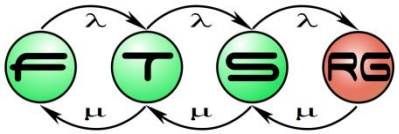
Zustandsmodellierung

Budapesti Műszaki és Gazdaságtudományi Egyetem
Hibatűrő Rendszerek Kutatócsoport



Inhalt

- Vorkenntnisse
- Zustandsräume
- Einfache (Mealy) Zustandsmaschinen
- Zusammengesetzte Zustandsmaschinen
- Ausblick, Softwaren

VORKENNTNISSE

Strukturelle und Verhaltensmodellierung

- Die Struktur (*structural*)
  - statisch
  - Teil und Ganzheit, Bestandteile
  - Verhältnisse, Verbindungen
- Verhalten (*behavioral*)
  - dynamisch
  - zeitlicher Verlauf
  - Zustände, Prozesse
  - Reaktionen auf die Außenwelt

Hauptteile des Roboterstaubsaugers sind das Steuerwerk, das Laufwerk und der Staubsauger.

Auf dem Befehl „rechts” wechselt das Laufwerk seine Betriebsart auf „Abbiegen”.

Hauptfragen der Verhaltensmodelle

- Was „macht” das System?

Ereignisbasierte Modelle

Prozessbasierte Modelle

…

- „Wie” ist das System aktuell und wie verändert es sich?

Zustandsbasierte Modelle

Motivationsbeispiel: virtuelle Tastatur

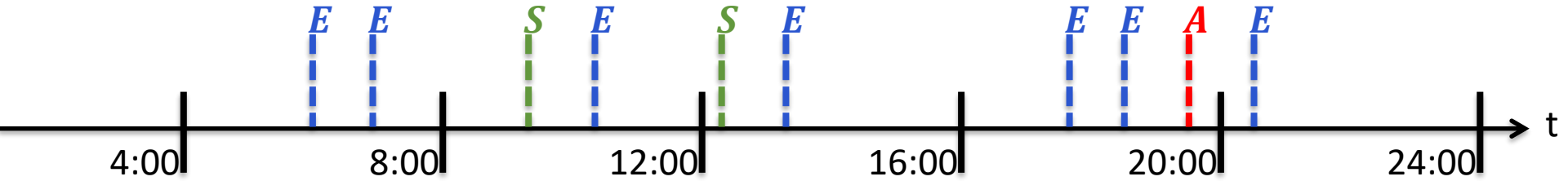
- Was passiert, wenn die linke obere Ecke berührt wird?
  - Q, q, 1 oder =
  - Es ist nur durch „die Vergangenheit” völlig bestimmt

Grundbegriffe: diskrete Ereignisreihe

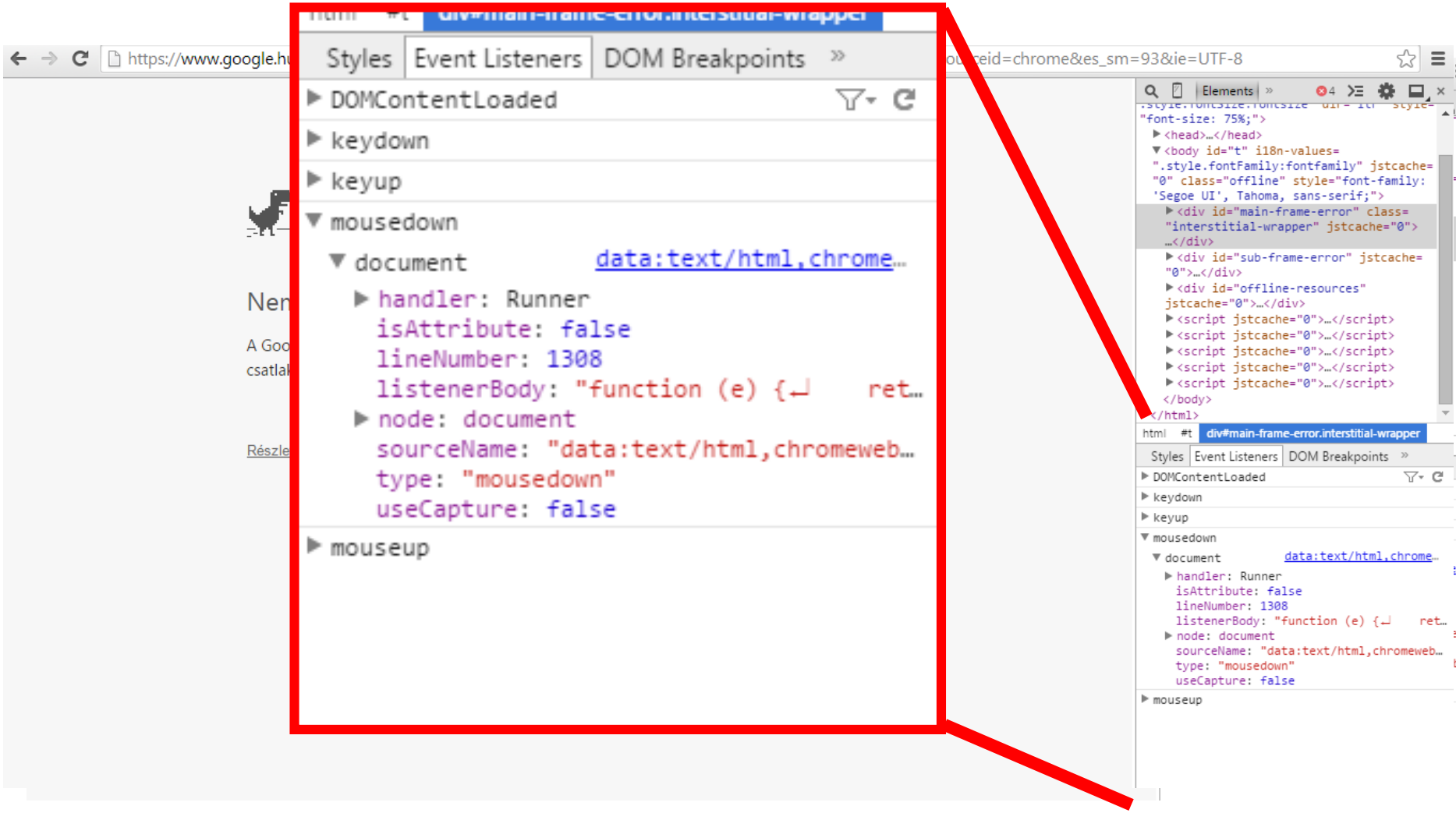
- **Ereignis**
  - **plötzliche Änderung (z.B. am Ein-/Ausgang des Systems)**
- **Ereignisfluss**
  - **z. B. ein für jede Eingang/Ausgang-Datenquelle**
- **Ereignisraum – {erlaubte Ereignisse}**
  - **Einlesbare Eingabewerte/emittierbare Ausgabewerte**
- **Reihe von (plötzlichen) Ereignissen, gleichzeitig ≤ 1**

Ereignisfluss: Handy-Statusnachrichten

Ereignisraum: {*Email*, *SMS*, ***schwache Batterie***}



Ereignisorientierte Programmierung



ZUSTANDSRÄUME

Definition: Zustandsraum

Der **Zustandsraum**

- ist eine Menge voneinander unterscheideter Systemzustände,
- von der gleichzeitig immer genau ein Element (der **aktuelle Zustand**) für das System charakteristisch ist.

- Beispiele: Zustandsräume
  - Tage: {*Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag, Sonntag*}
  - Zustände des Mikrowellengerätes: *{höchste Stufe, Auftauen, Aus}*
- Beispiele: Aktueller Zustand
  - Heute ist *Mittwoch*.
  - Das Mikrowellengerät ist *Aus*.

Eigenschaften des Zustandsraumes

„ immer genau ein Element für das System charakteristisch ist”

- Nicht jede Zustandsmenge kann Zustandsraum sein!

- **Vollständigkeit**
  - Zu jeder Zeit besteht mindestens ein Zustand
  - Gegenbeispiel (nicht geeignet für Zustandsraum!)
    - {*Montag, Dienstag, Donnerstag, Samstag*} nicht vollständig

- gegenseitiger **Ausschluss**
  - zu jeder Zeit besteht höchstens ein Zustand
  - Gegenbeispiel (nicht geeignet für Zustandsraum!)
    - {*Alltag, Wochenende, Nachmittag*} nicht ausschließend
    - Mikrowelle {*die Tür ist offen, ausgeschaltet*}

Warum sind diese Eigenschaften wichtig?

- Schalttag am Flughafen Düsseldorf

29. Februar 2016 | 13.46 Uhr

29. Februar

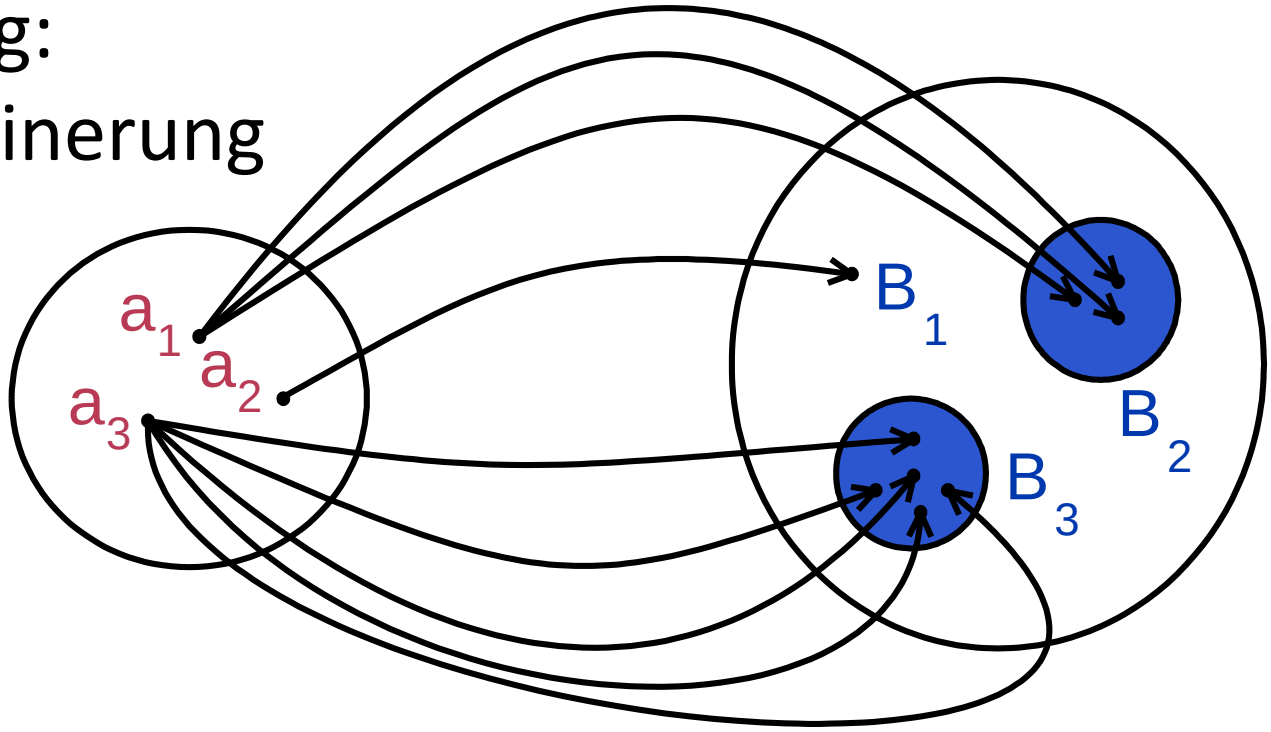
Schalttag legt Gepäckförderband am Flughafen lahm

Zustandsverfeinerung, Zustandsabstraktion

Zustandsverfeinerung bzw. **Zustandsabstraktion** ist eine auf dem Zustandsraum durchgeführte **Mengenverfeinerung** bzw. **Mengenabstraktion**, die als Ergebnis einen neuen Zustandsraum gibt.

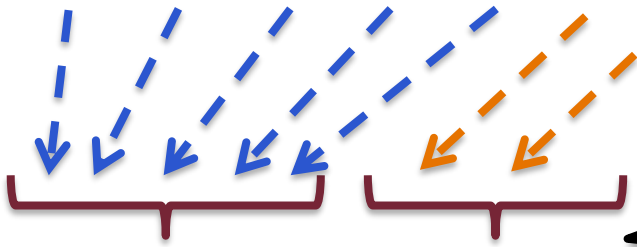
- (auch andere Abstraktionen kommen später vor…)

Wiederholung:
Mengenverfeinerung

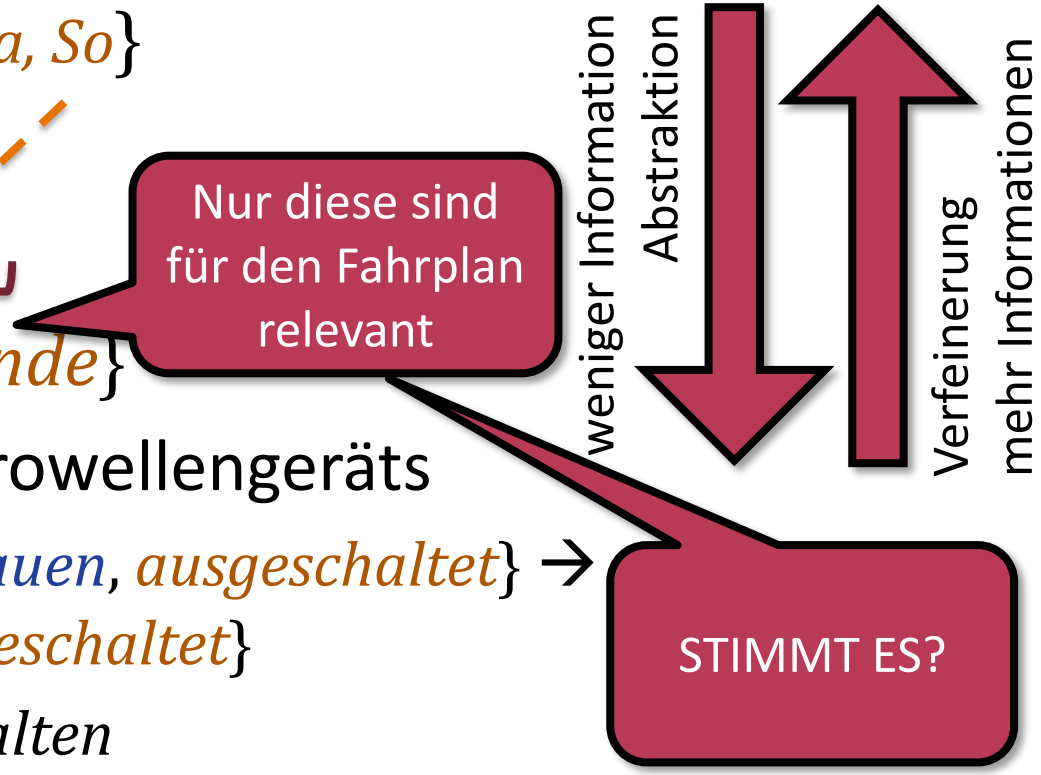


Zustandsverfeinerung, Zustandsabstraktion

- Zustandsverfeinerung/Abstraktion:
  Mengenverfeinerung/Abstraktion des Zustandsraumes
  - {*Mo, Di, Mi, Do, Fr, Sa, So*}



  - {*Werktag*, *Wochenende*}
  - Abstraktion des Mikrowellengeräts
    - {*höchste Stufe*, *Auftauen*, *ausgeschaltet*} → {*eingeschaltet*, *ausgeschaltet*}
    - *„ausgeschaltet" erhalten*



Motivation der Verfeinerung

- Zustandsverfeinerung: warum?
  - Fortschritt der Planung, mehr Implementierungsdetail
    - z.B. zur Starkstromplanung sind die Stufen der Mikrowelle relevant
  - Spezialisation/Erweiterung
    - z.B. ein teureres Mikrowellengerät enthält auch ein Timer
  - Gleichzeitige Untersuchung von mehreren Systemen (siehe später)
- Mehr Informationen, mehr Kenntnisse
  - Ist es immer nützlich?

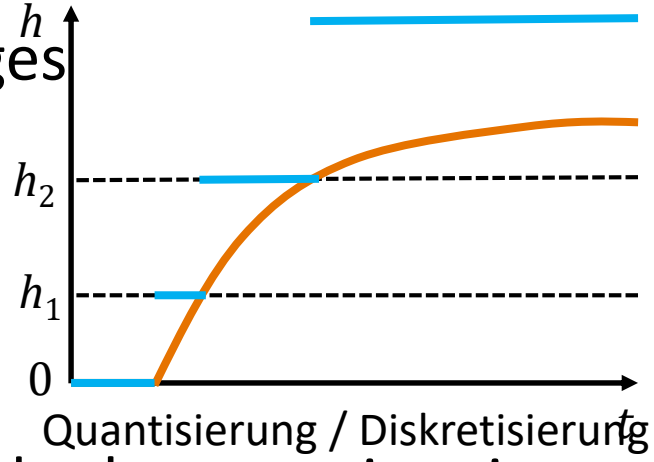
Motivation der Verfeinerung

- Zustandsabstraktion: warum?
  - Nützlich, wenn die abstrakte Zustände „einheitlich” sind
    - Aus irgendwelchem Aspekt sind die vereinten Zustände gleich
    - Siehe „Ersatz-Eigenschafts-Partition” → Digitaltechnik
  - Für bestimmte Aufgaben reicht weniger Information
    - Kleinerer, einfacherer Zustandsraum vereinfacht die Planung
    - Speicherung, Verarbeitung der Zustände sind leichter
    - Verborgene Details sind frei veränderbar
    - Extremer Fall: **zustandsfreies** Modell (|S|=1)
  - Manchmal darf nur weniger Information offengelegt werden

Verbreitet **bei struktureller Dekomposition** (siehe später)

Kontinuierliche Variablen

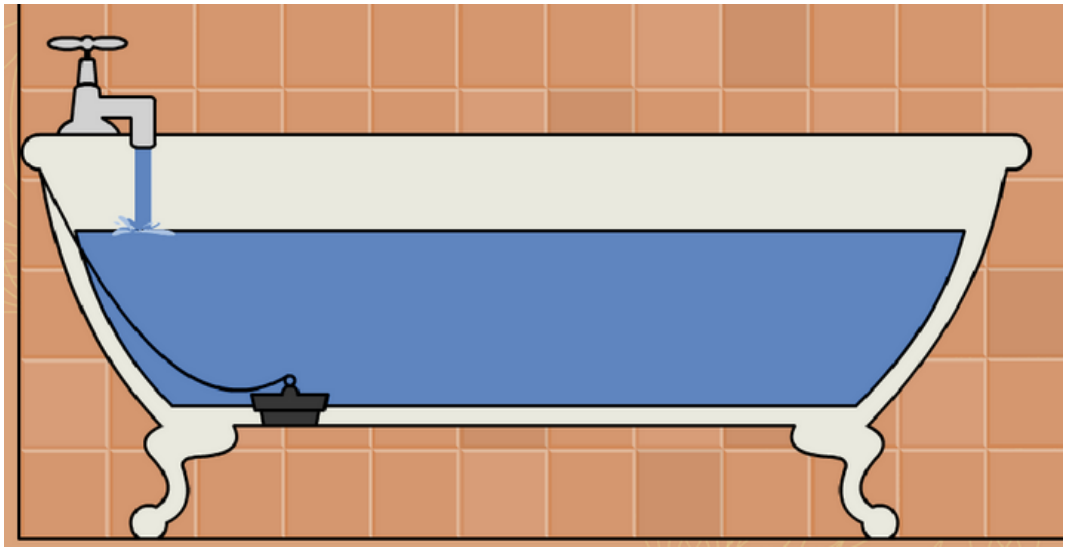
- Potenziell unendliche (sogar Kontinuum-) Zustandsräume
  - z.B. Zustandsvariablen eines Flugzeuges
    - $v \in \mathbb{R}$ Geschwindigkeit
    - $h \in \mathbb{R}$ Flughöhe
    - $\alpha \in [{}^{-\pi}/_{2}, {}^{\pi}/_{2}]$ Anstiegswinkel



  - Die zeitliche Veränderung des Zustandes kann stetig sein
    - z. B. Aufstieg des Flugzeuges: $\partial h/\partial t = v \sin\alpha$
- Aber bei typischen IT-Systemmodellen
  - **diskrete Zustände** (kein stetiger Übergang)
  - häufig **endlicher Zustandsraum** (Gegenbeispiel: Zähler $\in \mathbb{N}$)
  - zeitlose **Zustandsübergänge,** inzwischen fixe Zustände

Hybride Systeme

- Kontinuierliche und diskrete Veränderungen
- Beispiel: Badewanne
  - Diskrete Variable: Hahn {offen;zu}, Stöpsel {ein;aus}
  - Kontinuierliche Variable: Pegel $-\ 0 \leq h \leq 60\ cm$
  - Diskrete Variablen haben Einfluss auf die Änderung der kontinuierlichen



Prädikatabstraktion

- Abstraktion der Wertebereich einer (kont.) Variable entsprechend Prädikaten
- Bespiel: Pegel
  - 0 – Boden der Wanne
  - 40 –„Max” Zeichen
  - 60 – Rand der Wanne

Prädikate

$h = 0, h > 0$

$h \leq 40, h > 40$

$h < 60, h = 60$

- Abstrakter Zustandsraum: Kombinationen der Prädikate

$$\{h = 0, h > 0 \wedge h \leq 40, h > 40 \wedge h \leq 60, h = 60\}$$

Wanne leer

Normaler Pegel

Kritischer Pegel

Wanne übergelaufen

Prädikatabstraktion

- Prädikatabstraktion:
  - Aus einem gegebenen Aussagensatz (Prädikatenmenge)
    - auf welchen Zustand
    - welche Aussage ist wahr
  - Gleiche Aussagen wahr→ gemeinsamer abstr. Zustand

Vollständig: jeder Zustand wird in irgendeiner Gruppe

Ausschließend: keine Aussage kann gleichzeitig wahr und falsch sein

Die **Prädikatabstraktion** ist eine Abstraktion, die

- die konkrete Zustände in Klassen einteilt
- anhand logischer Aussagen (**Prädikate**)
- und beschreibt sie mit den von den Elementen erfüllten Aussagen

Partitionierung nach mehreren Aspekten

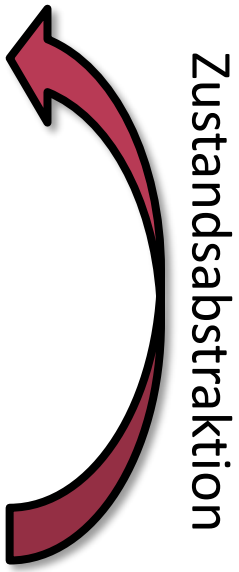
- Ein System – Auch mehrere richtige Zustandsräume
- z.B. zwei verschiedenen Zustandsräume des Mikrowellengerätes
  - bezüglich der Betriebsleistung:
    - {*höchste Stufe*, *Auftauen*, *ausgeschaltet*}
  - Offensein der Tür:
    - {*offen*, *zugeschlossen*}
  - nicht vollständig unabhängig: falls offen, dann ausgeschaltet

- Zustandsraum des Teilsystems→ Ohne Kenntnisse über den Systemzustand ist es entscheidbar, welcher Zustand aktuell ist

(Direktes) Produkt von Zustandsräumen

- Gleichzeitiger Betrachtung von zwei Zustandsräumen
  - S_1 = {*höchste Stufe*, *Auftauen*, *ausgeschaltet*}
  - S_2 = {*offen*, *zugeschlossen*}

$S_1 \times S_2$	*offen*	*zugeschlossen*
höchste Stufe	*höchste Stufe* und *offen*	*höchste Stufe* und *zugeschlossen*
Auftauen	*Auftauen* und *offen*	*Auftauen* und *zugeschlossen*
ausgeschaltet	*ausgeschaltet* und *offen*	*ausgeschaltet* und *zugeschlossen*



Bemerkung: $|S_1 \times S_2| = |S_1| \cdot |S_2|$

(Direktes) Produkt von Zustandsräumen

Direktes Produkt von Zustandsräumen

- **Komposition-Operation** auf den Komponentzustandsräumen**,**
- dessen Ergebnis ein neuer Zustandsraum (**Produkt-Zustandsraum**) ist,
  - welche als Descartes-Produkt der Mengen von Komponentzustandsräume entsteht.

