# Zustandsorientierte Modellierung

Modellbildung in der Entwicklung

Prof. Dr. Dr. h.c. Manfred Broy
Gemeinsam mit Dr. Bernhard Schätz
Fakultät für Informatik, TU München

SS 2007

TECHNISCHE UNIVERSITÄT MÜNCHEN

## Inhalte

Vorlesungsinhalte:

1. Modellbegriff
2. Grundlagen
   1. Datenmodellierung
   2. Zustandsmodellierung
      1. Grundlagen
      2. Ausführungsmodellierung
      3. Interaktionsmodellierung
      4. Nebenläufigkeitsmodellierung
      5. Kommunikationsmodellierung
   3. Abläufe und Prozesse
   4. Architekturen
   5. Schnittstellen
   6. Modellintegration
3. Anwendung
4. Modellgetriebene Entwicklung
5. Domänenspezifische Modellierung



## Grundlagen: Endlicher Automat

Ein nichtdeterministischer, endlicher Automat ist ein 5-Tupel $(S, \Sigma, \delta, s_0, F)$:

- $S$, nicht-leere, endliche Menge von Zuständen.
- $\Sigma$, nicht-leere, endliche Menge von Eingabezeichen
- $\delta \subseteq S \times \Sigma \times S$, Zustandsübergangsrelation
- $s_0 \in S$, Startzustand
- $F \subseteq S$, Menge der akzeptierenden Endzustände

Der Automat akzeptiert $w \in \Sigma^*$ genau dann, wenn $\delta^*(s_0, w) \cap F \neq \emptyset$
mit $\delta^*(q, \epsilon) = \{q\}$ und $\delta^*(q, a \circ w') = \bigcup_{(q,a,p)\in\delta} \delta^*(p, w')$.



## Grundlagen: Endlicher Automat

Ein deterministischer, endlicher Automat ist ein 5-Tupel $(S, \Sigma, \delta, s_0, F)$:

- $S$, nicht-leere, endliche Menge von Zuständen.
- $\Sigma$, nicht-leere, endliche Menge von Eingabezeichen
- $\delta : S \times \Sigma \rightarrow S$, Zustandsübergangsfunktion
- $s_0 \in S$, Startzustand
- $F \subseteq S$, Menge der akzeptierenden Endzustände

Der Automat akzeptiert $w \in \Sigma^*$ genau dann, wenn $\delta^*(s_0, w) \in F$
mit $\delta^*(q, \epsilon) = q$ und $\delta^*(q, a \circ w') = \delta^*(\delta(q, a), w')$.

## Grundlagen: Mealy-Automat

Eine Mealy-Maschine ist ein 7-Tupel $(S, \Sigma, \Omega, \delta, \lambda, s_0, F)$:

- $S$, nicht-leere, endliche Menge von Zuständen.
- $\Sigma$, nicht-leere, endliche Menge von Eingabezeichen
- $\Omega$, nicht-leere, endliche Menge von Ausgabezeichen
- $\delta : S \times \Sigma \rightarrow S$, Zustandsübergangsfunktion
- $\lambda : S \times \Sigma \rightarrow \Omega$, Ausgabefunktion
- $s_0 \in S$, Startzustand
- $F \subseteq S$, Menge der akzeptierenden Endzustände



## Grundlagen: Moore-Automat

Eine Moore-Maschine ist ein 7-Tupel $(S, \Sigma, \Omega, \delta, \lambda, s_0, F)$:

- $S$, nicht-leere, endliche Menge von Zuständen.
- $\Sigma$, nicht-leere, endliche Menge von Eingabezeichen
- $\Omega$, nicht-leere, endliche Menge von Ausgabezeichen
- $\delta : S \times \Sigma \rightarrow S$, Zustandsübergangsfunktion
- $\lambda : S \rightarrow \Omega$, Ausgabefunktion
- $s_0 \in S$, Startzustand
- $F \subseteq S$, Menge der akzeptierenden Endzustände



## Motivation und Abgrenzung

Zustandsmodellierung:

- Motivation: Modellierung von Systemverhalten
  - Sequentielle Programme
  - Interaktive/reaktive Systeme
- Ansatz: Zustandsbasierte Modellierung
- Kernkonzepte: Zustand, Übergang, Interaktion
- Abgrenzung: Automatentheorie
  - Fokus: Verhaltensmodellierung vs. Wortschatzbeschreibung
  - Erweiterung: Modellierung interaktiven, nebenläufigen Verhaltens



