

## 6. Modelle für Nebenläufigkeit

Was sind nebenläufige Systeme?

**Ganz allgemein:** Systeme, bei denen mehrere Komponenten/Prozesse nebenläufig arbeiten und typischerweise durch Nachrichtenaustausch miteinander kommunizieren. Das englische Wort für nebenläufig ist "concurrent".

Nebenläufigkeit versus Parallelität:

Parallelität

Zwei Ereignisse finden **parallel** statt, wenn sie gleichzeitig ausgeführt werden.

Nebenläufigkeit (engl. concurrency)

Zwei Ereignisse sind **nebenläufig**, wenn sie parallel ausgeführt werden können (jedoch nicht müssen), das heißt, wenn zwischen ihnen keine kausale Abhängigkeit besteht.

Das bedeutet: Nebenläufigkeit ist der allgemeinere Begriff.

Beispiele

- Prozesswechsel auf einem Prozessor (Nebenläufigkeit, aber keine Parallelität)
- Rechner mit mehreren Prozessoren
- Netzwerk (LAN, WAN, Internet)
- Nebenläufige/parallele Anwendungen

Hierbei ist zu unterscheiden:

- Parallelität zum Effizienzgewinn (Beispiel: paralleles Sortierverfahren)
- Nebenläufigkeit durch Nutzung verteilter Ressourcen (Beispiel: Verwendung von Web Services)

## (Mögliche) Charakteristika von nebenläufigen Systeme

- Nebenläufigkeit/Parallelität
- Offenheit (Erweiterbarkeit, Interaktion mit der Umgebung)
- Modularität
- Nicht-terminierendes Verhalten (unendlich lange Abläufe)
- Nicht-Determinismus
- Temporale Eigenschaften (z.B.: "irgendwann wird etwas passieren")

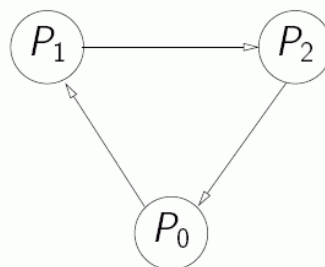
## Probleme beim Umgang mit nebenläufigen Systemen

- Deadlocks (Verklemmungen)
- Einhaltung des wechselseitigen Ausschlusses (engl. mutual exclusion)
- Unendlicher bzw. sehr großer Zustandsraum
- Stark dynamisches Verhalten/wechselnde Anzahl von Prozessen
- Variable Topologie/Mobilität

**Daher:** Methoden zur Modellierung, Analyse und Verifikation solcher Systeme sind erforderlich.

## Einführendes Beispiel: Prozesskommunikation

Wir betrachten das folgende System mit drei Prozessen  $P_0$ ,  $P_1$ ,  $P_2$ :



Dabei schickt  $P_0$  Nachrichten an  $P_1$ ,  $P_1$  schickt Nachrichten an  $P_2$  und  $P_2$  schickt Nachrichten an  $P_0$ .

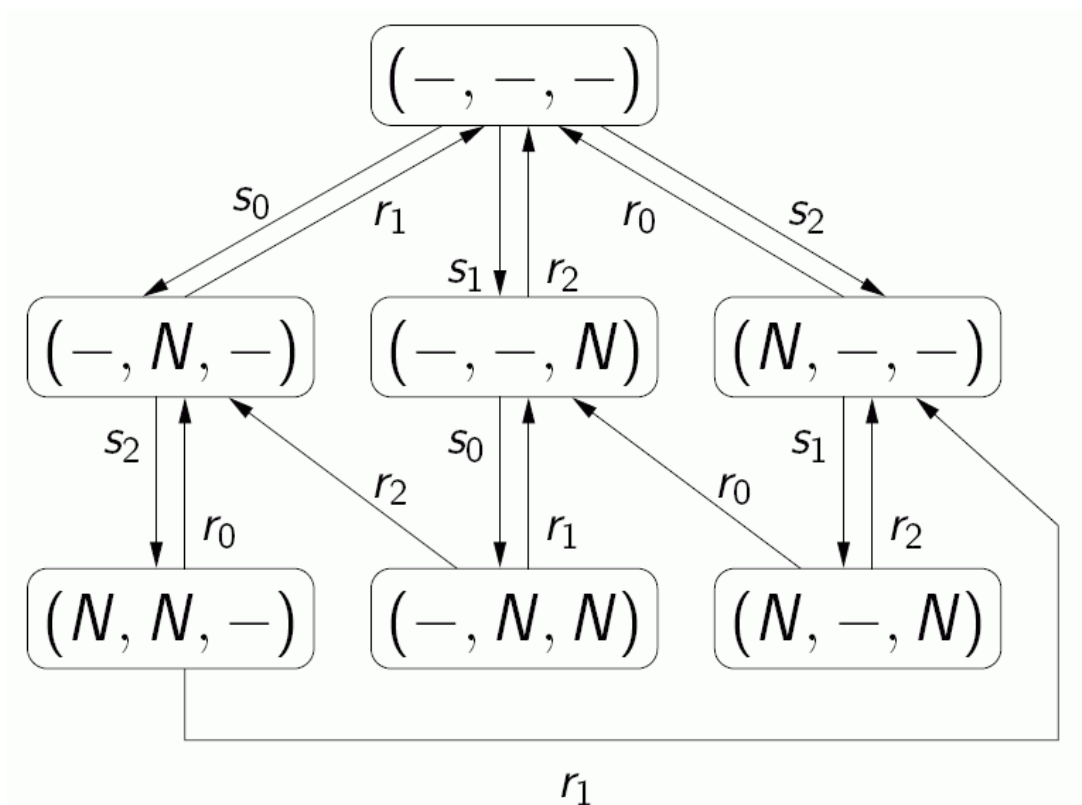
Ein Prozess  $P_i$  verhält sich wie folgt:

