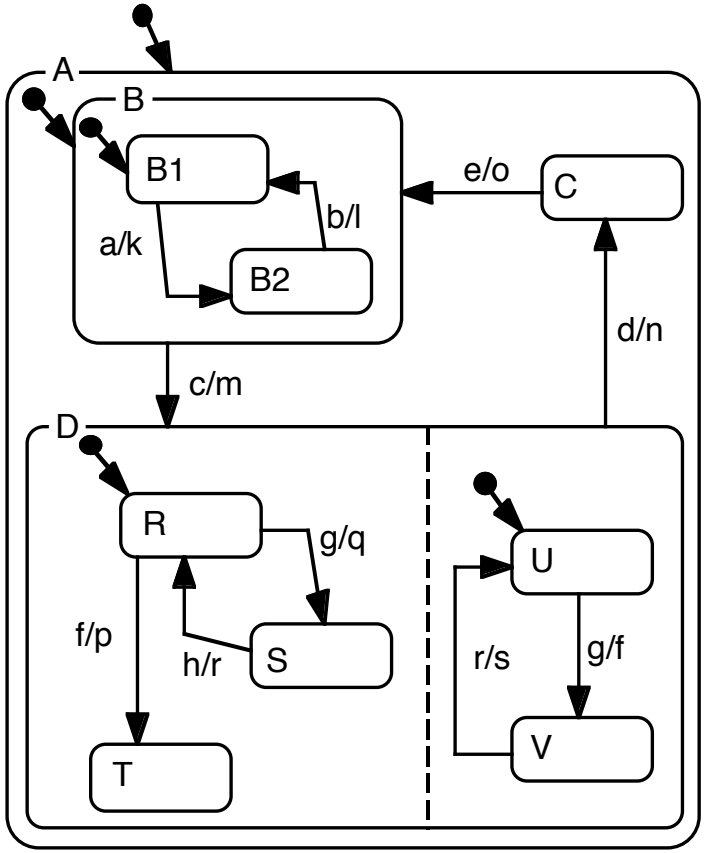
Modellieren Sie einen Kaffeeautomaten, der wie folgt funktioniert:
Als erstes wird eine Wertmünze eingeworfen.
Als nächstes können Zusätze gewählt werden: mit Milch, mit Zucker.
Als drittes wird die Art des Kaffees gewählt: Normal oder Espresso.
Dann wird das gewählte Getränk zubereitet.
Sobald der gefüllte Becher entnommen ist, ist das Gerät wieder bereit.

# 7.3 Statecharts

**Statechart** – Ein Zustandsdiagramm, dessen Zustände hierarchisch und parallel zerlegbar sind.

- Statecharts =
  - \+ Hierarchische Zerlegung
  - \+ Parallelität

(Harel 1987)



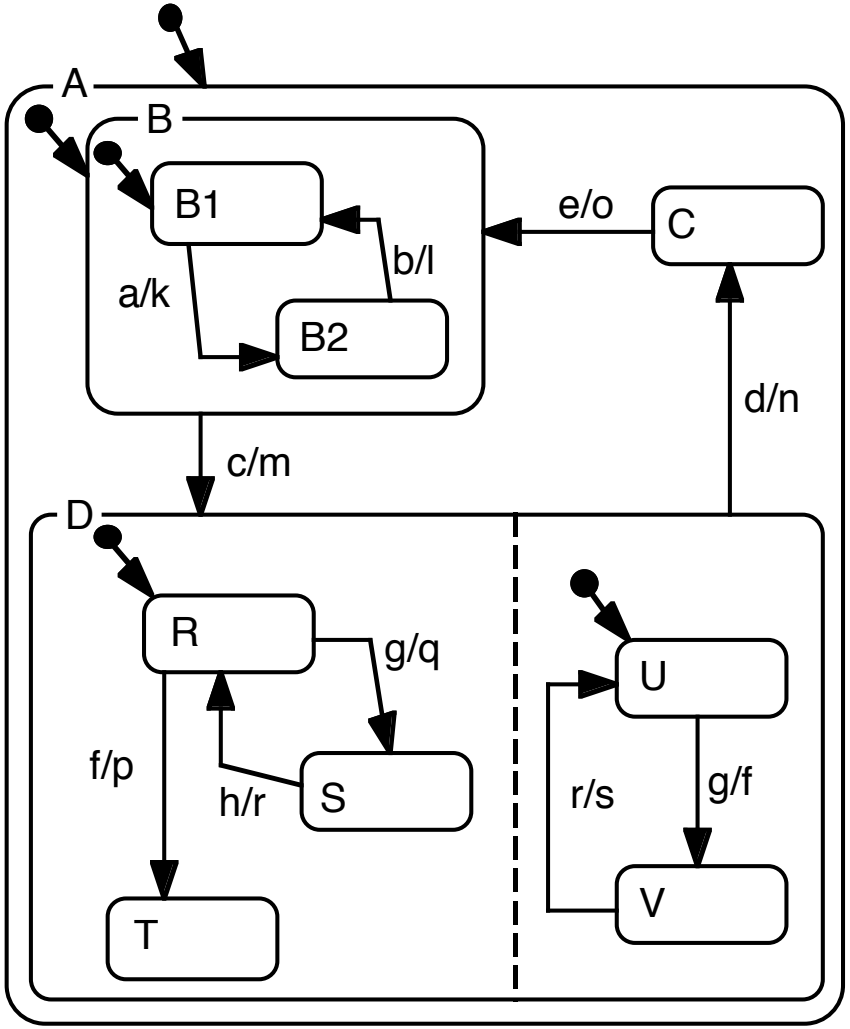
# Statecharts – 2

Jeder Zustand in einem Statechart ist

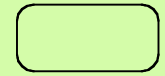
- entweder elementar
- oder hierarchisch zerlegt durch ein anderes Statechart
- oder parallel zerlegt in mehrere parallele Statecharts

- Vermeiden die Probleme von Zustandsautomaten
- Theoretisch äquivalent mit Zustandsautomaten

# Notation und Interpretation



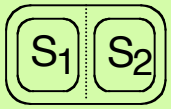

Notation:



Zustand oder Statechart

Zustandsübergang

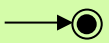
Initialzustand




Zustand besteht aus parallelen Statecharts

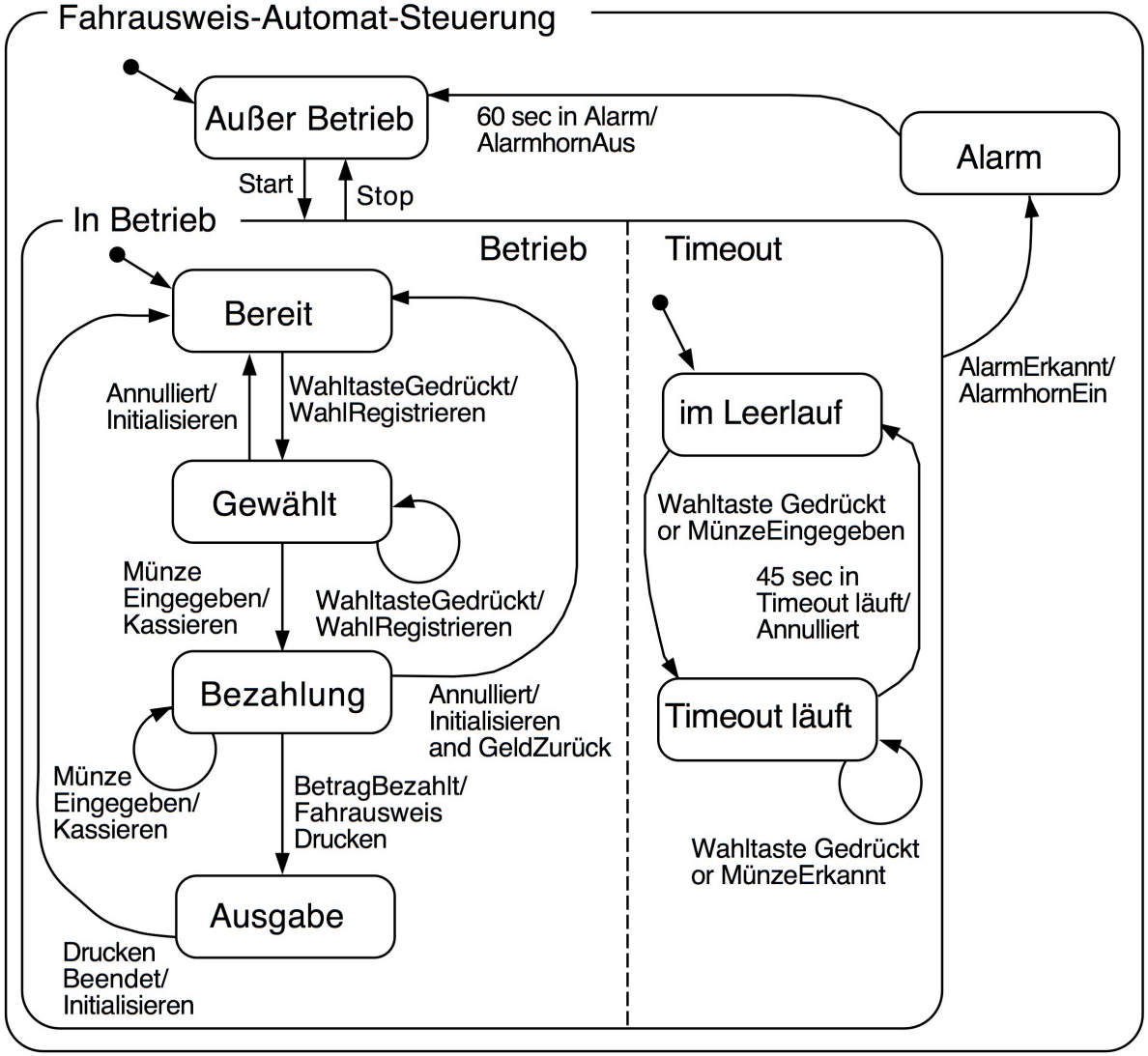
e/a

auslösendes Ereignis e und ausgelöste Aktion a



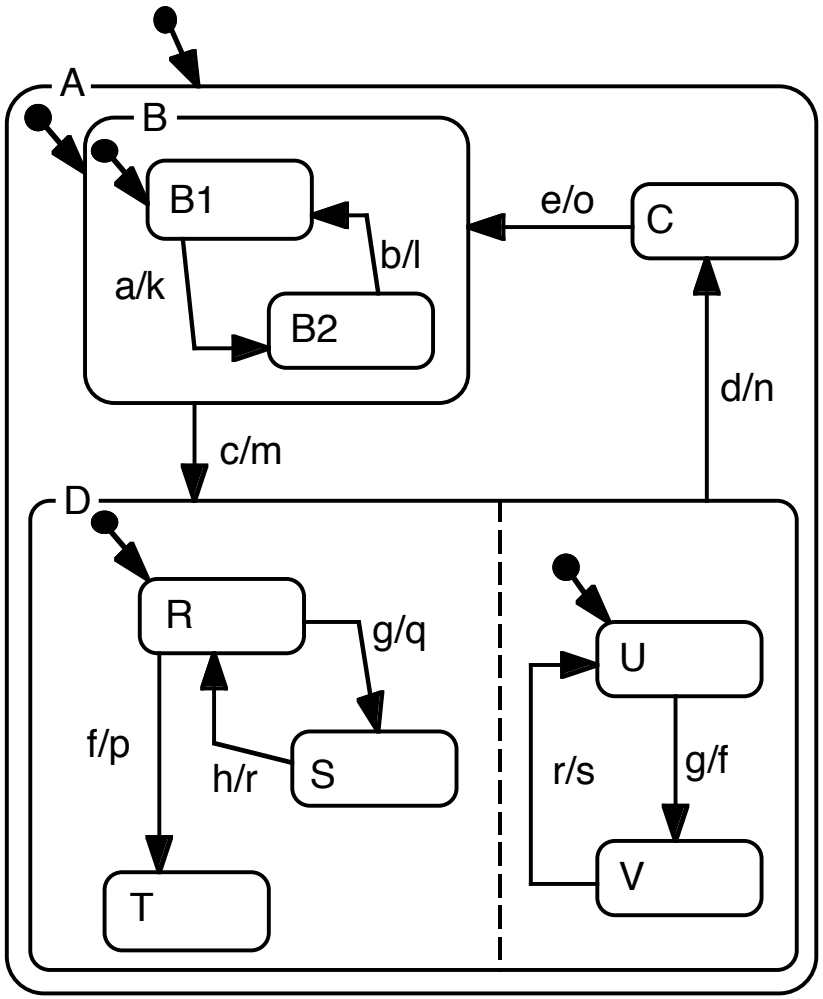
Endzustand

# Beispiel: Erweitertes Fahrausweis-Automat-Modell

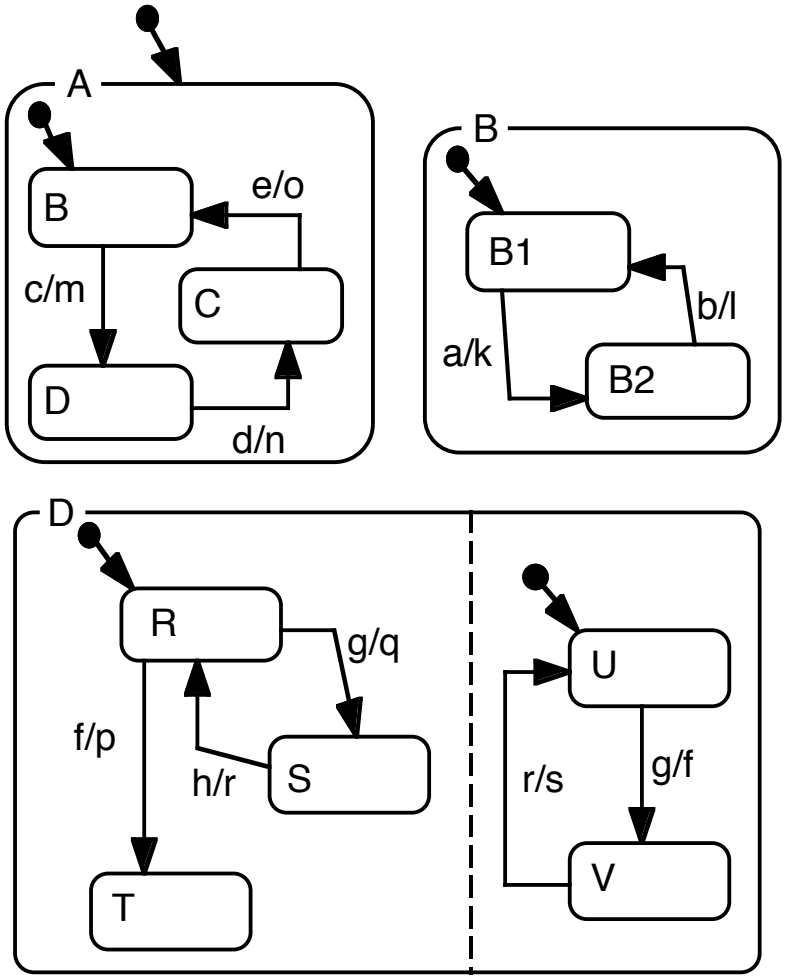


# Verschachtelte oder flache Darstellung möglich




verschachtelt



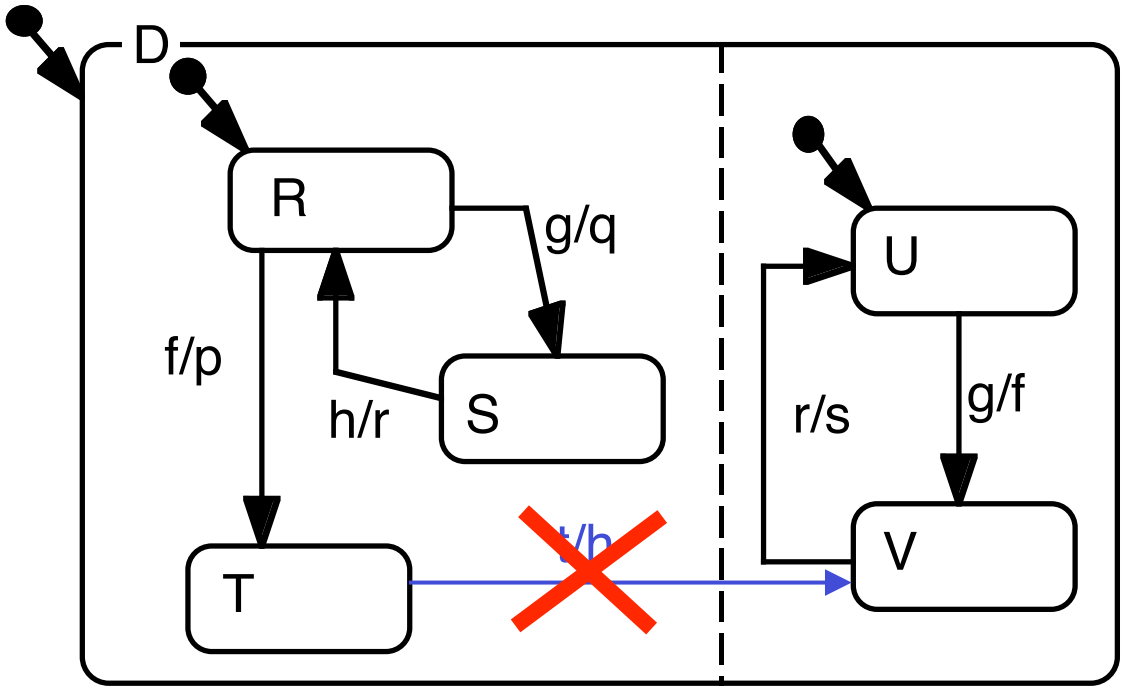

flach

# Interpretation hierarchischer Statecharts (Details)

- Zustandsübergang in einen elementaren Zustand:
  wie bei Zustandsautomaten
- Verlassen eines elementaren Zustands:
  wie bei Zustandsautomaten
- Zustandsübergang in einen Zustand, der aus einem Statechart besteht:
  dieses Statechart geht in den Initialzustand; bei mehrstufiger Hierarchie gilt diese Regel rekursiv
- Verlassen eines Zustands, der aus einem Statechart besteht:
  alle Zustände dieses und aller darin hierarchisch enthaltenen Statecharts werden verlassen

# Interpretation paralleler Statecharts

- Bei der Initialisierung geht dieses Statechart in den Zustand (R,U)
- Die Ereignis-Sequenz g h f




führt zur folgenden Zustandssequenz:

(S,V) (R,U) (T,U)

Wichtig: Zustandsübergänge zwischen parallelen Statecharts sind verboten!

# Interpretation paralleler Statecharts (Details)

- Ein Zustand Z kann aus mehreren parallelen Statecharts $S_1,...S_n$ bestehen (Statechart D in Beispiel oben). Dabei ...