## Ausführungsmodellierung

**Ziel**: Modellierung des Verhaltens von Berechnungen

- Adressiert Aspekte zur Darstellung von Berechnungsfolgen
- Beschreibt Aspekte unabhängig von einer spezifischen Programmiersprache
- Abstrahiert von Aspekten wie Ausführungsdauer oder Speichergröße

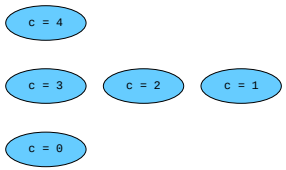
**Konzepte**: Zustands, Zustandsübergang, Ausführung, Verhalten

**Modelle**: Zustandsübergangssysteme

## Zustand

```
int c = 4;

while(c>0) do
  c = c - 1;
```




Ansatz: Modellierung der Berechnungseffekte (Programmschritte)

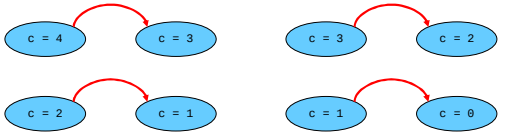
Konzept: Zustand = Menge von Belegungen

- Datenzustand, z.B. Variablen
- Kontrollzustand, z.B. Programmzähler

Beispiel: { c = 3 }, { c = 0 }



## Zustandsübergang




Ansatz: Modellierung des Verhaltens eines (sequentiellen) Programms

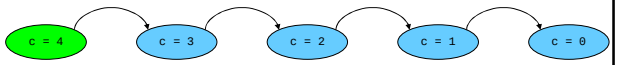
Konzept: Übergang = Zustandsrelation (Zustandspaar)

- Beschreibt den Wechsel zwischen Programmzuständen
- Entspricht einer atomaren Aktion des Programms

Beispiel: { c = 3 } → { c = 2 } or ({ c = 3 },{ c = 2 })



## Zustandsübergangssystem




Ansatz: Verhaltensbeschreibung eines (terminierenden) Programms

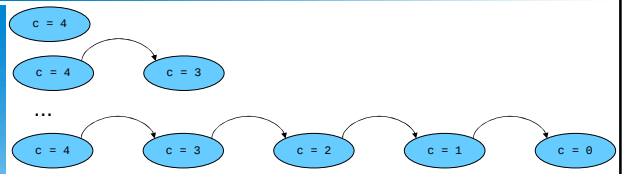
Modell: Zustandsübergangssystem $(S,s_0,T)$

- Menge S möglicher Programmzuständ
- Initialer Zustand $s_0 \subseteq S$ (Variante: Menge von Anfangszuständen $S_0$)
- Menge möglicher Übergänge $T \subseteq S \times S$

- Beispiel:
  - $S = \{ \{c = 4\}, \{c = 3\}, \{c = 2\}, \{c = 1\}, \{c = 0\} \}$
  - $s_0 = \{c = 4\}$
  - $T = \{ (\{c = 4\},\{c = 3\}), (\{c = 3\},\{c = 2\}),...,(\{c = 1\}, \{c = 0\}) \}$



## Ausführung



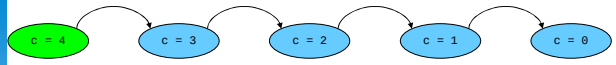

Ansatz: Modellierung einer (teilweisen) Ausführung eines Programms

Konzept: Spur = Sequenz aufeinander folgender Zustände $s_1 \bullet s_2 \bullet s_3 \bullet s_4 \bullet \ldots$

- Erster Zustand ist initialer Zustand: $s_1 = s_0$
- Aufeinander folgende Zustände definiert per Transitionen: $(s_i,s_{i+1}) \in T$

Beispiel: { c = 4 } • { c = 3 } • { c = 2 }

## Erreichbarkeit




Ansatz: Charakterisierung der Zustände eines Programms

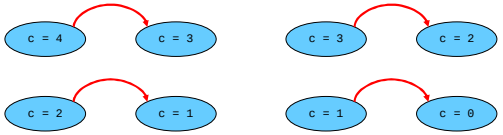
Konzept: $s_i$ erreichbar = Teil einer Spur $s_0 \bullet s_1 \bullet s_2 \bullet s_3 \bullet s_4 \bullet \ldots \bullet s_i$

Alternativ: $s_i$ erreichbar = $(s_0,s_i) \in T^*$

Beispiel: T = { ({c = 4},{c = 3}), ({c = 4},{c = 2}),...,({c = 3}, {c = 2}) }),
({c = 3},{c = 1}),...,({c = 0}, {c = 0}) }



