0.4pt1pt
0pt1pt

Lernkarten
Theoretische Informatik

Modelle für Rechensysteme
Theoretische Grundlagen der Programmierung

Zur Vorbereitung auf die Kerngebietsprüfung in theoretischer Informatik bei Prof. Dr. Rüdiger Valk.

Stand: 12. März 2011

Christoph Haas / Kai Kinne / Claas Rink

Es ist alles so unglaublich deprimierend, daß ich mir ein Grinsen nicht verkneifen kann.

Garfield

Alles sollte so einfach wie möglich gemacht werden — aber nicht einfacher.

Albert Einstein

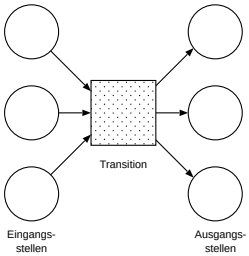
Tu es — oder tu es nicht. Es gibt kein ‚versuchen'.

Yoda

▷ Was ist ein Petri-Netz?

(Petri-)Netz 9

- Tripel: (S, T, F)
- S ist Menge von Stellen
  T ist (disjunkte, nicht-leere) Menge von Transitionen
  F ist Flußrelation $F \subset (S \times T) \cup (T \times S)$ — (Kantenmenge des Netzgraphen)



Stellen

- Zustände in einem Netz
- Bedingungen, die für eine Handlung notwendig sind
- vor einer Handlung: Vorbedingungen
  nach einer Handlung: Nachbedingungen
  vor und nach einer Handlung: Nebenbedingungen

Transitionen

- Zeitlose Handlung, die den Gesamtzustand in einem Netz verändert

▷ Was sind Eingangs- und Ausgangselemente?

▷ Was ist eine Nebenbedingung?

▷ Wann ist ein Netz schlingenfrei?

▷ Was ist der Rand einer Menge?

▷ Wann ist ein Netz stellen- oder transitionsberandet?

Eingangselemente / Ausgangselemente 10

Für eine Stelle oder Transition x ist:

- ${}'x = \{x|(x,y) \in F\}$ die Menge der Eingangselemente (${}'t$ sind Vorbedingungen, ${}'s$ sind Eingangstransitionen)
- $x' = \{y|(x,y) \in F\}$ die Menge der Ausgangselemente (t' sind Nachbedingungen, s' sind Ausgangstransitionen)

Nebenbedingung 10

Eine Stelle, die gleichzeitig Eingangs- und Ausgangsstelle für eine einzige Transition ist.

schlingenfrei / nebenbedingungsfrei 10

Ein Netz enhält keine Nebenbedingungen. ($\forall t \in T: \quad t' \cup {}'t = \emptyset$)

Rand 10

- die Menge der Elemente von A, die mit einem Element außerhalb von A verbunden sind
- formal: $R(A) = \{x \in A \mid \exists y \notin A : y \in {}'x \cup x'\}$

stellen-/transitionsberandet 11

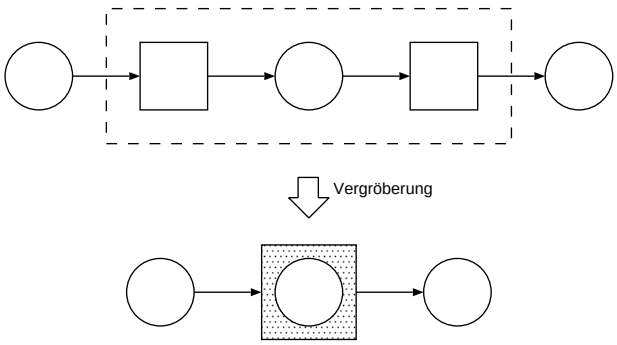
- stellenberandet: der Rand der Menge besteht nur aus Stellen $\Rightarrow R(A) \subset S$
- transitionsberandet: der Rand der Menge besteht nur aus Transitionen $\Rightarrow R(A) \subset T$

▷ Vergröberung bzw. Verfeinerung eines Netzes?

Vergröberung/Verfeinerung eines Netzes 11

Man verfeinert ein Teilnetz $N[A]$, indem man eine Menge A (Abstraktion) von Stellen und Transitionen durch eine Stelle (falls stellenberandet) oder eine Transition (falls transitionsberandet) ersetzt und die Flußrelation anpaßt.

Umgekehrt vergröbert man eine Stelle/Transition, indem man sie durch ein passendes Teilnetz ersetzt und die Flußrelation anpaßt.

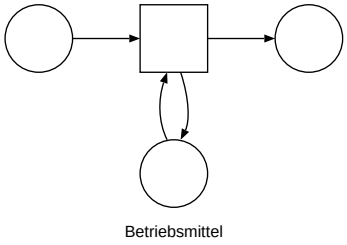


▷ Was sind Betriebsmittel?

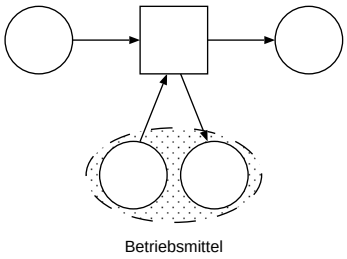
Betriebsmittel 17

Betriebsmittel sind Ressourcen, die zum Durchführen einer Handlung nötig sind. Exklusive Betriebsmittel werden zwischen zwei Handlungen zeitlich geteilt. Entziehbare Betriebsmittel können temporär anderen Handlungen zugewiesen werden.

wiederverwendbares Betriebsmittel



verbrauchbares Betriebsmittel



▷ Was ist eine Funktionseinheit?

▷ Was ist ein Auftrag?

▷ Was ist eine Auftragsbeschreibung?

Funktionseinheit 19

Ein Gebilde, das durch seine Aufgaben und Handlungen definiert ist.

