**Script** **generated by TTT**

Title: Grundlagen_Betriebssysteme (02.11.2012)

Date: Fri Nov 02 08:30:47 CET 2012

Duration: 91:28 min

Pages: 27

# Modellierung paralleler Systeme

In diesem Abschnitt behandeln wir grundlegende Mechanismen zur Modellierung und Beschreibung parallel ablaufender Systeme.





***Ziel*** ist die einfache Analyse und Beschreibung von parallelen Systemen.

Spezifikation eines Modells im Sinne von Abstraktion: Vereinfachung, Weglassen von Details, Beschränken auf interessierende Eigenschaften.

Beispiele interessanter Eigenschaften sind:

- Determiniertheit.
- Störungsfreiheit.
- wechselseitiger Ausschluss.
- Endloses Blockieren (engl. Deadlock).
- Verhungern (engl. Starvation).

Die Eigenschaften können in zwei Klassen eingeteilt werden.

***Sicherheitseigenschaften*** (engl. safety): Sicherstellen, dass gewisse unerwünschte Zustände und unerwünschte Aktionsverläufe nicht auftreten; Formulierung durch Invarianten (z.B. wechselseitig ausgeschlossene Nutzung der CPU).

***Lebendigkeitseigenschaften*** (engl. liveness): Garantieren, dass gewisse erwünschte Zustände bzw. Aktionsmuster in den Abläufen irgendwann auch auftreten (z.B. erwünschter Ablauf: rechenbereiter Prozess macht irgendwann auch Fortschritte).



Beschreibung der Eigenschaften eines dynamischen Systems und deren zeitliche Veränderungen. Die Verhaltensbeschreibung setzt sich aus den Basiseinheiten Aktionen und Zustände zusammen.

Eigenschaftenveränderungen sind Ausführungen von elementaren Aktionen. Es sind zwei Sichten möglich.

Die Sicht auf die Tätigkeiten, die auszuführen sind; in dieser Sicht sind die Basiseinheiten die atomaren ***Aktionen*** .

Die Sicht auf die Veränderungen, die erfolgen und beobachtet werden können; in dieser Sicht sind die Basiseinheiten die atomaren ***Ereignisse*** , die eintreten.

Aktionen bzw. Ereignisse sind zeitlich geordnet.

Zustände des Systems anhand der Werte der Datenobjekte. Folge von Zuständen gemäß der Zeitachse.

**Spuren**

Es sind zwei Varianten von Verhaltensbeschreibungen zweckmässig.

1. ***Ereignisspuren*** : Sie beschreiben den zeitlichen Ablauf der Berechnungen eines Systems mit Ereignissen, die linear geordnet sind.
2. ***Zustandsspuren*** : Sie beschreiben den zeitlichen Ablauf der Berechnungen eines Systems anhand der auftretenden Zustände der Datenobjekte.

# Modellierung paralleler Systeme

In diesem Abschnitt behandeln wir grundlegende Mechanismen zur Modellierung und Beschreibung parallel ablaufender Systeme.





# Prozess

Gegeben seien eine Menge (das "Universum") $E^*$ von ***Ereignissen*** (engl. events) und eine Menge A von ***Aktionen*** (engl. actions).

**Definition**

Ein Triple $p = (E, \leq, \alpha)$ nennen wir einen ***Prozess*** oder auch eine ***Aktionsstruktur***, falls folgende Aussagen gelten:

$E \subseteq E^*$, E heißt die Ereignismenge.

$\leq$ ist eine partielle Ordnung über E, $\leq$ ist die Kausalitätsrelation.

$\alpha$: $E \rightarrow A$ ist die Aktionsmarkierung des Prozesses p.

Die Abbildung $\alpha$ ordnet jedem Ereignis eine Aktion zu. Leerer Prozess = Aktionsstruktur mit leerer Ereignismenge.





Wir betrachten einen Prozess mit 14 Ereignissen. Jedem Ereignis werden Aktionen zugeordnet.