  - bilden die Zustände $S_1,...S_n$ zusammen den Zustand Z
  - können in parallelen Statecharts parallel und unabhängig voneinander Zustandsübergänge stattfinden
  - sind Zustandsübergänge zwischen parallelen Statecharts verboten (graphisch: die gestrichelten Trennlinien dürfen nie von einem Zustandsübergangspfeil gekreuzt werden)
  - können sich parallele Statecharts über Ereignisse gegenseitig beeinflussen
- Beim Zustandsübergang nach Z gehen die Statecharts $S_1,...S_n$ in ihren jeweiligen Initialzustand
- Beim Verlassen des Zustands Z werden alle Zustände aller Statecharts $S_1,...S_n$ verlassen

# Etwas Theorie: Statechart-Semantik

- Die Semantik der Zustandsübergänge in Statecharts wird durch Makro- und Mikroschritte definiert
- Makroschritt: Vollständige Reaktion auf das Auftreten eines äußeren Ereignisses
- Der Ablauf eines Makroschritts wird in Mikroschritte unterteilt
- Während der Bearbeitung eines Makroschritts treten keine äußeren Ereignisse ein (d.h. die Dauer eines Makroschritts ist null oder beliebig klein)
- Es gibt eine Menge verschiedener Semantiken
  - Unterschiede sind meist subtil
  - Hier wird eine einfache, intuitive Semantik verwendet [Glinz 2002]

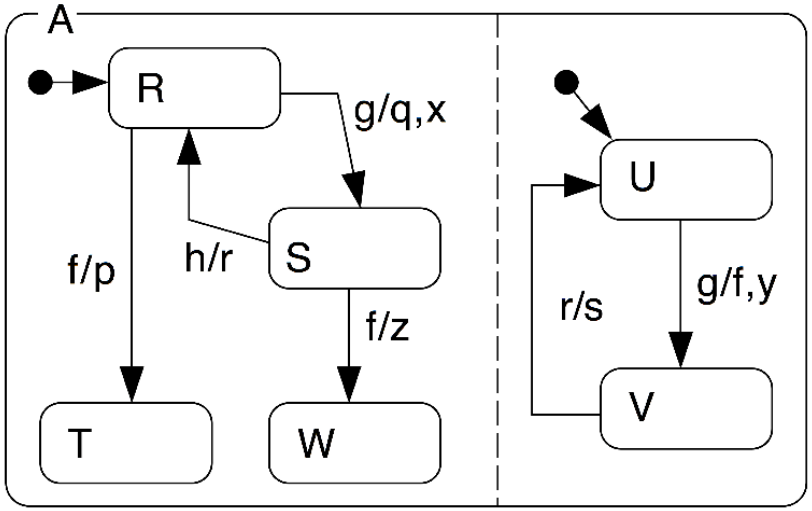
# Hier verwendete Semantik

Makroschritt: Reaktion auf Ereignis g im Zustand (R, U)

Mikroschritte:

1. R und U werden verlassen
2. S und V werden betreten
3. q wird erzeugt
4. x wird erzeugt
5. f wird erzeugt,
   S wird verlassen
6. W wird betreten
7. z wird erzeugt
8. y wird erzeugt

Der Makroschritt endet.