In dem Produkt-Zustandsraum entspricht

- jeder Zustandskombination der Komponentzustandsräume
- ein zusammengesetzter Zustand (**Zustandsvektor**) .

$$\{\text{AM,PM}\} \times \{\text{1h..12h}\} \times \{\text{0m..59m}\}$$
$$\Downarrow$$
$$\langle PM, 12h, 08m \rangle$$

Projektion des Zustandsraums auf ein Komponent

Projektion auf Komponente ist

- eine Zustandsabstraktion-Operation,
- die aus dem Zustandsraum des Produktes
  - ein oder mehrere Komponente erhält,
  - die anderen werden vernachlässigt.

$$\{\text{AM,PM}\} \times \{\text{1h..12h}\} \times \{\text{0m..59m}\}$$

$$\Downarrow$$

$$\langle PM, 12h, 08m \rangle$$

$$\Downarrow$$

$$\langle PM, 12h \rangle$$

(Wie bei der Projektion auf Tabellen)

Verfeinerung der Komp. der Zust.variablen

- Zust.variablen können voneinander abhängig sein
  - nicht jede Kombination kommt tatsächlich vor
  - kompositer Zustandsraum ist feiner als das Produkt

Der **potentielle Zustandsraum** ist ein Zustandsraum, der den **echten Zustandsraum** enthält.

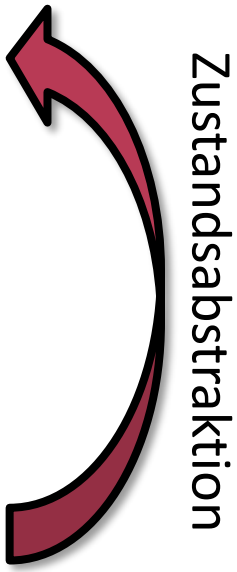
Merke, daß der potentieller Zustandsraum ist **eine Abstraktion** des echten Zustandsraumes:

- „es wird vergessen", ob ein Zustand tatsächlich vorkommen kann
- das direkte Produkt kann kompakter angegeben werden
  - Kleinerer Informationsgehalt: *n+m* Element (Wertebereiche der Variablen) ist kleiner als *n*m* (Wertebereich des Produktes)

Verfeinerte Komposition der Zustandsvariablen

- Nicht unabhängige Zustandsvariablen
  - Kommt nicht jede Kombination tatsächlich vor
  - Der komposite Zustandsraum ist feiner als das Produkt

$S \subseteq S_1 \times S_2$	*offen*	*zugeschlossen*
höchste Stufe	*höchste Stufe* und *offen*	*höchste Stufe* und *zugeschlossen*
Auftauen	*Auftauen* und *offen*	*Auftauen* und *zugeschlossen*
ausgeschaltet	*ausgeschaltet* und *offen*	*ausgeschaltet* und *zugeschlossen*



Bemerkung: Projektion als Abstraktionsverhältnis bleibt erhalten

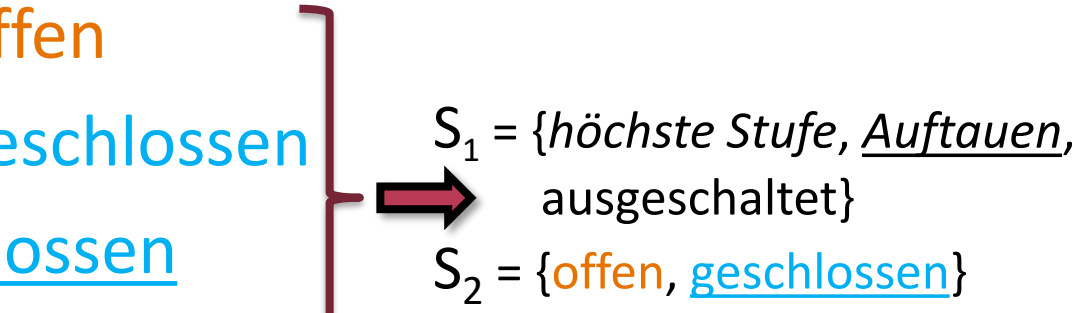
Verfeinerte Komposition der Zustandsvariablen

- „Der komposite Zustandsraum ist **feiner** als das Produkt”
  - … da das Vorkommen von zwei kompositen Zuständen ausgeschlossen wurden
  - Hier hat der verfeinerte Zustandsraum *weniger* Elemente!
    - Im Fall der Zustandsverfeinerung wurde die Anzahl der Zustände höher
    - Nach der Verfeinerung kann sich die Größe des Zustandsraumes sowohl erhöhen als auch verringern
    - das Wesentliche: **über das System wird mehr gewusst**
    - Also: **weniger Systeme entsprechen dem Modell**

$S \subseteq S_1 \times S_2$	*offen*	*zugeschlossen*
höchste Stufe	*höchste Stufe* und *offen*	*höchste Stufe* und *zugeschlossen*
Auftauen	*Auftauen* und *offen*	*Auftauen* und *zugeschlossen*
ausgeschaltet	*ausgeschaltet* und *offen*	*ausgeschaltet* und *zugeschlossen*

Dekomposition von Zustandsvariablen

- Dekomposition: Umkehrung des Produkt / der Komposition
  - ausgeschaltet und offen
  - ausgeschaltet und geschlossen
  - Auftauen und geschlossen
  - *höchste Stufe und geschlossen*



- Die projizierte Zustandsvariablen sind Abstraktionen
- Ihr Produkt ist nur ein potentieller Zustandsraum
- Warum wird dekomponiert?
  - um Zustandsvariablen getrennt zu behandeln
  - um Zustandsvariablen getrennt zu speichern

Ausblick: Beispiele aus unserem Beruf

- Wo kommt die Zustandsmodellierung in der IT vor?
- Soziale Netzwerke – Bekanntschaft zw. Hänsel und Gretel
  - (davon hängen manche Funktionen ab, z.B. welche Bilder gezeigt werden)
  - Zustandsvariablen:
    - Hänsel hat Gretel angefragt
    - Vice versa

Speichern:
- Im Speicher
- In einer Datenbank (langfristig)

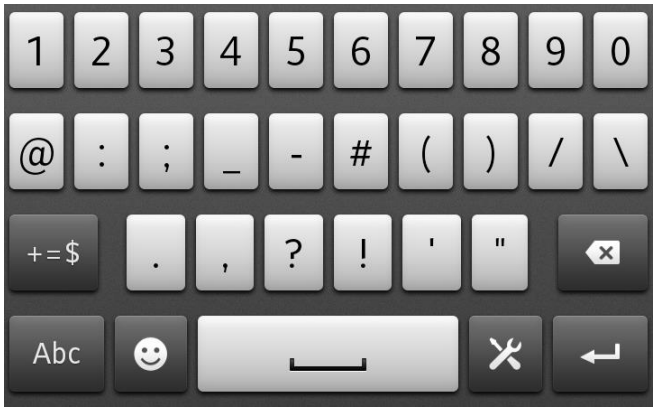
$S_1 \times S_2$		
	Keine Bekanntschaft	Hänsel hat Gretel angefragt
	Gretel hat Hänsel angefragt	Bekanntschaft

Ausblick: Beispiele aus unserem Beruf

- Wo kommt die Zustandsabstraktion vor?
- Soziale Netzwerke – nur Teile der Datenbank sind präsentiert
  - Unnötige Information, ob Hänsel Gretel kennt
    - Unbefugte dürfen es auch nicht wissen!
  - Viel weniger Datenverkehr
  - Dekomposition des Softwaresystems
    - Einfachere Präsentationsschicht (HTML + CSS + JavaScript), falls es nur mit den für mich bestimmten Informationen arbeitet
    - Danach sind verschiedene Mobilklienten leichter anzufertigen
    - Die Ingenieure des sozialen Netzwerkes müssen die Abfrage der für mich relevanten Daten nur einmal, nur an einer Stelle verwirklichen

Ausblick: Beispiele aus unserem Beruf

- Virtuelle Tastatur fürs Touchscreen
  - Zustandsvariablen?



Ausblick: Beispiele aus unserem Beruf

- Programmierung: wie speichern wir den Zustand?
  - Variable mit entsprechendem Wertebereich (Objektfeld, usw.)

```
enum VirtualKeyboardState {
   LOWER_CASE,
   UPPER_CASE_ONCE,
   UPPER_CASE_LOCK,
   NUMBERS_COMMON_SYMBOLS,
   RARE_SYMBOLS
}
// ...
VirtualKeyboardState keyboardState;
```

  - Ergänzung: erinnern wir uns an den SHIFT-Zustand!
    - der alfanumerische Modus kehrt mit dem SHIFT-Zustand zurück, in dem er verlassen wurde

Ausblick: Beispiele aus unserem Beruf

- Programmierung: wie speichern wir den Zustand?
  - Ergänzung: erinnern wir uns an den SHIFT-Zustand!

```
enum VirtualKeyboardStateWithMemory {
    LOWER_CASE,
    UPPER_CASE_ONCE,
    UPPER_CASE_LOCK,
    NUMBERS_COMMON_SYMBOLS_WITH_LOWER_CASE,
    NUMBERS_COMMON_SYMBOLS_WITH_UPPER_CASE_ONCE,
    NUMBERS_COMMON_SYMBOLS_WITH_UPPER_CASE_LOCK,
    RARE_SYMBOLS_WITH_LOWER_CASE,
    RARE_SYMBOLS_WITH_UPPER_CASE_ONCE,
    RARE_SYMBOLS_WITH_UPPER_CASE_LOCK
}
// ...
VirtualKeyboardStateWithMemory keyboardStateWithMemory;
```

  - Das Phänomen heißt **Zustandsraum-Explosion**

Ausblick: Beispiele aus unserem Beruf

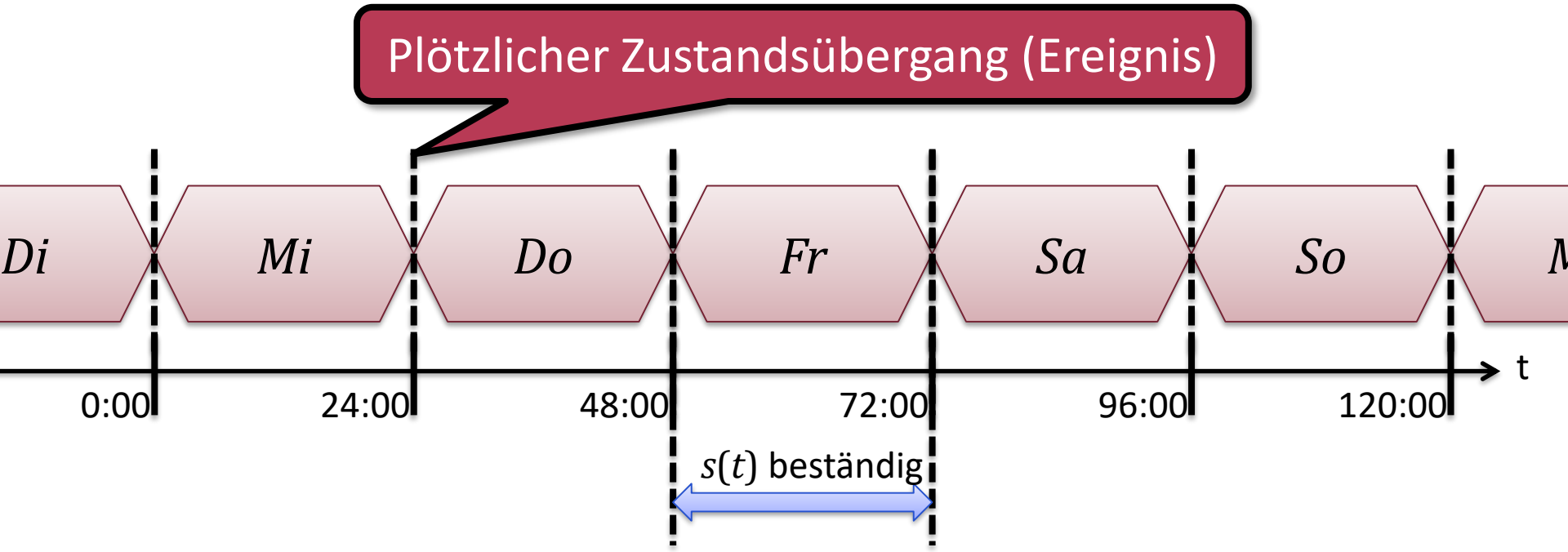
- Programmierung: wie speichern wir den Zustand?
  - kompakte Lösung: mit mehreren Zustandsvariablen

```
enum VirtualKeyboardFacet {
    ALPHABETIC,
    NUMBERS_COMMON_SYMBOLS,
    RARE_SYMBOLS
}
enum CapsState {
    LOWER_CASE,
    UPPER_CASE_ONCE,
    UPPER_CASE_LOCK
}
// ...
VirtualKeyboardFacet keyboardFacet;
CapsState capsState;
```

EINFACHE (MEALY) ZUSTANDSMASCHINEN

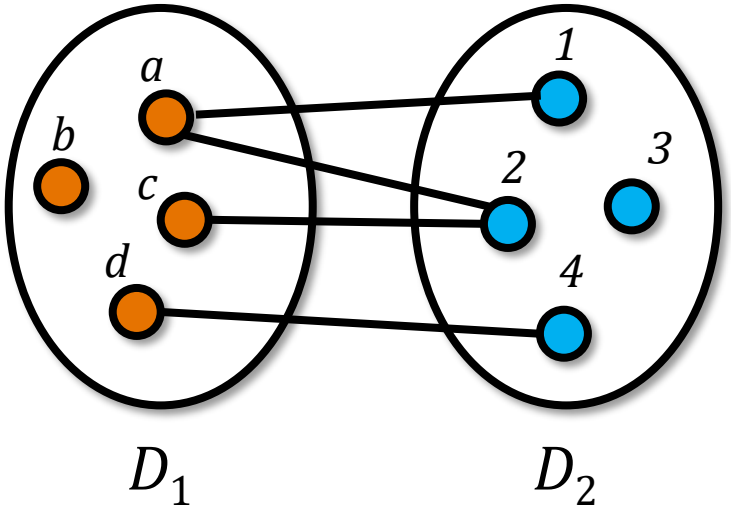
Zustandsübergänge

- Zustandsraum: S
  - z.B. $S = \{Mo, Di, Mi, Do, Fr, Sa, So\}$
- $s(t) \in S$
  - Der aktuelle Zustand nach der Zeit dargestellt



Wiederholung: binäre Relation

- binäre Relation :
  - Teilmenge des Descartes-Produktes zweier Mengen
  - $R \subseteq D_1 \times D_2$



$$R = \{(a, 1), (a, 2), (c, 2), (d, 4)\}$$

Zustandsübergänge

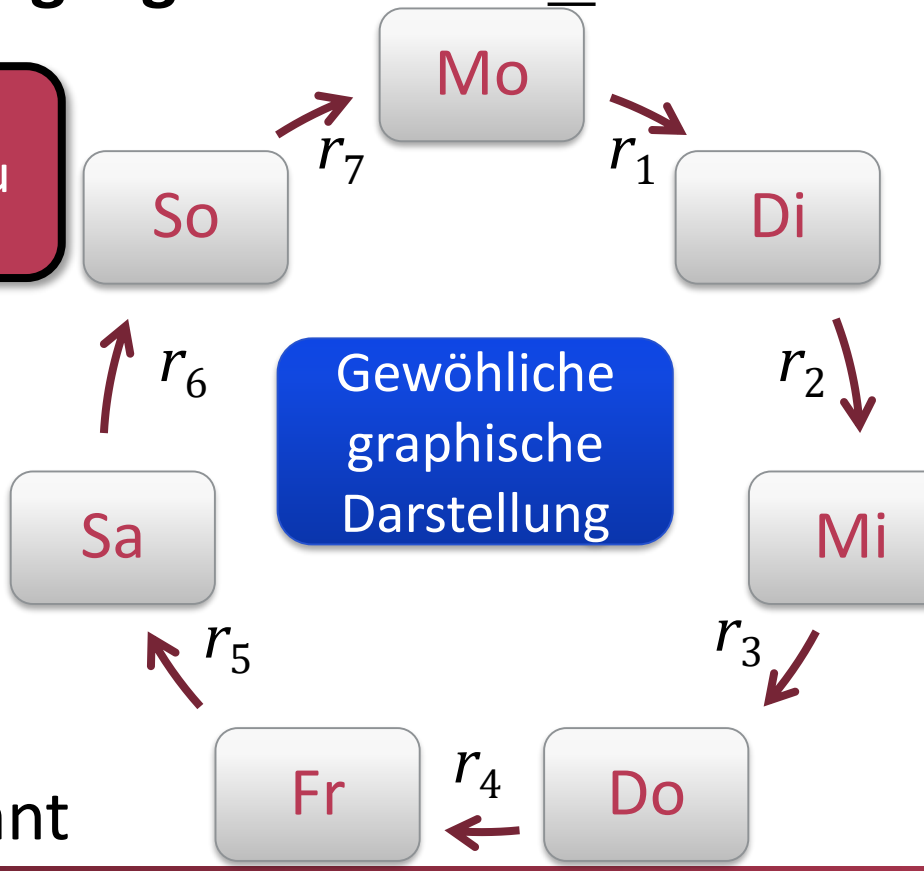
- Welche Zustände können welche Zustände folgen?
  - Zustandsraum $S = \{Mo, Di, Mi, Do, Fr, Sa, So\}$
  - Ereignisraum: **Zustandsübergangsrelation** $R \subseteq S \times S$

Zustandsübergang

Regeln $r \in R$

- $r_1 = (Mo, Di)$
- $r_2 = (Di, Mi)$
- $r_3 = (Mi, Do)$
- $r_4 = (Do, Fr)$
- $r_5 = (Fr, Sa)$
- $r_6 = (Sa, So)$
- $r_7 = (So, Mo)$

Von Do kann das System zu Fr übergehen



  - Auch **Zustandsgraph** genannt

Bemerkungen über den Zustandsgraphen

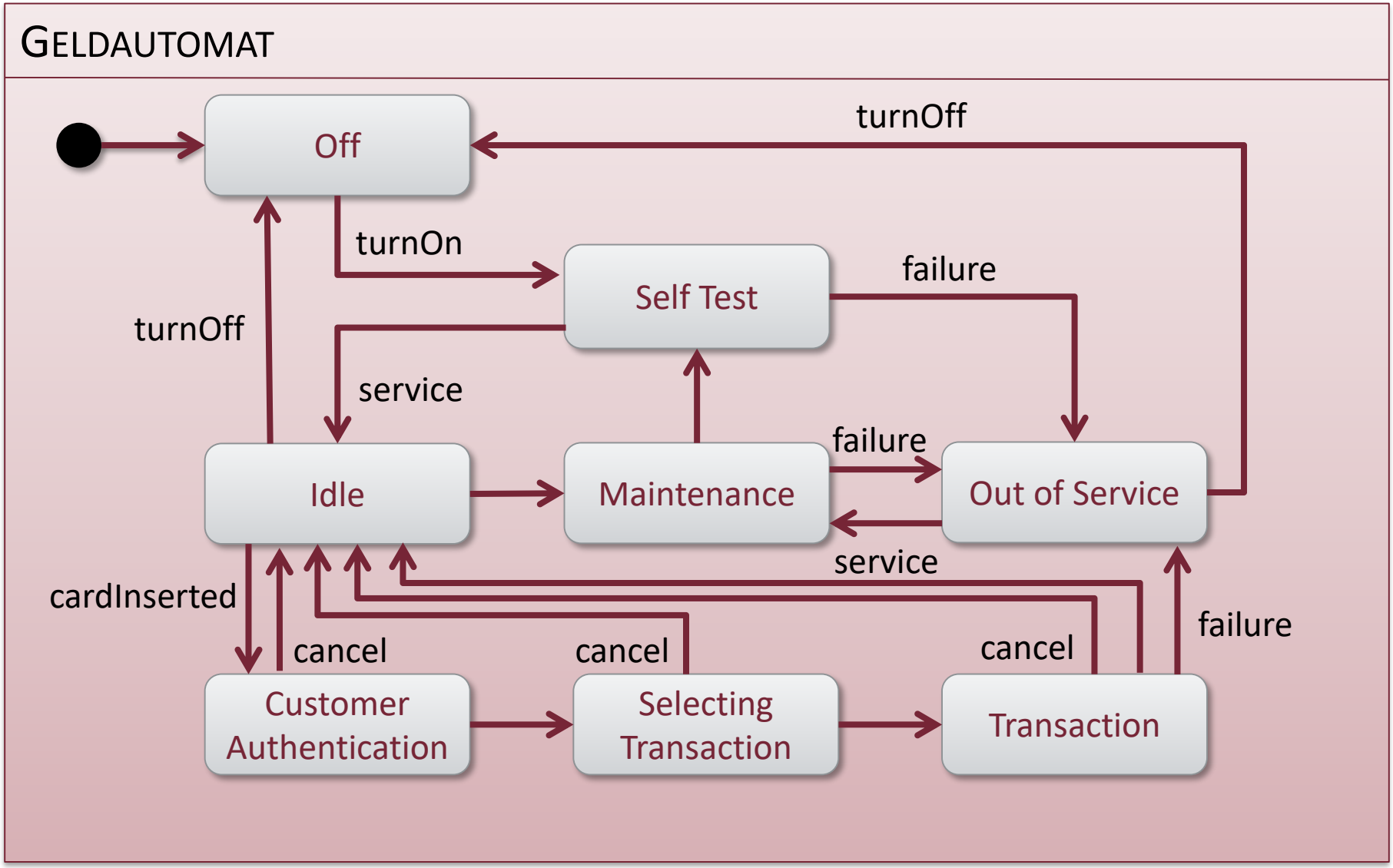
- Es ist möglich, dass…
  - der Graph vollständig ist → jeder Übergang ist erlaubt
    - $S_{\text{Kätzchen}} = \{schläft, spielt, trinkt\}$
  - bestimmte Zustände nicht aus jedem Zustand erreichbar sind
    - Pl. $S_{\text{Glas}} = \{leer, voll, gebrochen\}$ → kein Pfad $gebrochen \rightsquigarrow leer$
  - bestimmte Zustände auch mehrere Nachfolger haben