## (Nicht-)Terminierung




Ansatz: Identifikation (potentiell) unendlicher Berechnungen

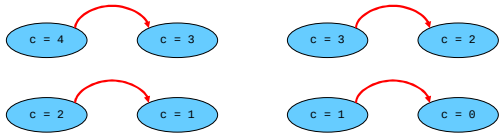
Konzept: Zustandsübergangssystem $(S,s_0,T)$ partiell bzw. total

- Total: Jeder Zustand hat einen Nachfolger: { s | (s,s') ∈ T } = S
- Partiell: Es gibt Zustände ohne Nachfolger: { s | (s,s') ∈ T } ≠ S

Beispiel: {c = 0} ist nachfolgeloser Zustand



## (Nicht)Determinismus



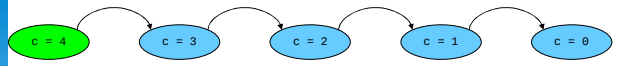

Ansatz: Modellierung des Verhaltens eines (sequentiellen) Programms

Konzept: Zustandsübergangssystem $(S,s_0,T)$ deterministisch bzw. nichtdeterministisch

- Deterministisch: Max. einen Nachfolger: (s,s'), (s,s'') ∈ T ⇒ s' = s''
- Nichtdeterministisch: Es gibt Zustände mit mehreren Nachfolgern



## Zusammenfassung




Konzepte:

- Zustand: Beobachtung eines Systems zu einem Zeitpunkt
- Zustandsübergang: Atomare, den Zustand ändernde Aktion
- Übergangsrelation: Menge möglicher Aktionen eines Programms
- Spur: Sequenz von Zuständen während einer Ausführung

Modell: Transitionssystem $(S,s_0,T)$

## Interaktionsmodellierung

**Ziel**: Modellierung des Verhaltens von interaktiven Systemen
- Addressiert Aspekte der Darstellung von Interaktionen
- Beschreibt Aspekte unabhängig von spezifischen Interaktionsschnittstellen
- Abstrahiert von Aspekten wie Berechnungsdauer und Schnittstellenbreite

**Konzepte**: Interaktion, Beobachtung, Verhalten

**Modell**: Markierte Transitionssysteme



## Interaktion

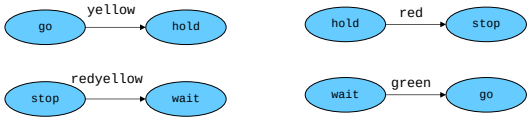
Ansatz: Modellierung der Interaktionen zwischen System und Umgebung

Konzept: Interaktion = gemeiname Aktion an der Systemschnittstelle
- Einache Interaktion (atomar und unmittelbar)
- Komplexe Interaktion (Kombination of atomarer Interaktionen)

Beispiel: “Ampel schaltet auf Rot”, “Ampel schaltet auf Grün”



## Markierte Transition




Ansatz: Modellierung der Aktionen eines (reaktiven) Systems

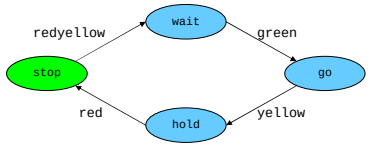
Konzept: Markierte Transition = Interaktion und Zustandsänderung
- Beschreibt den Übergang zwischen Zuständen des Systems
- Beschreibt die dem Übergang entsprechende Interaktion an der Systemschnittstelle

Beispiel: $go \xrightarrow{yellow} hold$ $wait \xrightarrow{green} go$

or (go,yellow,hold), (wait,green,go)



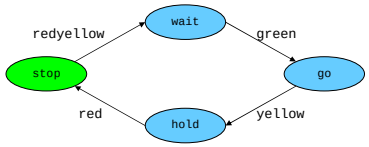
## Markiertes Transitionssystem




Ansatz: Modellierung des Verhaltens eines reaktiven Systems

Modell: Markiertes Transitionssystem $(S,A,S_0,T)$
- Menge der möglichen Systemzustände S
- Menge der möglichen einfachen Interaktionen A (Alphabet)
- Menge möglicher Startzustände $S_0 \subseteq S$
- Menge möglicher Transitionen $T \subseteq S \times A \times S$

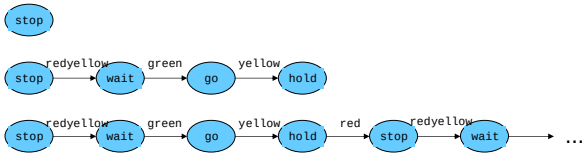
## Markiertes Transitionssystem




- Beispiel:
  - S = { stop, wait, go, hold }
  - A = { green, yellow, yellowred, red }
  - $S_0$ = { stop }
  - T = { (stop,redyellow,wait), (wait,green,go), (go,yellow,hold), (hold,red,stop) }
  - Or: $T = \{stop \xrightarrow{redyellow} wait,\ wait \xrightarrow{green} go,\ go \xrightarrow{yellow} hold,\ hold \xrightarrow{red} stop\}$