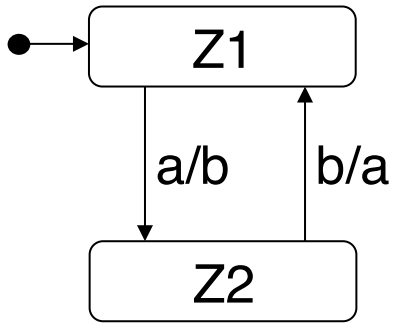



[Glinz 2002]

# Hier verwendete Semantik – 2

- Die Reihenfolge der Bearbeitung paralleler Zustandsübergänge ist zufällig: die Reihenfolgen
  - q, x, f, z, y
  - f, z, y, q, x

  sind beide möglich
- In dieser Semantik kann es vorkommen, dass ein Makroschritt nicht terminiert



# Andere Semantiken für Statecharts

- Beispiel: Originale Semantik von Harel

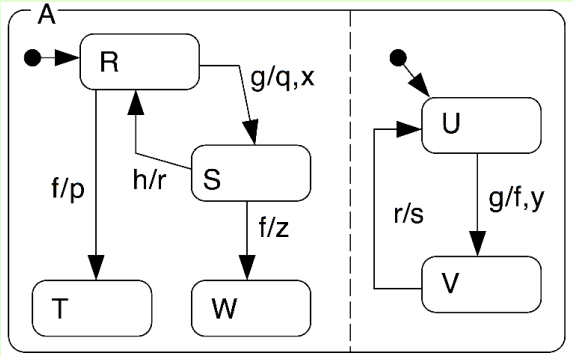
  Makroschritt: Reaktion auf Ereignis g im Zustand (R, U)

  Ereignismenge = { g }

  Mikroschritte:

  1. Die Ereignisse q, x, f, y werden erzeugt und zur Ereignismenge hinzugefügt
  2. In jedem parallelen Statechart wird zufällig ein auf Grund der Ereignismenge möglicher Zustandsübergang ausgewählt:
     R → S und U → V oder R → T und U → V

  Der Makroschritt endet

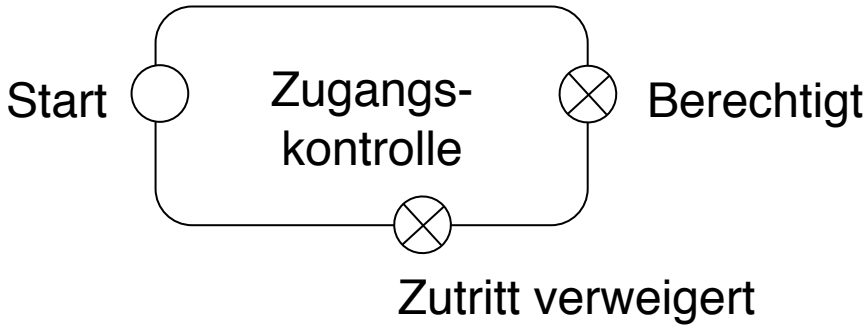



  Problem: diese Semantik führt zu kontra-intuitivem Verhalten

- UML verwendet ebenfalls eine eigene Semantik für Statecharts

# Statecharts – Weitere Elemente

- Es gibt eine Reihe weiterer Elemente in Statecharts, die hier nicht behandelt werden
- Merkzustände (history states) gibt es schon in den Originalarbeiten von Harel
- UML 2 erweitert Statecharts um diverse weitere Elemente
  - Nützlich sind Ein- und Austrittspunkte für Komponentenzustände:



# Aufgabe 7.3

Die Haustür eines Gebäudes ist aus Sicherheitsgründen nur über eine automatische Tür zugänglich.

Diese Tür hat folgende Eigenschaften: Die Tür öffnet automatisch, wenn auf der Außenseite eine Schlüsselkarte mit einprogrammierter Zutrittsberechtigung gesteckt wird, wenn ein Annäherungssensor auf der Innenseite anspricht oder wenn ein Öffnungsknopf auf der Innenseite gedrückt wird. Vier Sekunden nachdem die Tür sich vollständig geöffnet hat, schließt sie automatisch wieder.

An der Tür befinden sich Sensoren, welche folgende Ereignisse melden: Tür vollständig geschlossen, Tür vollständig geöffnet, Hindernis in der Türöffnung. Wird beim Schließen der Tür ein Hindernis in der Türöffnung registriert, so wird der Schließvorgang unterbrochen und die Tür öffnet wieder.

# Aufgabe 7.3 (Fortsetzung)

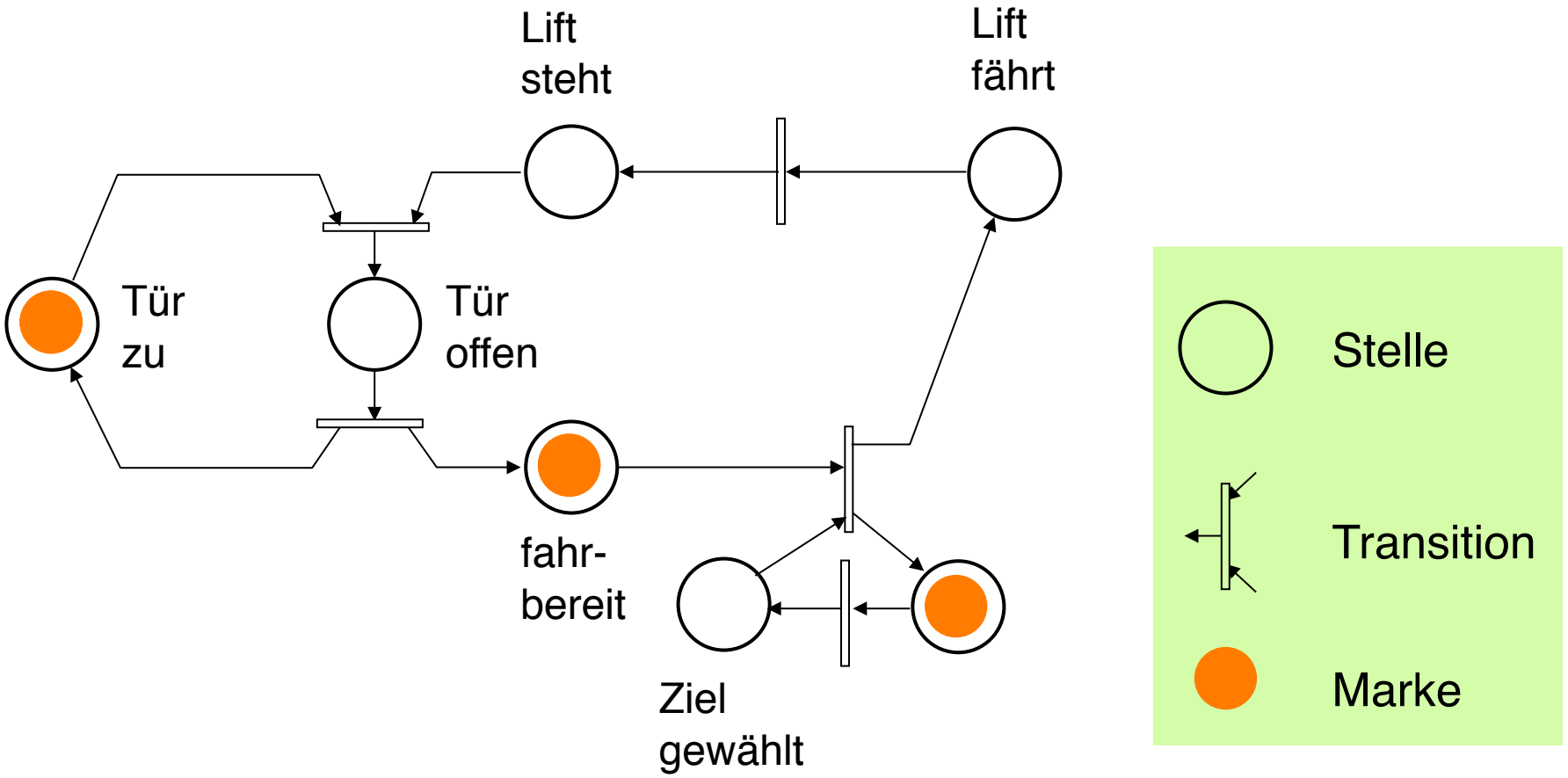
An der Innenseite der Tür befindet sich ein plombierter Notöffnungsknopf. Wird dieser gedrückt, so muss die Tür sofort öffnen und offen bleiben.