- Falls eine Nachricht für  $P_i$  anliegt, so empfängt  $P_i$  diese Nachricht (Aktion  $r_i$  – receive).
- Falls keine Nachricht für  $P_i$  anliegt und auch keine Nachricht am Nachbarn  $P_j$  (mit  $j = (i + 1) \bmod 3$ ) anliegt, so sendet  $P_i$  eine Nachricht an  $P_j$  (Aktion  $s_i$  – send).

Zu jedem Zeitpunkt kann ein beliebiger Prozess des Systems die nächste Aktion ausführen.

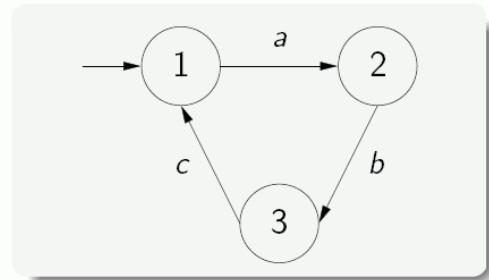
Beschreibung durch ein **Transitionssystem**:

- **System-Zustände** der Form  $(x, y, z)$  mit  $x, y, z \in \{N, -\}$ .  
**Beispiel:**  $(N, N, -)$  bedeutet, dass an den Prozessen  $P_0$  und  $P_1$  jeweils eine Nachricht anliegt.
- Transitionen/Zustandsübergänge mit Aktionen  $r_i$  (Prozess  $i$  liest eine Nachricht) und  $s_i$  (Prozess  $i$  sendet eine Nachricht).



# Transitionssysteme

- Transitionssysteme repräsentieren Zustände und Zustandsübergänge von Systemen.
- Echte Parallelität wird nicht direkt repräsentiert.
- Starke Ähnlichkeit zu Automaten, aber hier sind wir weniger an der akzeptierten Sprache interessiert.



## Aktionen und Aktionsfolgen

Wir verwenden folgende Notation:

- $Act$ : Menge von **atomaren Aktionen**, oft mit  $a, b, c, \dots$  bezeichnet. Manchmal verwenden wir auch die interne Aktion  $\tau$ , die von außen her nicht beobachtbar sein soll.
- $Act^*$ : Die Menge aller **endlichen Wörter** (engl. **finite words**) über dem Alphabet  $Act$ .
- $Act^\omega$ : Die Menge aller **unendlichen Wörter** (engl. **infinite words**) über dem Alphabet  $Act$ .  
Ein unendliches Wort  $w \in Act^\omega$  kann durch eine Abbildung  $w: \mathbb{N}_0 \setminus \{0\} \rightarrow Act$  dargestellt werden.
- $Act^\infty = Act^* \cup Act^\omega$ : Die Menge aller **endlichen und unendlichen Wörter** über  $Act$ .
- $Pref(L)$ : Für eine Sprache  $L \subseteq Act^\infty$  definieren wir die Menge aller **Präfixe** von  $L$  wie folgt:

$$Pref(L) = \{u \in Act^* \mid \exists w \in L, v \in Act^\infty : w = uv\}.$$

### Transitionssystem (Definition)

Gegeben sei eine feste Menge von Aktionen  $Act$ . Ein **Transitionssystem**  $T = (Z, \rightarrow)$  über  $Act$  besteht aus

- Einer Menge von Zuständen  $Z$  und
- einer Menge  $\rightarrow \subseteq Z \times Act \times Z$  von Zustandsübergängen.

- Ein Transitionssystem heißt **endlich** (engl. **finite**), wenn sowohl die Zustandsmenge als auch die Übergangsmenge endlich sind.
- Statt  $(z, \ell, z') \in \rightarrow$  schreiben wir in Zukunft  $z \xrightarrow{\ell} z'$ .
- Außerdem werden wir  $z \xrightarrow{a_1} \dots \xrightarrow{a_n} z'$  im folgenden mit  $z \xrightarrow{a_1 \dots a_n} z'$  abkürzen. Es gilt außerdem, dass  $z \xrightarrow{\varepsilon} z$  für jeden Zustand  $z$ .
- Für einen Zustand  $z$  schreiben wir  $z \xrightarrow{a}$ , falls es einen Zustand  $z'$  mit  $z \xrightarrow{a} z'$  gibt und  $z \not\xrightarrow{a}$ , wenn es einen solchen Zustand nicht gibt.