## Ablauf




Ansatz: Modellierung eines (partiellen) Ablauf eines Programms

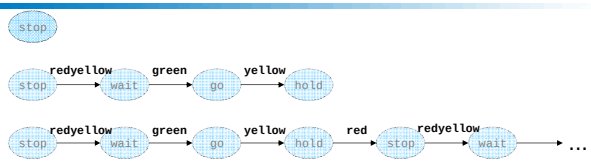
Konzept: Spur = (endliche/unendliche) Folge aufeinander folgender Zustände und zugehöriger Interaktionen $s_1 \bullet a_1 \bullet s_2 \bullet a_2 \bullet s_3 \bullet a_3 \bullet s_4 \bullet \ldots$

- Erster Zustand ist ein Startzustand: $s_1 \in S_0$
- Folgezustände verknüpft über Transitionen: $(s_i, a_i, s_{i+1}) \in T$

Beispiel: stop • redyellow • wait • green • go



## Beobachtung




Ansatz: Modellierung eines (partiellen) Verhaltens eines Systems an seiner Schnittstelle

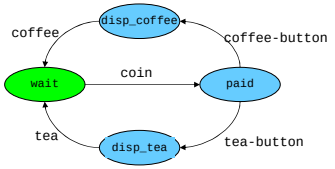
Konzept: Spur = (endliche/unendliche) Sequenz aufeinander folgender Interaktionen a1 • a2 • a3 • ... eines Ablaufs $s_1 \bullet a_1 \bullet s_2 \bullet a_2 \bullet s_3 \bullet a_3 \bullet s_4 \bullet \ldots$

- Erster Zustands ist Initialzustand: s1 ∈ I
- Folgezustände verknüpft über Transitionen: $(s_i, a_i, s_{i+1}) \in T$

Beispiel: redyellow • green • yellow • red • redyellow • green • yellow • ....



## Auswahl



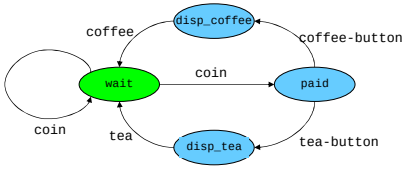

Ansatz: Modellierung kontrollierbarer Verhaltensalternativen eines reaktiven Systems

Konzept: (Deterministische) Auswahl im Zustand s

- $(s, a_1, s_1) \in T$
- $(s, a_2, s_2) \in T$

Beispiel: (paid,coffee-button,disp_coffee), (paid,tea-button,disp_tea) ∈ T

## Nichtdeterminismus




Ansatz: Modellierung nicht kontrollierbarer Verhaltensalternativen eines reaktiven Systems

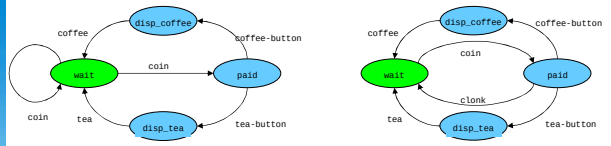
Konzept: Nichtdeterministische Auswahl im Zustand s

- $(s, a, s_1) \in T$
- $(s, a, s_2) \in T$

Beispiel: (wait,coin,paid), (wait,coin,wait) ∈ T



## (Nicht)deterministische Auswahl




Alternatives Verhalten eines Systems:

- Determ. Auswahl: Kontrolliert durch die Umgebung (Nutzer)
- Nichtdeterm. Auswahl: Kontrolliert durch das System

Varianten: Wer kontrolliert eine Interaktion?

- Symmetrisch: Gemeinsame Interaktion
- Asymmetrisch: Unterscheidung zwischen Ein-/Ausgabe



## Eingabe und Ausgabe

Ansatz: Trrennung zwischen System- und Umgebungsaktionen

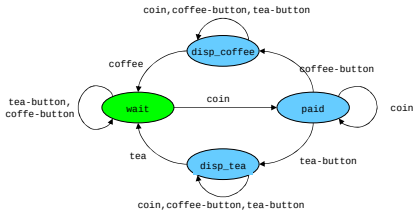
Konzept: Alphabet A von Aktionen, getrennt nach Ein- und Ausgabe

- Eingabe I: Von der Umgebung kontrollierte Aktionen
- Ausgabe O: Vom System kontrollierte Aktionen