▷ Controller, CPU, Programm, Betriebssystem

Auftrag 19

- Verpflichtung einer Funktionseinheit zur Ausführung einer Handlung
- Auftragsbeginn: Zeitpunkt der Auftragszuteilung
  Auftragsende: Zeitpunkt des Abschlusses der Handlung
- formal: $a = (\alpha, R)$
  α ist die Anweisung
  R ist die verpflichtete Funktionseinheit

Auftragsbeschreibung 19

Beschreibung der auszuführenden Handlung eines Auftrages (formuliert in einer Auftragssprache)

▷ **Was ist ein Auftragssystem?**

▷ **Wann sind zwei Aufträge direkt präzedent?**

▷ **Was ist ein Präzedenzgraph?**

▷ **Was ist ein Konnektivitätsgraph?**

Auftragssystem 24

- formal: AS$= (A, \prec)$
- A ist Menge von Aufträgen
- $\prec$ ist Präzedenzrelation, die die kausale Abfolge der Aufträge angibt
  (irreflexiv, weil kein Auftrag seine eigene Ausführung voraussetzen kann und
  transitiv, weil die kausalen Abfolgen der Aufträge dargestellt werden)

direkt präzedent 24

- formal: $a_1 \prec a_2$ und es gibt kein a_x mit $a_1 \prec a_x \prec a_2$
- zwei Auträge stehen in der Präzedenzrelation, sind aber über keinen anderen Weg zu erreichen

Präzedenzgraph 25

Grafische Darstellung der Präzedenzen von Aufträgen in einem Auftragssystem

Konnektivitätsgraph 24

Graphdarstellung aller direkten Präzedenzen von Aufträgen (Hasse-Diagramm der Präzedenzrelation)

▷ **inverse Relation?**

▷ **komplementäre Relation?**

▷ **identische Relation?**

▷ **symmetrische und reflexive Hülle?**

▷ **transitive Hülle?**

▷ **transitive und reflexive Hülle?**

Inverse Relation 25

$R^{-1} = \{(b, a) \mid (a, b) \in R\}$ (Umkehrung der Kantenrichtungen)

Komplementäre Relation 25

$\overline{R} = \{(a, b) \mid (a, b) \notin R\}$ (alle anderen Kanten in der Obermenge)

Identische Relation 25

$id_{|A} = \{(a, a)\}$ (nur Schleifen)

Symmetrische und reflexive Hülle 25

$\hat{R} = R \cup R^{-1} \cup\ id_{|A}$ (Schleifen und richtungslose Kanten)

Transitive Hülle 25

$R^+ = \bigcup_{i \geq 1}^{\infty} R$ (Menge der Paare, die ein Pfad verbindet)

Transitive und reflexive Hülle 25

$R^* = \bigcup_{i \geq 0}^{\infty} R$

▷ **Was ist ein Pfad?**

▷ **Was ist ein Zyklus?**

▷ **Was ist ein Kausalnetz?**

Pfad 26

- formal: $(a_1, a_2), (a_2, a_3), (a_3, a_4), \ldots, (a_{n-1}, a_n)$
- Folge von Knotenpaaren, die von einem Knoten zu einem anderen führen

Zyklus 26

Pfad, bei dem Starknoten und Endknoten identisch sind

Kausalnetz 26

Netz $N = (S, T, F)$ mit den Eigenschaften:

- N ist zyklenfrei $(F^+ \cap id = \emptyset)$
- jede Stelle hat höchstens eine Eingangs- und Ausgangstransition (keine Entscheidungen)

  $\forall s \in S : |'s| \leq 1 \wedge |s'| \leq 1$
- Die transitive Hülle der Flußrelation heißt Kausalrelation. Die Kausalrelation ist wie die Präzedenzrelation irreflexiv und transitiv.

  $\prec := F^+_{|(S \cup T)}$

▷ Kausalnetze und Auftragssysteme sind äquivalent

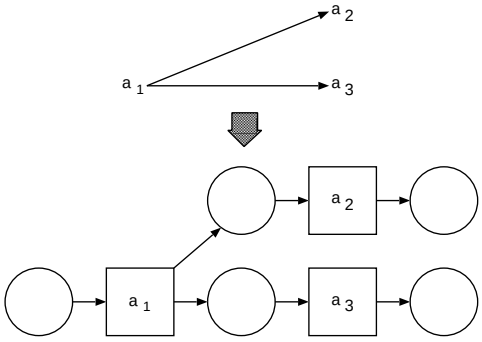
▷ notwendig zur Konstruktion von Prozessen für S/T-Netze oder B/E-Netze

▷ **Wie ordnet man einem Auftragssystem ein Kausalnetz zu?**

▷ **Wie ordnet man einem Kausalnetz ein Auftragssystem zu?**

Auftragssystem → Kausalnetz 26

- Aufträge werden zu Transitionen
- Zwischen den Transitionen entsprechend der direkt präzedenten Aufträge sowie am Anfang und Ende werden Stellen eingefügt
- Die Flußrelation wird angepaßt



Auftragssystem

Kausalnetz

Kausalnetz → Auftragssystem 27

Die Transitionen bilden die Aufträge.

Beim endlichen Kausalnetz ist $AS = (T, \prec_{|T \times T})$ das dem Kausalnetz zugeordnete Auftragssystem

▷ **Was bedeuten die Relationen li und co?**

▷ **Wann ist ein Kausalnetz sequentiell?**

▷ **Wann ist ein Auftragssystem sequentiell?**

Relation li 27

In einem Kausalnetz stehen alle sequentiellen Elemente in der Relation li.

$li = \prec \cup \prec^{-1} \cup id$

li ist nicht transitiv!

Relation co 27

In einem Kausalnetz stehen alle nebenläufigen Elemente in der Relation co.

$co = \overline{li} \cup id$

co ist nicht transitiv!

Achtung! li und co funktionieren nur bei Kausalnetzen, weil es keine Zyklen gibt.

Sequentielles Kausalnetz 29

. . . wenn alle Elemente in li liegen

Sequentielles Auftragssystem

. . . wenn das zugehörige Kausalnetz sequentiell ist

▷ **Wann sind Elemente bezüglich einer Relation unabhängig?**

▷ **Wann bilden Elemente eine Klique in einer Relation?**

▷ **Wann bilden Elemente eine Elementeüberdeckung von einer Relation?**

▷ **Wann bilden Elemente einen Bezirk?**

unabhängig bezüglich einer Relation 28

- wenn keine zwei Elemente durch eine Kante verbunden sind (bzw. in Relation zueinander stehen)
- $(\forall x, y \wedge x \neq y) \quad \Rightarrow \quad (x, y) \notin R$

Klique 28

- wenn jedes Element mit jedem anderen durch eine Kante verbunden sind
- $\forall x, y \in K \quad \Rightarrow \quad (x, y) \in R$

Elementeüberdeckung 28

- wenn jede Kante der Relation R zu einem Element der Überdeckung K führt
- $(x, y) \in R \wedge x \neq y \quad \Rightarrow \quad x \in K \vee y \in K$

Bezirk (maximale Klique) 28

- wenn die Elemente eine Klique K bilden, die sich nicht durch Hinzunahme weiterer Elemente k erweitern läßt
- $\forall a \notin K \Rightarrow \exists k \in K : \quad (a, k) \notin R$

▷ Was sind Linien und Schnitte?