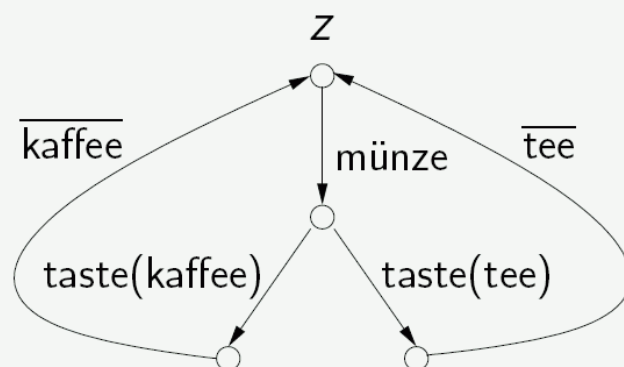
## Beispiel

Ein **klassisches Beispiel**: die Tee- und Kaffee-Maschine.

Wir wollen eine ganz einfache Maschine modellieren, die

- nach Münzeinwurf und Tastendruck entweder Tee oder Kaffee ausgibt,
- Fehlverhalten zeigen kann *und*
- sich unter Umständen nicht-deterministisch verhält.

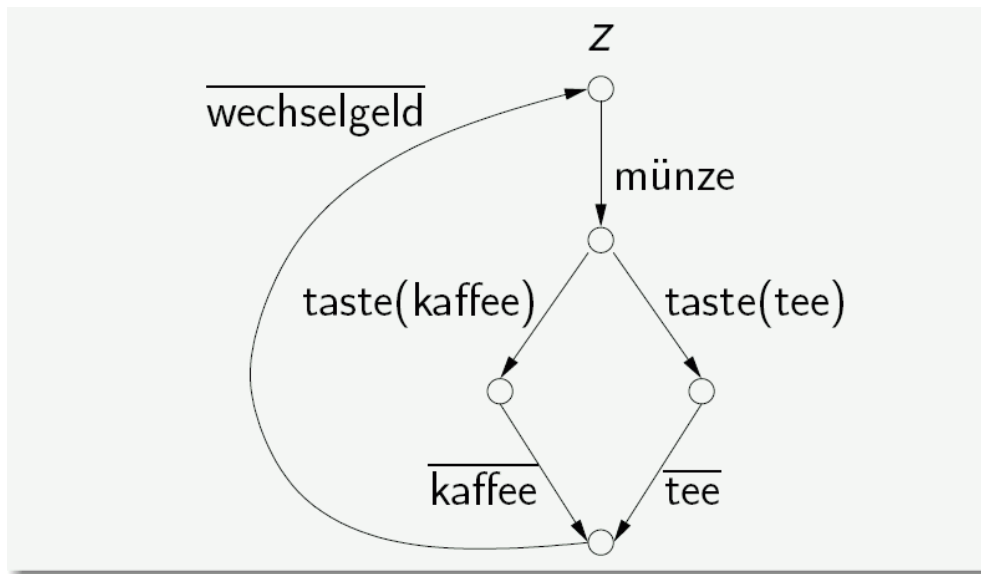
### 1. Version



Eine Tee-/Kaffee-Maschine.

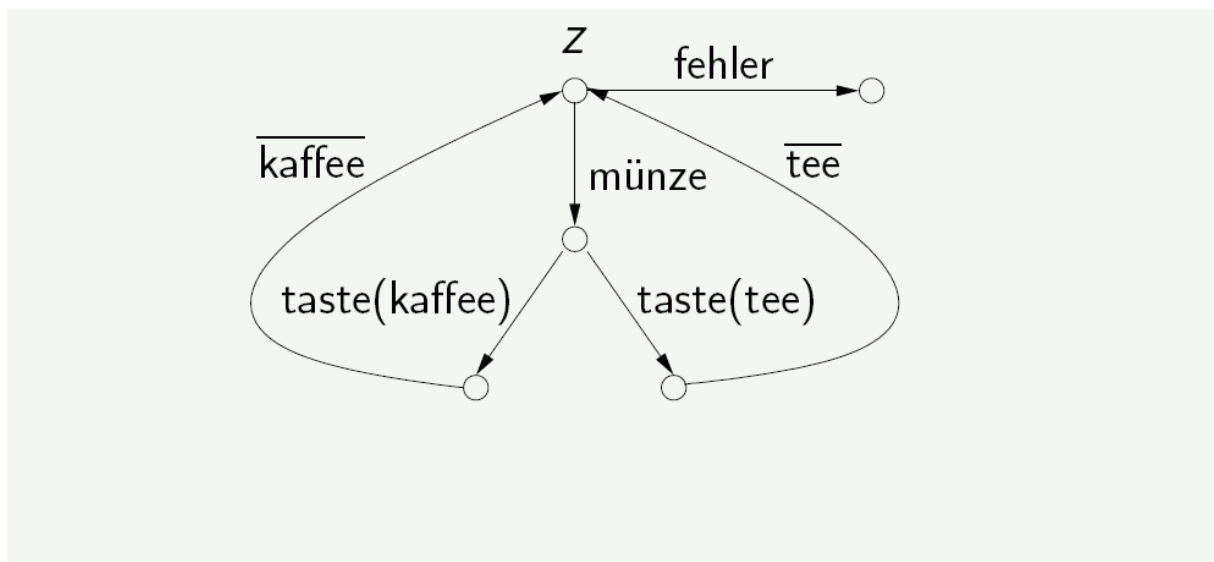
Wir unterscheiden Eingabeaktionen der Form  $a$  und Ausgabeaktionen (auch Ko-Aktionen genannt) der Form  $\bar{a}$ .