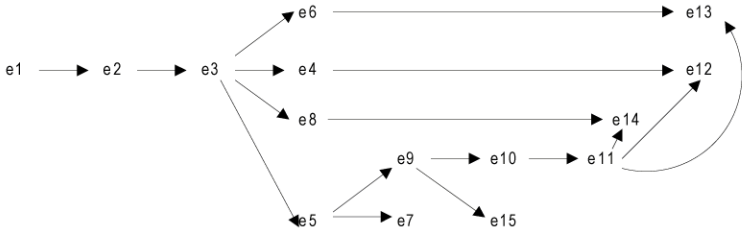
| Ereignis | Aktion |
|---|---|
| e1 | Knopf wird gedrückt |
| e2 | Ampel für Autofahrer schaltet auf Gelb |
| e3 | Ampel für Autofahrer schaltet auf Rot |
| e4 | Auto hält auf Spur 1 |
| e5 | Ampel für Fußgänger schaltet auf Grün |
| e6 | Auto hält auf Spur 2 |
| e7 | Fußgänger benutzt Fußgängerübergang |
| e8 | Auto hält auf Spur 3 |
| e9 | Ampel für Fußgänger schaltet auf Rot |
| e10 | Ampel für Autofahrer schaltet auf Rot und Gelb |
| e11 | Ampel für Autofahrer schaltet auf Grün |
| e12 | Auto fährt an auf Spur 1 |
| e13 | Auto fährt an auf Spur 2 |
| e14 | Auto fährt an auf Spur 3 |
| e15 | Knopf wird gedrückt |

Die Relation der kausalen Abhängigkeiten sei gegeben durch:

$e1 < e2$, $e2 < e3$, $e3 < e4$, $e3 < e5$, $e3 < e6$, $e5 < e7$

# Graphische Darstellung

Endliche Prozesse lassen sich anschaulich durch knotenmarkierte, gerichtete, zyklenfreie Graphen darstellen, Die Knoten repräsentieren die Ereignisse und sind durch die zugeordneten Aktionen markiert.



## Charakterisierung von Prozessen

Ereignisse haben eine eindeutige Identität.

**Parallel, nebenläufig**

Für einen Prozess p = (E, ≤, α) nennen wir zwei Ereignisse e1, e2 ∈ E ***parallel*** oder ***nebenläufig*** (engl. concurrent), falls sie im Prozess p nicht in einer kausalen Relation stehen, d.h. wenn gilt:

¬(e1 ≤ e2 oder e2 ≤ e1)

Parallele Ereignisse sind ***kausal unabhängig*** , sie können zeitlich nebeneinander oder in beliebiger Reihenfolge stattfinden.

**Sequentiell**

Ein Prozess p = (E, ≤, α) heißt ***sequentiell*** , wenn in ihm kein Paar von parallelen Ereignissen auftritt, d.h. die Kausalitätsrelation ≤ eine lineare Ordnung ist.

Der Ablauf eines Programmes kann auch als Prozess im Sinne der Aktionsstrukturen gedeutet werden. Sequentielle Programme erzeugen sequentielle Prozesse.

**Endlich**

Ein Prozess p = (E, ≤, α) heißt ***endlich*** , falls seine Ereignismenge eine endliche Menge ist.



## Kausale Abhängigkeiten

Unterschiedliche Ursachen für die kausale Beziehung e ≤ d.

**Echt kausale Beziehungen**

das Ereignis e ist kausal für das Ereignis d in dem Sinn, dass d ohne e niemals auftreten kann.

**Beispiel**

e = Geldeinwurf, d = Kartenausgabe; e ≤ d, d.h. der Fahrkartenautomat gibt erst dann eine Fahrkarte aus, wenn der passende Geldbetrag eingeworfen wurde.

**zeitliche Beziehungen**

das Ereignis e endet, bevor das Ereignis d beginnt.

Beispiel: Nachricht muss gesendet sein, bevor sie empfangen werden kann.