▷ Was ist eine Markierung und wie wird sie dargestellt?

Linien, Schnitte

- $L = \{K|K$ ist Bezirk bezüglich $li\}$ist Menge der Linien (lines) von N
- $C = \{K|K$ ist Bezirk bezüglich $co\}$ist Menge der Schnitte (cuts) von N
- $S = \{K|K \subset S \wedge K \in C\}$ ist Menge der S-Schnitte von N
- $T = \{K|K \subset T \wedge K \in C\}$ ist Menge der T-Schnitte von N

Linien sind maximale Mengen sequentieller Handlungen.
Schnitte sind maximale Mengen nebenläufiger Handlungen.

Markierung [36]

. . . eine Verteilung von Marken auf den Stellen
formal: $m : S \to \mathbb{N}$
$m(S_1)$ steht für die Anzahl der Marken in der Stelle S_1

Darstellung einer Markierung

- als Abbildung: $m(1) = 3, m(2) = 2, m(3) = 2$
- als Vektor: $m = (3, 2, 2)$
- als Folge: $m = s_1^3 s_2^2 s_3^2$

▷ Was ist ein S/T-Netz?

▷ Was ist ein B/E-Netz?

S/T-Netz [37]

$N = (S, T, F, K, W, m_0)$

- S ist Menge von Stellen
- T ist Menge von Transitionen
- F ist Flußrelation $(S \times T) \cup (T \times S)$
- K ist Kapazitätsfunktion $K : S \to N \cup \{\omega\}$ (ω bedeutet: keine Kapazitätsbeschränkung)
- W ist Kantenbewertung $W : F \to N$
- m_0 ist Anfangsmarkierung $m_0 : S \to N$

B/E-Netz [38]

$N = (S, T, F, m_0)$

Ein Bedingungs/Ereignis-Netz ist ein S/T-Netz mit den Einschränkungen:

- die Kapazität jeder Stelle ist 1
- jede Kantenbewertung ist 1

▷ **Wann ist eine Transition aktiviert?**

▷ **Wann schaltet eine Transition?**

Transition aktiviert 39

$m \xrightarrow{t}$

- wenn genügend Marken in den Eingangsstellen liegen
  $\forall s \in S: \quad m_1(s) \geq W(s,t)$
- und die Kapazitäten der Ausgangsstellen durch das Schalten nicht überschritten werden
  $\forall s \in S: \quad m_1(s) - W(s,t) + W(t,s) \leq K(s)$

Transition schaltet 39

$m_1 \xrightarrow{t} m_2$

Die Markierung aller Stellen ändert sich entsprechend der Kantenbewertungen:
$\forall s \in S: \quad m_2(s) = m_1(s) + W(t,s) - W(s,t)$
▷ kann schalten, wenn die Transition aktiviert ist

▷ **Was ist ein Erreichbarkeitsgraph/Fallgraph?**

▷ **Was ist $R(N)$**

▷ **Was ist $F(N)$ bzw. $F(N, M_E)$?**

▷ **Was ist $Er(N)$?**

Erreichbarkeitsgraph/Fallgraph 40

Der Erreichbarkeitsgraph gibt an, welche Markierungen durch Schaltfolgen erreicht werden können.
$Er(N) = (V, E, m_0)$

- $V = R(N)$
  (die Knoten sind die Markierungen)
- $E = \{(m_1, t, m_2) \mid m_1, m_2 \in R(N),\ t \in T: \quad m_1 \xrightarrow{t} m_2\}$
  (die Kanten sind die schaltenden Transitionen)
- m_0 ist der Anfangsknoten (Anfangsmarkierung)

R(N) 40

- Menge der erreichbaren Markierungen (reachability set) (Erreichbarkeitsmenge)
- $\{m \in M \mid \exists w \in T^*: m_0 \xrightarrow{w} m\}$

F(N) 40

- Menge der Schaltfolgen (firing sequences)
- $\{w \in T^* \mid \exists m \in M: m_0 \xrightarrow{w} m\}$

F(N,M$_E$) 40

- Menge der terminalen Schaltfolgen (Schaltfolge führt zu einer Endmarkierung)
- $\{w \in T^* \mid \exists m_E \in M_E: m_0 \xrightarrow{w} m_E\}$

▷ **Was ist ${}^0N$ bzw. N^0?**

${}^0N$ **41**

- Menge der Stellen ohne Eingangstransitionen
- $\{s \in S \mid {}^\cdot s = \emptyset\}$

N^0 **41**

- Menge der Stellen ohne Ausgangstransitionen
- $\{s \in S \mid s^\cdot = \emptyset\}$

▷ **Was ist ein asynchroner Prozeß?**

▷ **Wie wird ein asynchroner Prozeß dargestellt?**

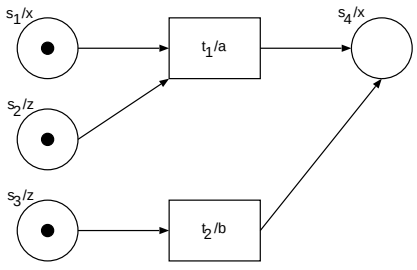
(asynchroner) Prozeß **45**

Kausalnetz, das für eine mögliche Schaltfolge die Bewegung der Marken in einem S/T-Netz beschreibt

formal: (N_K, p)

- N_K ist ein markiertes Kausalnetz
- p ist eine Abbildung $S_K \cup T_K \to S \cup T$ mit den Eigenschaften:
  - passende Struktur
  - Anfangsmarkierungen entsprechen sich
  - Anzahl der Marken einer Stelle in N entspricht der Anzahl der kopierten Stellen in N_K

Darstellung



▷ Ein Prozeß eines endliches Netzes kann unendlich sein!