## 2. Version



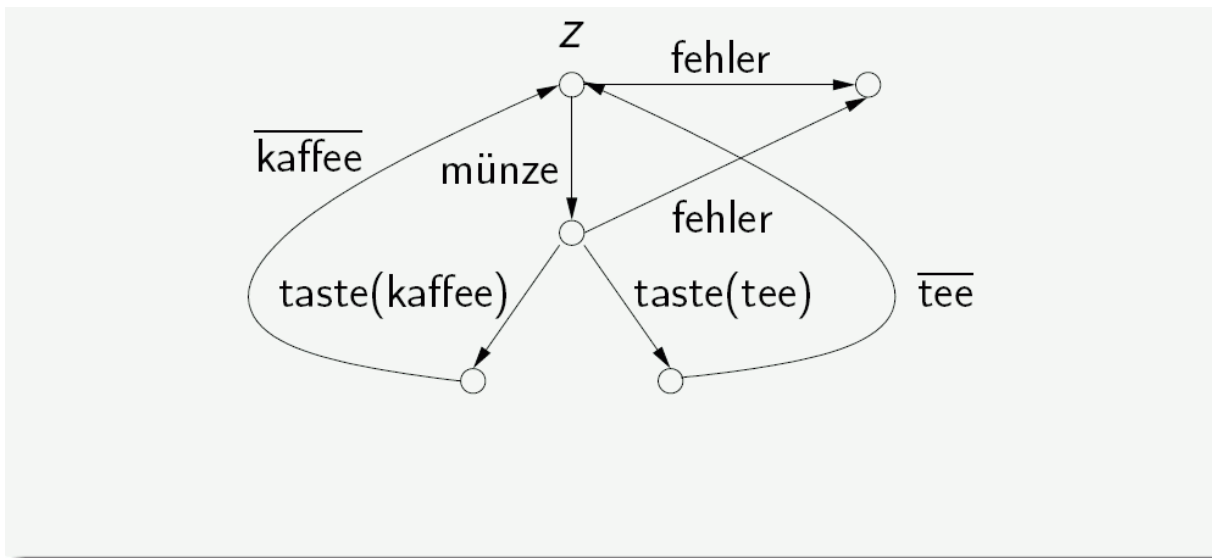
Eine Maschine, die sogar wechselt.

## 3. Version



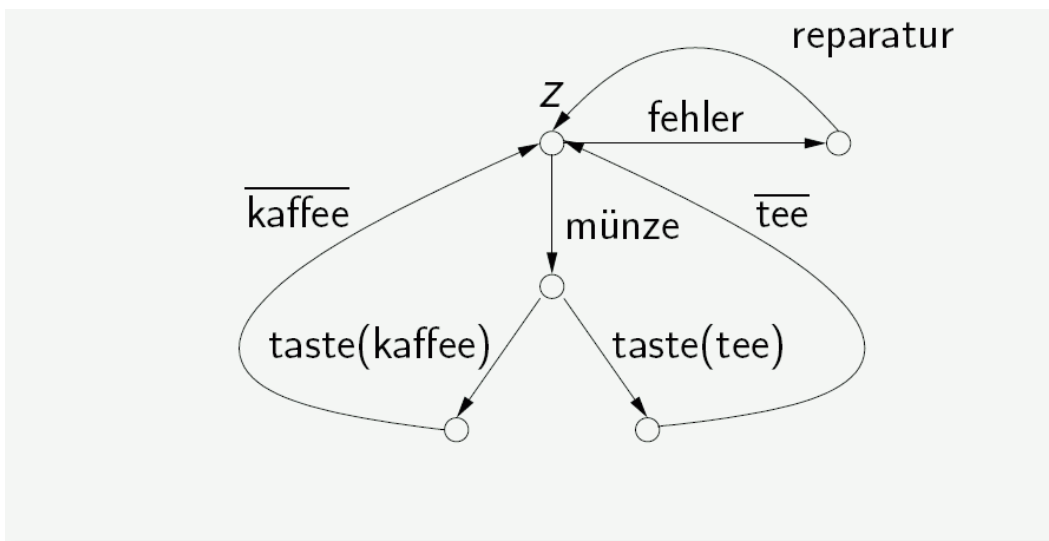
Eine Maschine mit Fehler. Das Auftreten eines Fehlers entspricht eher einer internen Aktion und könnte alternativ auch mit einem  $\tau$  modelliert werden.