Beispiel: I = {coin, coffee-button, tea-button}, O = {coffee, tea}, A = I ∪ O



## Eingabebereitschaft



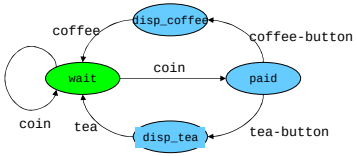

Ansatz: Unterscheidung zwischen Eingabe und Ausgabe

Konzept: Eingabebereitschaft von (S,s,I,O,T)

- Das System ist immer bereit, eine Eingabe anzunehmen
- Für beliebige s, i: $(s,i,s') \in T$

Beispiel: {(paid,coin,paid),(paid,coffee-button,disp-coffee),(paid,tea-button,disp_tea)} ⊆ T

## Interaktionsmodellierung




Konzepte:

- Interaktion (Alphabet): Gemeinsame Aktionen von System/Umgebung
- Beobachtung: Spur von Interaktionen
- Auswahl: Verhaltensalternativen eines Systems
- Nichtdeterminismus: Systemkontrollierte Auswahl

Modell: Markiertes Transitionssystem $(S,A,S_0,T)$



## Nebenläufigkeitsmodellierung

**Ziel**: Modellierung des Verhaltens von nebenläufigen reaktiven Systemen

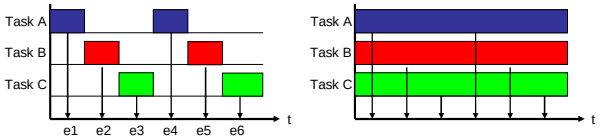
- Addressiert Aspekte der Synchronisierung
- Beschreibt Aspekte unabhängig von spezifischen Kommunikations/Synchronisierungstechniken
- Abstrahiert von Aspekten wie Schnittstellenbreite, Ausführungsgeschwindigkeit

**Konzepte**: Gemeinsame/unabhängige Interaktion, gemeinsame/unabhängige Transition, Nebenläufigkeit

**Modell**: Synchronisierte Transitionssysteme



## Nebenläufig und Parallel




Ansatz: Modellierung gleichzeitiger Abläufe in einem System

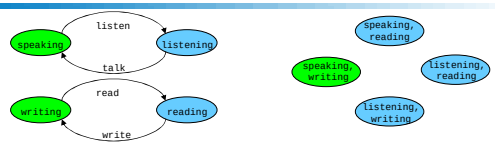
Konzept: Simultane Ausführung

- Nebenläufig: Logisch simultan (simultan bei sequentieller Beobachtung)
- Parallel: Technisch simultan (simultan bei gleichzeitiger Beobachtung)

Beispiel: Pre-emptives Multitasking vs. Multiprozessorausführung



## Nebenläufiges System




Ansatz: Modellierung nebenläufiger Systemausführungen

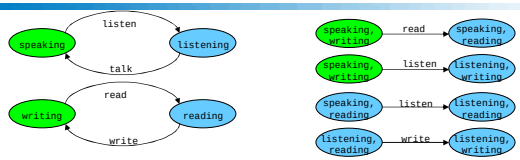
Konzept: Produkt = Parallele Zustände von $(S,A,S_0,T)$ und $(S',A',S'_0,T')$

- Kombination aller Zustände jedes MTS: Produktraum $S \times S'$
- Startzustände: $S_0 \times S'_0$

Beispiel: ({speaking,listening},A,{speaking},T) und ({writing,reading},A',{writing},T')

- Produktraum: {speaking,listening} × {reading,writing} = { (speaking,writing),(speaking,reading),(listening,writing),(listening,reading)}
- Startzustand = {speaking} × {writing} = {(speaking,writing)}

## Unabhängige Interaktion




Ansatz: Modellierung unabhängiger Interaktionen von Systemen

Konzept: Nichtsynchronisierte Transition von $(S,A,S_0,T)$ and $(S',A',S'_0,T')$

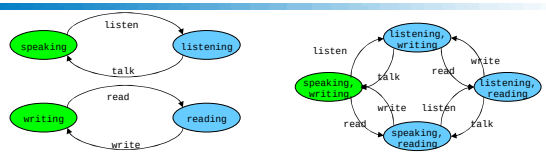
- $((s,s'),a,(t,s'))$ für alle $(s,a,t) \in T$ und $a \in A \setminus A'$
- $((s,s'),a',(s,t'))$ für alle $(s',a',t') \in T'$, and $a' \in A' \setminus A$

Beispiel:

- ((speaking,writing),read,(speaking,reading))
- ((listening,reading),talk,(speaking,reading))



## Unsynchronisierter Produktautomat



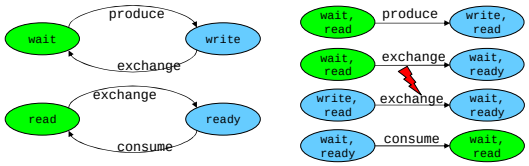

Ansatz: Modellierung von unabhängiger Nebenläufigkeit

Modell: Produktautomat von $(S,A,S_0,T)$ and $(S',A',S'_0,T')$ mit $A \cap A' = \emptyset$

- Zustände: $S \times S'$
- Alphabet: $A \cup A'$
- Startzustände: $S_0 \times S'_0$
- Transitionsrelation $T||T'$:
  - $((s,s'),a,(t,s'))$ für $(s,a,t) \in T$ und $a \in A$
  - $((s,s'),a',(s,t'))$ für $(s',a',t') \in T'$ und $a' \in A'$



## Synchronisierte Interaktion




Ansatz: Modellierung synchronisierter Interaktionen zwischen Systemen

Konzept: Synchronisierte Transitionen von $(S,A,S_0,T)$ und $(S',A',S'_0,T')$

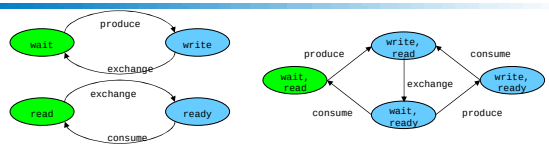
- $((s,s'),a,(t,t'))$ für $(s,a,t) \in T$, $(s',a,t') \in T'$ und $a \in A \cap A'$

Beispiel:

- (write,read),exchange,(wait,ready)



## Synchronisierter Produktautomat

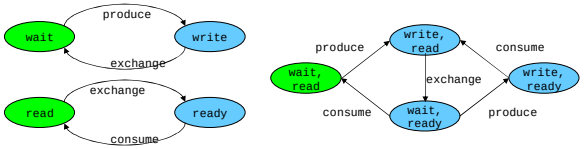



Ansatz: Modellierung von synchronisierter Nebenläufigkeit

Model: Produktautomat von $(S,A,S_0,T)$ und $(S',A',S'_0,T')$ mit $A \cap A' \neq \emptyset$

- Zustände: $S \times S'$
- Alphabet: $A \cup A'$
- Startzustände: $S_0 \times S'_0$
- Transition relation $T||T'$:
  - $((s,s'),a,(t,s'))$ für $(s,a,t) \in T$ und $a \in A \setminus A'$
  - $((s,s'),a',(s,t'))$ für $(s',a',t') \in T'$ und $a' \in A' \setminus A$
  - $((s,s'),a,(t,t'))$ für $(s,a,t) \in T$, $(s',a,t') \in T$ und $a \in A \cap A'$

## Nebenläufigkeitsmodellierung




Konzepte:

- Nebenläufige Ausführung: Synchronized alternative execution
- Unabhängige Interaktion: Verschränkte Ausführung
- Synchronisierte Interaktion: Simultane Ausführung

Modell: Synchronisierter Produktautomat



## Kommunikationsmodellierung

**Ziel**: Beschreibung von kommunizierenden Systemen

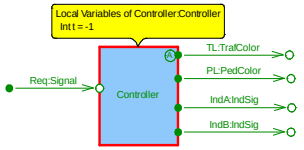
- Unterscheidung zwischen System und Umgebung
- Erweiterte Modellierung mit Ein- und Ausgabe

**Konzepte**: Signal, zeitsynchrone Kommunikation, nachrichtensynchrone Kommunikation

**Modell**: Synchronisierte Erweitere Zustandsübergangssysteme



## Daten- und Kontrollzustand




Ausführungsmodell: $S \subseteq L \times V_1 \times ... \times V_k \times I_1 \times ... \times I_m \times O_1 \times ... \times O_n$

- Kontrollzustand L (e.g., Init, Green, Yellow, Red, RedYellow)
- Variablen $V_1,...,V_k$ (e.g., t:Int)
- Eingabe: $I_1,...,I_m$ (e.g., Req:Signal ∪ { - } )
- Ausgabe: $O_1,...,O_n$ (e.g., TL:TrafColor ∪ { - })