▷ **Wie konstruiert man einen Prozeß zu einem S/T-Netz?**

▷ **Was bedeutet**

▷ **. . . nebenläufig?**

▷ **. . . sequentiell?**

▷ **. . . kollateral?**

▷ **. . . seriell?**

▷ **. . . synchron?**

Konstruktion S/T-Netz $\rightarrow$ Prozeß 46

1. Man zeichnet soviele neue Stellen mit je einer Marke, wie Marken in der alten Stelle waren,
2. man übernimmt die alten Transitionen
3. und zeichnet soviele Kannten, wie die Kantenbewertung vorher war.

nebenläufig 47

zwei Transitionen sind in der Relation co

sequentiell 47

zwei Transitionen sind in der Relation li

kollateral ($\neq$ seriell) 47

die Ausführungen zweier Transitionen überschneiden sich zeitlich

synchron ($\neq$ asynchron) 47

die Transitionen haben konkrete Anfangszeitpunkte $a(t)$ und Endzeitpunkte $e(t)$ mit:
$a(t_1) \leq e(t_1) \quad \wedge \quad e(t_1) \leq a(t_2)$
wobei $t_1 < t_2$

▷ **Wann ist ein Netz kontaktfrei?**

▷ **Was ist eine Konfliktstelle?**

▷ **Wozu braucht man Komplementärstellen?**

▷ **Wie bildet man Komplementärstellen?**

kontaktfrei 50

Bei jeder Markierung ist jede Transition genau dann aktiviert, wenn genügend Marken in den Eingangsstellen ${}^{\cdot}t$ liegen:
$\forall t \in T,\ \forall s \in {}^{\cdot}t : \quad m(s) \geq W(s,t)$
Bei kontaktfreien Netzen ist in die Kapazität der Ausgangsstellen immer unendlich, so daß die Transition immer schalten kann, wenn die Vorbedingungen erfüllt sind.

Konfliktstelle 49

Stelle mit mindestens zwei Ausgangstransitionen (Entscheidung)

wozu Komplementärstellen 50

Zur Konstruktion von Prozessen. Das zum Prozeß gehörige Kausalnetz darf keine Kapazitäten haben. Durch die Einführung von Komplementärstellen an den Ausgangsstellen betroffener Transitionen wird ein Netz kontaktfrei gemacht und die Kapazitäten werden entfernt.

Bildung von Komplementärstellen 51

1. man negiert alle Kantenbewertungen, die eine Stelle betreffen und
2. die Anfangsmarkierung der Komplementärstelle ist die Kapazität minus der Anfangsmarkierung der Ursprungsstelle:
   $m_0(\overline{s}) = K(s) - m_0(s)$

▷ **Was ist bei einer Funktionseinheit...**

▷ **die Füllung?**

▷ **die relative Füllung?**

▷ **die mittlere Füllung?**

▷ **die Kapazität?**

▷ **die Belegtzeit?**

Füllung einer Funktionseinheit f 62

Anzahl der Aufträge, die der Funktionseinheit übergeben, aber noch nicht erledigt wurden
$f = 0$: Funktionseinheit ist frei
$f > 0$: Funktionseinheit ist beschäftigt
$f = k$: Funktionseinheit ist belegt

relative Füllung ϕ 62

$\phi = \frac{f}{k}$ (phi) $[\%]$

mittlere Füllung (Littlesches Gesetz)

$\overline{f} = \overline{y} \cdot d =$ mittlere Verweilzeit $\times$ Durchsatz $[\%]$

Kapazität k 62

Maximale Füllung einer Funktionseinheit
$k = 1$: Funktionseinheit ist einfach

Belegtzeit x 66

Summe der Verweilzeiten (Zeit, in der die Funktionseinheit belegt ist)

$x = \sum y_i$

▷ **Was ist bei einer Funktionseinheit...**

▷ **die Bedienzeit?**

▷ **die Verweilzeit?**

▷ **der Durchsatz?**

▷ **der Grenzdurchsatz?**

▷ **die Auslastung?**

Bedienzeit b 66

Angenommene Verweilzeit bei Füllung 1 (d.h., wenn nur der eine Auftrag von der Funktionseinheit bearbeitet würde)

Verweilzeit y 65

$y = t_2 - t_1$

▷ die Zeit zwischen Auftragsbeginn und Auftragsende

Durchsatz d 65

$d(t_1, t_2) = \frac{n}{t_2 - t_1} = \frac{\text{Aufträge}}{\text{Zeitintervall}}$ $\left[\frac{\text{Aufträge}}{\text{sek}}\right]$

▷ Anzahl beendete Aufträge pro Zeitintervall

Grenzdurchsatz c 65

$c = \frac{n}{\sum y} = \frac{\text{Aufträge}}{\text{Verweilzeiten}}$ $\left[\frac{\text{Aufträge}}{\text{sek}}\right]$

▷ maximal möglicher Durchsatz

Auslastung / relativer Durchsatz (ρ) 65

$\rho = \frac{d}{c} = \frac{\text{Durchsatz}}{\text{Grenzdurchsatz}} = \frac{\lambda}{m \cdot \mu}$ (in Wartesystemen) $[\%]$

▷ momentane Auslastung

▷ Was ist ein Funktionseinheitensystem?

▷ Was ist eine Transitions-Anschrift eines Netzes?

▷ Was ist eine Typfolge?

Funktionseinheitensystem 73

Ein S/T-Netz $N = (S, T, F, K, W, m_0, l)$ mit Transitionsanschrift.

Transitions-Anschrift 72

Zuordnung von Transitionen auf Typen (Funktionseinheiten). X ist die Menge von Funktionseinheiten.
$l : T \to X \cup \{\lambda\}$

Typfolge 72

Folge von Typen (entspricht Schaltfolge beim S/T-Netz)
$L(N) = \{l^*(w) \mid w \in F(N)\}$
$L(N)$ ist die Menge von Typfolgen die den Schaltfolgen w zugeordnet werden.

▷ Was ist der Kruskal-Algorithmus?

Kruskal-Algorithmus 101

Algorithmus zur Berechnung eines kostenminimalen Gerüstes
Ablauf:

1. Kante mit minimalen Kosten auswählen und streichen
2. ausgehend von durchstrichenen Teilbaum die nächste Kante mit minimalen Kosten suchen
3. falls noch Kanten übrig sind $\Rightarrow$ (2.)

▷ Durch Kruskal erhält man nicht unbedingt das optimale Ergebnis

Komplexität
$O(n \cdot \log n)$

▷ Was ist Synchronisation?