#### 4. Version



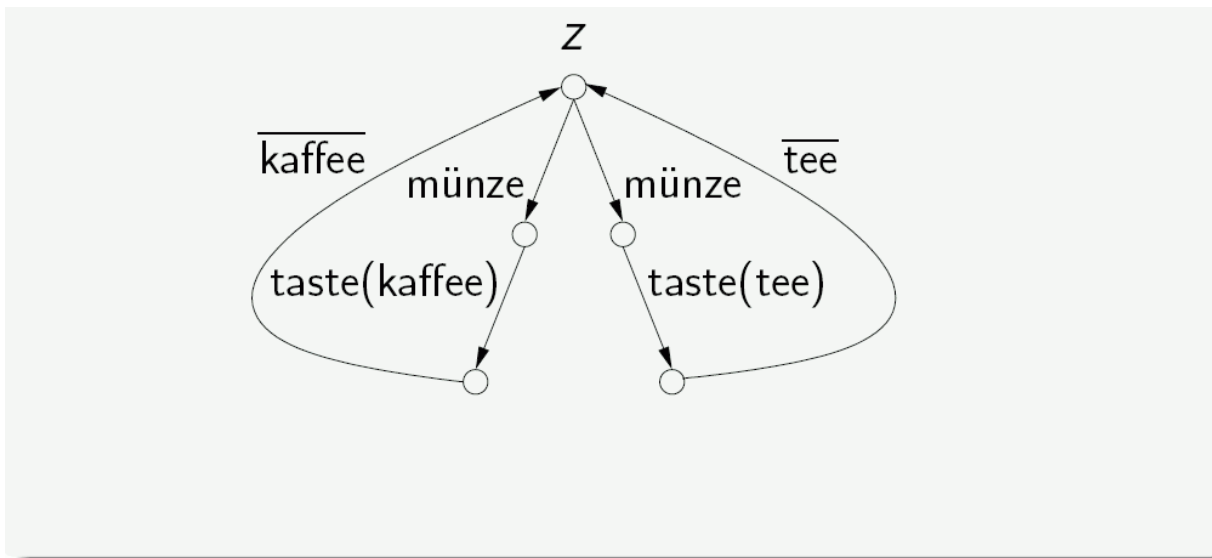
Eine (unfaire) Maschine mit Fehler, die nach Einwurf einer Münze in den Fehlerzustand übergehen kann.

#### 5. Version



Eine Maschine mit Fehler, die wieder repariert werden kann.

## 6. Version



Eine Maschine mit nicht-deterministischem Verhalten, die die Getränkeauswahl für den Benutzer trifft.

## Deterministische Transitionssysteme:

### Deterministisches Transitionssystem (Definition)

Ein Transitionssystem  $T = (Z, \rightarrow)$  heißt **deterministisch**, wenn für jeden Zustand  $z \in Z$  gilt:

Falls  $z \xrightarrow{a} z_1$  und  $z \xrightarrow{a} z_2$ , dann gilt  $z_1 = z_2$ .

### Bemerkungen:

- Alle Tee-/Kaffee-Maschinen bis auf die letzte sind deterministisch.
- Im Gegensatz zu deterministischen Automaten wird hier nicht gefordert, dass bei einem deterministischem Transitionssystem von jedem Zustand aus auch jede Aktion möglich ist.



# Verhaltensäquivalenzen von Transitionssystemen

## Sprachäquivalenz

### Sprache eines Zustands (Definition)

Mit der **Sprache eines Zustands**  $z$  bezeichnen wir alle Wörter über  $Act$ , die den von  $z$  ausgehenden Pfaden entsprechen. Es gibt drei verschiedene Möglichkeiten, Sprachen zu definieren:

- $S(z) = \{w \in Act^* \mid \exists z' \in Z: z \xrightarrow{w} z'\}$
- $S^\omega(z) = \{a_1 a_2 a_3 \dots \in Act^\omega \mid z \xrightarrow{a_1} z_1 \xrightarrow{a_2} z_2 \xrightarrow{a_3} \dots\}$
- $S^\infty(z) = S(z) \cup S^\omega(z)$

### Sprachäquivalenz (Definition)

Zwei Zustände  $z, z' \in Z$  heißen **S-sprachäquivalent**, wenn  $S(z) = S(z')$  gilt. Analog ist  **$S^\omega$ -** und  **$S^\infty$ -Sprachäquivalenz** definiert.

**Sprachäquivalenz** wird auf Englisch auch als **trace equivalence** bezeichnet.

Die Sprachäquivalenz hat folgende Eigenschaft:

Falls  $z, z'$   $S^\infty$ -sprachäquivalent sind, dann sind sie auch  $S$ - und  $S^\omega$ -sprachäquivalent.

### Begründung:

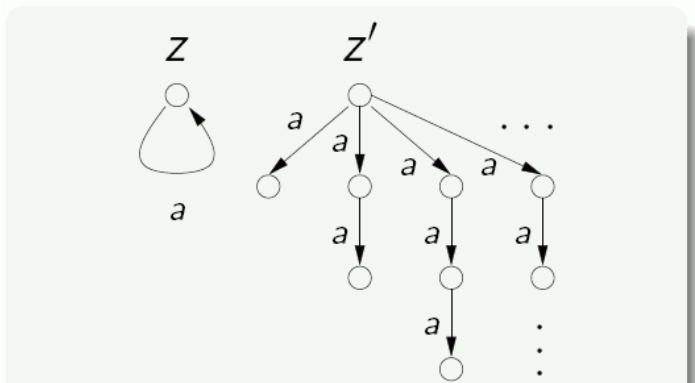
- Aus  $S^\infty(z) = S^\infty(z')$  folgt  
 $S(z) = S^\infty(z) \cap Act^* = S^\infty(z') \cap Act^* = S(z')$ .
- Und analog:  
 $S^\omega(z) = S^\infty(z) \cap Act^\omega = S^\infty(z') \cap Act^\omega = S^\omega(z')$ .