Im Hinblick auf eine zu entwerfende automatische Steuerung soll das Verhalten der Tür präzise modelliert werden.

Verwenden Sie zur Modellierung Statecharts.

# 7.4 Petrinetze

Eine (stark vereinfachte) Liftsteuerung:



# Petrinetze: Definitionen, Grundlagen

- Petrinetze (Petri nets) modellieren ereignisgesteuerte, parallele Abläufe (Petri 1962, Reisig 1982, 1985, Murata 1988)
- Petrinetze basieren auf einer fundierten mathematischen Theorie.

- Ein **Petrinetz** ist ein bipartiter gerichteter Graph, d.h. ein Netz aus Knoten und Kanten (gerichteten Verbindungen zwischen Knoten) mit zwei sich abwechselnden Sorten von Knoten.
- Jeder Knoten ist entweder eine Stelle oder eine Transition.
- Stellen und Transitionen wechseln einander im Netz ab: es dürfen nie zwei Stellen oder zwei Transitionen direkt benachbart sein.

# Petrinetze: Definitionen, Grundlagen – 2

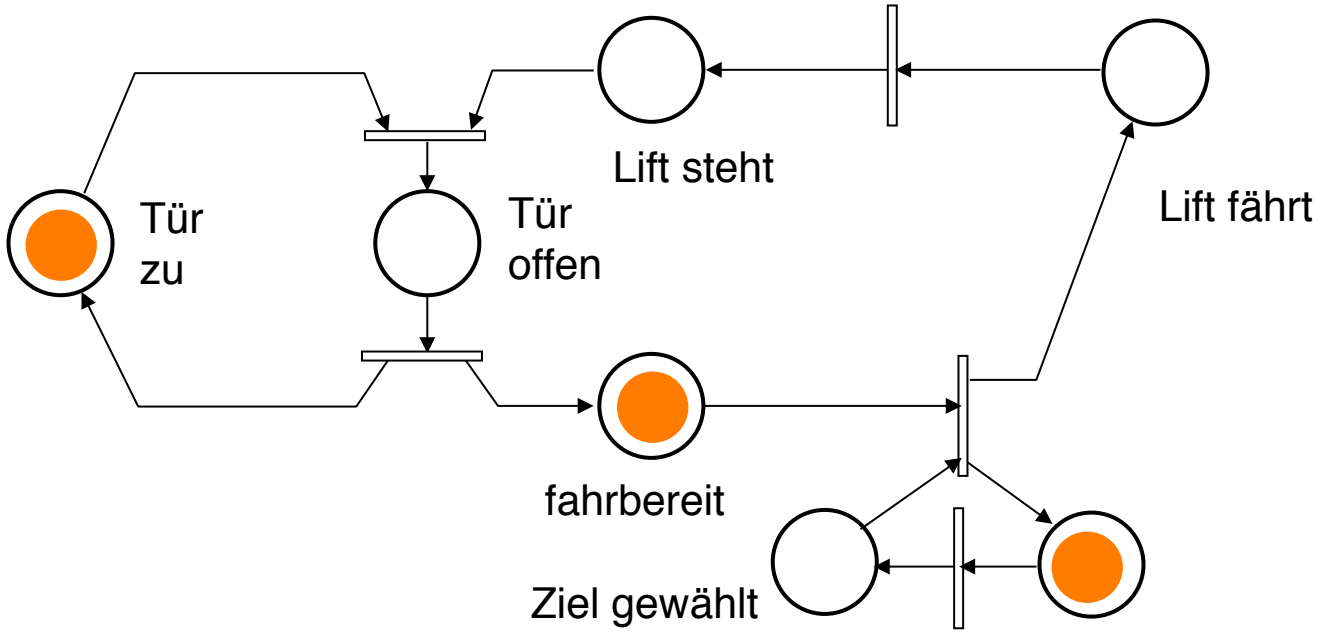
- **Stellen** (places) speichern Information mit Marken und sind Träger des Zustands des modellierten Systems
- **Transitionen** (transitions) modellieren Verarbeitungen und Zustandsveränderungen: Sie entnehmen Information (d.h. Marken) aus Stellen, transformieren diese und legen sie in neuen Stellen ab.
- Eine **Flussrelation** (flow relation) legt fest, welche Transitionen mit welchen Stellen verbunden sind und umgekehrt.
- **Marken** (tokens), die auf den Stellen des Netzes abgelegt sind, modellieren den aktuellen Systemzustand.
- Durch die Benennung von Stellen, Transitionen und Marken werden begriffliche Bezüge zum modellierten Original hergestellt.

# Notation

- Stelle — Kreis
- Transition — Schmales Rechteck oder breiter Strich
- Fluss — Pfeil in Flussrichtung
- Marke — schwarzes oder farbiges Plättchen

# Einfache Petri-Netze

- Einfache Petrinetze verwenden anonyme Marken, lassen pro Stelle höchstens eine Marke zu und modellieren Transitionen ohne Bedingungen.
- Formal handelt es sich um Stellen-Transitionsnetze (siehe dort) mit einer auf eins beschränkten Stellenkapazität