Happend-before

**Systembeschränkungen**

Systembeschränkungen, z.B. wechselseitiger Ausschluss. Das Ereignis e darf nicht parallel zum Ereignis d auftreten.

**Beispiel**

e = Fußgänger überquert die Fahrbahn beim Übergang, und

d = Auto überfährt den Übergang.

e und d dürfen nicht parallel ausgeführt werden, aber es besteht keine echte kausale Abhängigkeit zwischen e und d.



## Happend-before

Die kausale Beziehung impliziert also eine ***zeitliche Relation*** ; das ist die bekannte ***happened before*** -Beziehung von L. Lamport. Die Umkehrung gilt nicht. Die zeitliche Relation zwischen Ereignissen sagt nichts über die Kausalität aus.

Wichtig: Ereignisse wurden als atomare Ereignisse modelliert. D.h. Beginn und Ende fallen auf einen Zeitpunkt zusammen.

Die Happend-before Relation spielt im Kontext von Verteilten Anwendungen eine wichtige Rolle (siehe Vorlesung Verteilte Anwendungen ).



## Kausale Abhängigkeiten

Unterschiedliche Ursachen für die kausale Beziehung e ≤ d.

**Echt kausale Beziehungen**

das Ereignis e ist kausal für das Ereignis d in dem Sinn, dass d ohne e niemals auftreten kann.

**Beispiel**

e = Geldeinwurf, d = Kartenausgabe; e ≤ d, d.h. der Fahrkartenautomat gibt erst dann eine Fahrkarte aus, wenn der passende Geldbetrag eingeworfen wurde.

**zeitliche Beziehungen**

das Ereignis e endet, bevor das Ereignis d beginnt.

Beispiel: Nachricht muss gesendet sein, bevor sie empfangen werden kann.

Happend-before

**Systembeschränkungen**

Systembeschränkungen, z.B. wechselseitiger Ausschluss. Das Ereignis e darf nicht parallel zum Ereignis d auftreten.

**Beispiel**

e = Fußgänger überquert die Fahrbahn beim Übergang, und

d = Auto überfährt den Übergang.

e und d dürfen nicht parallel ausgeführt werden, aber es besteht keine echte kausale Abhängigkeit zwischen e und d.

Generated by Targeteam

# Sequentialisierung

Idee: vereinfachte Darstellung paralleler Abläufe, aus der Sicht eines Beobachters. Vollständige Sequentialisierung: partielle Kausalitätsordnung zu linearer Ordnung ergänzen.

**Definition**

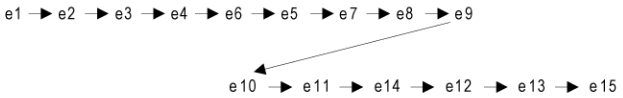
Sequentieller Beobachter hält Ereignisse bzw. Aktionen in einem sequentiellen Ablaufprotokoll fest ⇒ Sequentialisierung eines Prozesses mit parallelen Aktionen.

Jeder endliche Prozess p ist durch die Menge ihrer vollständigen Sequentialisierungen eindeutig bestimmt.

Darstellung einer Programmausführung als Prozess.

**Beispiel**

Für das Beispiel des Fußgängerübergangs ist folgendes eine vollständige Sequentialisierung.




**Spuren**



# Definition

Ein Prozess $p_1 = (E_1, \leq_1, \alpha_1)$ heißt eine Sequentialisierung eines Prozesses $p_2 = (E_2, \leq_2, \alpha_2)$, falls gilt:

$E_1 = E_2$

$\forall e, d \in E_1 : e \leq_2 d \Rightarrow e \leq_1 d$

$\alpha_1 = \alpha_2$

Ist $p_1$ sequentiell, so heißt die Sequentialisierung ***vollständig*** .



# Sequentialisierung

Idee: vereinfachte Darstellung paralleler Abläufe, aus der Sicht eines Beobachters. Vollständige Sequentialisierung: partielle Kausalitätsordnung zu linearer Ordnung ergänzen.