Nichtdeterminismus

- Mögliche Ursprünge:
  - Funktion des modellierten Systems
  - Während der Modellierung eingeführte Abstraktion

z.B. interne Variable, die nicht betrachtet wird, Steuerungsinput, …

Zustandsgraph-Beispiel: ATM



Zustandsübergänge und Ereignisse

- Markierung der Übergangsregel
  - plötzliches Ereignis
  - der Übergang kann nur mit dem Ereignis zusammen auftreten

- Verschiedene Interpretierung: das Ereignis kann …
  - die Folgerung des Übergangs sein (Post-Kondition)

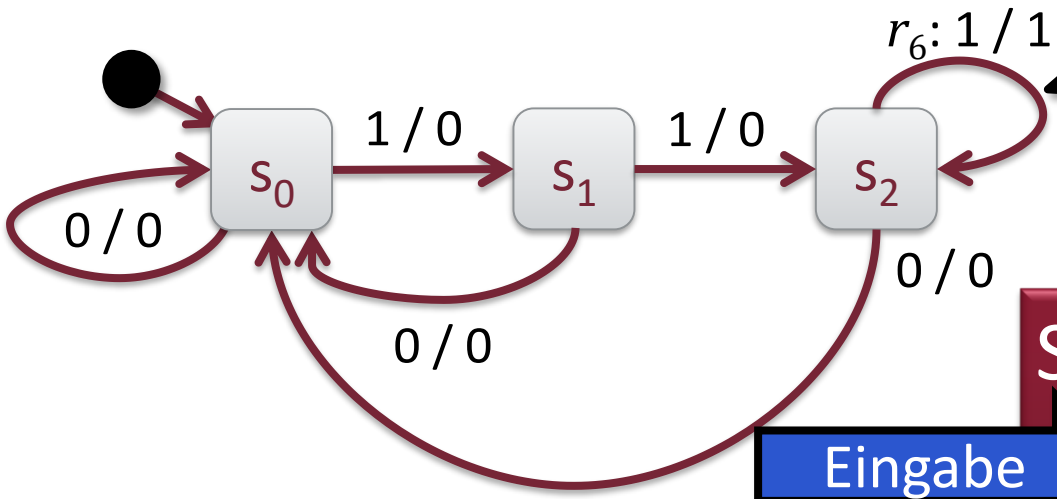
  - die Ursache des Übergangs sein (Pre-Kondition)

- Eine Regel kann mit mehreren Ereignissen beschriftet werden

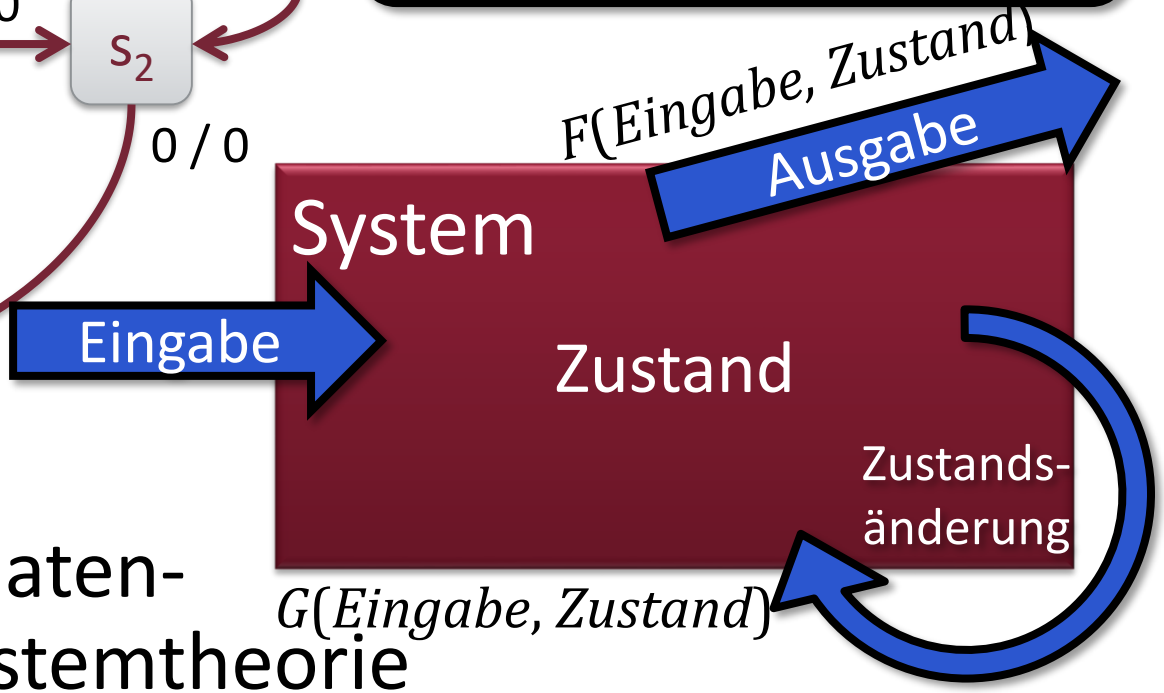
  - z.B. gelesene Eingabe(n) / geschriebene Ausgabe(n)

Gedächtnisstütze: endlicher Mealy-Automat

- markierter Anfangszustand → $s_0 = s_{t=0}$
- Jeder Schritt ist deterministisch, liest die Eingabe und schreibt eine Ausgabe



zum Nachdenken:
- Was macht diese Maschine?
- Falls r_6 in s_0 bringen würde?

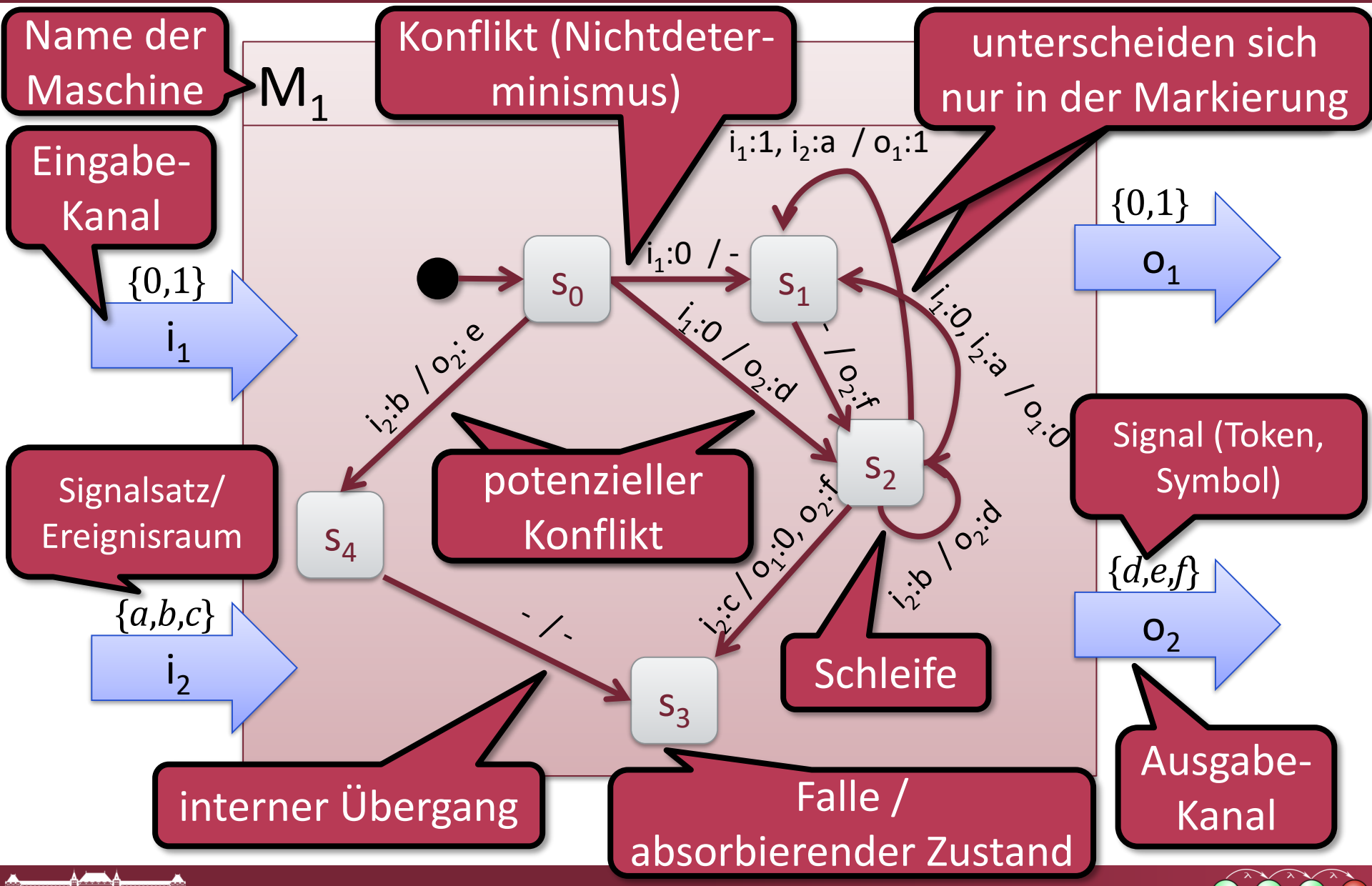


- Klassischer Automaten-Modell aus der Systemtheorie

Erweiterungen der Mealy-Automaten

- Nichtdeterministisches Modell (bez. Eingabe)
- (**Spontane**) Schritte, die kein Lesen von Eingaben erfordern
  - aufgrund eines internen, nicht modellierten Ereignisses
  - z.B. Mikrowellengerät ist fertig, stoppt → Timer nicht modelliert
- Mehrere Ausgabekanäle (getrennte Ereignisflüsse!)
  - mit getrenntem Signalsatz
  - jede Regel schickt entsprechende Signale auf manche Kanäle
- Mehrere Eingabekanäle (getrennte Ereignisflüsse!)
  - jede Regel liest Signale von manchen Kanälen
  - es kann zu Nichtdeterminismus führen

Erweiterte Zustandsmaschine

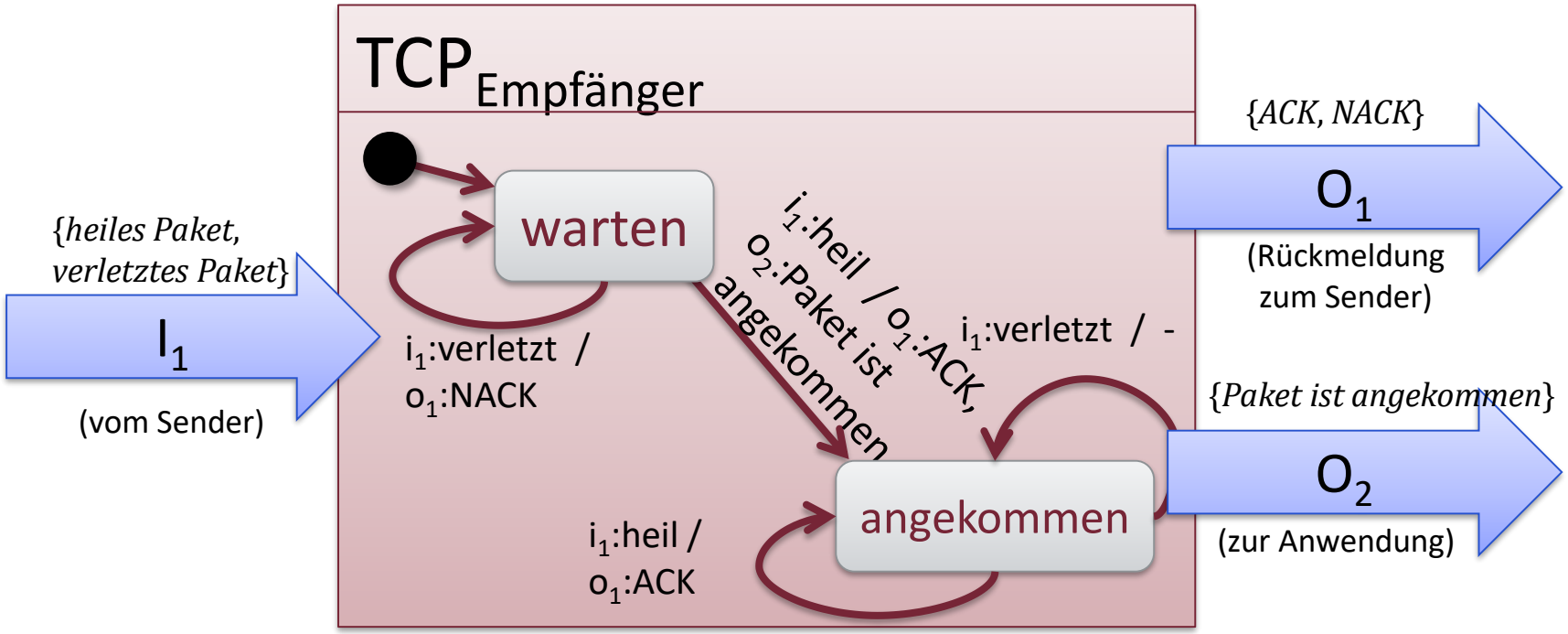


Unspezifizierte Eingabe

- Braucht jeder Zustand für jede Eingabe eine Regel?
  1. Falls keine Regel→ Ereignis kann nicht auftreten
     - Mathematisch richtig, aber für Eingaben keine realistische Annahme
     - Was passiert, falls es dennoch auftritt?
  2. Falls keine Regel → unsichtbare Schleife
     - „Alle andere” Eingaben einlesen, außer Acht lassen
  3. Falls keine Regel→ ungültiges Modell
     - z.B. bei sicherheitskritischen eingebetteten Systemen
     - Auch ungültig, falls es mehrere Regel gibt (Determinismus erfordert)
- Falls es immer Regel gibt -> vollständig spezifiziert

Ausblick: Beispiele aus unserem Beruf

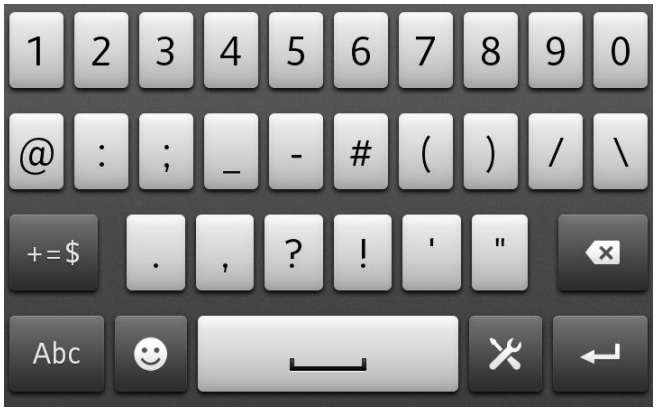
- Paketbasiertes Datenübertragungsprotokoll
  - Pakete können verloren gehen oder verletzt werden
  - Empfangsbestätigung, Wiederholung



- (weiter siehe bei Computernetzwerke)

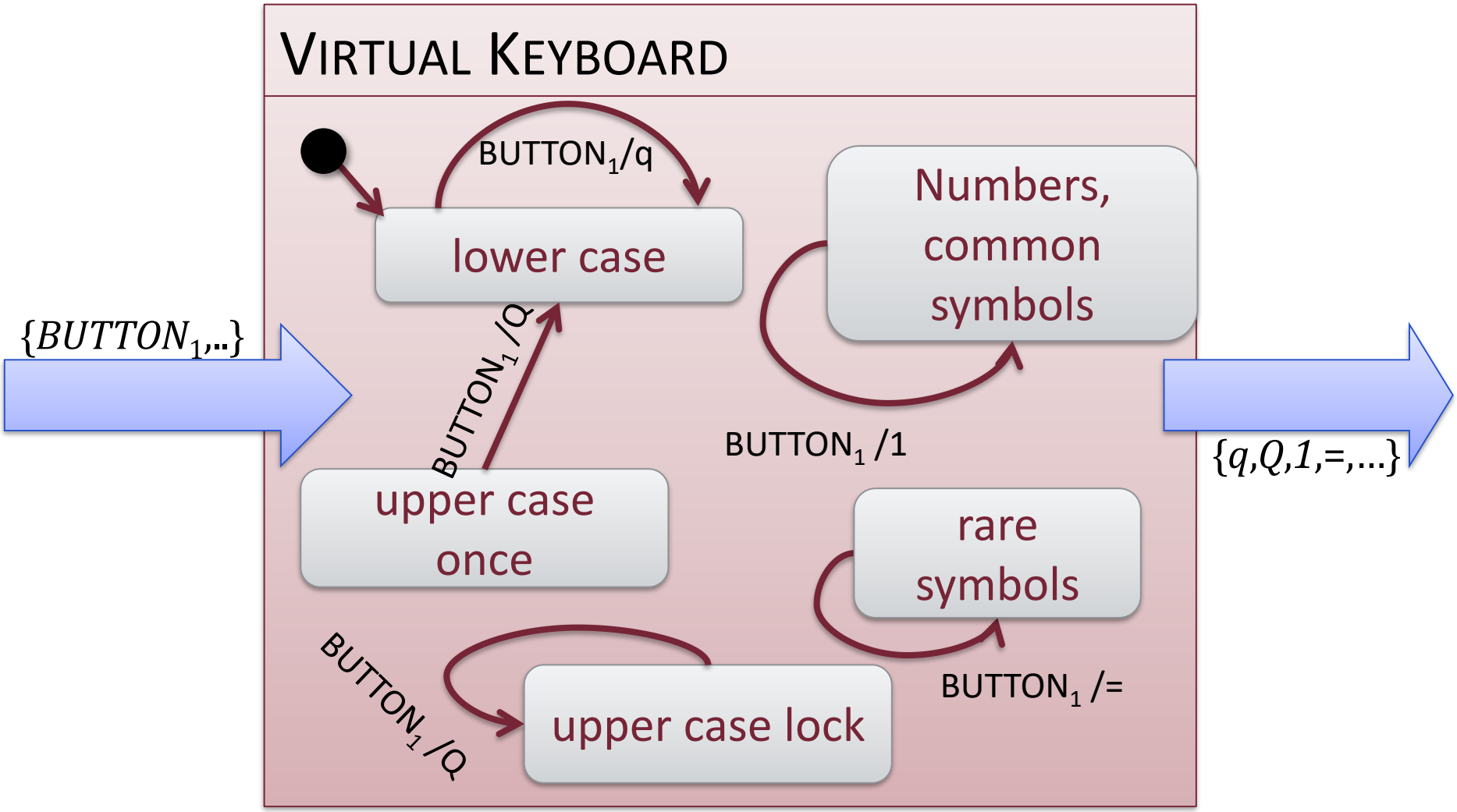
Denkzettel

- Virtuelle Tastatur fürs Touchscreen



Ausblick: Beispiele aus unserem Beruf

- Virtuelle Tastatur: Teil der Zustandsmaschine



Ausblick: Beispiele aus unserem Beruf

- Programmierung:
  Implementierung
  einer Zustandsmaschine
  → Grundl. der Prog. I.
  - Verzweigungsbedingung:
    - aufgrund Zustandsvariablen und Eingabe
  - Auf jedem Zweig:
    - Ausgabe (falls existiert)
    - Zustandswechsel (falls nötig)

```
void handleKey(KeyCode input) {
    switch(input) {
    case BUTTON_1:
        switch(keyboardState) {
        case LOWER_CASE:
            emit('q');
            break;
        case UPPER_CASE_ONCE:
            keyboardState = LOWER_CASE;
        case UPPER_CASE_LOCK:
            emit('Q');
            break;
        case NUMBERS_COMMON_SYMBOLS:
            emit('1');
            break;
        case RARE_SYMBOLS:
            emit('=');
        }
        break;
    case SWITCH_1:
        //...
```

Ausblick: Beispiele aus unserem Beruf

- Falls das (Zustandsmaschinen-)Modell…
  - … wohldetailliert (deterministisch), und
  - … in einer Form ist, die verarbeitet werden kann
    - siehe fachbereichspezifische Zielsprachen (z.B. Protokollplanung)
    - siehe standardisierte Modellierungsformate (z.B. UML)
  - … dann kann es automatisch zum Programkode übersetzt werden
    - z.B. Kodegenerierung zur Kommunikationsprotokolle
    - z.B. Modellbasierte Entwicklung von eingebetteten Steuergeräten
  - … oder mit allgemeinem Interpreter ausgeführt werden
    - z.B. Automatisierung des IT-Systembetriebes

ZUSAMMENGESETZTE (HAREL) ZUSTANDSMASCHINEN

Motivation: Hierarchische Modellverfeinerung

- Elementare Zustände werden auf Teilzustände aufgeteilt
- Die in den neuen Zuständen verbrachte Gesamtzeit = die Zeit verbracht in dem alten Zustand

Auflösung:„eins zu eins” ersetzen

KOMPOSITIONALITÄT

Beispiel: Tee zubereiten

statt

„Filter ziehen lassen + Würzen”

wird

„Filter einlegen + Süßen + Rühren”

Statechart-Sprachen

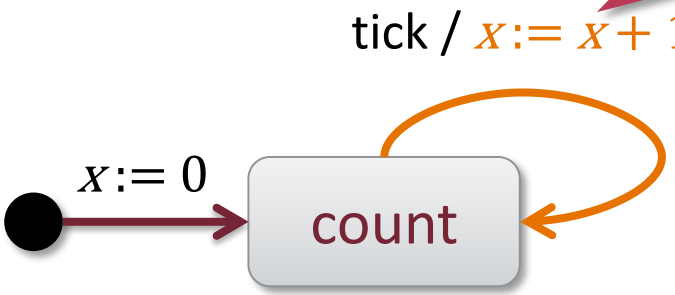
- Statechart: Zustandsmaschine +
  - Zustandshierarchie
  - Ortogonalität
  - Variablen
  - Pseudo-Zustände
  - …
- zum Beispiel
  - Yakindu (Hausaufgabe)
  - UML (Softwaretechnologie)

Variablen

- Unendlicher Zähler
  - $S = \mathbb{N}$

- Einführung einer Variable: x

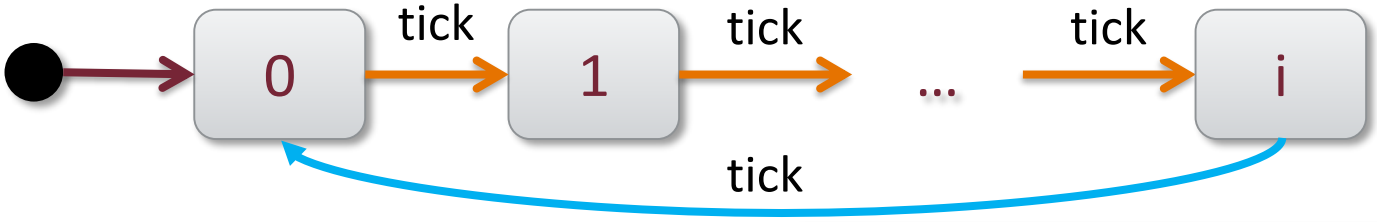
x bildet eigentlich eine andere Zustandsregion…



$$(count, \{x \mapsto 0\}) \rightarrow (count, \{x \mapsto 1\}) \rightarrow \ \ldots$$