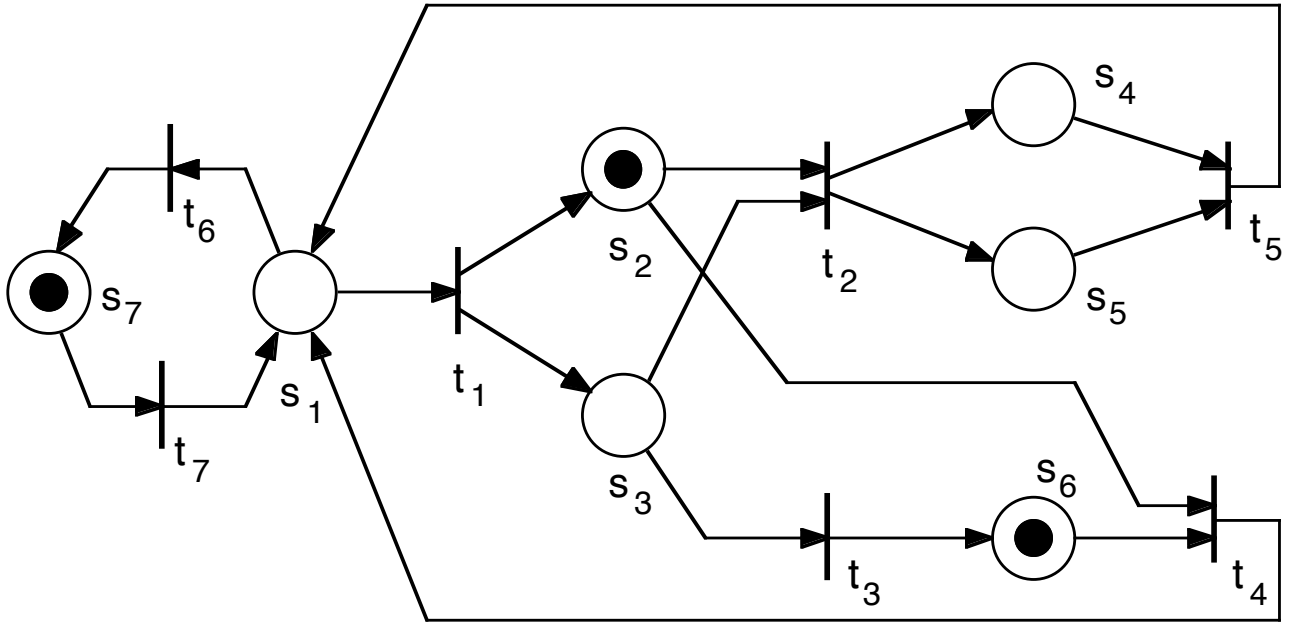


# Interpretation einfacher Petrinetze – 1

- Eine Initialbelegung mit Marken stellt den Startzustand des Netzes dar
- Jede Transition, deren Eingangsstellen alle mit Marken belegt sind, und deren Ausgangsstellen alle frei sind, kann schalten oder „feuern“
- Eine Transition schaltet, indem sie alle Marken von ihren Eingangsstellen entnimmt und jede ihrer Ausgangsstellen mit je einer Marke belegt
- Verschiedene Transitionen können gleichzeitig schalten
- Sind konkurrierende Transitionen (d.h. solche mit mindestens einer gemeinsamen Eingangsstelle) schaltbar, so kann höchstens eine schalten (bestimmt durch Zufall)
- Wann eine schaltbare Transition tatsächlich schaltet, ist nicht bestimmt; sie muss nicht zum frühest möglichen Zeitpunkt schalten

# Aufgabe 7.4

Gegeben sei das folgende einfache Petrinetz:




a) Spielen Sie mögliche Markierungen durch.

b) Können die Stellen s4, s5 und s6 jemals gleichzeitig markiert sein? Begründen Sie Ihre Aussage.

# Aufgabe 7.4 (Fortsetzung)

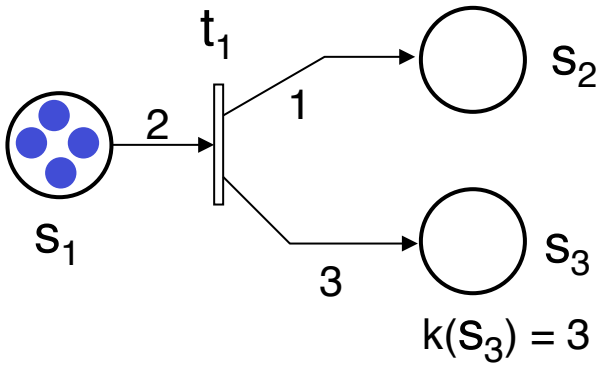
c) Zeigen Sie, dass Sie von der gegebenen Initialbelegung zu einer Markenbelegung gelangen können, in der zwei Transitionen gleichzeitig feuern können.

d) Welche Transitionen konkurrieren miteinander?

# Stellen-Transitionsnetze – 1

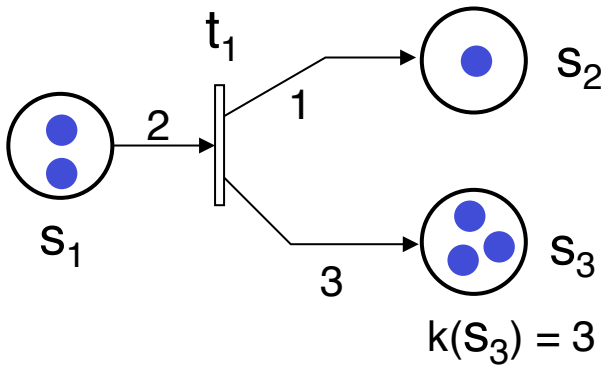
- Stellen-Transitionsnetze sind eine Verallgemeinerung einfacher Petrinetze:
  - jede Stelle hat eine maximale Kapazität k ($1 \leq k \leq \infty$)
  - jeder Fluss hat ein Gewicht (die Zahl der konsumierten bzw. produzierten Marken)

vorher:




$t_1$ schaltet nur, wenn in $s_1$ mind. zwei Marken liegen

nachher:




Wegen der Kapazitätsbeschränkung von $s_3$ kann $t_1$ nicht mehr schalten

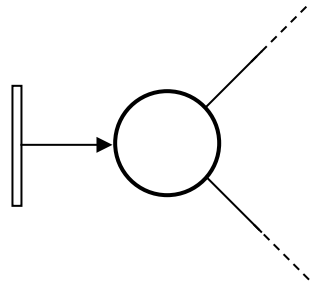
# Stellen-Transitionsnetze – 2

Ein Stellen-Transitionsnetz (place/transition net) ist ein Tupel $(S, T, F, k, w, m_0)$ mit

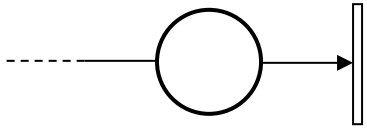
- S ist die Menge der Stellen
- T ist die Menge der Transitionen, $S \cap T = \varnothing$
- F ist die Flussrelation: $F \subseteq (S \times T) \cup (T \times S)$
- $k: S \rightarrow IN$ bezeichnet die Stellenkapazitäten (k ist eine partielle Funktion: Stellen, für die k nicht definiert ist, haben unbeschränkte Kapazität)
- $w: F \rightarrow IN$ bezeichnet die Gewichte der Flusspfeile (Flusspfeile, für die kein expliziter Gewichtswert spezifiziert ist, haben das Gewicht 1)
- $m_0: S \rightarrow IN$ ist die initiale Markierung, wobei für jede kapazitätsbeschränkte Stelle s gilt: $m_0(s) \leq k(s)$

# Quellen und Senken

- Eine Transition ohne eingehende Flüsse heißt Quelle (source). Sie kann jederzeit schalten.
- Eine Transitionen ohne ausgehende Flüsse heißt Senke (sink). Wenn eine Senke schaltet, so konsumiert sie Marken, ohne welche zu produzieren.



Quelle

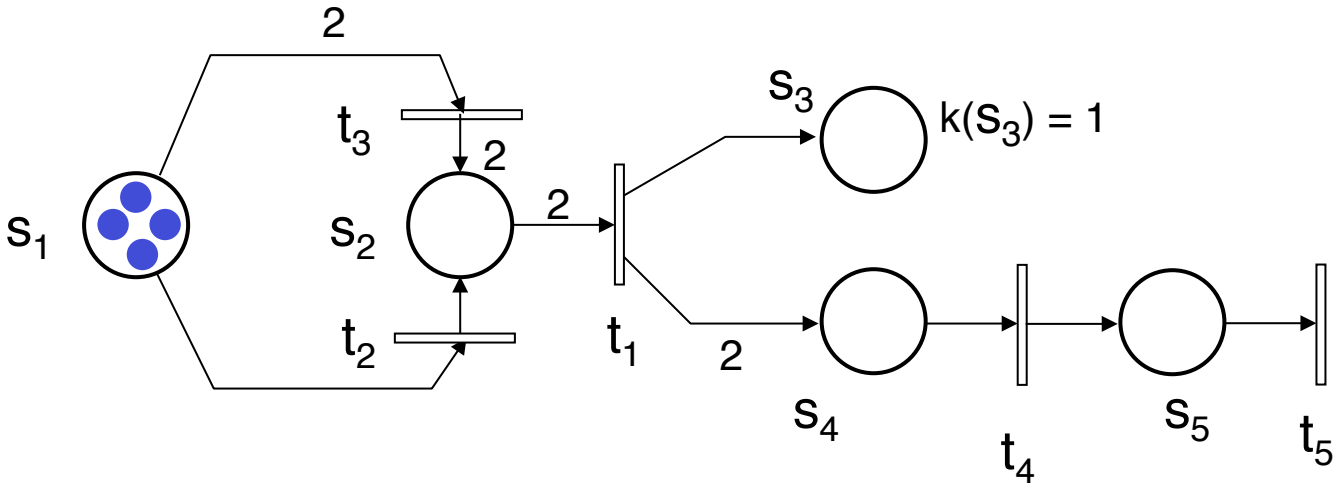


Senke

# Aufgabe 7.5

Bestimmen Sie den Endzustand des folgenden Stellen-Transitionsnetzes. Geben Sie einen Ablauf an, der zu diesem Endzustand führt. Gibt es nur einen möglichen Ablauf?