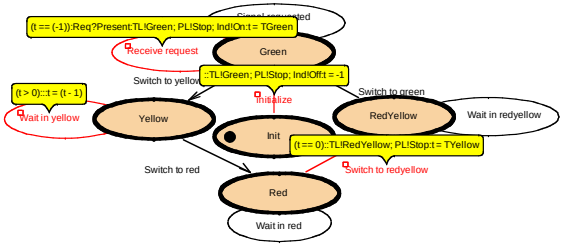
## Erweiterte Transition


| Pre-State | | | Post-State | | | | |
|---|---|---|---|---|---|---|---|
| State | t | Req | TL | PL | Ind | t | State |
| Green | -1 | Present | Green | Stop | On | TGreen | Green |

Model: Extended Transition System

- Vorgängerzustand: Kontrollzustand, Datenzustand, Eingabe
- Nachfolgerzustand: Ausgabe, Datenzustand, Kontrollzustand

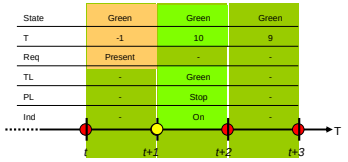
## Beispiel: Erweiterte Transition




| Pre-State | | | Post-State | | | | |
|---|---|---|---|---|---|---|---|
| State | t | Req | TL | PL | Ind | t | State |
| Init | | | Green | Stop | Off | -1 | Green |
| Green | -1 | Present | Green | Stop | On | TGreen | Green |
| Yellow | >0 | | - | - | - | t-1 | Yellow |
| Red | 0 | | RedYellow | Stop | - | TYellow | RedYellow |



## Beispiel: Erweiterte Transition





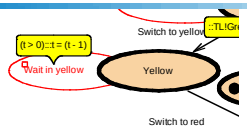
| Pre-State | | | Post-State | | | | |
|---|---|---|---|---|---|---|---|
| State | t | Req | TL | PL | Ind | t | State |
| Green | -1 | Present | Green | Stop | On | TGreen | Green |

Transitionsausführung: Anwendung der Pre-State/Post-State Relation

- Pre-State: Kontrollzustand, Datenzustand, Eingabe
- Post-State: Ausgabe, Datenzustand, Kontrollzustand



## Eingabebereitschaft



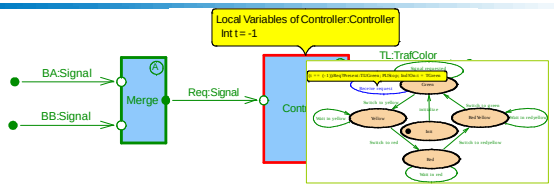

| Pre-State | | | Post-State | | | | |
|---|---|---|---|---|---|---|---|
| State | t | Req | TL | PL | Ind | t | State |
| Yellow | >0 | - | - | - | - | t-1 | Yellow |
| Yellow | ==0 | - | Red | Go | Off | 0 | Red |
| **Yellow** | **>0** | **Present** | **Yellow** | **-** | **-** | **T-1** | **Yellow** |

Eingabebereitschaft:

- System blockiert keine Eingaben der Umgebung
- Transitionsrelation ist für alle Pre-States definiert



## Synchronisiertes Übergangssystem

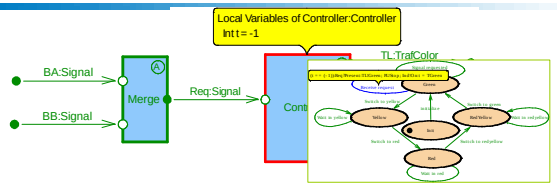



Synchronisiertes Übergangssystem:

- $S_1 \subseteq L_1 \times V_{1,1} \times \ldots \times V_{1,k} \times I_{1,1} \times \ldots \times I_{1,m} \times O_{1,1} \times \ldots \times O_{1,n}$
- $S_2 \subseteq L_2 \times V_{2,1} \times \ldots \times V_{2,r} \times I_{2,1} \times \ldots \times I_{2,s} \times O_{2,1} \times \ldots \times O_{2,t}$
- Ein-/Ausgabe: $I_{1,1} = O_{2,1}, \ldots, I_{1,v} = O_{2,v}$, $I_{2,1} = O_{1,1}, \ldots, I_{2,w} = O_{1,w}$
- Produktraum $S \subseteq$

$$L_1 \times L_2 \times V_{2,1} \times \ldots \times V_{2,k} \times V_{2,1} \times \ldots \times V_{2,r} \times O_{1,1} \times \ldots \times O_{1,w} \times O_{2,1} \times \ldots \times O_{2,v} \times I_{1,v+1} \times \ldots \times I_{1,m} \times I_{2,w+1} \times \ldots \times I_{2,s} \times O_{1,w+1} \times \ldots \times O_{1,n} \times O_{2,v} \times \ldots \times O_{2,n}$$