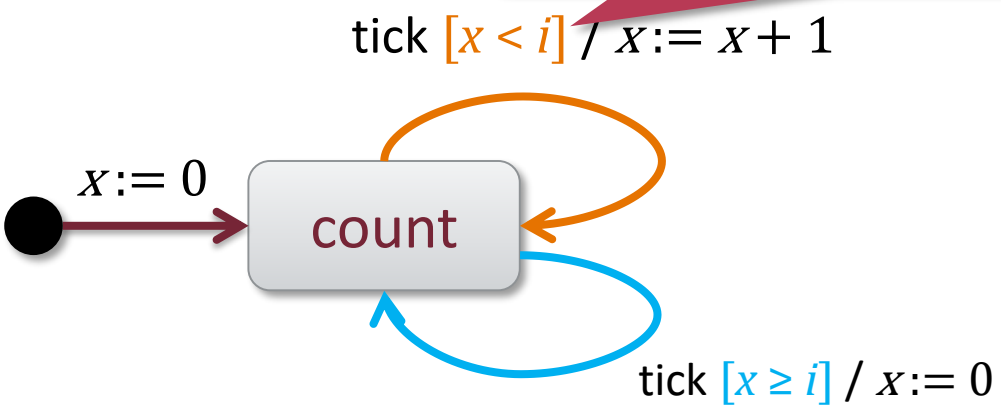
Variablen + Wächterkriterien

- Zyklischer Zähler
  - $S = \{0,1, \dots, i\}$



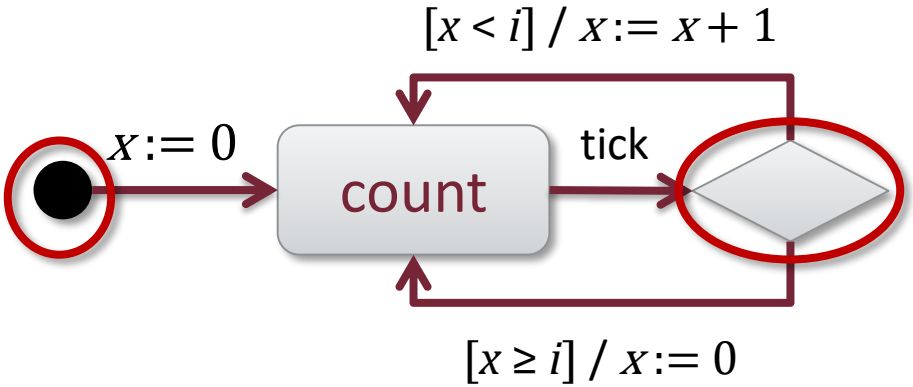
- mit Wächterkriterien :

Wert einer Variable oder aktueller Zustand einer Region



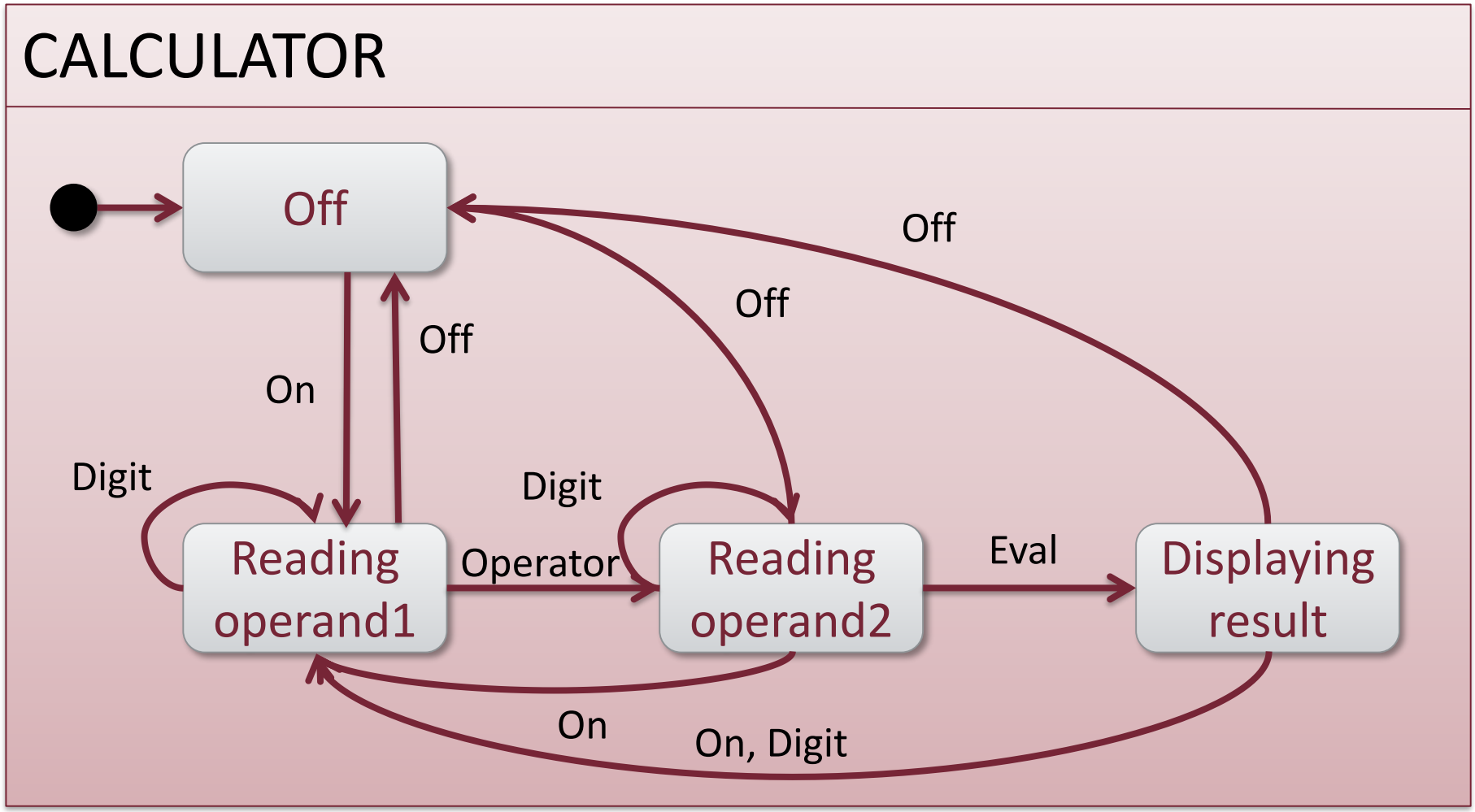
Pseudozustände

- Pseudozustand:
  - semantisch gilt als kein Zustand:
    - gibt es keinen Zeitpunkt, wann das System ihn aufnimmt
  - syntaktisch gilt als Zustand:
    - kann Anfangs- oder Zielzustand eines Überganges sein

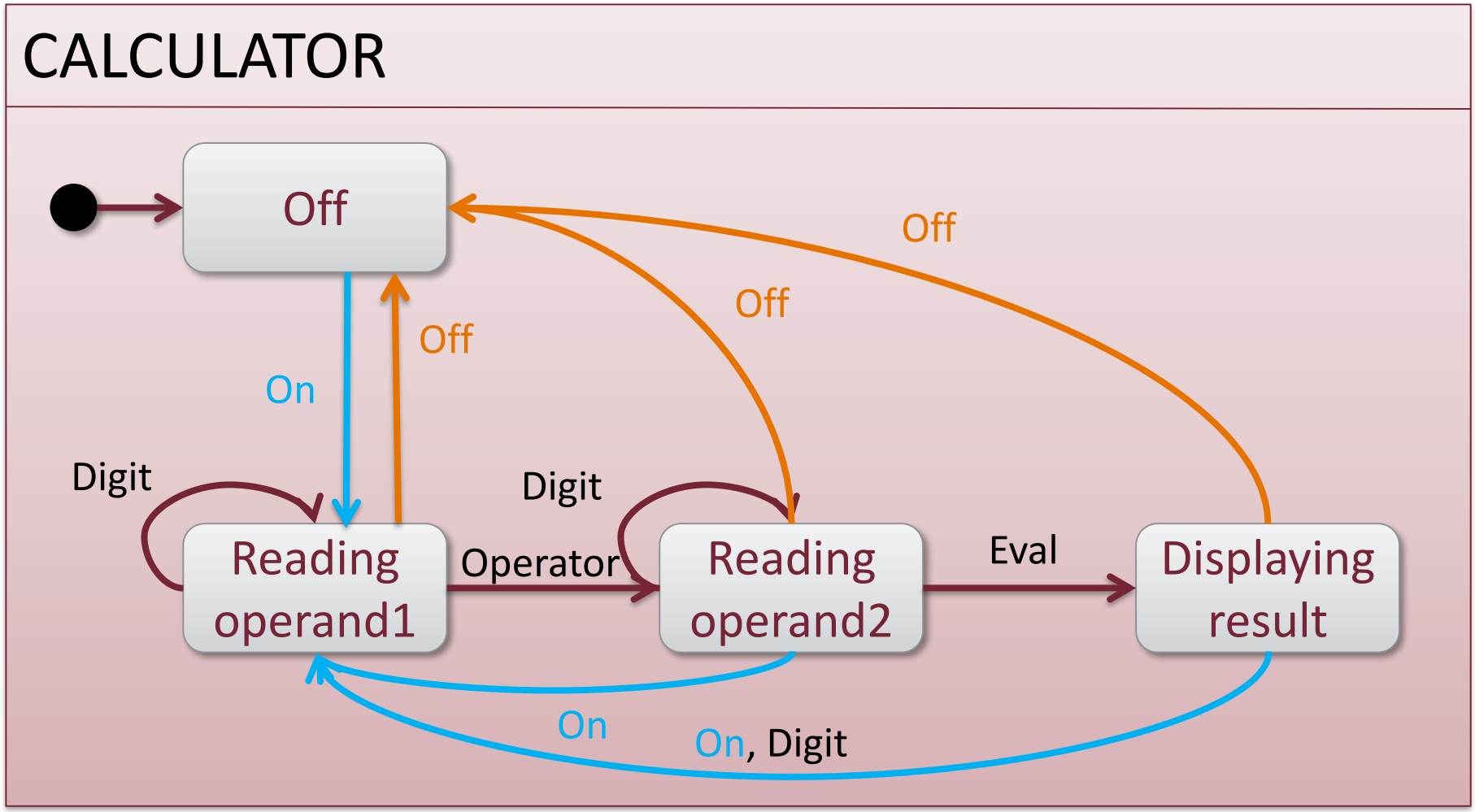


Statechart-Beispiel: Taschenrechner

Statechart-Beispiel: Zustandshierarchie

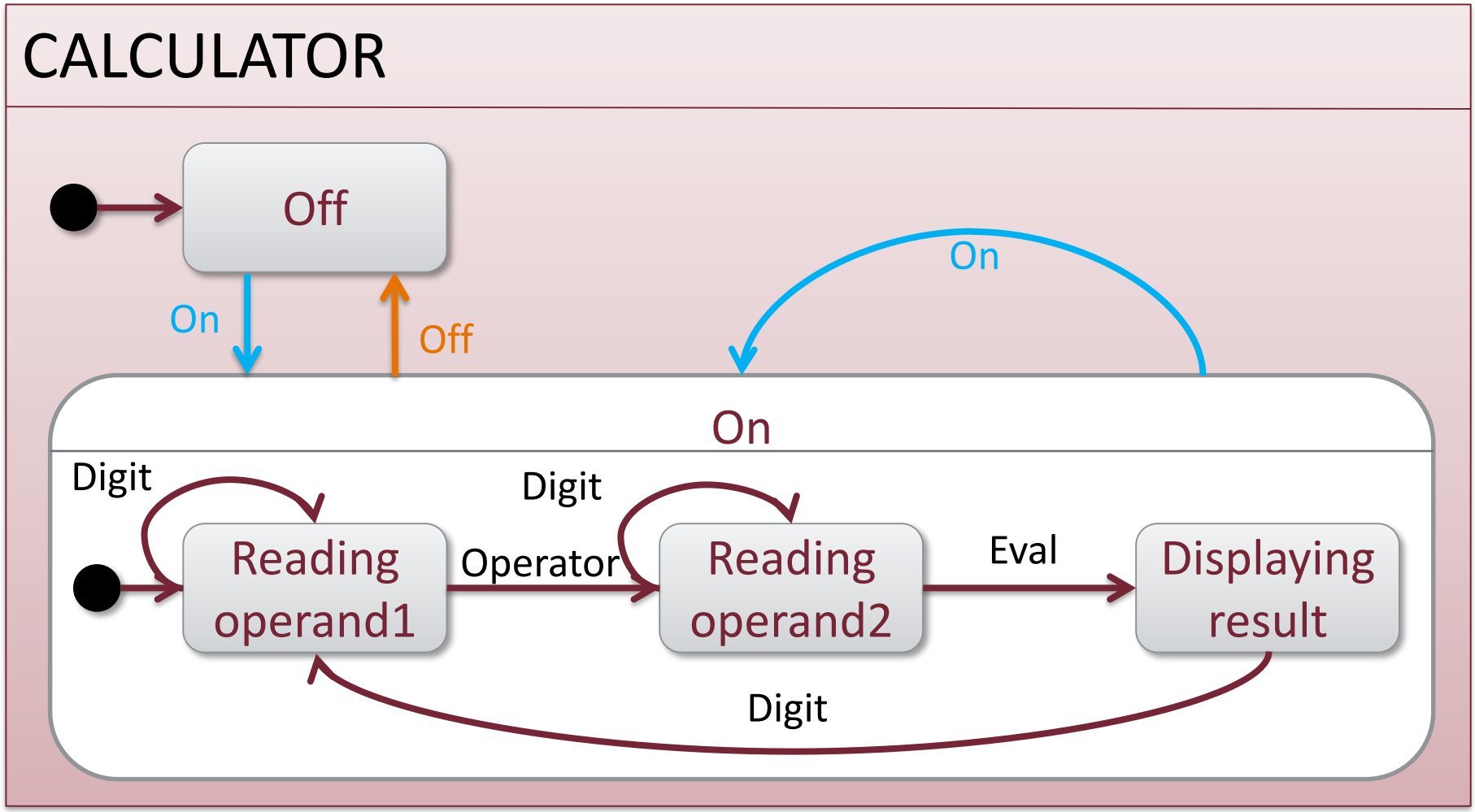


Statechart-Beispiel: Zustandshierarchie



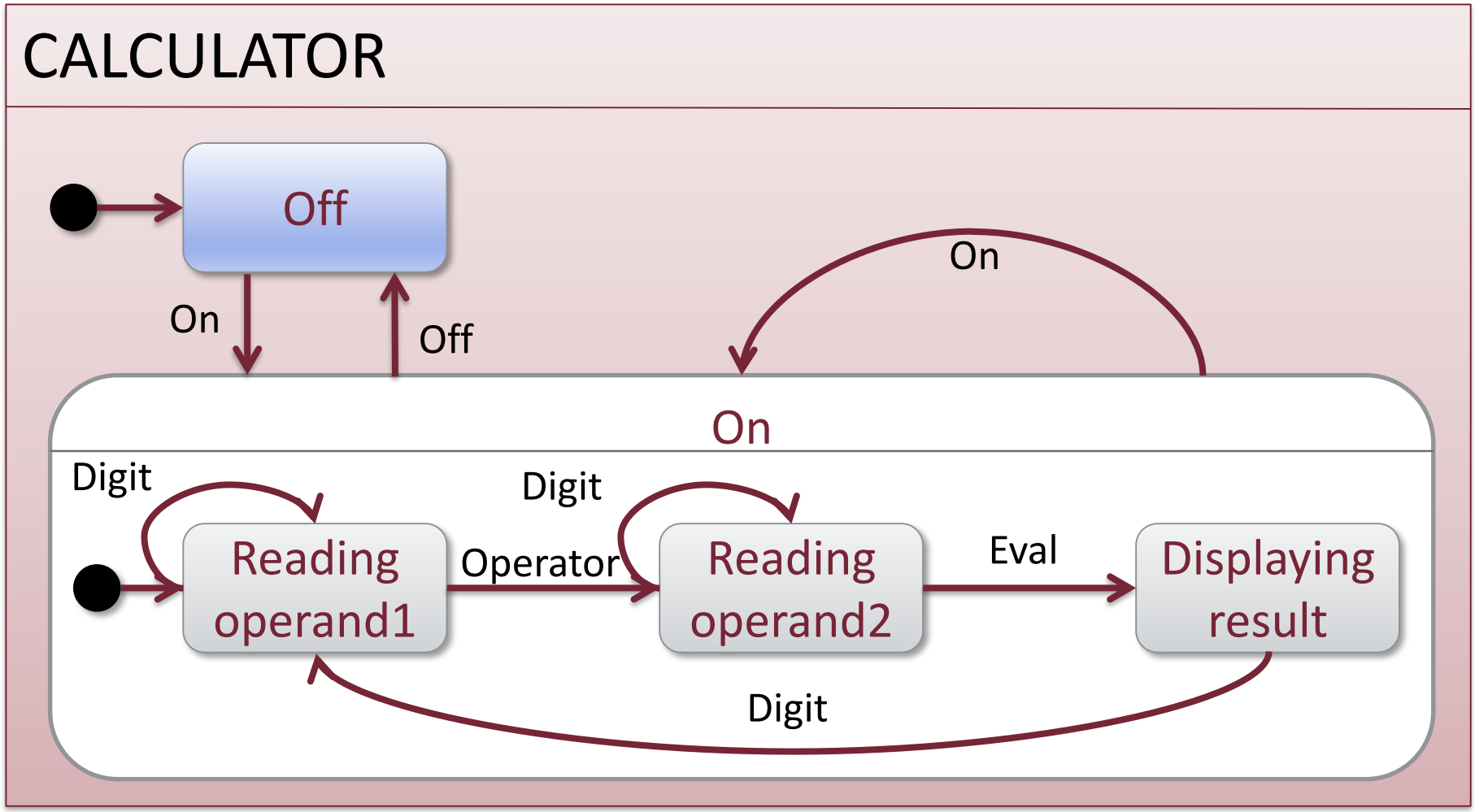
- Eingaben: {*on, off, digit, operand, eval*}
- Annahme: Die Operationen haben immer genau zwei Operanden

Statechart-Beispiel: Zustandshierarchie



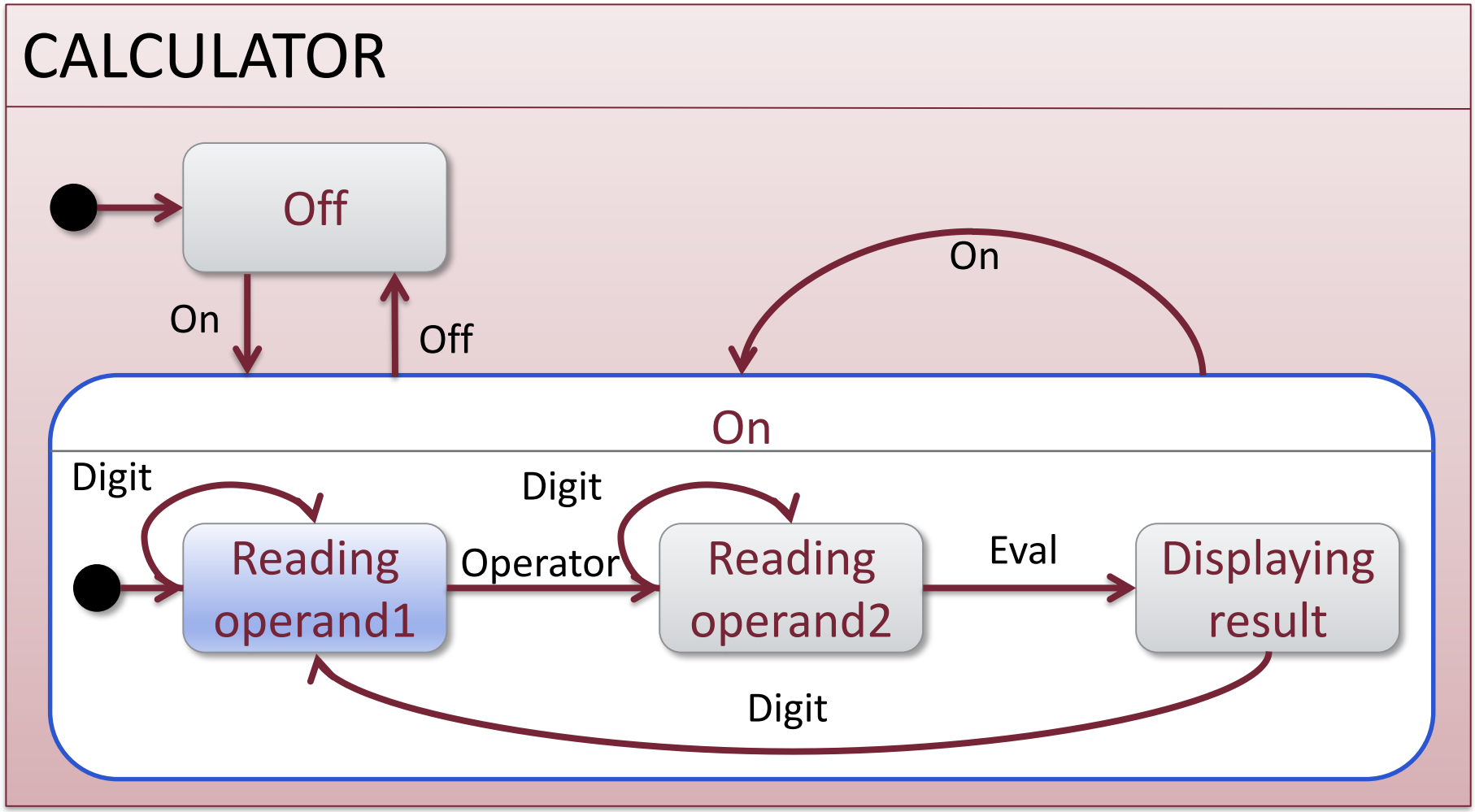
- Das gemeinsame Verhalten wird abgetrennt