Hinweis: ein Endzustand ist eine Markenbelegung, bei der keine Transition schalten kann.



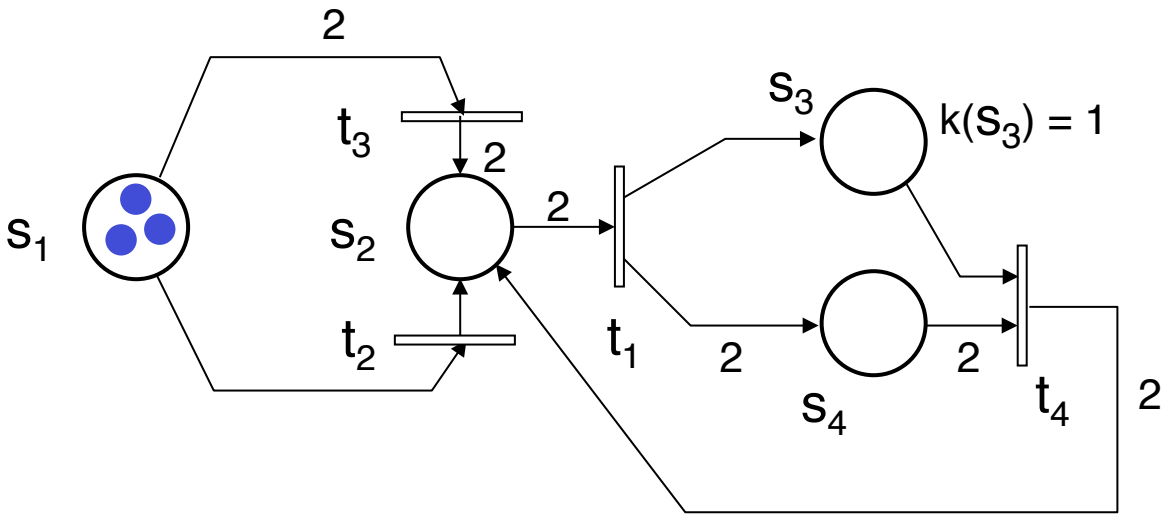
# Formale Analyse von Petrinetzen

- Die Gültigkeit einer Reihe von Eigenschaften eines gegebenen Netzes ist formal analysierbar, beispielsweise
  - Erreichbarkeit einer bestimmten Markierung
  - Unmöglichkeit bestimmter Markierungen
  - Verklemmungsfreiheit: es ist keine Markierung erreichbar, bei der keine Transition schaltbar ist
  - Lebendigkeit: von jeder erreichbaren Markierung aus gibt es eine Folge von Schaltvorgängen, welche alle Transitionen des Netzes umfasst
  - Beschränktheit der maximal auftretenden Marken auf einer Stelle
- Die formale Analyse von Petrinetzen ist kein Gegenstand dieser Vorlesung. Sie wird beispielsweise in [Murata 1988] behandelt

# Aufgabe 7.6

a) Begründen Sie, warum das untenstehende Netz verklemmungsfrei, aber nicht lebendig ist.

b) Was würde geschehen, wenn man das Gewicht des Flusses von $t_4$ nach $s_2$ von 2 auf 1 reduziert?



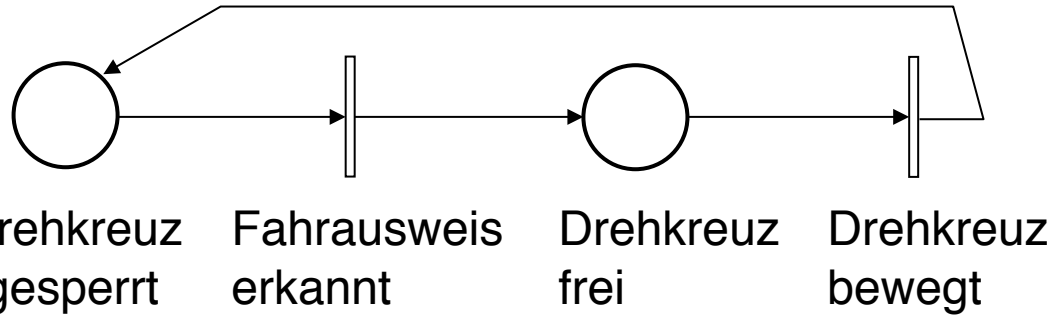
# Prädikat-Transitionsnetze

- Ein Prädikat-Transitionsnetz ist ein Stellen-Transitionsnetz mit folgenden zusätzlichen Eigenschaften:
  - Marken können benannt werden
  - Transitionen können mit Bedingungen versehen werden
  - Namen von Marken an den Flusspfeilen geben an, welche Marken über diesen Pfad zu transportieren sind

# Schaltregeln für Prädikat-Transitionsnetze

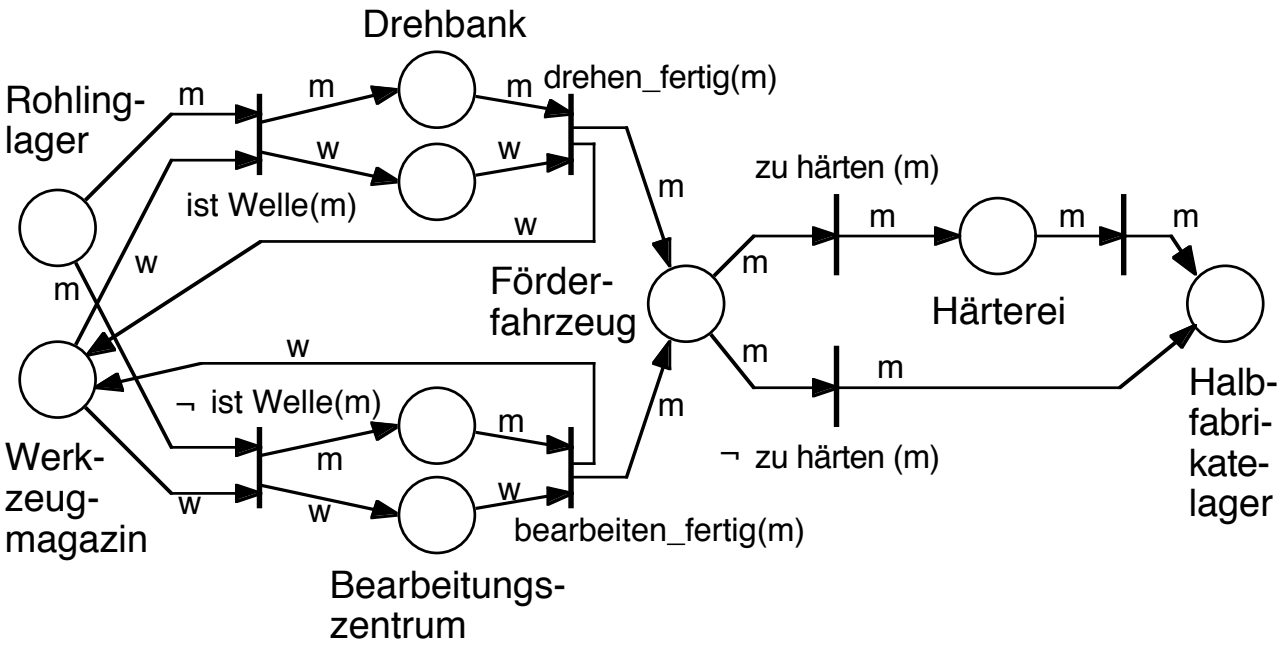
- Eine Transition in einem Prädikat-Transitionsnetz kann schalten, wenn
  - sie nach den Regeln für Stellen-Transitionsnetze schaltbar ist
  - zusätzlich die mit der Transition verbundene Bedingung erfüllt ist
- Häufig werden Ereignisse als Transitionsbedingung modelliert mit der Interpretation, dass die Transition genau dann schaltet, wenn
  - sie nach den Regeln für Stellen-Transitionsnetze schaltbar ist
  - das zugehörige Ereignis eintritt