Es gilt aber auch:

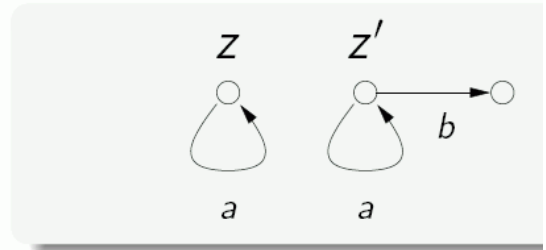
Falls  $z, z'$   $S$ -sprachäquivalent sind, dann sind sie nicht notwendigerweise  $S^\omega$ -sprachäquivalent (und umgekehrt).

### Gegenbeispiele:

Zustände  $z, z'$  sind  $S$ -äquivalent, aber nicht  $S^\omega$ -äquivalent. (Von  $z'$  aus gibt es nur endliche Pfade.)

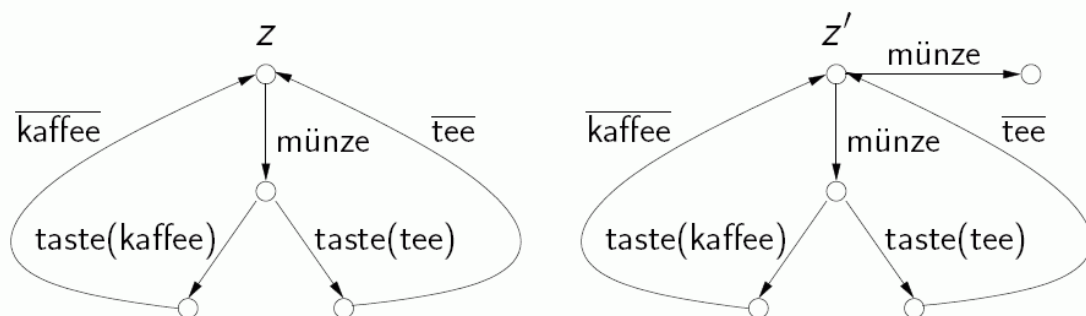


Zustände  $z, z'$  sind  $S^\omega$ -äquivalent, aber nicht  $S$ -äquivalent.



## Verweigerungsäquivalenz

**Motivation:** die Zustände  $z, z'$  der beiden folgenden Tee-/Kaffee-Maschinen sind sprachäquivalent.



Allerdings möchte man diese Zustände nicht wirklich als "gleich" ansehen. In dem einen Fall funktioniert die Maschine korrekt, in dem anderen verschluckt sie eine Münze, ohne etwas auszugeben.

**Folgerung:** Sprachäquivalenz ist nicht ausreichend. Wir benötigen auch noch die Möglichkeit zu sagen, dass die erreichbaren Zustände dieselben Aktionen erlauben bzw. nicht erlauben sollen.

## Verweigerungsäquivalenz (Definition)

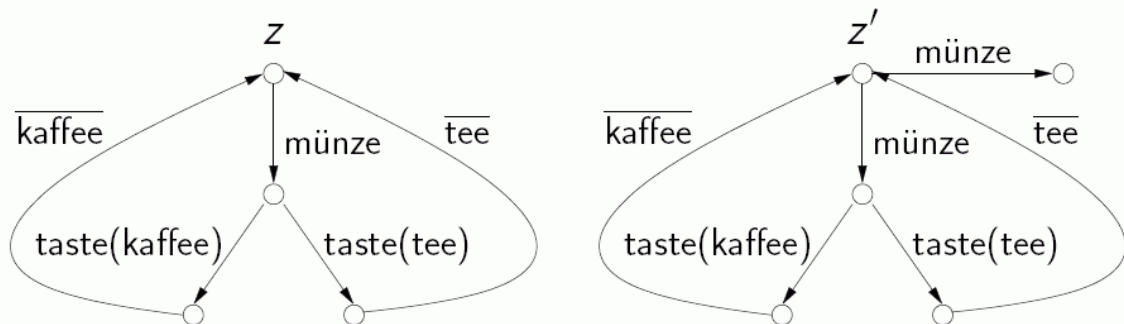
Sei  $T = (Z, \rightarrow)$  ein Transitionssystem. Die Menge der **Verweigerungspaare** eines Zustands  $z \in Z$  ist definiert als:

$$\mathcal{F}(z) = \{(w, A) \in \text{Act}^* \times \mathcal{P}(\text{Act}) \mid \exists z' \in Z : z \xrightarrow{w} z' \text{ und } z' \not\xrightarrow{a} \text{ für alle } a \in A\}$$

Zwei Zustände  $z, z' \in Z$  heißen **verweigerungsäquivalent**, wenn  $\mathcal{F}(z) = \mathcal{F}(z')$  gilt.

**Verweigerungsäquivalenz** wird auf Englisch als **failures equivalence** bezeichnet.

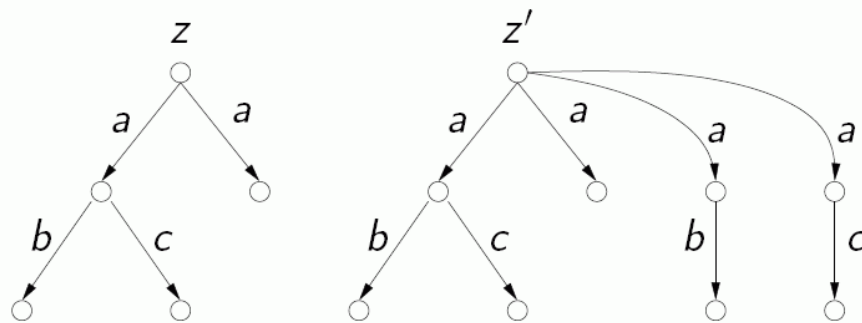
**Motivation:** die Zustände  $z, z'$  der beiden folgenden Tee-/Kaffee-Maschinen sind sprachäquivalent.