Statechart-Beispiel: Zustandshierarchie



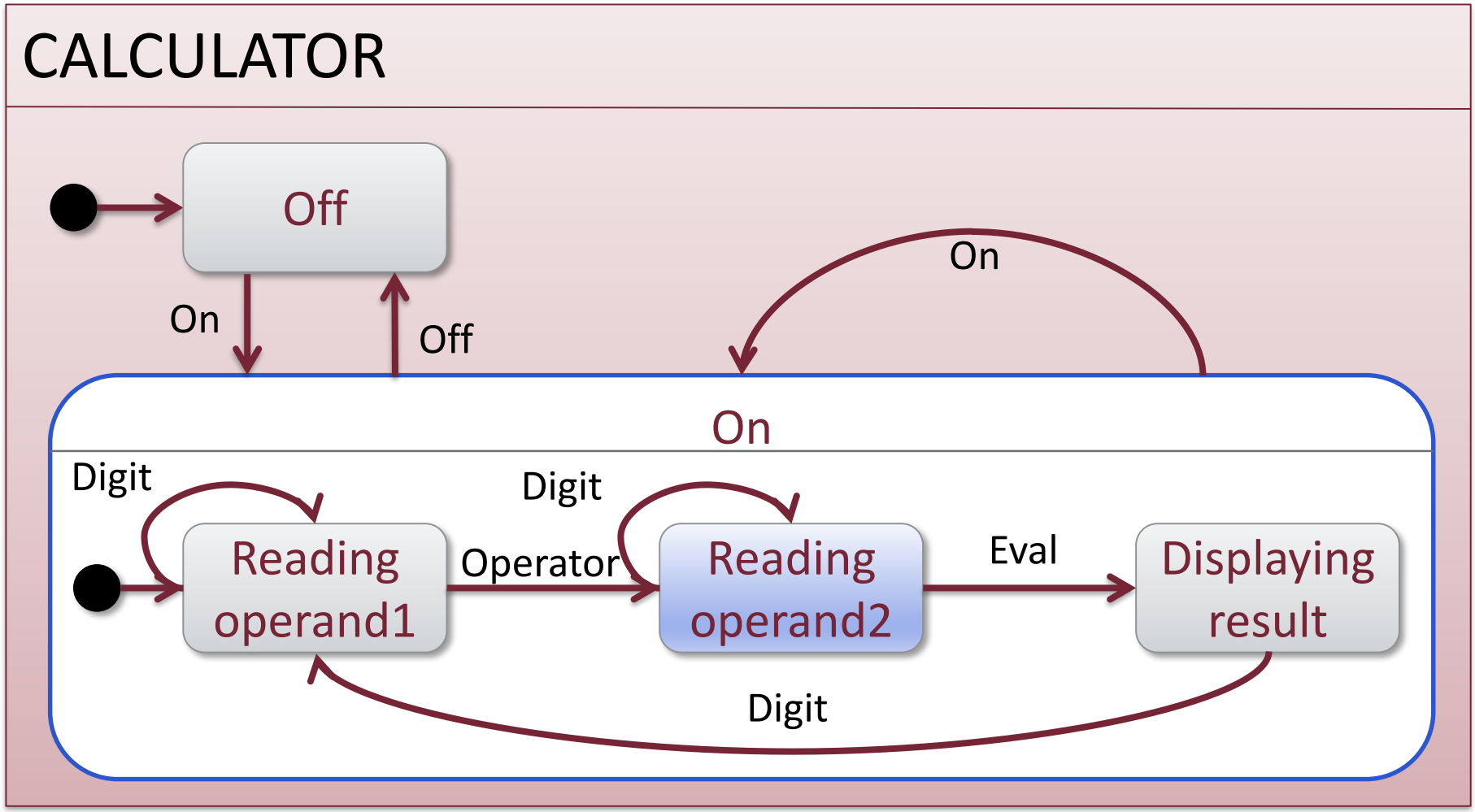
- Zustandskonfiguration: $\{Off\}$

Statechart-Beispiel: Zustandshierarchie



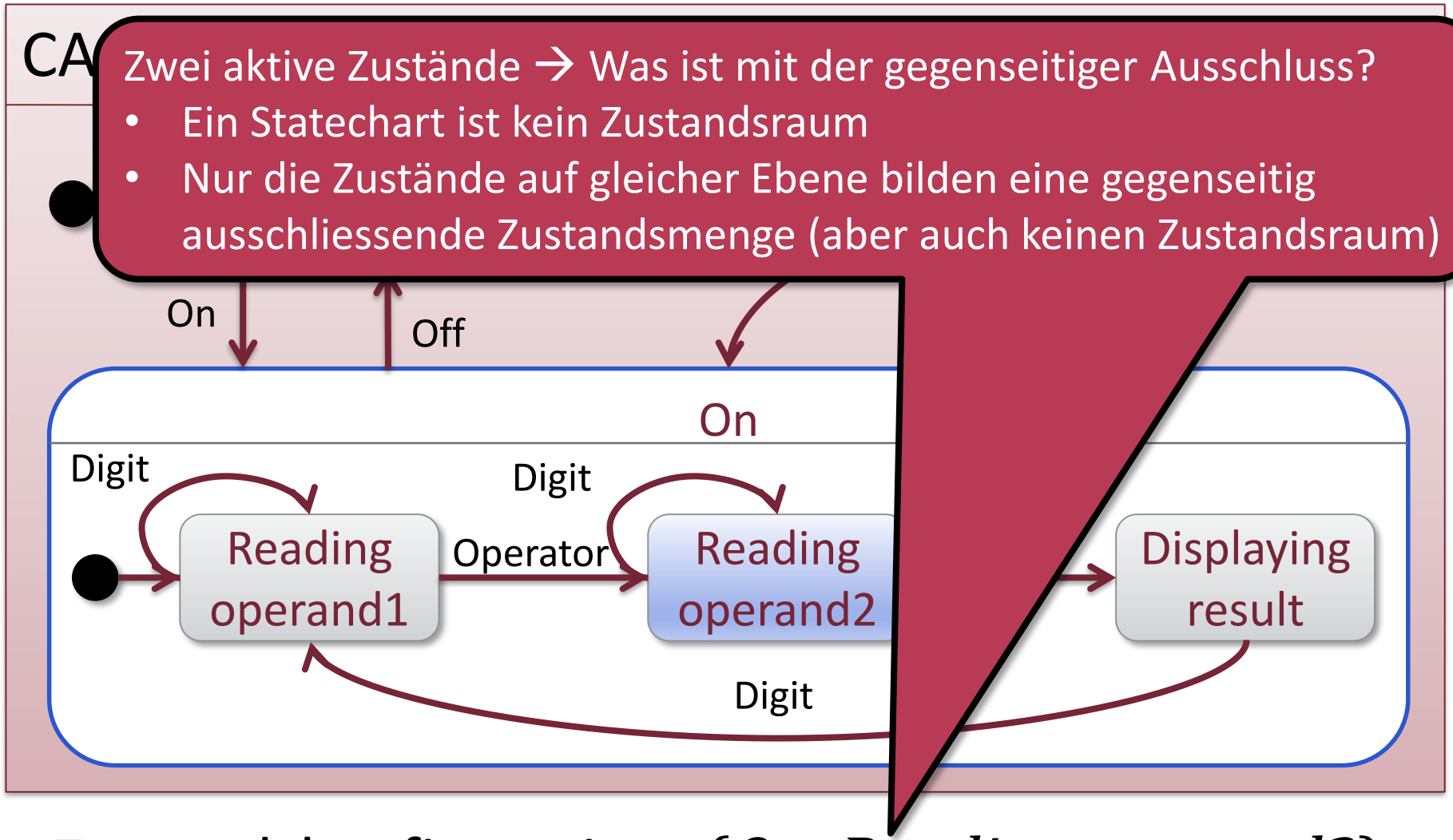
- Zustandskonfiguration: $\{On, Reading\ operand1\}$

Statechart-Beispiel: Zustandshierarchie



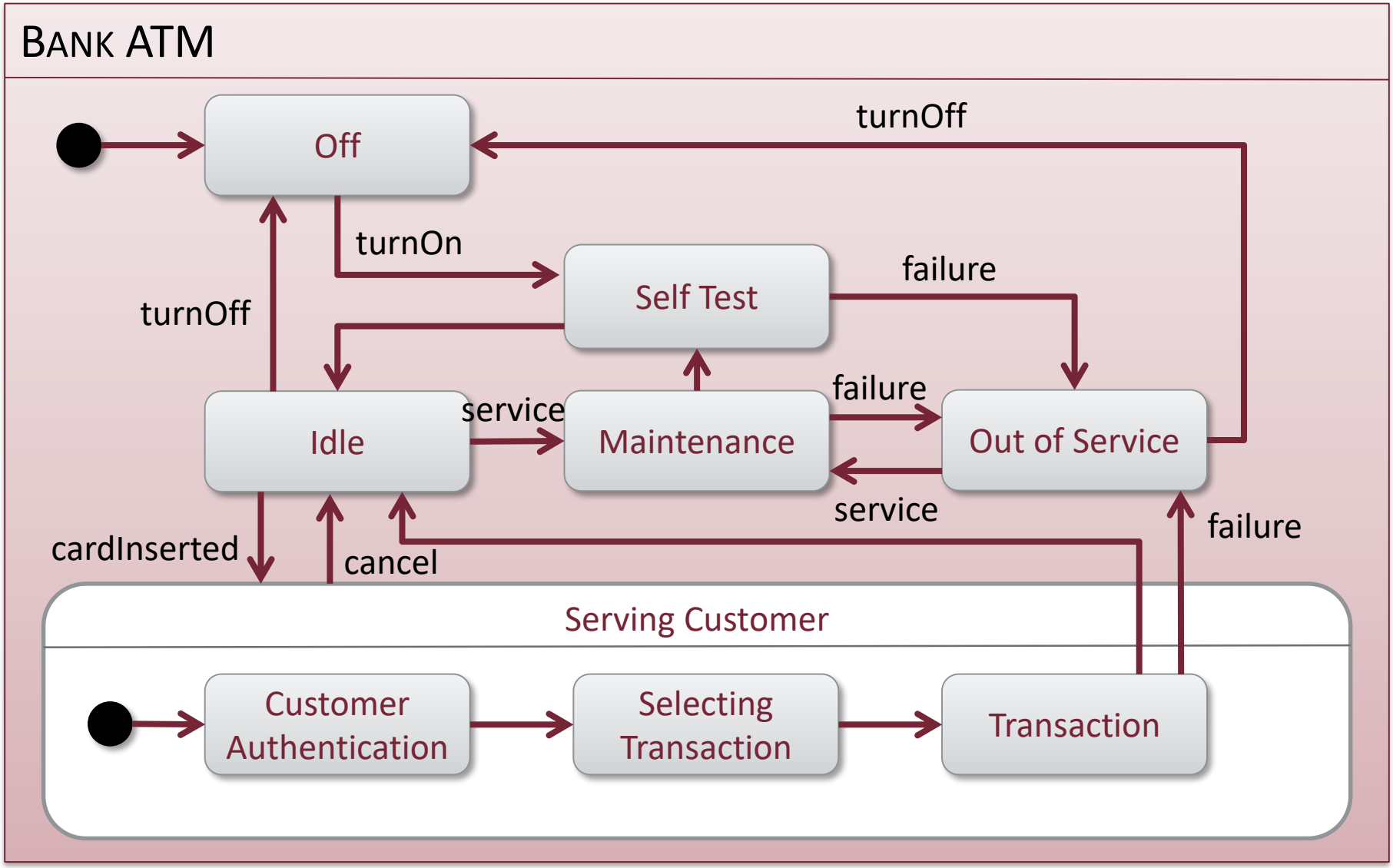
- Zustandskonfiguration: $\{On, Reading\ operand2\}$

Statechart-Beispiel: Zustandshierarchie



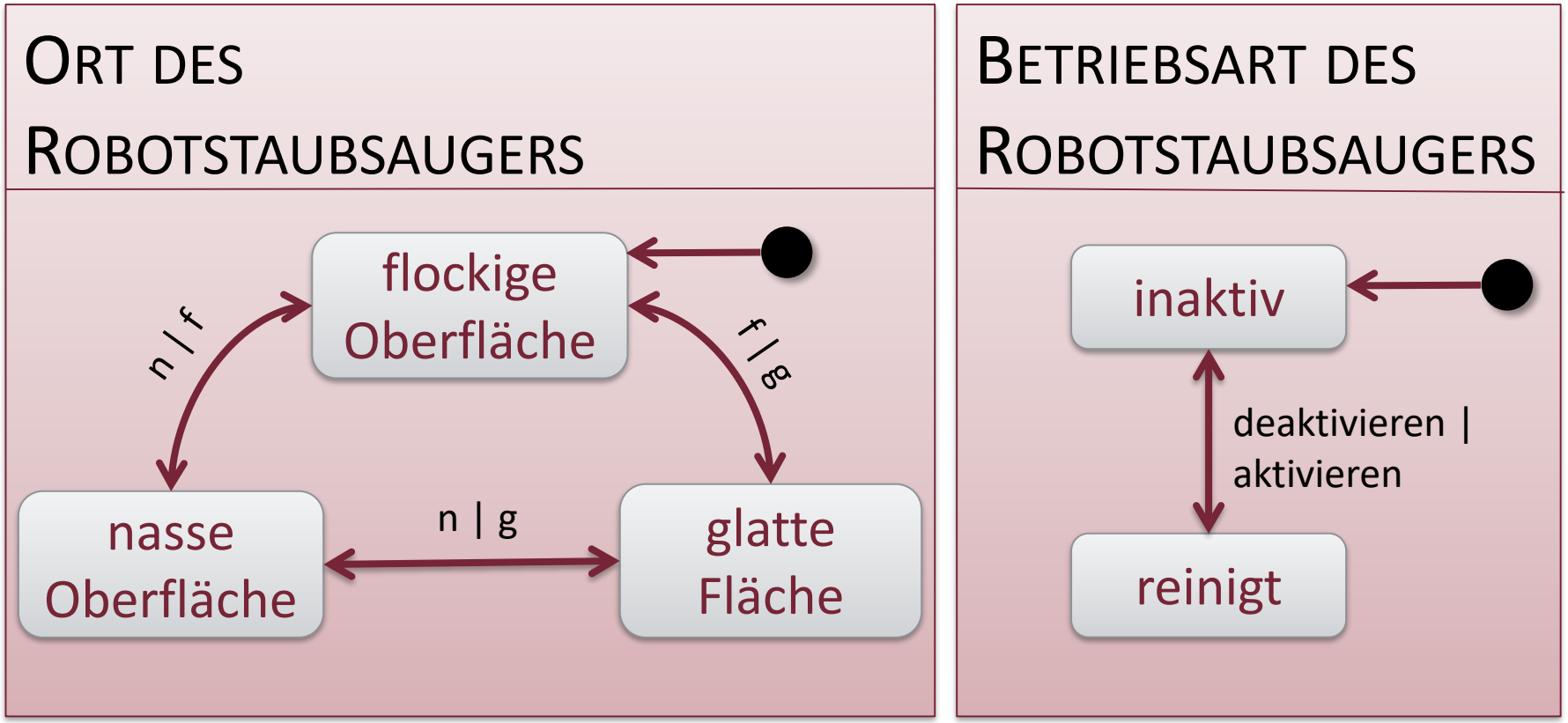
- Zustandskonfiguration: $\{On, Reading\ operand2\}$

Statechart-Beispiel 2: Geldautomat



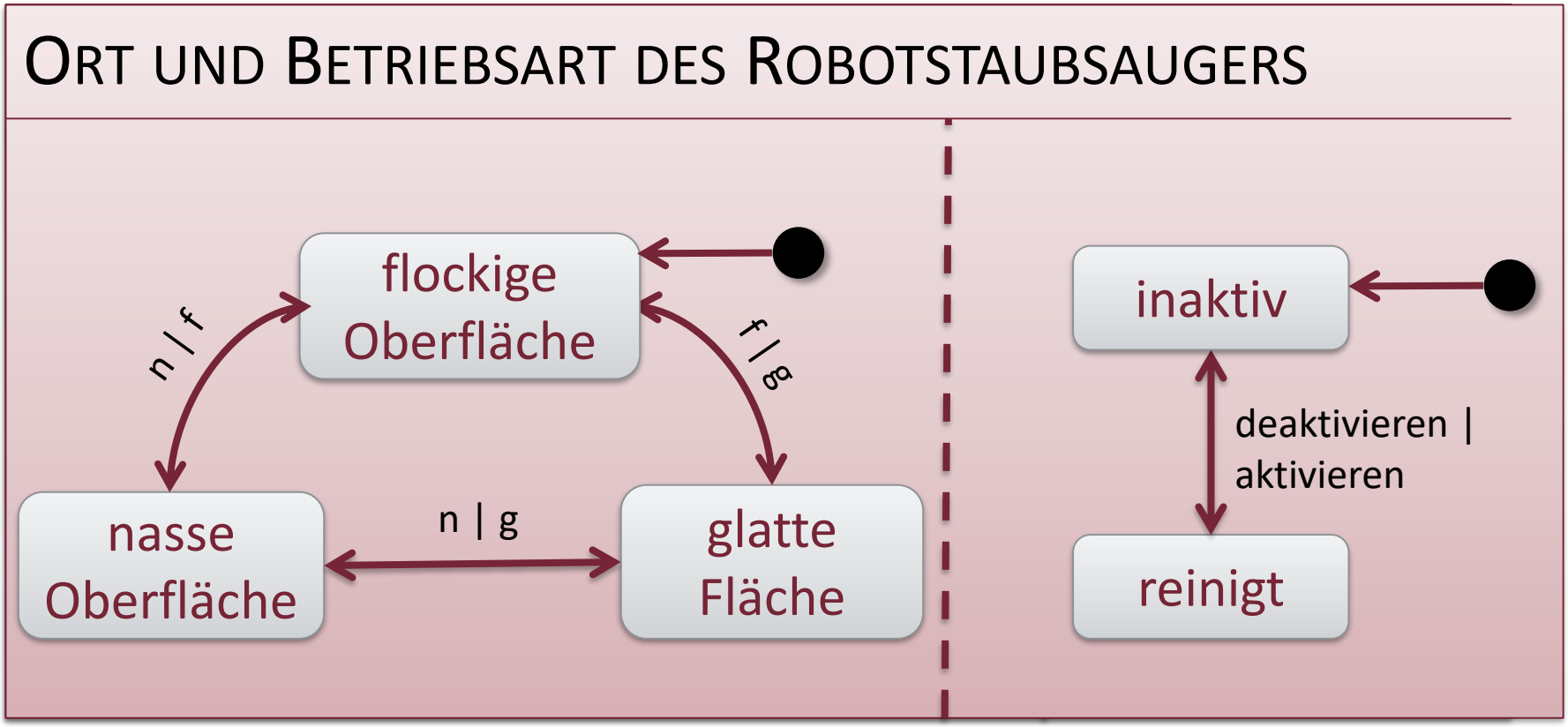
Statechart-Beispiel 3: Robotstaubsauger

Ortogonalität

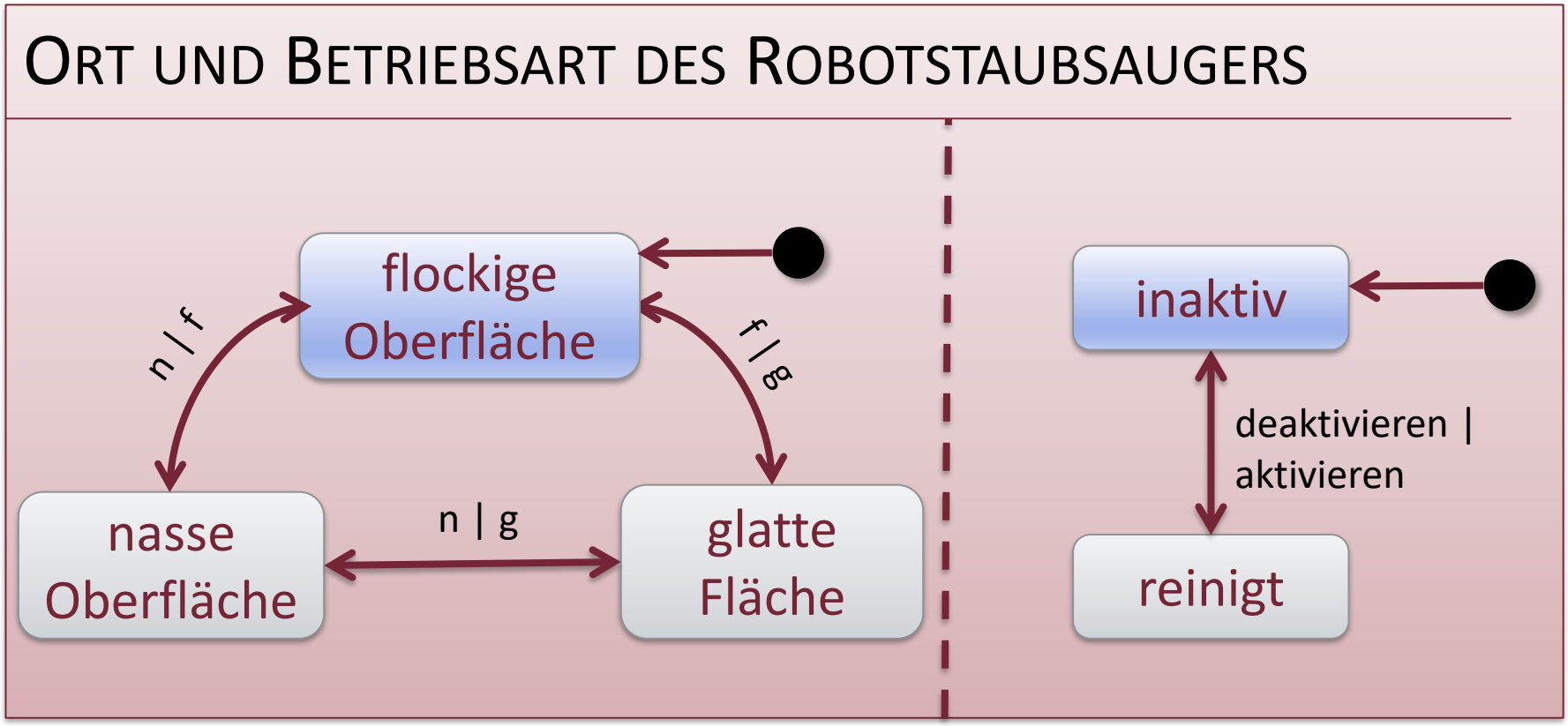


Anmerkung: beidseitiger Pfeile sind nicht üblich, auch hier nur als Abkürzung gedacht …