Beispiel: U-Bahn-Zugang



# Beispiel eines Prädikat-Transitionsnetzes

Stark vereinfachtes Modell einer Produktionsstraße




Zwei Markensorten:
m: Material
w: Werkzeug

Initiale Markierung:
Rohlinglager und Werkzeugmagazin werden mit der gewünschten Zahl von Materialien und Werkzeugen belegt

Kapazitäten:

k(Drehbank) = k(Bearbeitungszentrum) = k(Förderfahrzeug) = 1

Die Transitionen schalten dann, wenn ihre Eingangsstellen mit den erwarteten Marken belegt sind und das Transitionsprädikat wahr ist bzw. wahr wird.

# Aufgabe 7.7

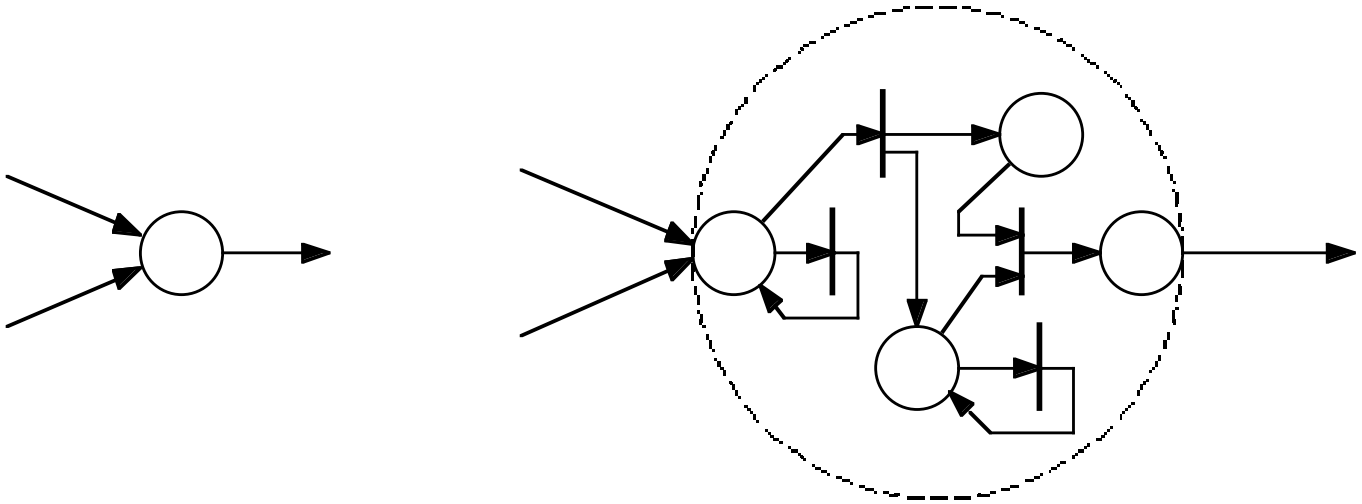
In einer Bankfiliale ist die Beratung von Laufkundschaft (d.h. von Leuten, die sich nicht angemeldet haben) wie folgt organisiert:

Es gibt fünf Bankangestellte, die an zwei Beratungsplätzen Beratungen durchführen. Daneben führen sie auch noch Arbeiten an anderen Arbeitsplätzen in der Filiale aus. Wenn ein Kunde beraten werden will, so wird er einem Beratungsplatz zugewiesen, wenn ein Platz und ein Berater frei sind. Andernfalls muss der Kunde warten.

Modellieren Sie diesen Ablauf mit einem Prädikat-Transitionsnetz.

# Verfeinerung von Petrinetzen

- Petrinetze können hierarchisch verfeinert werden, indem eine Stelle oder eine Transition durch ein Subnetz ersetzt wird
- Beispiel: Verfeinerung einer Stelle



- Dabei sind Konsistenzbedingungen zu beachten, beispielsweise darf ein Subnetz, welches eine Stelle verfeinert, die Zahl und Art der erhaltenen Marken nicht verändern

# Abstraktion von Petrinetzen

- Petrinetze können vergröbert werden, indem man
  - auf eine Belegung mit Marken verzichtet
  - und nur noch Netze modelliert mit
    - Kanälen (aktiv, transportierend/transformierend)
    - Instanzen (passiv, speichernd)
- Kanal-Instanzennetze eignen sich als Übersichtsmodelle
- Sie sind nicht ausführbar

# Literatur


Glinz, M. (2002). Statecharts For Requirements Specification – As Simple As Possible, As Rich As Needed. *ICSE International Workshop on Scenarios and State Machines: Models, Algorithms and Tools*, Orlando, May 2002.

Hagelauer, R. (2002). Schaltkreise und digitale Logikschaltungen, Kapitel 1.3.1: Endliche Automaten. In: P. Rechenberg, G. Pomberger (Hrsg.): *Informatik-Handbuch*, 3. Auflage. München, Wien: Hanser. 276-278.

Harel, D. (1987). Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming* **8** (1987). 231-274.

Harel, D. (1988). On Visual Formalisms. *Communications of the ACM* **31**, 5 (May 1988). 514-530.

Murata, T. (1988). Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, **77**, 4. 541-580.

Object Management Group (2009). *Unified Modeling Language: Superstructure,* version 2.2. OMG document formal/2009-02-02. http://www.omg.org/spec/UML/2.2/Superstructure/PDF/

Petri, C.A. (1962). *Kommunikation mit Automaten*. Dissertation, Universität Bonn.

Reisig, W. (1982). *Petrinetze - Eine Einführung*. Berlin-Heidelberg-New York-Tokyo: Springer. Reisig, W. (1985). *Systementwurf mit Netzen*. Berlin-Heidelberg-New York-Tokyo: Springer.

Von der Beeck, M. (1994). A Comparison of Statechart Variants. In H. Langmaack, W.-P. de Roever, J. Vytopil (eds.): Formal Techniques in Real-Time and Fault-Tolerant Systems. Berlin: Springer. 128-148.

Ward, P.T., S.J. Mellor (1985). *Structured Development for Real-Time Systems*, Vol. I-III. Englewood Cliffs, N.J.: Prentice-Hall.