## Synchronisiertes Übergangssystem




Synchronisiertes Übergangssystem:

- Synchronisierte Transitionen von $(S_1,S_{1,0},T_1)$ und $(S_2,S_{2,0},T_2)$
- $((l_1,l_2,v_1,v_2,h_1,h_2,i_1,i_2,o_1,o_2),(l'_1,l'_2,v'_1,v'_2,h'_1,h'_2,i'_1,i'_2,o'_1,o'_2))$ für
  - $((l_1,v_1,i_1,h_1,o_1),(l'_1,v'_1,'_1,h'_1,o'_1)) \in T_1$
  - $((l_2,v_2,i_2,h_2,o_2),(l'_2,v'_2,'_2,h'_2,o'_2)) \in T_2$



## Literatur


- Manfred Broy, Ralf Steinbrüggen: Modellbildung in der Informatik, Springer-Verlag, Berlin, 2004
- David Harel: Statecharts: A visual approach to complex systems, CS84-05, Department of Applied Mathematics, The Weizmann Institute of Science, 1984
- Jeff Magee, Jeff Kramer: Concurrency – State Models & Java Programs, John Wiley & Sons Ltd., Chichester, 1999
- G. H. Mealy: *A Method for Synthesizing Sequential Circuits*, Bell System Tech. J. 34, pp. 1045–1079, September 1955
- Nancy Lynch, Frits Vaandrager. Forward and backward simulations for timing-based systems. In J. W. de Bakker, W. P. de Roever, C. Huizing, and G. Rozenberg, editors, Proceedings of Real-Time: Theory in Practice (REX Workshop, Mook, The Netherlands, June 1991), volume 600 of Lecture Notes in Computer Science, pages 397-446. Springer-Verlag 1992.



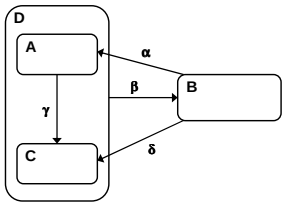

## Statecharts von David Harel / UML

- Visueller Formalismus für Zustandsmodellierung
- Konzepte:
  - Clustern von Zuständen / Zustandshierarchie
  - Verfeinerung
  - Allgemeine Transitionen und Reaktionen
    - Ereignisse
    - Bedingungen
    - Aktionen
    - Aktivitäten
  - Unabhängigkeit / Orthogonalität

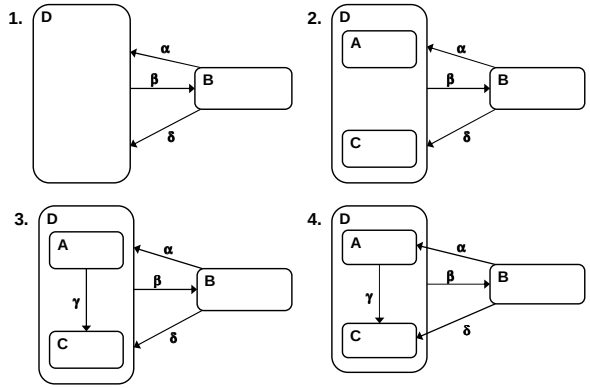


## Cluster / Hierarchie

- Zusammenfassen von
  - Sich ausschließenden Zuständen (XOR Semantik)
  - Transitionen von Unterzuständen
- Beispiel
  - D ist XOR-Zustand von A und C
  - β führt von A zu B, aber auch von C zu B

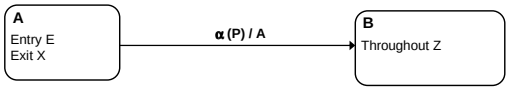


## Verfeinerung / Top-Down Definition






## Allgemeine Transition

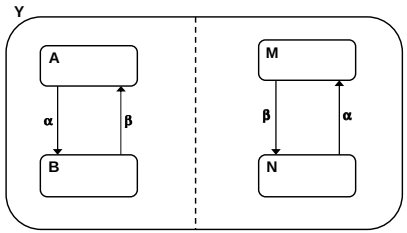



- Ereignis α: Transitionsauslöser
- Bedingung P: Einschränkungen der Transition
- Aktion A: Transitionsausgabe
- Aktivität E: Eingangsaktivität
- Aktivität X: Ausgangsaktivität
- Aktivität Z: Zustandsaktivität



## Unabhängigkeit / Orthogonalität

- Zusammenfassen von
  - Parallelen Zuständen (AND-Semantik)






## Statecharts Beispiel: Armbanduhr