**Definition**

Sequentieller Beobachter hält Ereignisse bzw. Aktionen in einem sequentiellen Ablaufprotokoll fest ⇒ Sequentialisierung eines Prozesses mit parallelen Aktionen.

Jeder endliche Prozess p ist durch die Menge ihrer vollständigen Sequentialisierungen eindeutig bestimmt.

Darstellung einer Programmausführung als Prozess.

**Beispiel**

Für das Beispiel des Fußgängerübergangs ist folgendes eine vollständige Sequentialisierung.

e1 e2 e3 e4 e6 e5 e7 e8 e9
e10 e11 e14 e12 e13 e15

**Spuren**

Generated by Targeteam

Darstellung sequentieller Prozesse mittels Spuren.

Sei A eine Menge von Aktionen, dann bezeichnen wir mit $A^+$ die Menge der endlichen Folgen von Aktionen aus A und mit $A^\infty$ die Menge der unendlichen Folgen von Aktionen.

Jedem sequentiellen Prozess können wir eindeutig eine Folge von Aktionen zuordnen, wie sprechen von der ***Spur*** (engl. trace). Sei $p = (E, \leq, \alpha)$ ein sequentieller Prozess. Wir definieren:

$spur : \{p \mid p \text{ ist sequentieller Prozess}\} \rightarrow A^+ \cup A^\infty$

$spur(p) = empty$, falls $E = \emptyset$

$spur(p) = \langle a \rangle \cdot spur(p \mid E \setminus \{e\})$, falls $E \neq \emptyset$, wobei e das gemäß der Kausalitätsordnung kleinste Ereignis in p ist und $\alpha(e) = a$ gilt.

$p \mid E \setminus \{e\}$ bezeichnet den Teilprozess, der durch die Einschränkung auf die Ereignismenge $E \setminus \{e\}$ entsteht.

Für einen nichtsequentiellen Prozess p gilt:

$Spuren(p) = \{spur(q) : \text{Prozess q ist eine vollständige Sequentialisierung von p}\}$

In diesem Abschnitt wird der Begriff ***Prozess*** anhand von Aktionsstrukturen mathematisch gefasst. Dies ermöglicht die mathematische Modellierung beliebiger, nichtkontinuierlicher ("diskreter") Abläufe verteilter Systeme, insbesondere von Prozessen in Rechenanlagen.

**Prozess**

**Kausale Abhängigkeiten**

**Sequentialisierung**



# Zustandsautomat

Ein nichtdeterministischer Zustandsautomat ist gegeben durch:

S, eine Menge von Zuständen, genannt ***Zustandsraum*** ,

A, eine Menge von ***Transitionsaktionen*** ,

$R \subseteq S \times A \times S$ eine ***Zustandsübergangsrelation*** ,

Seien $s_0$ , $s_1 \in S$ und $a \in A$ gegeben. $(s_0 , a, s_1 ) \in R$ bedeutet, dass im Zustand $s_0$ die Aktion a ausgeführt werden kann und dies zum Nachfolgezustand $s_1 \in S$ führen kann.

Diese Art von Automaten heißt ***nichtdeterministisch*** , da in einem Zustand mehrere Transitionsaktionen möglich sein können und eine Transitionsaktion zu unterschiedlichen Nachfolgezuständen führen kann.

Wir schreiben (für gegebene Relation R) $s_0 \rightarrow_a s_1$ , um auszudrücken, dass $(s_0 , a, s_1 ) \in R$ gilt.

S0 eine Menge von möglichen Anfangszuständen.



# Beispiel Fahrkartenautomat

Akzeptiert werden 1- und 2 DMark Münzen. Mittels zweier Knöpfe kann man zwischen einer Kurzstrecke für 1 DM oder einer normalen Fahrt für 2 DM wählen. Der Automat gibt Wechselgeld zurück.