▷ Was ist ein Fall/Transitions-System?

Synchronisation 119

- Einordnung einer nebenläufigen Handlung in ein sequentielles Schema
- Einschrankung der Nebenläufigkeit (z.B. der Relation co)

Rendezvous-Synchronisation

Synchronisation zur Datenübertragung zwischen Prozessen ohne globalen Speicher. Dazu signalisiert ein Prozeß, daß er bereit ist, Daten zu versenden oder zu empfangen. Er wartet dann solange, bis der Partner ebenfalls bereit ist.

Fall/Transitions-System 120

$\Sigma = (B, C, T, r)$

- B ist Menge von Bedingungen
- C ist Menge von Fällen ($C \subset P(B)$) – also eine Menge von Bedingungsmengen
  Fälle = Transitionen
- T ist Menge von Transitionen
- r ist Übergangsrelation ($r \subset C \times T \times C$)

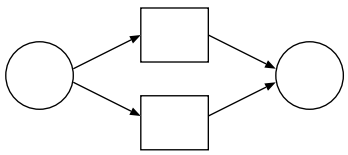
▷ entspricht Erreichbarkeitsgraph beim B/E-Netz

▷ Wann ist ein Netz schlicht?

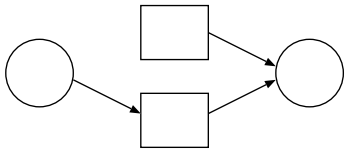
schlicht 121

- keine zwei Transitionen haben dieselben Eingangs- und Ausgangsstellen
- Handlungen werden eindeutig durch Transitionen beschrieben (Extensionalitätsprinzip ist erfüllt)
- $\forall t_1, t_2 \in T$:

$$\dot{t}_1 = \dot{t}_2 \quad \wedge \quad \dot{t}_1 = \dot{t}_2 \quad \Rightarrow \quad t_1 = t_2$$



nicht schlicht



schlicht

▷ Wann erfüllt ein Fall/Transitions-System das Extensionalitätsprinzip?

Extensionalitätsprinzip 121

Ein Fall/Transitions-System Σ erfüllt das Extensionalitätsprinzip, wenn es 3 Abbildungen $pre(t)$, $side(t)$ und $post(t)$ gibt, die T nach $P(B)$ abbilden

. . . und gilt:

1. Vor-/Nachbedingungen gelten nur vor/nach der Ausführung. Nebenbedingungen gelten vorher und hinterher

   $\forall(c_1, t, c_2) \in r :$
   $pre(t) = c_1 - c_2 \quad \wedge$
   $post(t) = c_2 - c_1 \quad \wedge$
   $side(t) \subset c_1 \cap c_2$

2. t kann schalten, wenn

   $c_1 \in C, t \in T :$
   $[pre(t) \cup side(t)] \subset c_1 \wedge post(t) \cap c_1 = \emptyset$
   $\Rightarrow \exists c' : (c_1, t, c') \in r$

3. Eine Handlung wird eindeutig durch die Extension bestimmt $pre(t_1) = pre(t_2) \wedge post(t_1) = post(t_2)$ $\wedge \quad side(t_1) = side(t_2) \Rightarrow t_1 = t_2$

▷ Was bedeutet unteilbar?

▷ Was ist ein Monitor?

unteilbar 128

Eine Handlungen heißt unteilbar, wenn andere nebenläufige Handlungen sie nicht beeinflussen dürfen.

Um die Konsistenz von Ergebnissen zu sichern, benutzt man Semaphore oder Monitore.

PROG

In der Programmiersprache PROG wird Unteilbarkeit mittels <Anweisung, Anweisung, Anweisung> ausgedrückt.

Netze

In Netzen werden unteilbare Handlungen in einer Transition zusammengezogen.

Monitor 235

Hochsprachliche Implementation einer unteilbaren Handlung. Ein Monitor ist ein Unterprogramm, bei dem eine Instanz (z.B. der Compiler) darauf achtet, daß es nur im wechselseitigen Ausschluß aufgerufen wird.

- `delay(x)`:
  wartet (kein busy waiting) bis x mit `continue(x)` freigegeben wurde
- `continue(x)`:
  gibt den Semaphor x für die Routine frei, die mit `delay(x)` darauf wartet

Die Operationen `delay` und `continue` benutzen Semaphoren.

▷ **Was ist ein Kommunikationsprotokoll?**

▷ **Was ist ein Semaphor?**

Kommunikationsprotokoll 131

Anweisungen oder Spezifikationen, die die Kommunikation zwischen Programmen regeln

Semaphor 143

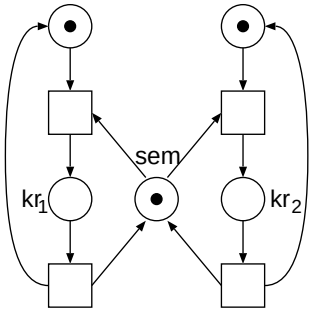
Ganzzahlige Signalvariable, die die Anzahl der Prozesse beschränkt, die sich gleichzeitig in einem kritischen Abschnitt befinden dürfen. Der Semaphor wird mit dieser Anzahl initialisiert.

Dann gibt es zwei im Betriebssystem verankerte Funktionen, die ungeteilt ausgeführt werden:

- P(sem):
  ```
  IF (sem > 0)
  THEN sem=sem-1
  ELSE wait
  ```
- V(sem):
  ```
  sem=sem+1
  ```

Ein Programm geht folgendermaßen in den kritischen Abschnitt:

```
P(sem);
kritischer Abschnitt
V(sem);
```



▷ **Was ist ein schematisches Auftragssystem?**

▷ **Warum heißt es auch uninterpretiert?**

▷ **Wie vergröbert man ein schematisches Auftragssystem?**

▷ **Welche Entscheidbarkeitsfragen gibt es bei schematischen Auftragssystemen?**

Schematisches Auftragssystem 149

Auftragssystem $(A, \prec)$ mit den Besonderheiten:

- die Aufträge bestehen aus je einem Lese- und Schreibauftrag: $a_i = (l_i, s_i)$
- l_i und s_i sind direkt präzedent
- es gibt eine Menge von globalen Variablen $V = \{v_1, v_2, \ldots, v_m\}$
- zu jedem Auftrag a_i gibt es eine Menge von Eingangsvariablen $ein_i = \{e_1, \ldots, e_{m_i}\} \subseteq V$ und Ausgangsvariablen $aus_i = \{a_1, \ldots, a_{m_i}\} \subseteq V$

Darstellung der Aufträge:

$A = \{l_1[ein_1], s_1[aus_1], l_2[ein_2], s_2[aus_2] \ldots\}$

warum uninterpretiert?