In diesem Beispiel enthält  $\mathcal{F}(z')$  das Paar  $(\text{münze}, \{\text{taste(kaffee)}, \text{taste(tee)}\})$ , wobei dieses Paar in  $\mathcal{F}(z)$  nicht vorkommt.

$\Rightarrow z$  und  $z'$  sind nicht verweigerungsäquivalent.

Ein weiteres Beispiel für Verweigerungsäquivalenz:



Es gilt:

$$\begin{aligned} \mathcal{F}(z) &= \{(\varepsilon, A) \mid A \subseteq \{b, c\}\} \cup \{(a, A) \mid A \subseteq \{a, b, c\}\} \cup \\ &\quad \{(ab, A) \mid A \subseteq \{a, b, c\}\} \cup \{(ac, A) \mid A \subseteq \{a, b, c\}\} \\ &= \mathcal{F}(z') \end{aligned}$$

und  $z, z'$  sind damit verweigerungsäquivalent.

## Bisimilarität

Es gibt noch eine weitere wichtige Äquivalenz, die etwas feiner als Verweigerungsäquivalenz und im allgemeinen auch leichter zu mechanisieren ist: die **Bisimilarität** oder **Bisimulationsäquivalenz**.

Dabei verlangt man anschaulich, dass die beiden Zustände  $z, z'$  in der Lage sind, sich gegenseitig zu **simulieren**.

### Bisimulation (Definition)

Sei  $T = (Z, \rightarrow)$  ein Transitionssystem. Eine Relation  $R \subseteq Z \times Z$  auf den Zuständen heißt **Bisimulation**, wenn für jedes Paar  $(z_1, z_2) \in R$  und für jede Aktion  $a \in Act$  gilt:

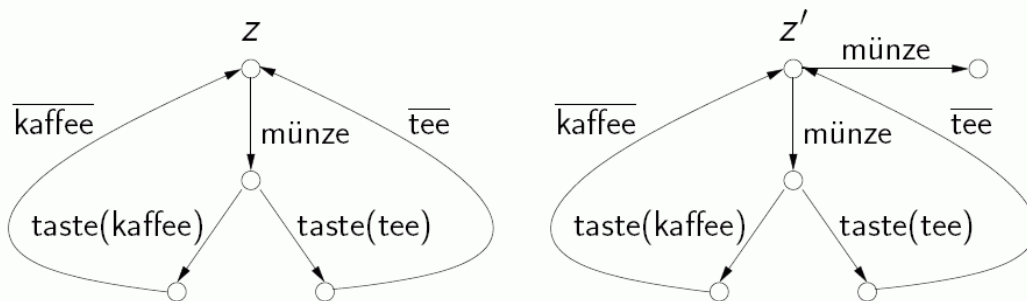
- für jedes  $z'_1$  mit  $z_1 \xrightarrow{a} z'_1$  gibt es ein  $z'_2$  mit  $z_2 \xrightarrow{a} z'_2$  und  $(z'_1, z'_2) \in R$ .
- für jedes  $z'_2$  mit  $z_2 \xrightarrow{a} z'_2$  gibt es ein  $z'_1$  mit  $z_1 \xrightarrow{a} z'_1$  und  $(z'_1, z'_2) \in R$ .

## Bisimilarität (Definition)

Zwei Zustände  $z_1, z_2$  heißen **bisimilar**, wenn es eine Bisimulation  $R$  gibt mit  $(z_1, z_2) \in R$ . In diesem Fall schreiben wir  $z_1 \sim z_2$ . Die Relation  $\sim$  heißt dann **Bisimilarität** oder **Bisimulationsäquivalenz**.

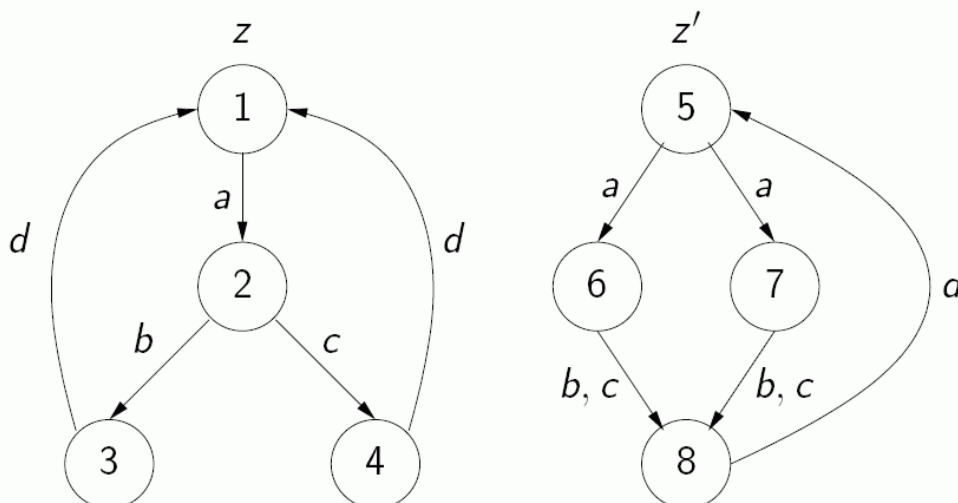
Bisimilarität wird auf Englisch auch als **bisimilarity** oder **bisimulation equivalence** bezeichnet.