Ortogonalität

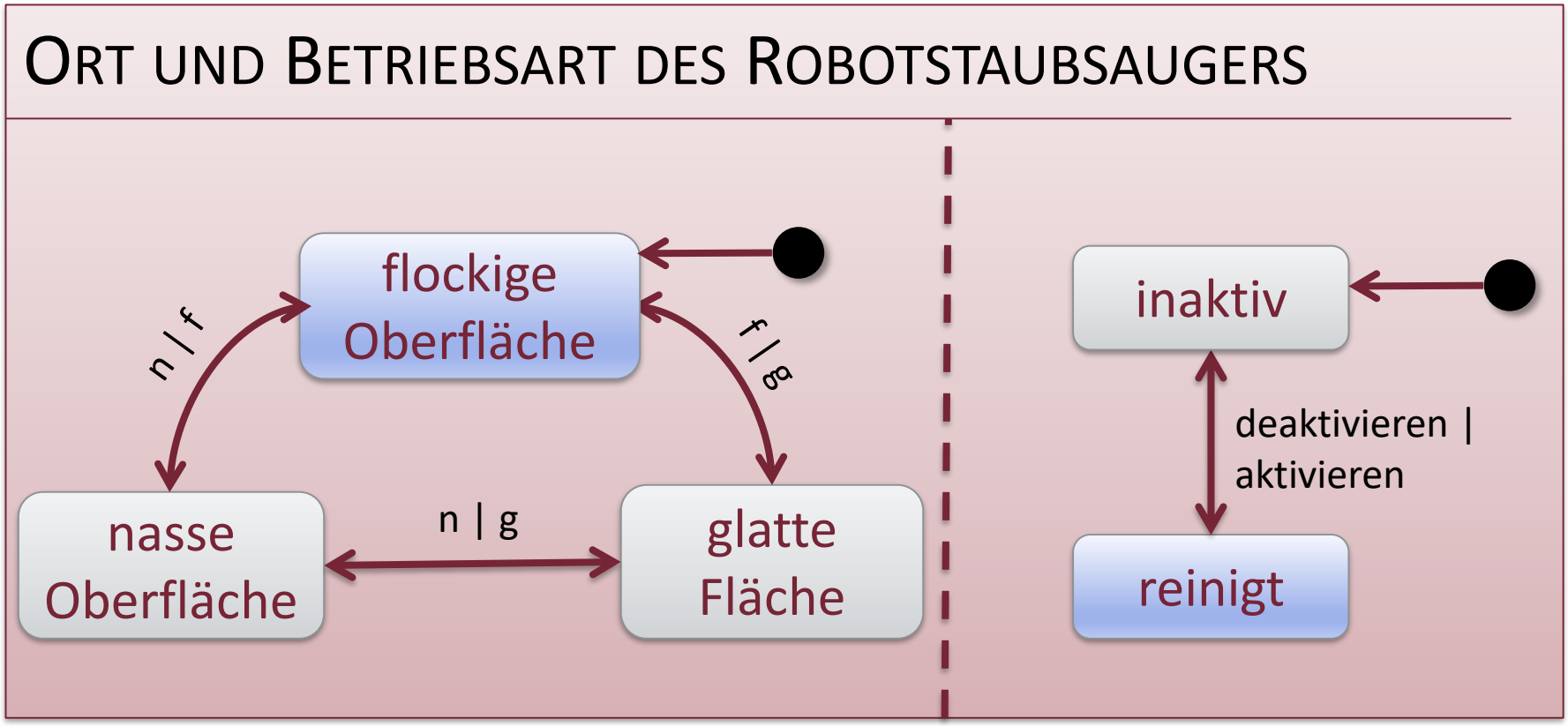


Ortogonalität



- Zustandskonfiguration: {*flockige Oberfl.*, *inaktiv*}

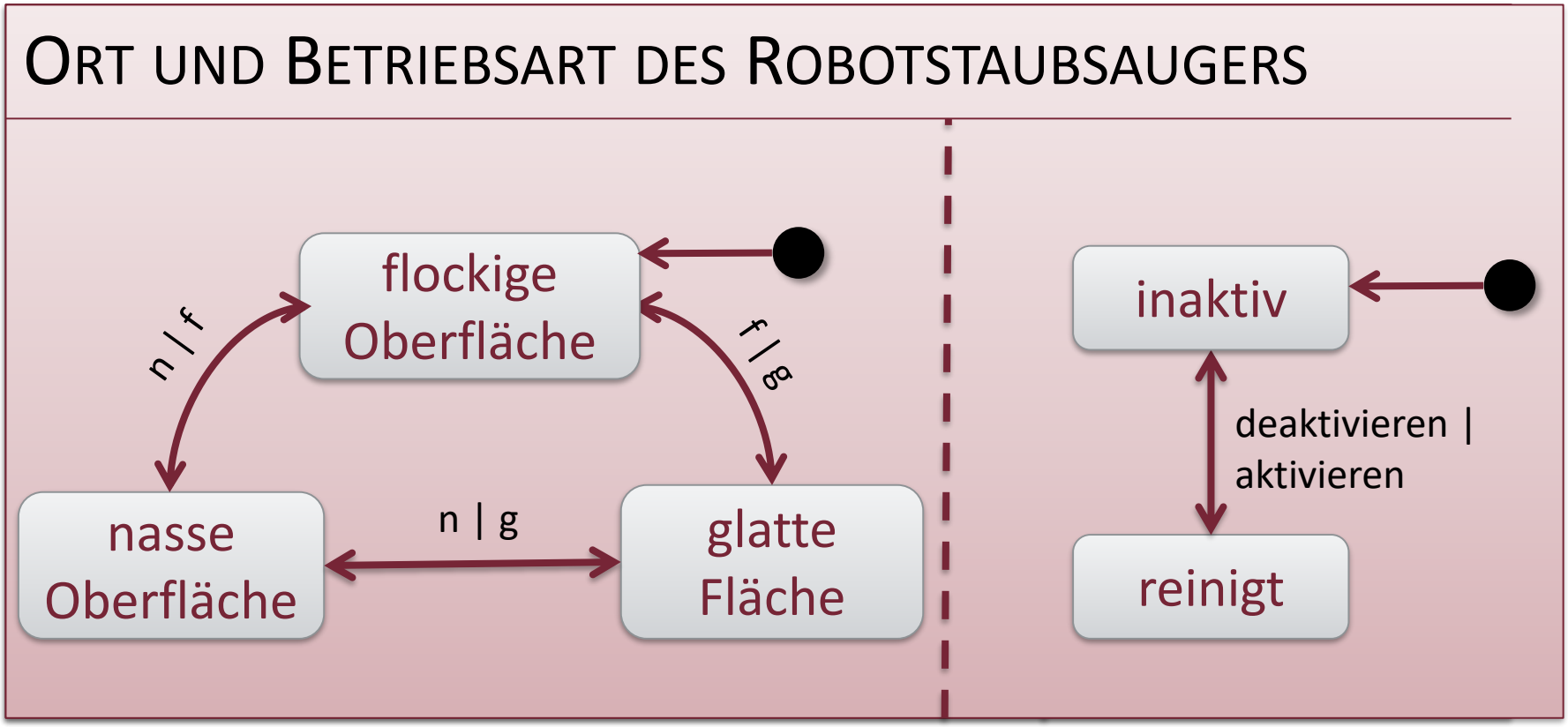
Ortogonalität



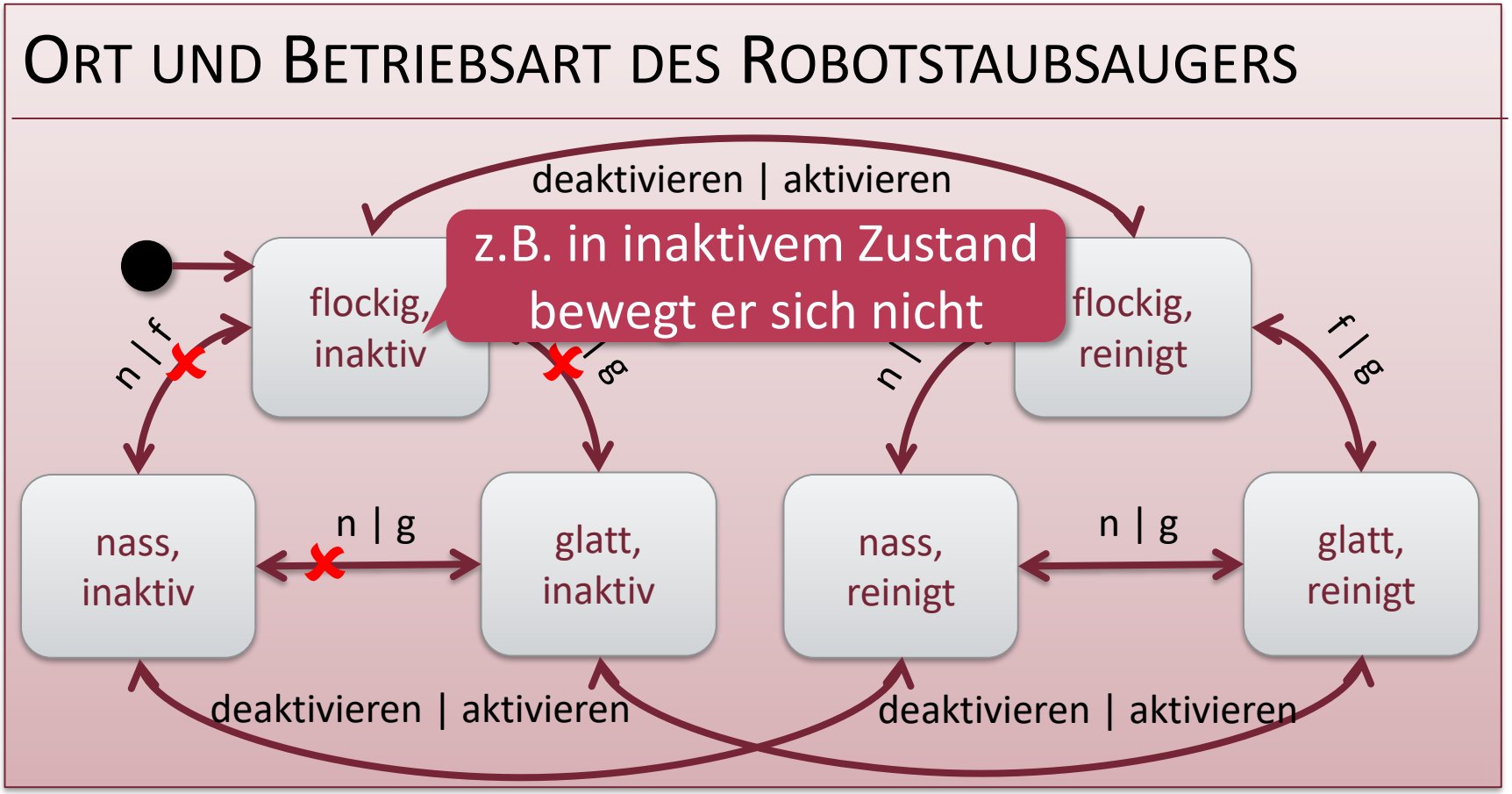
- Zustandskonfiguration: {*flockige Oberfl.*, *reinigt*}

KOOPERATION: PRODUKTE

Asynchrones Produkt der Regionen



Asynchrones Produkt der Regionen

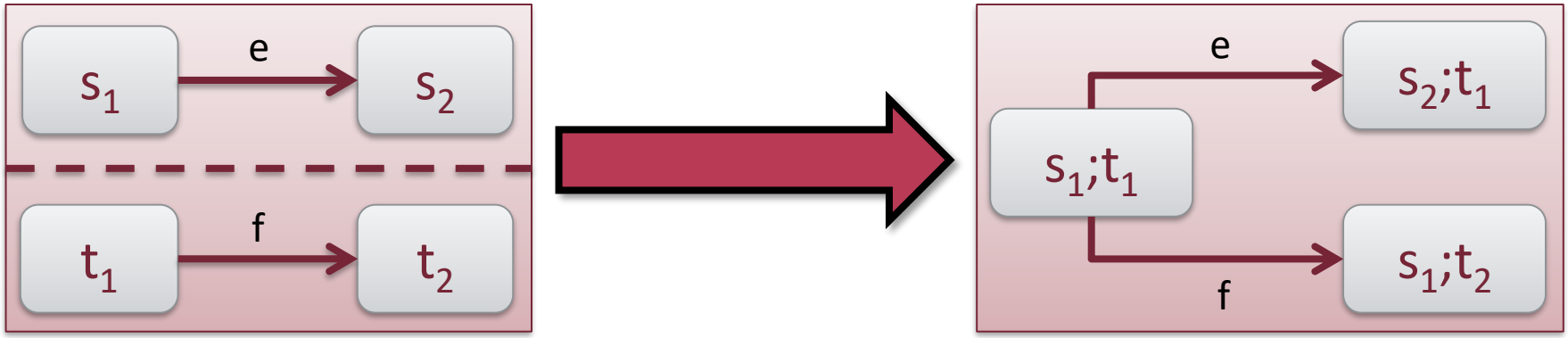


- Verfeinerung notwendig: Überg. können wegfallen
  → Zustände können auch unerreichbar werden

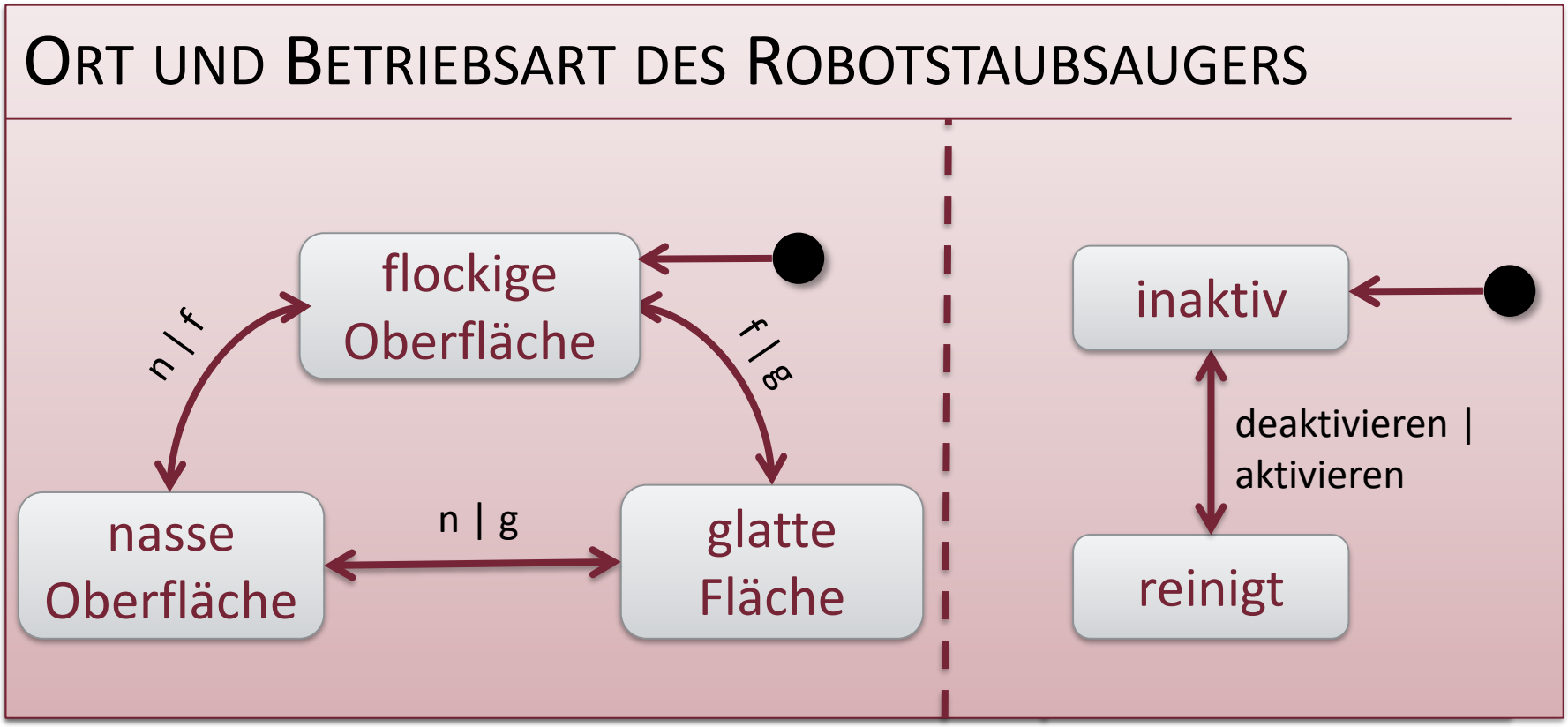
Definition: Asynchrones Produkt

Asynchrones Produkt der (Mealy-) Zustandsmaschinen ist eine auf den Komponenten-Zustandsmaschinen (auch Regionen genannt) durchgeführte **Kompositionsoperation**. Das **Ergebnis** der Komposition ist eine (Mealy-) Zustandsmaschine,

- deren Zustandsraum das direkte Produkt der Zust.räume der Regionen ist,
- in derem Anfangszustand jede Region in ihrem Anfangszustand ist,
- und deren Übergangsregel alle Übergänge bilden, in denen **genau eine Region** einen Übergang macht, weil alle anderen ihren aktuellen Zustand erhalten.

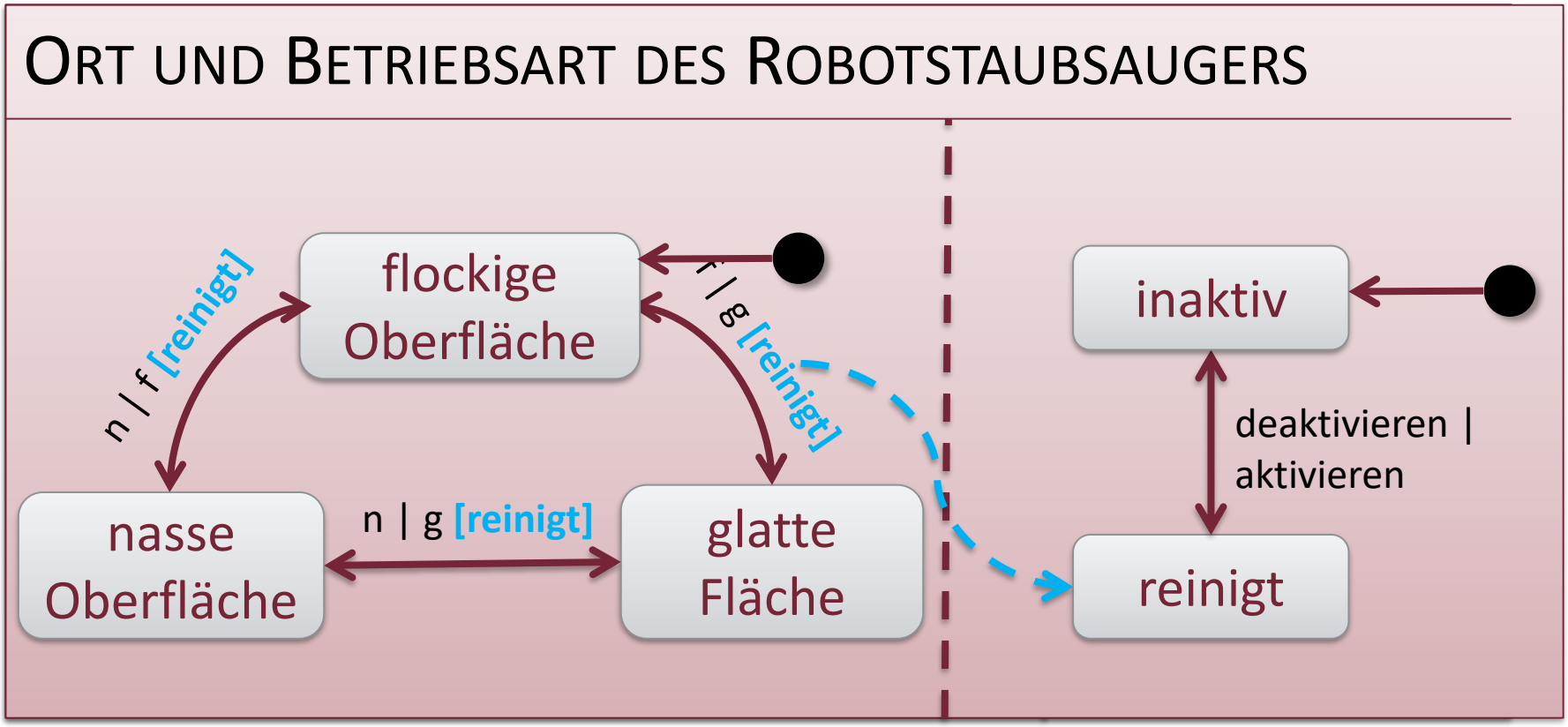


Kooperation bei einem Asynchronen Produkt



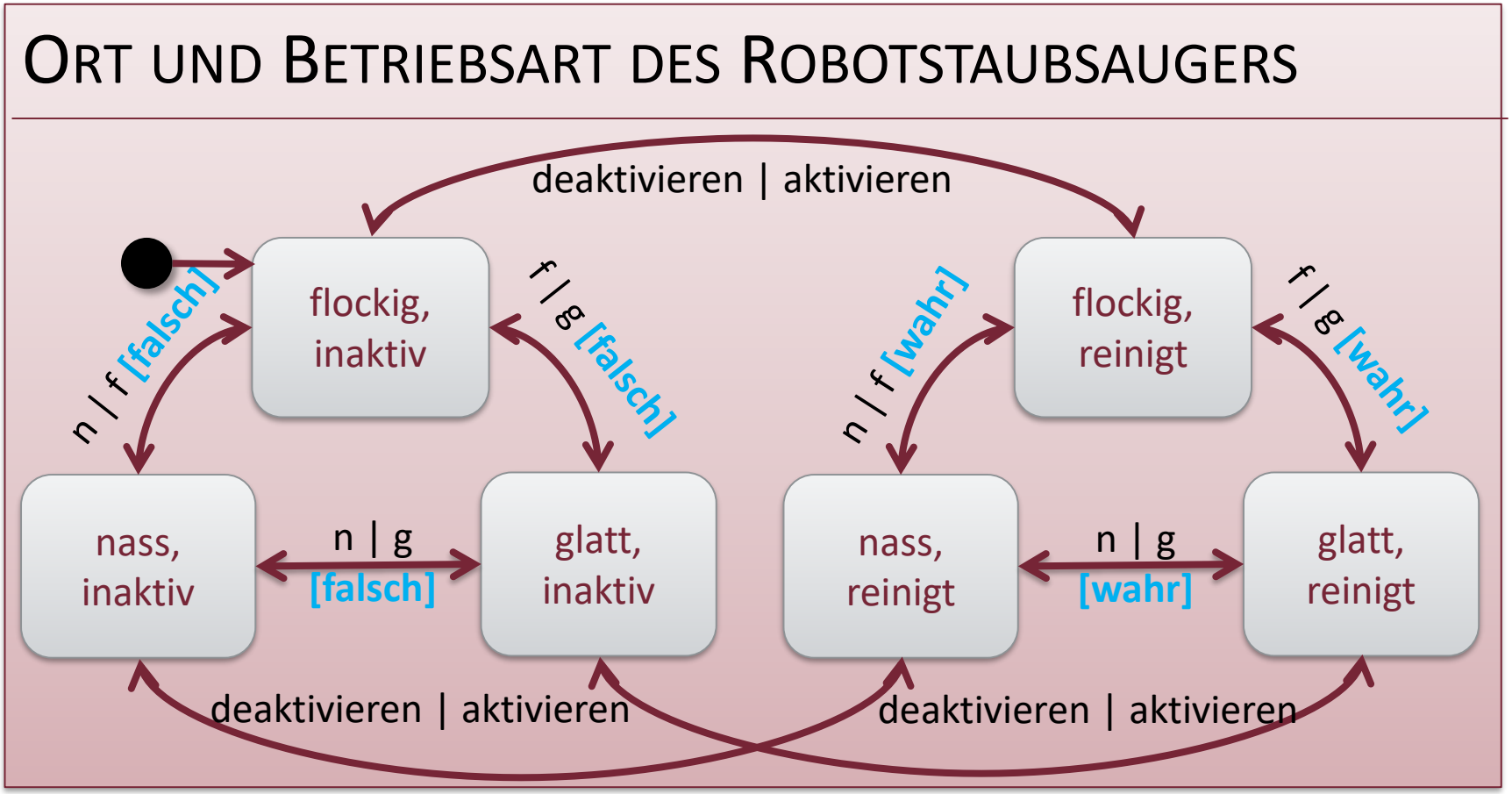
- Wie wird Kooperation modelliert?
  - Genau wie sind die zwei Regionen **nicht unabhängig**?