Die Interpretation (Funktionsweise) der einzelnen Aufträge ist nicht bekannt. An einem schematischen Auftragssystem untersucht man die Konflikte der Variablen und mögliche Serialisierungen.

Vergröberung

Lese- und Schreibaufträge werden zusammengefaßt.

Entscheidbarkeitsfragen

- Serialisierbarkeit
- Funktionalität
- äquivalente Ausführungsfolgen

▷ **Was ist eine Interpretation?**

▷ **Wann ist ein schematisches Auftragssystem vollständig?**

Interpretation 151

Konkrete Festlegung der Anfangswerte und der Handlungen (Funktionen) der Aufträge.

Besteht aus

- einer Folge von Vektoren $d_0, d_1, d_2, \ldots$
  (Inhalte der Variablen zum Zeitpunkt jedes Auftrags)
- Funktionen $f_i^j : D^{p_i} \to D \qquad (j = 1 \ldots q_i)$
  (i=Auftrag / j=Variable / p_i=Anzahl der Eingangsvariablen)
  (Berechnungen, die ein Auftrag auf den Variablen ausführt)
- einem Anfangszustand d_0 (Anfangsbelegung der Variablen)

Bei einem Leseauftrag werden die Eingangsvariablen $e_1 \ldots e_{p_i}$ in lokale Variablen $lo_1 \ldots lo_{p_i}$ übernommen.

Bei einem Schreibauftrag ergeben sich die Ausgangsvariablen durch die Funktionen: $a_1 = f_i^1(lo_1, \ldots, lo_{p_i}) \quad a_2 = f_i^2 \ldots$

res(w, I) ist das Resultat der Ausführungsfolge w bezüglich der Interpretation I.

Vollständigkeit schem. Auftragssystems 150

Es gibt mindestens drei Aufträge, wobei

- der erste Auftrag (Initialisierungsauftrag) a_1 keine Variablen liest, aber alle Variablen (aus V) schreibt
- der letzte Auftrag (Ausgabeauftrag) a_n alle Variablen liest, aber keine Variablen (in andere) schreibt

▷ **Was bedeutet „l_i liest v von s_j “?**

▷ **Was ist die Werteübertragungsrelation?**

▷ **Welche Aufträge sind relevant?**

▷ **Wann ist ein Auftragssystem relevant?**

▷ **Was ist eine Ausführungsfolge?**

l_i liest v von s_j ? 152

- $s_j <_w l_i$
- s_j schreibt v und l_i liest v
- kein anderer Auftrag zwischen s_j und l_i schreibt v

Werteübertragungsrelation 152

WÜ$(w) := \{(a_i, a_j) \mid \exists v \in V : l_j$ liest v von s_i in $w\}$

relevante Aufträge ($\neq$ nutzlos) 153

- Ausgabeauftrag
- Aufträge, die über die WÜ (von hinten aus berechnet) zu einem relevanten Auftrag führen

  a_j relevant $\wedge$ $(a_i, a_j) \in$ WÜ$(w) \quad \Longrightarrow \quad a_i$ relevant

Auftragssystem relevant 153

wenn alle Aufträge relevant sind

Ausführungsfolge 42

Reihenfolge der Ausführung von Aufträgen

▷ **Wann ist eine Ausführungsfolge seriell?**

▷ **Wann sind zwei Ausführungsfolgen äquivalent?**

▷ **Wann ist eine Ausführungsfolge eine Serialisierung?**

▷ **Wann ist eine Ausführungsfolge serialisierbar?**

▷ **Wann ist eine Ausführungsfolge schwach serialisierbar?**

▷ **Was ist ein Unterauftragssystem?**

▷ **Was ist die Motivation für Serialisierungen?**

Ausführungsfolge seriell 154

- Lese- und Schreibauftrag werden (ungeteilt) hintereinander ausgeführt
- $w_i = l_j \implies w_{i+1} = s_j$

Ausführungsfolgen äquivalent 153

- Das Resultat ist für alle Interpretationen gleich
- gleiche Anzahl von relevanten Aufträgen und gleiche Werteübertragungsrelation

▷ entscheidbar: P

Serialisierung 155

serielle äquivalente Ausführungsfolge

serialisierbar 155

es gibt (mindestens) eine Serialisierung

schwach serialisierbar 155

es gibt eine Serialisierung eines Unterauftragssystems

Unterauftragssystem 150

Ein auf eine Teilmenge der Aufträge reduziertes Auftragssystem.

Motivation für Serialisierungen

Bei nebenläufiger Ausführung von Aufträgen sollen Inkonsistenzen vermieden werden.

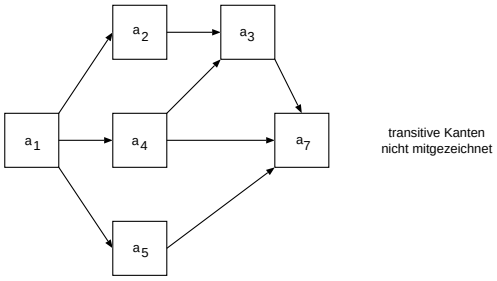
▷ **Was ist ein Serialisierungsgraph?**

Serialisierungsgraph 158

Darstellung einer Serialisierung einer Ausführungsfolge

SG$(w) = (A', \prec_0)$

- A' ist die Menge der relevanten Aufträge
- $\prec_0$ ist die transitive Hülle von $\prec' \cup \prec_1 \cup \prec_2$
  - $\prec'$=$\prec_{|A'}$
  - $\prec_1$= $W\ddot{U}(w)_{|A'}$
  - $\prec_2$= $\{(a_x, a_y) \mid a_i$ liest v von a_j und $v \in aus_k$ mit: entweder $(a_x, a_y) = (a_k, a_j)$ (a_k schreibt vor a_j) oder $(a_x, a_y) = (a_i, a_k)\}$ (a_i liest vor a_k)



MSG(w) ist die Menge der Serialisierungsgraphen der Ausführungsfolge w.
SER(w) ist die Menge der Serialisierungen von w.