### Beispiele:



Die beiden Zustände  $z, z'$  sind nicht bisimilar.

### Beispiele:



Die beiden Zustände  $z = 1, z' = 5$  sind bisimilar.

Bisimulationsrelation:

$$R = \{(1, 5), (2, 6), (2, 7), (3, 8), (4, 8)\}$$

## Vergleich der verschiedenen Verhaltensäquivalenzen

Wie stehen die verschiedenen Verhaltensäquivalenzen von Zuständen in Transitionssystemen untereinander in Beziehung?

### Verweigerungsäquivalenz impliziert $S$ -Sprachäquivalenz (Satz)

Seien  $z, z'$  zwei verweigerungsäquivalente Zustände, d.h., es gilt  $\mathcal{F}(z) = \mathcal{F}(z')$ . Dann sind  $z, z'$  auch  $S$ -sprachäquivalent, d.h., es gilt  $S(z) = S(z')$ .

**Bemerkung:** zwei verweigerungsäquivalente Zustände sind jedoch nicht notwendigerweise  $S^\omega$ -sprachäquivalent.

### Bisimilarität impliziert Verweigerungsäquivalenz und $S^\omega$ -Sprachäquivalenz (Satz)

Seien  $z, z'$  zwei bisimulare Zustände, d.h., es gilt  $z \sim z'$ . Dann sind  $z$  und  $z'$  auch verweigerungsäquivalent und  $S^\omega$ -sprachäquivalent.

## "Simulations"-Relation als Quasiordnung:

Den Begriff der Simulation erhält man aus der Definition der Bisimulation, wobei nur die erste Bedingung verwendet wird.

### Simulation (Definition)

Sei  $T = (Z, \rightarrow)$  ein Transitionssystem. Eine Relation  $R \subseteq Z \times Z$  auf den Zuständen heißt **Simulation**, wenn für jedes Paar  $(z_1, z_2) \in R$  und für jede Aktion  $a \in Act$  gilt:

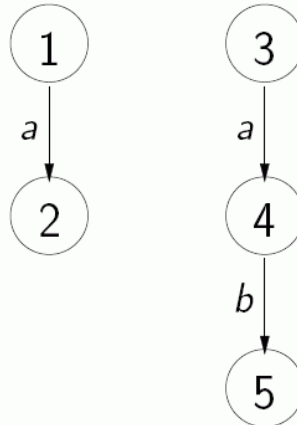
- für jedes  $z'_1$  mit  $z_1 \xrightarrow{a} z'_1$  gibt es ein  $z'_2$  mit  $z_2 \xrightarrow{a} z'_2$  und  $(z'_1, z'_2) \in R$ .

Wir sagen, dass ein Zustand  $z_2$  einen Zustand  $z_1$  simuliert (in Zeichen:  $z_1 \prec z_2$ ), wenn es eine Simulation  $R$  gibt mit  $(z_1, z_2) \in R$ .

**Bemerkung:** Aus  $z_1 \prec z_2$  folgt immer  $S(z_1) \subseteq S(z_2)$  und  $S^\omega(z_1) \subseteq S^\omega(z_2)$ .

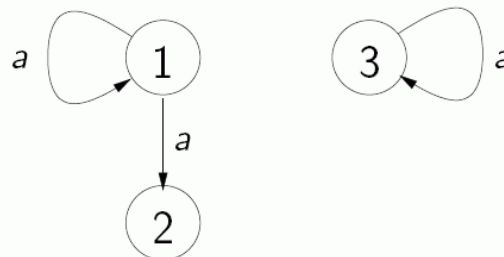
### Bemerkungen zur Simulation:

- Es gelte  $z_1 \prec z_2$ , d.h.  $z_2$  simuliert  $z_1$ . Dann gilt nicht notwendigerweise  $\mathcal{F}(z_1) \subseteq \mathcal{F}(z_2)$  (oder umgekehrt).



Da  $R = \{(1, 3), (2, 4)\}$  eine Simulation ist, gilt  $1 \prec 3$ .  
Es gilt jedoch weder  $\mathcal{F}(1) \subseteq \mathcal{F}(3)$  (wegen  $(a, \{b\}) \notin \mathcal{F}(3)$ ),  
noch  $\mathcal{F}(3) \subseteq \mathcal{F}(1)$  (wegen  $(ab, \emptyset) \notin \mathcal{F}(1)$ ).

- Aus  $z_1 \prec z_2$  und  $z_2 \prec z_1$  folgt nicht notwendigerweise  $z_1 \sim z_2$ .  
Das heißt, aus wechselseitiger Simulation folgt nicht immer Bisimilarität.



Es gilt  $1 \prec 3$ , da  $\{(1, 3), (2, 3)\}$  eine Simulation ist, und  $3 \prec 1$ , da  $\{(3, 1)\}$  eine Simulation ist. Jedoch:  $1 \not\sim 3$ .