Kooperation bei einem Asynchronen Produkt



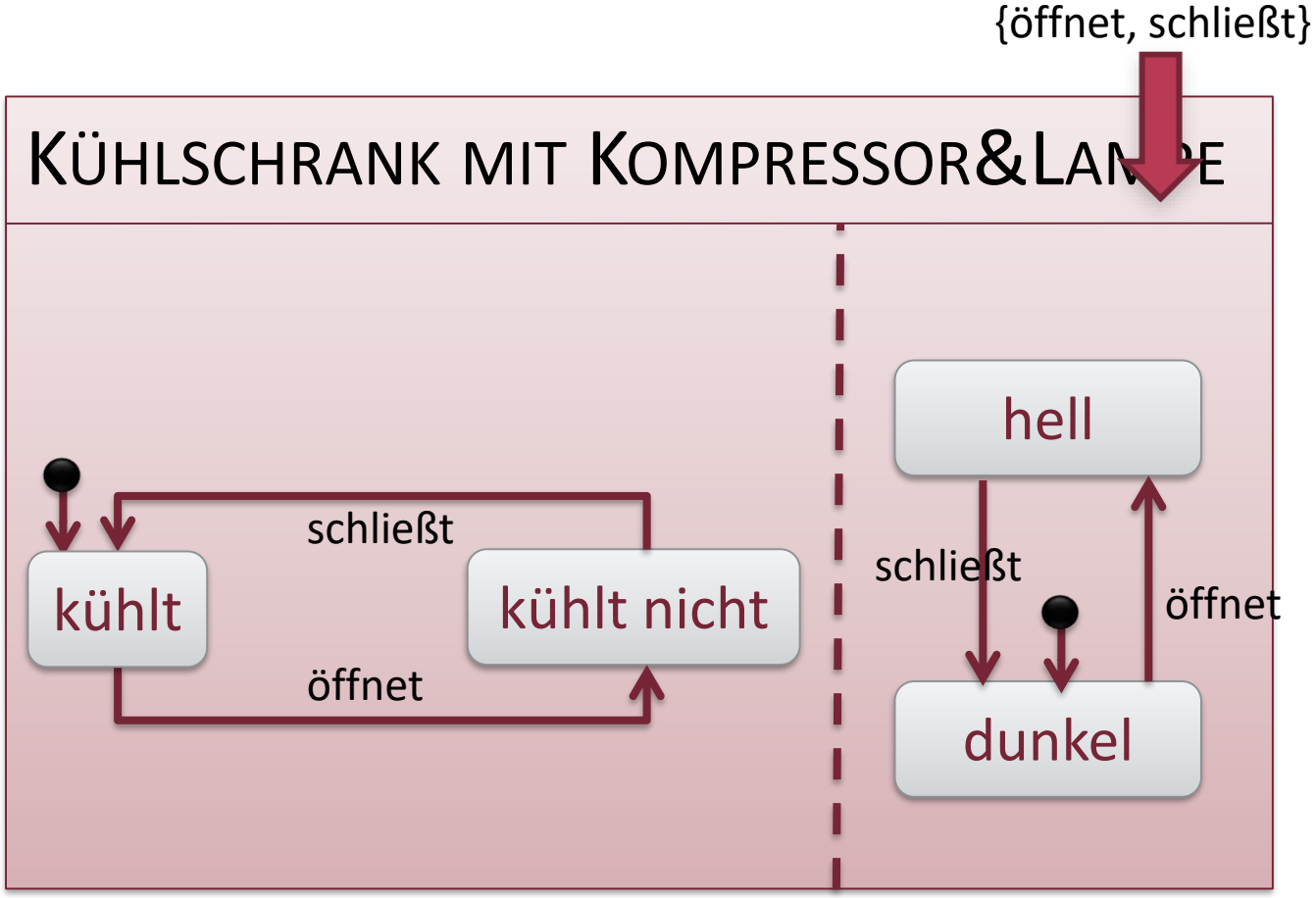
- Kooperation durch **Wächterkriterien**
  - Das Bestehen eines Zustandes in der einen Region ist ein Kriterium für einen Übergang in der anderen Region

Kooperation bei einem Asynchronen Produkt



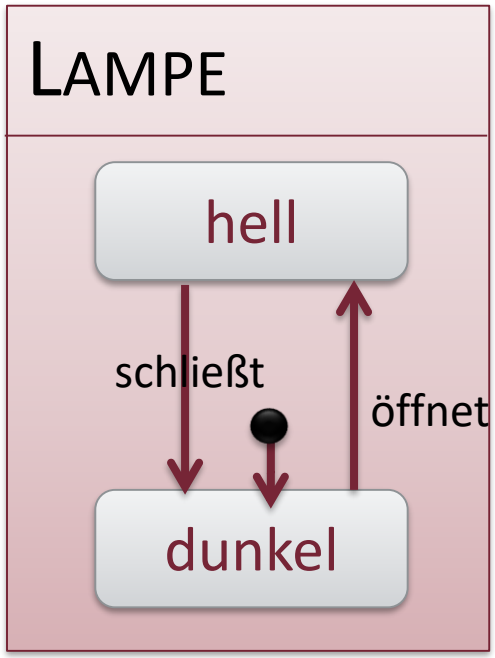
Beispiel: Synchrones Produkt

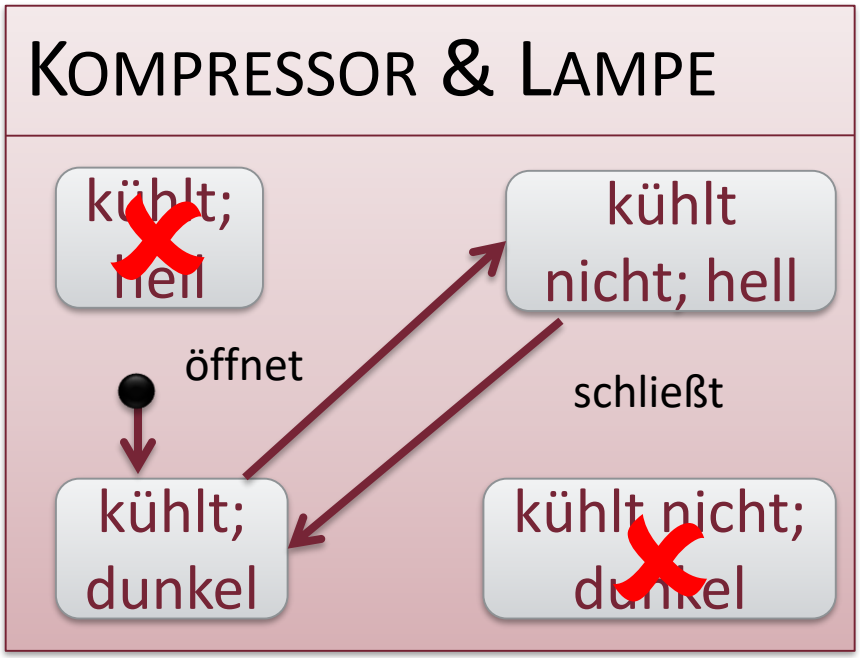
- Kühlschrank mit Kompressor&Lampe
  - Gemeinsame Eingabe: die Tür



Beispiel: Synchrones Produkt

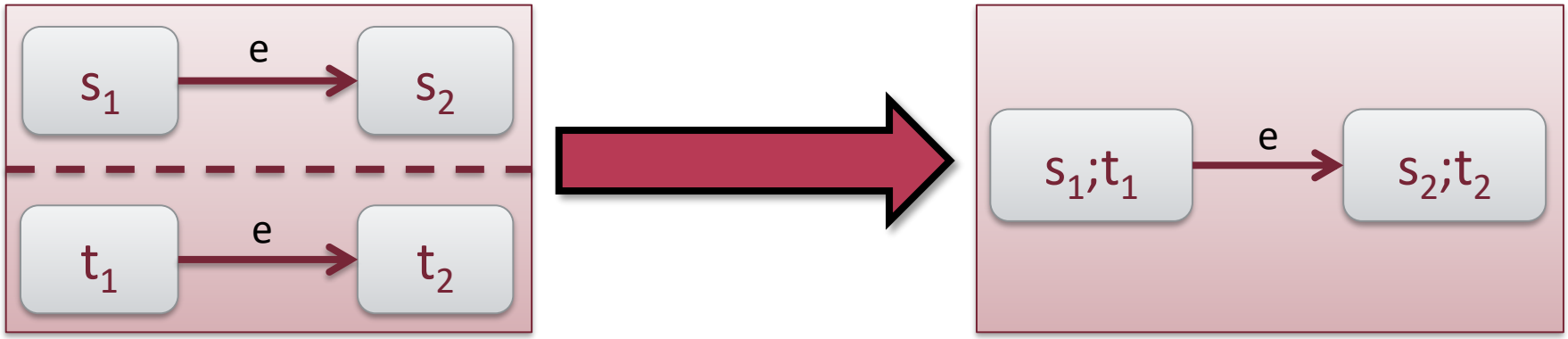
- Gemeinsames Lesen der Eingaben
  → synchrone Schritte
  - Zustände können unerreichbar werden





Synchrones Produkt

- Produkt der Zustandsmaschinen
  - Aufbau des Modells aus den Regionen (Komponenten)
  - Zustandsmenge: **direktes Produkt** der Zustandsräume
  - Anfangszustand: n-Tupel der Anfangszustände der Regionen
- Übergänge des **synchronen Produktes**: beide Zustandsvariablen ändern sich gleichzeitig



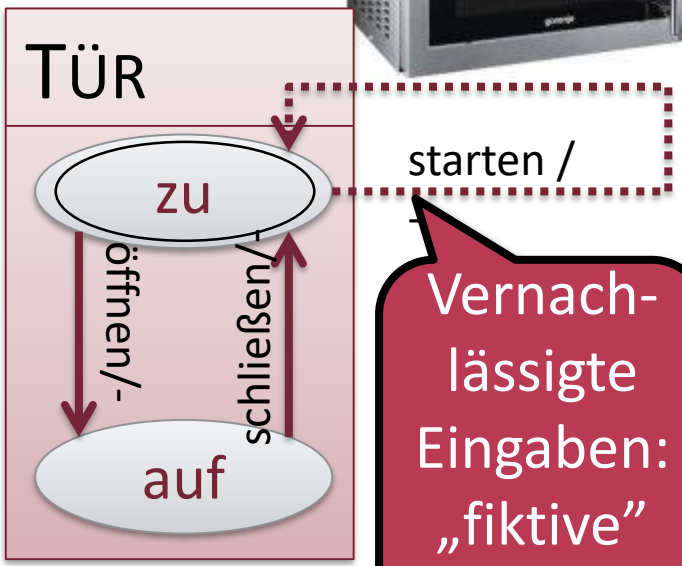
Gemischtes Produkt

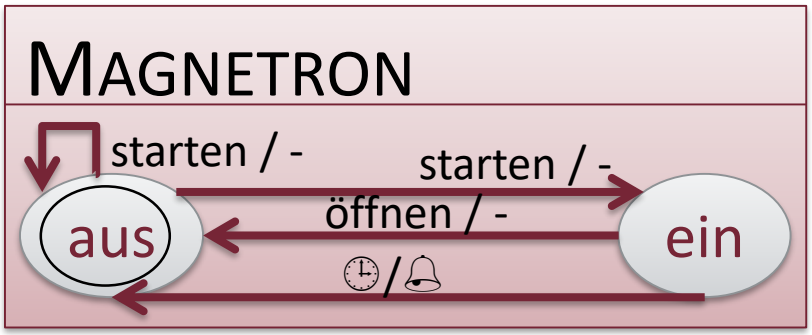
- Produkt der Zustandsmaschinen
  - Aufbau des Modells aus den Regionen (Komponenten)
  - Zustandsmenge: **direktes Produkt** der Zustandsräume der Regionen
  - Anfangszustand: n-Tupel der Anfangszustände der Regionen
- Übergänge des **synchronen Produktes**:
  - Grundsätzlich asynchrone Komposition …
  - …aber die Regionen können auch synchronisiert wechseln

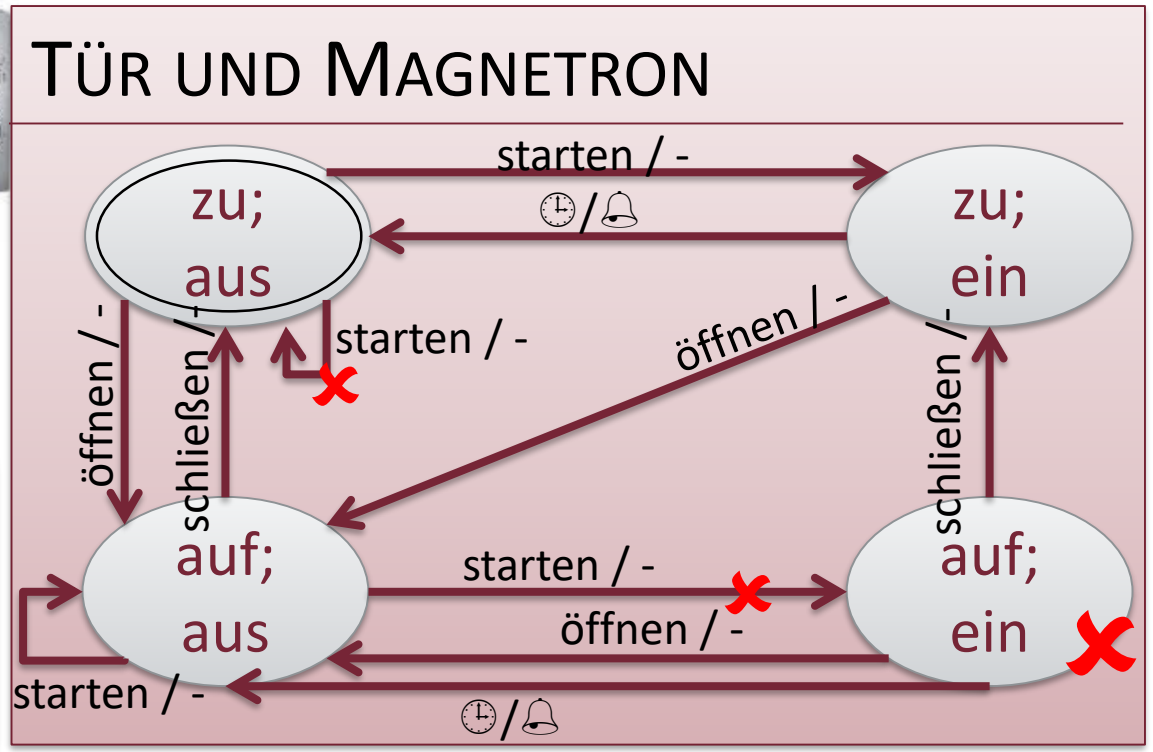
Die einfachste Synchronisation: (auch) gemeinsame Eingänge

Beispiel: Gemischtes Produkt

- Verfeinerung notwendig
  - Überg. können wegfallen
  - Zustände können auch unerreichbar werden







Gemischtes Produkt

- Produkt der Zustandsmaschinen
  - Aufbau des Modells aus den Regionen (Komponenten)
  - Zustandsmenge: **direktes Produkt** der Zustandsräume der Regionen
  - Anfangszustand: n-Tupel der Anfangszustände der Regionen
- Übergänge des **synchronen Produktes**:
  - Grundsätzlich asynchrone Komposition …
  - …aber die Regionen können auch synchronisiert wechseln

Die einfachste Synchronisation: (auch) gemeinsame Eingänge

Fortgeschrittene Kooperation: Rendezvous (internes synchr. Ereignis)

Übersicht der Produkte

- Produkt von Zustandsräumen
  - **Direktes Produkt**: $S_1 \times S_2 \times \ldots \times S_n$
    - zusammengesetzter Zustand: n-Tupel der Komponentenzust.
- Produkt von Zustandsmaschinen
  - Zust.raum immer das direkte Produkt (od. Verfeinerung)
  - **Synchrones Produkt**
    - Immer gleichzeitiger Zust.wechsel der Komponente/Regionen
  - **Asynchrones Produkt**
    - Immer getrennter Zustandswechsel der Komponente/Regionen
  - **Gemischtes Produkt**
    - Manchmal gleichzeitiger, manchmal getrennter Zust.wechsel

Übersicht der Produkte

- Produkt von Zustandsräumen
  - **Direktes Produkt**: $S_1 \times S_2 \times \ldots \times S_n$
    - zusammengesetzter Zustand: n-Tupel der Komponentenzust.
- Produkt von Zustandsmaschinen
  - Zust.raum immer das direkte Produkt (od. Verfeinerung)
  - **Synchrones Produkt**
    - Immer gleichzeitiger Zust.wechsel der Komponente/Regionen
  - **Asynchrones Produkt**
    - Immer getrennter Zustandswechsel der Komponente/Regionen
  - **Gemischtes Produkt**
    - Manchmal gleichzeitiger, manchmal getrennter Zust.wechsel

Art der Kooperation

Wächterkriterien

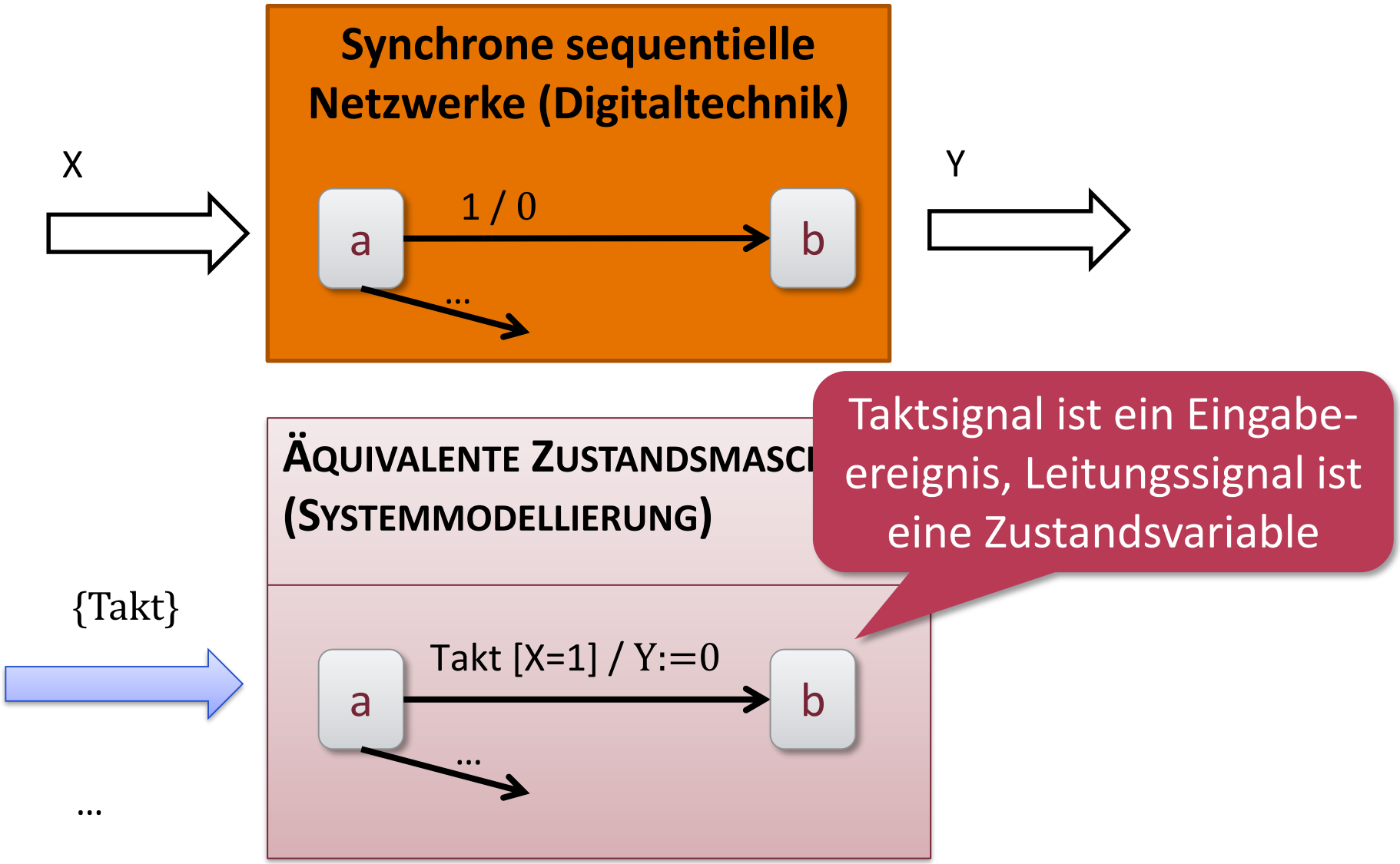
Rendezvous

AUSBLICK, WERKZEUGUNTERSTÜTZUNG

Bedeutung eines Zustandüberganges

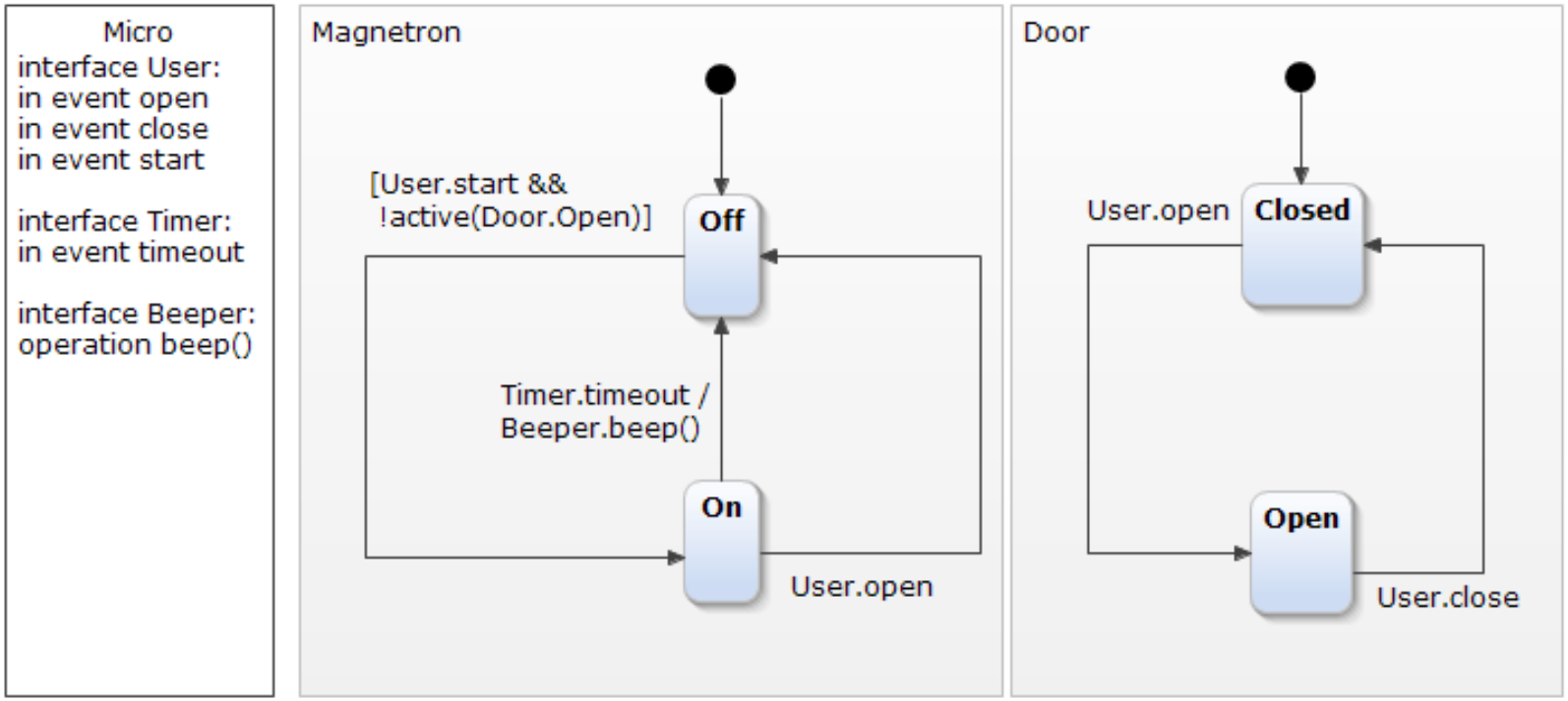
- Beim Entwurf digitaler Schaltnetzwerke
  - Synchrone sequentielle Netzwerke (siehe *Digitaltechnik*)
  - Ein einziges Eingabesignal (Taktsignal)
  - Beschreibung der Übergänge mit Eingabebedingungen
    - Übergang für jedes Taktsignal → wird nicht ausgeschrieben
    - Eingabebedingung: bezieht sich auf *Zustände* der Leitungen
- Bei **Systemmodellierung** allgemein
  - Verschiedene Eingabeereignisse (Signale) modelliert
  - Beschreibung der Übergänge mit Eingabebedingungen
    - Primäre Bedingung: auslösendes **Eingabeereignis**
    - Optional: **Wächterkriterien**, vom Zustand anderer Systeme abhängend