**Menge der Zustände**

$S \subseteq S_1 \times S_2$ mit

$S_1$ = {Wahl, kurz, normal} und $S_2$ = {-1,0,1,2}

mit der folgenden Interpretation eines Zustandes s = (x, y):

x= Wahl: Automat wartet auf die Wahltaste

x= kurz: eine Kurzstreckenkarte wurde gewählt und ist noch auszugeben

x= normal: eine normale Fahrkarte wurde gewählt und ist noch auszugeben

y= -1 : es ist noch 1 DM zurückzugeben

y= 0 : es ist kein Geld mehr einzuwerfen oder zurückzugeben

y= 1 : es muss noch 1 DM eingeworfen werden

y= 2 : es muss noch 2 DM eingeworfen werden

**Menge der Transitionsaktionen**

A = {Wk, Wn, E1, E2, R1, Ak, An }, mit

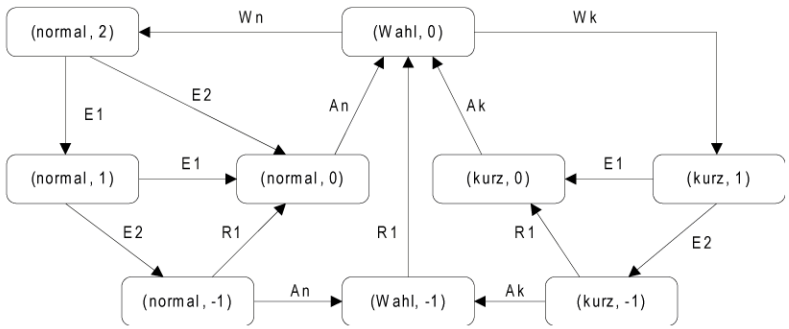
Wk: Wahl einer Kurzstreckenkarte

Wn: Wahl einer normalen Fahrkarte

E1: Einwurf eines 1DM Stückes

E2: Einwurf eines 2DM Stückes

# Zustandsübergänge



# Aktionsspur

Jedem Zustandsautomaten lassen sich ausgehend von der gegebenen Menge von Anfangszuständen Spuren zuordnen.

**Definition**

Gegeben sei ein Zustandsautomat Z = (S, A, R, S0). Eine Folge $a_i$ , wobei $1 \leq i \leq k$ mit $k \in \mathbb{N} \cup \{\infty\}$, ist eine endliche oder unendliche ***Aktionsspur*** des Zustandsautomaten, falls eine Folge von Zuständen $s_i$ existiert, mit $s_i \in S$, $s_1 \in S0$ und

$s_{i-1} \rightarrow_{a_i} s_i$ für alle i mit $1 < i \leq k$

Eine Aktion a heißt ***deterministisch*** , wenn für jeden Zustand $s \in S$ höchstens ein Nachfolgezustand $s_f \in S$ existiert mit

$s \rightarrow_a s_f$

Beispiel



# Beispiel

Auswertung eines Programms mit Zustandsänderung. Das Programm

```
y = 1;
x = 10;
while (x > 0) {
    y = x * y; x = x - 1;
}
```

berechnet die Fakultätsfunktion. Nach seiner Ausführung gilt 10! = y.

| Ereignis | Aktion | Zustand x | Zustand y |
|---|---|---|---|
| a0 | y = 1 | undefiniert | 1 |
| b0 | x = 10 | 10 | 1 |
| a1 | (x > 0) ? | 10 | 1 |
| b1 | y = x * y | 10 | 10 |
| c1 | x = x -1 | 9 | 10 |
| ... | ... | ... | ... |
| a10 | (x > 0)? | 1 | 10! |
| b10 | y = x * y | 1 | 10! |
| c10 | x = x - 1 | 0 | 10! |
| a11 | (x > 0)? | 0 | 10! |



Aktionen und deren Zustandsänderungen können in Konflikt zueinander stehen.