▷ Serialisierbarkeit ist NP-vollständig

▷ Was ist ein Transaktionen-System?

▷ Wann ist eine Ausführungsfolge virtuell seriell?

Transaktionen-System 160

Schematisches Auftragssystem, wobei $l_i \prec s_i$ die einzigen Präzedenzen sind

virtuell seriell [Transaktionen-Systeme] 161

Es gibt Serialisierungs-Zeitpunkte $t_1 \dots t_n \quad (\in R)$:

1. $\pi(l_i) < t_i < \pi(s_i)$
   (t_i liegt innerhalb des Auftrags a_i)
2. $\pi(l_i) < \pi(s_j) \Rightarrow t_i < t_j$
   (a_i liest v bevor a_j schreibt)
3. $\pi(s_i) < \pi(s_j) \Rightarrow t_i < t_j$
   (a_i schreibt v vor a_j, dann $\underline{t_i} < t_j$)

$\pi(a)$ ist die Position des Auftrags a in der Ausführungsfolge.

Bei den Serialisierungs-Zeitpunkten kann eine Transaktion als ungeteilt ausgeführt betrachtet werden.

▷ zweiphasig $\rightarrow$ virtuell seriell $\rightarrow$ serialisierbar $\rightarrow$ schwach serialisierbar

▷ Wann ist eine Ausführungsfolge eines Transaktionen-Systems zweiphasig?

zweiphasig [Transaktionen-System] 164

Es gibt Sperrzeitpunkte $sp_1 \dots sp_n \quad (\in R)$:

- $\pi(l_i) < sp_i < \pi(s_i)$
  (sp_i liegt innerhalb des Auftrags a_i)
- $\pi(l_i) < \pi(s_j) \Rightarrow sp_i < sp_j$
  (a_j liest v von a_i, dann $sp_i < sp_j$)
- $\pi(s_i) < \pi(s_j) \Rightarrow \underline{\pi(s_i)} < sp_j$
  (a_i schreibt v vor a_j, dann liegt s_i vor sp_j)

Ein Sperrzeitpunkt gibt an, wann ein Auftrag alle benötigten Betriebsmittel erworben hat und ungeteilt ausgeführt werden könnte. Die erste Phase reicht vom Leseauftrag bis zum Sperrzeitpunkt und die zweite Phase vom Sperrzeitpunkt bis zum Schreibauftrag.

▷ Jede zweiphasige Ausführungsfolge ist virtuell seriell.

▷ **Wann ist ein Auftragssystem funktional?**

▷ **Wann ist ein Auftragssystem spurfunktional?**

▷ **Wann sind zwei Ausführungsfolgen spuräquivalent?**

▷ **Was ist eine Spur?**

funktional 168

- Das Resultat eines Auftragssystems ist (trotz Nebenläufigkeit) eine Funktion der Anfangswerte. 166
- Alle Ausführungsfolgen sind zueinander äquivalent. 168
- Alle relevanten Aufträge sind paarweise störungsfrei. 169

▷ Funktionalität ist in P entscheidbar

spurfunktional 169

Ein Auftragssystem ist spurfunktional, wenn alle Ausführungsfolgen paarweise spuräquivalent sind.

▷ Alle verlustfreien Aufträge sind paarweise störungsfrei.

spuräquivalent 168

Zwei Ausführungsfolgen sind spuräquivalent, wenn ihre Spuren für alle Interpretationen gleich sind.

$spur(w_1, I) = spur(w_2, I)$

Spur [einer Variablen] 168

Folge von Vektoren, die die Werte der Variablen bei jedem Auftrag während einer Interpretation angeben.

Formal: spur$(w, v, I) := d_{i_0}(v)d_{i_1}(v)\ldots$
(wobei $d_0, d_1, \ldots$ die Folge von Zuständen ist)

▷ **Wann sind zwei Aufträge störungsfrei?**

▷ **Wann ist ein Auftragssystem störungsfrei?**

▷ **Was ist die Bernsteinrelation?**

▷ **Wann ist ein Auftrag verlustfrei?**

▷ **Wann ist ein Auftragssystem verlustfrei?**

störungsfrei [Aufträge] 169

Zwei Aufträge werden entweder nacheinander ausgeführt oder sind disjunkt (nach der Bernsteinrelation).

störungsfrei [Auftragssystem] 169

Wenn alle Aufträge paarweise störungsfrei sind.

Bernsteinrelation 169

dis$(a_i, a_j) := (aus_i \cap aus_j = ein_i \cap aus_j = aus_i \cap ein_j = \emptyset)$

Man sagt auch: zwei Aufträge sind *disjunkt*.

▷ Zwei Aufträge beeinflussen sich bezüglich ihrer Variablen nicht.

verlustfrei [Auftrag] 169

Ein Auftrag macht eine Ausgabe.

$aus_i \neq \emptyset$ (wenn es Ausgabevariablen gibt)

verlustfrei [Auftragssystem] 169

Wenn alle Aufträge verlustfrei sind.

▷ **Wann ist ein schem. Auftragssystem maximal nebenläufig?**

▷ **Was ist eine strukturiertes Auftragssystem?**

▷ **Was ist eine Vergröberungsebene?**

▷ **Wann ist ein strukt. Auftragssystem wohlstrukturiert?**

▷ **Wann ist ein Auftrag wohldefiniert?**

maximal nebenläufig 173

Das Auftragssystem hat eine Eigenschaft. Und: würde man eine Präzedenz entfernen, würde es die Eigenschaft verlieren.

z.B.: funktional, spurfunktional

strukturiertes Auftragssystem 179

$(AS, \mathcal{A})$ mit $\mathcal{A} \subset 2^A$

$\mathcal{A}$ ist die Menge von vergröberten Aufträgen und beinhaltet alle elementaren Aufträge A des ursprünglichen Auftragssystems.