Bedeutung eines Zustandüberganges



Yakindu Statechart Tools

- Komposite Zustandsmaschine sind konstruierbar
  - Statechartssprache → kompakt ausdrückbare Komposition



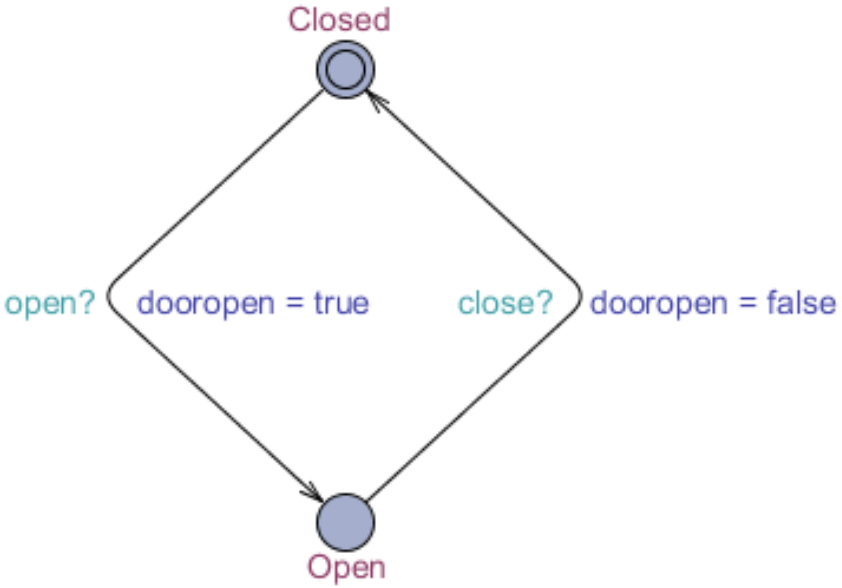
Yakindu Statechart Tools

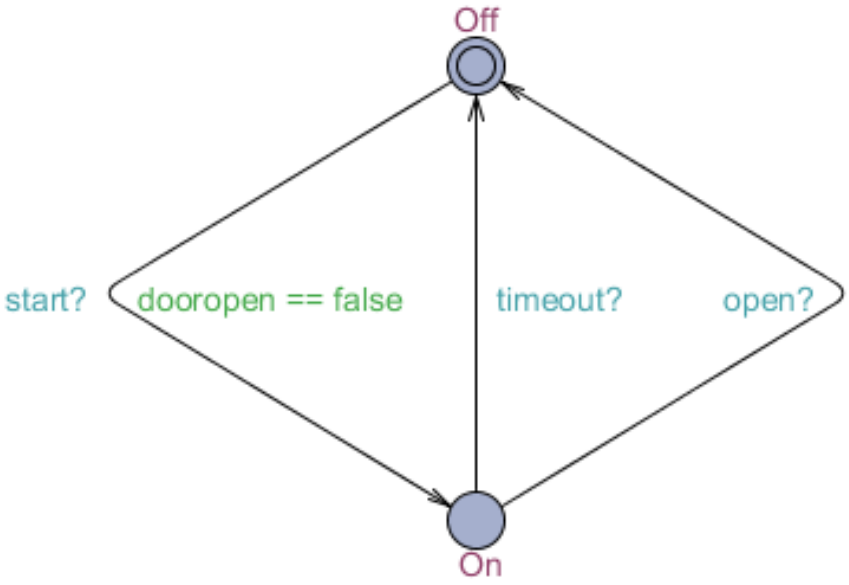
- Aus modellierten Zustandsmaschinen kann C/Java/…-Code generiert werden
  - Übergänge des *Magnetron* in *On*-Zustand (vereinfacht)

```
/* The reactions of state On. */
private void reactMagnetron_On() {
    if (sCITimer.timeout) {
        sCIBeeper.operationCallback.beep();
        stateVector[0] = State.magnetron_Off;
    } else {
        if (sCIUser.open) {
            stateVector[0] = State.magnetron_Off;
        }
    }
}
```

UPPAAL

- Gemischtes Produkt der Zustandsmaschinen





- Verhalten der kompositen Zustandsmaschine …
  - kann simuliert werden
  - kann verifiziert werden

```
A[](!(door.Open && magnetron.On))
```

ZUSAMMENFASSUNG

Definition: Zustandsraum

Der **Zustandsraum**

- ist eine Menge voneinander unterscheideter Systemzustände,
- von der gleichzeitig immer genau ein Element (der **aktuelle Zustand**) für das System charakteristisch ist.

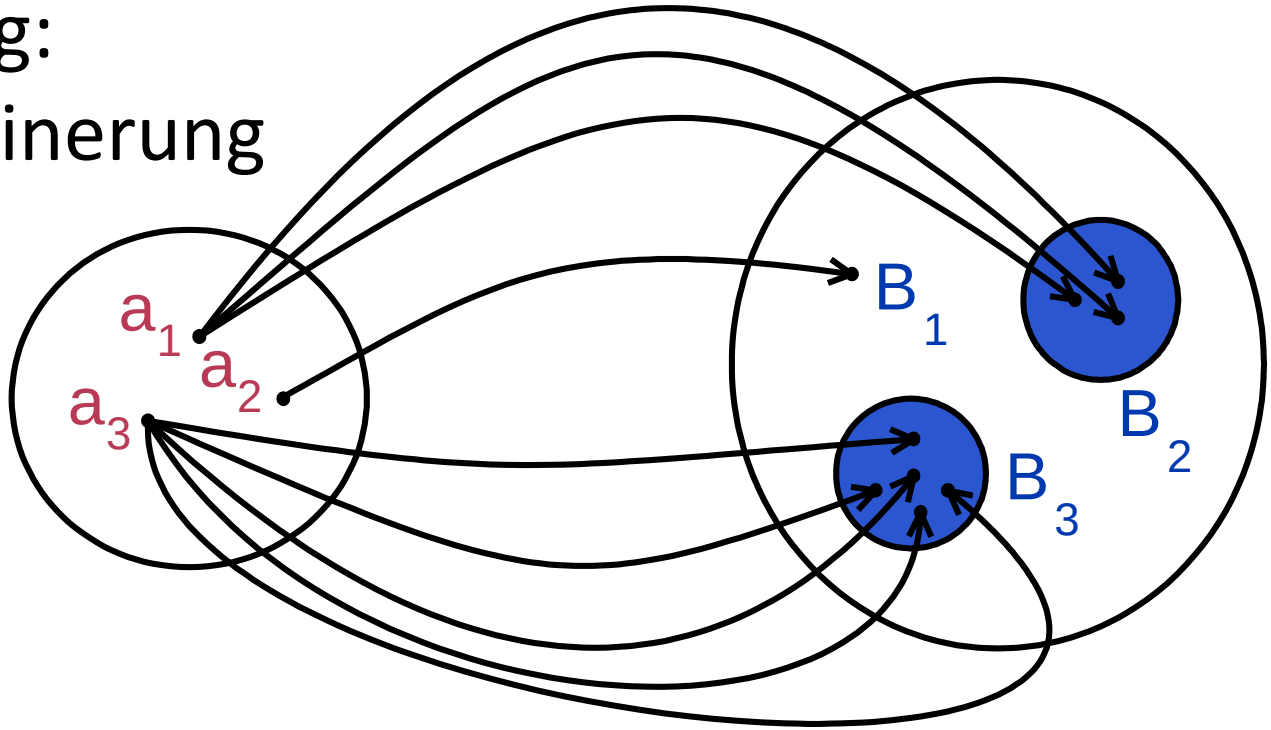
- Beispiele: Zustandsräume
  - Tage: {*Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag, Sonntag*}
  - Zustände des Mikrowellengerätes: *{höchste Stufe, Auftauen, Aus}*
- Beispiele: Aktueller Zustand
  - Heute ist *Mittwoch*.
  - Das Mikrowellengerät ist *Aus*.

Zustandsverfeinerung, Zustandsabstraktion

Zustandsverfeinerung bzw. **Zustandsabstraktion** ist eine auf dem Zustandsraum durchgeführte **Mengenverfeinerung** bzw. **Mengenabstraktion**, die als Ergebnis einen neuen Zustandsraum gibt.

- (auch andere Abstraktionen kommen später vor…)

Wiederholung:
Mengenverfeinerung



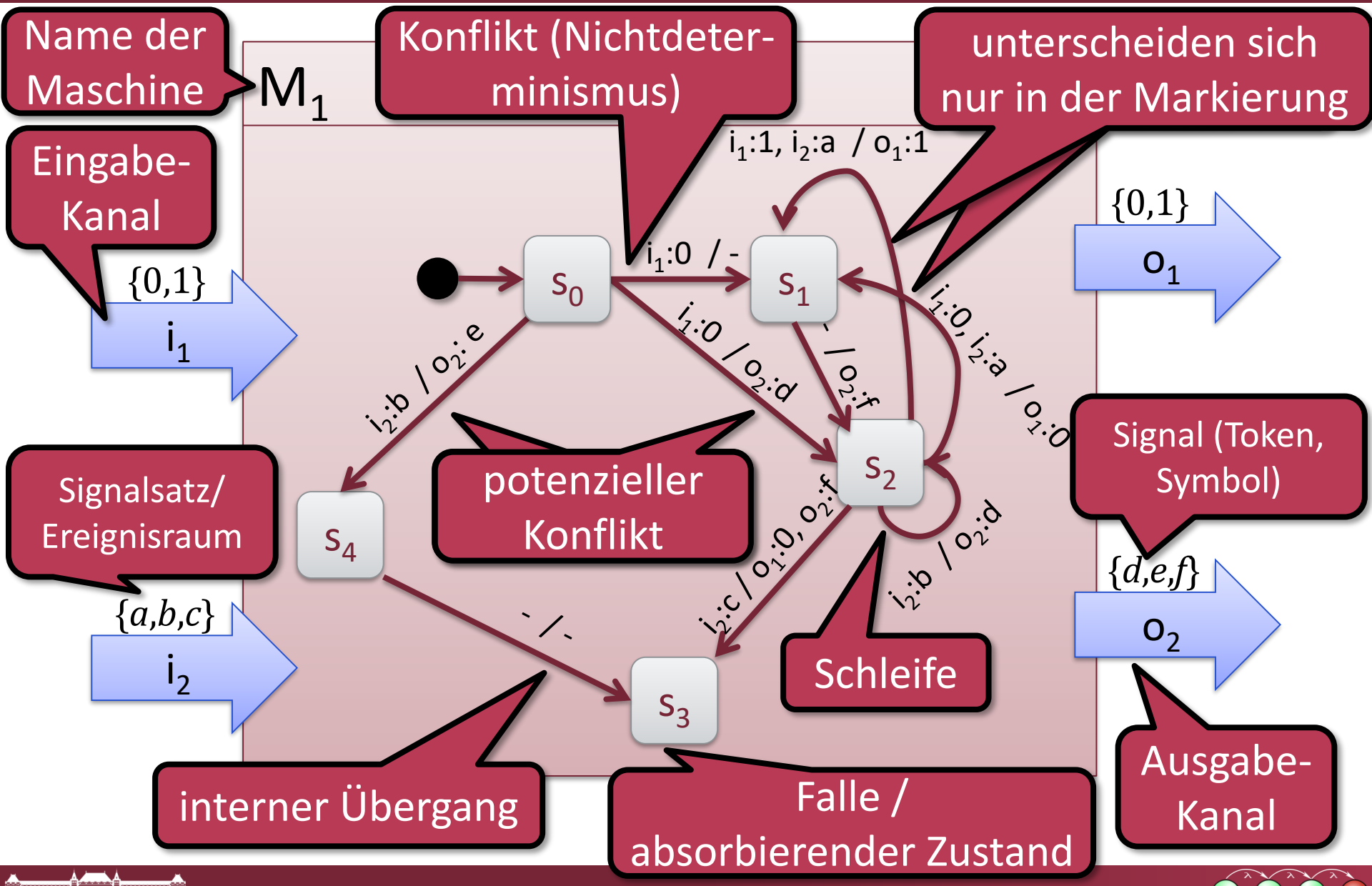
(Direktes) Produkt von Zustandsräumen

Direktes Produkt von Zustandsräumen ist eine **Komposition-Operation** auf den Komponentzustandsräumen**,** dessen Ergebnis ein neuer Zustandsraum (**Produkt-Zustandsraum**) ist, welche als Descartes-Produkt der Mengen von Komponentzustandsräume entsteht.

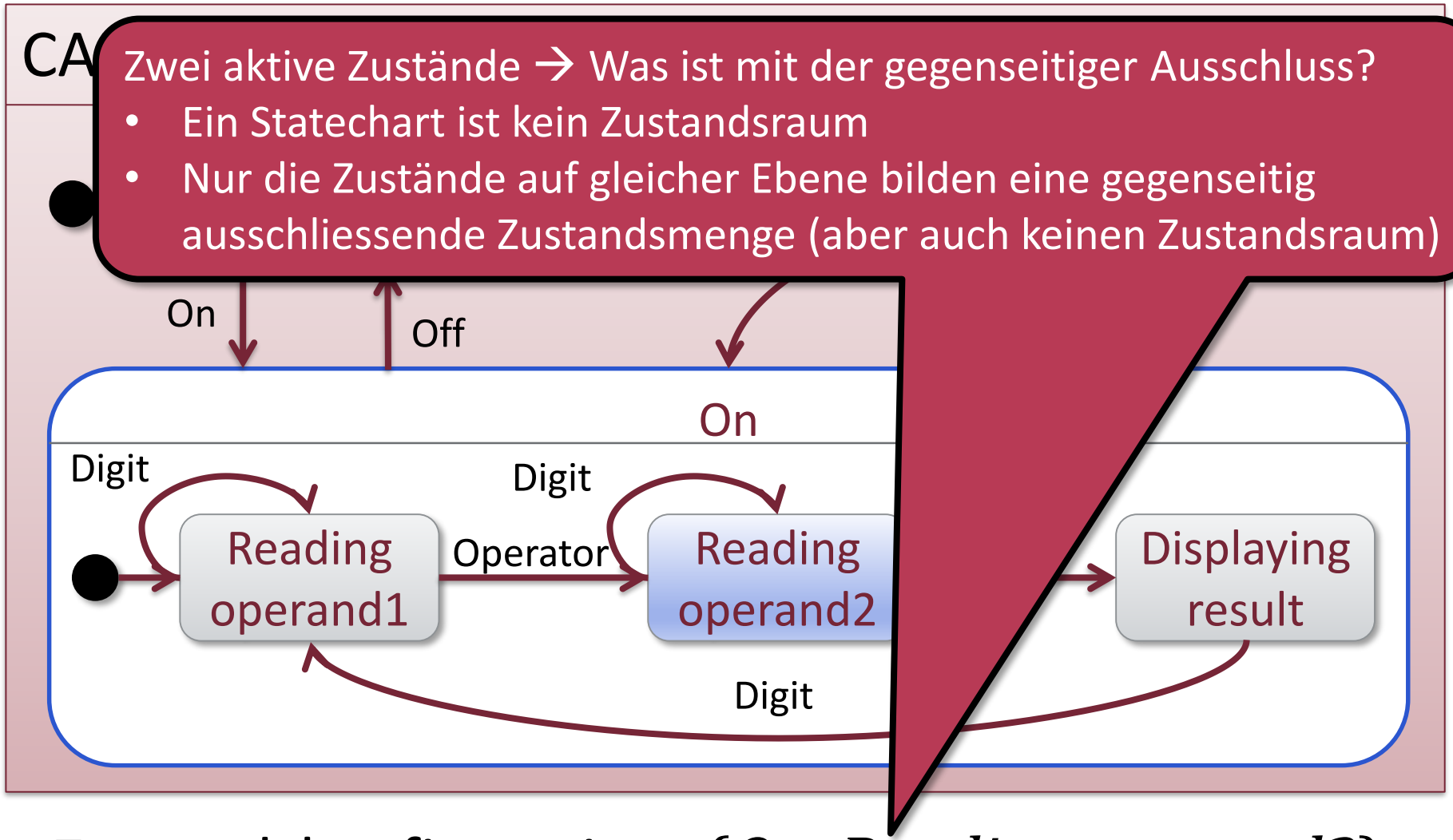
In dem Produkt-Zustandsraum entspricht jeder Zustands-Kombination der Komponentzustandsräume ein zusammengesetzter Zustand (**Zustandsvektor**).

Projektion auf Komponente ist eine Zustandsabstraktion-Operation, die aus dem Zustandsraum des Produktes ein oder mehrere Komponente erhält, die anderen werden vernachlässigt.

Erweiterte Zustandsmaschine



Statechart-Beispiel: Zustandshierarchie

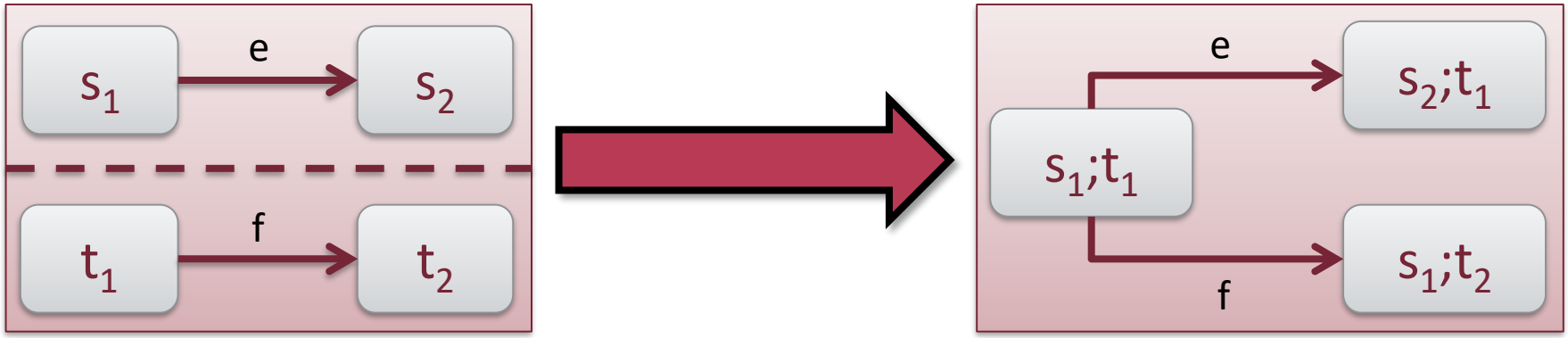


- Zustandskonfiguration: $\{On, Reading\ operand2\}$

Definition: Asynchrones Produkt

Asynchrones Produkt der (Mealy-) Zustandsmaschinen ist eine auf den Komponenten-Zustandsmaschinen (auch Regionen genannt) durchgeführte **Kompositionsoperation**. Das **Ergebnis** der Komposition ist eine (Mealy-) Zustandsmaschine,

- deren Zustandsraum das direkte Produkt der Zustandsräume der Regionen ist,
- in deren Anfangszustand jede Region in ihrer Anfangszustand ist,
- und deren Übergangsregel alle Übergänge bilden, in denen **genau eine Region** einen Übergang macht, weil alle anderen ihren aktuellen Zustand erhalten.



Deutsch - Ungarisch - Englisch

Strukturelles Modell	Felépítési modell	Structural model
Verhaltensmodell	Viselkedési modell	Behavioural model
Ereignis	Esemény	Event
Zeitlos	Pillanatszerű	Instantaneous
Ereignisfluss	Eseményfolyam	Event stream
Zustand	Állapot	State
Zustandbasiertes modell	Állapotalapú modell	State based model
Zustandsraum / -partition	Állapottér / Állapotpartíció	State space
Gegenseitiger Ausschluss	Kölcsönösen kizárólagos	Mutually exclusive
Vollständig	Teljes	Complete
Zustandsfrei	Állapotmentes	Stateless
Produkt der Zustandsräume	Állapotterek szorzata	Product of state spaces
Zustandsvariable	Állapotváltozó	State variable
Komposit	Összetett	Composite
Zustandsraum-Explosion	Állapottér-robbanás	State space explosion

Deutsch - Ungarisch - English

Endlich	Véges	Finite
Diskret	Diszkrét („szétválasztható”)	Discrete
Diskret/Dezent	*Diszkrét („nem pletykál”)*	*Discreet* ☺
Zustandsübergang	Állapotátmenet	Transition
Zustandsübergang-Regel	Állapotátmeneti szabály	Transition rule
Nichtdeterminismus	Nemdeterminizmus	Nondeterminism
Konflikt	Konfliktus	Conflict
Zustandsmaschine / Automat	Állapotgép / Automata	State machine / Automaton
Anfangszustand	Kezdőállapot	Initial state
Markierung	Címke	Label
Ursache / Pre-Kondition	Ok / Előfeltétel	Cause / Precondition
Folgerung / Post-Kondition	Következmény / Utófeltétel	Consequence / Postcondition
Schleife	Hurokél	Loop edge
Kanal	Csatorna	Channel
Signal / Zeichen / Symbol	Jel / Jel / Szimbólum	Signal / Token / Symbol

Deutsch - Ungarisch - English

Absorbierender Zustand / Falle	Nyelő / Csapda	Sink / Trap
Synchrones Produkt	Szinkron szorzat	Synchronous product
Asynchrones Produkt	Aszinkron szorzat	Asynchronous product
Wächterkriterie	Őrfeltétel	Guard condition