**Beispiel**

Mögliche Konflikte bei parallel durchzuführenden Zuweisungen. Wir betrachten die Aktionen, die den folgenden beiden Zuweisungen entsprechen:

```
(1) x = x + 1
(2) x = 2 * x
```

Die Zuweisungen stehen im Konflikt zueinander.

Werden sie parallel ausgeführt, so kann dem entsprechenden Prozess keine Zustandsänderung eindeutig zugeordnet werden; wir sagen auch, die Aktionen ***stören*** sich wechselseitig.

Störungsfreiheit



Frage: Gibt es Kriterien, um auf einfache Weise solche Konflikte zu erkennen, so dass sie durch eine Reihenfolgeregelung gelöst werden können?

**Vorbereitung**

Für eine Transitionsaktion a gelte:

V(a) $\subseteq$ S ist ***Vorbereich*** , d.h. die Menge der gelesenen Zustandsvariablen.

N(a) $\subseteq$ S ist ***Nachbereich*** , d.h. die Menge der geschriebenen Zustandsvariablen.

**Beispiel**

Sei S = {x, y, z}

```
(1)    a1: x = x+3y+z        V(a1)={x,y,z}, N(a1)={x}
(2)    a2: y = 2*x           V(a2)={x}, N(a2)={y}
```

Definition Störungsfreiheit

Für störungsfreie Systeme gilt, dass unter Einhaltung der mit $\leq$ festgelegten Reihenfolge, die Ausführungsreihenfolge der parallelen Ereignisse und deren Aktionen keinen Einfluss auf die berechneten Ergebnisse hat.

# Definition Störungsfreiheit

Gegeben sei ein Prozess p= (E, ≤, α). Der Prozess p heißt ***störungsfrei*** , genau dann, wenn für jedes Paar von Ereignissen e1, e2 ∈ E gilt:

1.) e1 ≤ e2 oder e2 ≤ e1 oder

2.) V(α(e1)) ∩ N(α(e2)) = N(α(e1)) ∩ V(α(e2)) = N(α(e1)) ∩ N(α(e2)) = Ø.

Die Bedingung (2) nennt man auch ***Bernstein-Bedingungen*** .



# Aktionen als Zustandsübergänge

Zustände und Zustandsänderungen zur Modellierung des Systemverhaltens; Zustandsraum = Menge aller Systemzustände

**Interpretierte Aktionen**

Aktionen werden Zustandsänderungen als Bedeutung zugeordnet.

**Modell**

Zu diesem Zweck definieren wir ***nichtdeterministische Zustandsautomaten mit Transitionsaktionen*** .

Zustandsautomat

Beispiel Fahrkartenautomat

Aktionsspur

Konflikte



Formalismus von C.A. Petri 1962 entwickelt. Ansatz führte zu einer Theorie zur Analyse und Entwicklung nebenläufiger Systeme.

**informelle Charakterisierung**

Ein ***Petri-Netz*** ist gerichteter Graph mit Kanten und zweierlei Knoten.

Knoten: Stellen (graphisch Kreise) und Transitionen (graphisch Rechtecke)

Kanten: von Stellen zu Transitionen oder von Transitionen zu Stellen

Belegung der Stellen mit Marken/Werten (Token);

In einem boolschen Netz sind als Werte nur 0 oder 1 zugelassen.

In einem Stellen/Transitionsnetz sind für die Belegung der Stellen natürlichzahlige Werte zugelassen; die maximale Belegung definiert die Stellenkapazität.

Zustand: definiert durch Belegung der Stellen;

Zustandsübergang durch sogenannte Schaltregeln (engl. firing rule); Belegung ändert sich. Man spricht in diesem Zusammenhang vom Schalten einer Transition.

Markierung der Kanten, Kantengewichte: Gewichtung gibt an, wieviele Marken beim Schalten einer Transition von den Eingangsknoten (Stellen) der Transition abgezogen und den Ausgangsknoten (Stellen) der Transition hinzugefügt werden.

Beispiel eines Petri-Netzes



grobe Modellierung einer Materialverwaltung.