Vergröberungsebene

Zusammenfassung von einzelnen Aufträgen oder anderen Vergröberungsebenen.

wohlstrukturiert 181

Alle Vergröberungsebenen sind Kausalnetze (ohne Zyklen!). [Durch Vergröberung von Aufträgen könnte es zu Zyklen kommen.]

wohldefiniert 182

Induktive Definition:

- alle vergröberten Aufträge bestehen aus einem einzelnen Auftrag
- alle Teilaufträge des vergröberten Auftrags sind bereits wohldefiniert und durch die Vergröberung entsteht kein neuer Zyklus

▷ **Welchen Zweck hat eine Synchronisation**

▷ **Wann ist eine unendliche Schaltfolge aktiviert?**

▷ **Verklemmung / verklemmungsfrei?**

Zwecke von Synchronisation 185

- Konsistenz-Synchronisation: Sicherstellen eines korrekten Ergebnisses (Serialisierung)
- Betriebsmittel-Synchronisation: Verwaltung beschränkter Betriebsmittel

Unendliche Schaltfolge aktiviert 196

$m \xrightarrow{w}$

Eine unendliche Folge von Transitionen ist aktiviert in m wenn alle Anfangsstücke aktiviert sind ($\forall n \geq 1 : \; m \xrightarrow{w[n]}$)

- $F_\omega(N)$ ist die Menge aller unendlichen Schaltfolgen von N $F_\omega(N) = \{w \in T^\omega \mid m_0 \xrightarrow{w}\}$
- $w(n)$ ist das n-te Element der Schaltfolge
- $w[n]$ ist das Anfangsstück von w mit der Länge n

Verklemmung 196

- Bei einer Markierung ist keine Transition mehr aktiviert.
- $\neg\exists\, t \in T : \quad m \xrightarrow{t}$

verklemmungsfrei 196

Keine erreichbare Markierung $m \in R(N)$ ist eine Verklemmung.

$\forall m \in R(N), \exists t \in T : \quad m \xrightarrow{t}$

▷ Wann ist eine Markierung sicher / ω-sicher?

▷ Wirkung einer Transition?

sicher (bzw. fortsetzbar) 196

Eine Markierung m ist sicher, wenn

- irgendeine Ausführungsfolge zu einer Endmarkierung führt
- $\exists\, w \in T^* \quad \exists\, m_e \in M_E : \quad m \xrightarrow{w} m_e$

ω-sicher (bzw. ω-fortsetzbar) 196

Eine Markierung m ist ω-sicher, wenn es mindestens eine unendliche Ausführungsfolge gibt, die in m aktiviert ist.

SICH(N) ist die Menge aller ω-sicheren Markierungen von N.

$\exists\, w \in T^\omega : \quad m \xrightarrow{w}$

Wirkung einer Transition 209

$\Delta_N(t)$ ist die Wirkung einer Transition (Vektor aus $N^{|S|}$) und beschreibt die Veränderung jeder Stelle durch das Schalten der Transition t. ($\Delta_N(t)(s) = w(t,s) - w(s,t)$)

$m_2 = m_1 + \Delta_N(t) \quad \Leftrightarrow \quad m_1 \xrightarrow{t} m_2$

Wirkungsmatrix / Inzidenzmatrix

Matrix der Kantenbewertungen eines Netzes. Spalten=Transitionen. Zeilen=Stellen.

N	t_1	t_2	$t_3 \ldots$
s_1	-1	0	$2 \ldots$
s_2	0	1	$0 \ldots$
s_3	1	0	$-1 \ldots$
$\vdots$	$\vdots$	$\vdots$	$\ddots$

▷ Was sind Invarianten?

Invarianten 210

Beziehungen zwischen Variablen, die während der Ausführung von Aufträgen immer gelten. Invarianten kann man ganzzahliges Lösen des Gleichungssystems (der Inzidenzmatrix) erhalten (NP-vollständig).

T-Invariante

ganzzahlige Lösung der Inzidenzmatrix des Netzes

T-Invarianten geben an, wie oft jede Transition schalten muß, um eine Markierung zu reproduzieren. Dies ist aber nur eine notwendige Bedingung! Wenn man die Anfangsmarkierung frei wählen kann, ist sie auch eine hinreichende Bedingung, denn möglicherweise kann nicht aus jeder Markierung jede Transition so oft schalten wie gefordert.

$\Delta_N \cdot i = \vec{0}$

S-Invariante

ganzzahlige Lösung (s. Satz von Lautenbach) der transponierten Inzidenzmatrix des Netzes

S-Invarianten geben Beziehungen zwischen Markierungen von Stellen an, die immer gelten.

Satz von Lautenbach

$\Delta'_N \cdot i = \vec{0} \quad \Longrightarrow \quad i' \cdot m_0 = i' \cdot m \quad \forall m \in R(N)$

Der Satz von Lautenbach definiert den Begriff der „S-Invarianten“.

▷ Entdeckung / Ausschluß / Umgehung von Verklemmungen?

Entdeckung 213

Um Verklemmungen zu entdecken, muß der Erreichbarkeitsgraph konstruiert werden. (Komplexität: NP)

Ausschluß 214

- genügend Betriebsmittel zur Verfügung stellen
- BM-Nachforderungen verbieten (beim Start eines Auftrags muß dieser alle benötigten Betriebsmittel belegen)
- Betriebsmittel entziehbar machen

Umgehung 216

Die Betriebsmittel müssen „intelligent“ zugewiesen werden (Residuen).

▷ Was ist ein Residuum?

▷ Was ist das Bankiersproblem?

▷ Was ist ein K-gesteuertes Netz?

Residuum 216

- kleinste Menge unvergleichbarer Vektoren eines Vektorraums
- formal: $res(k) \subset \mathbb{N}^n$ mit $res(k) + \mathbb{N}^n = K + \mathbb{N}^n$

Zweck des Residuums

Die Residuumsvektoren stellen eine untere Grenze der sicheren Markierungen dar.

Bankiersproblem 221

Beispiel zur Problematik von Verklemmungen und deren Umgehung. Der Bankier verleiht verschiedene Beträge aus seinem Kapital an seine Kunden. Ein Kunde muß den Kredit auf einmal nach endlicher Zeit zurückzahlen. Damit kann es für den Bankier zu Verklemmungen führen, da er Teilbeträge bereits ausgegeben haben kann, aber kein Kunde bereits den vollen Kredit erhalten hat und auch nicht verpflichtet ist, ihn zurückzuzahlen.

Hier empfiehlt es sich im Voraus das Residuum zu berechnen und unsichere Zustände zu vermeiden.

K-gesteuertes Netz 217

Ein S/T-Netz, bei dem die Transitionen erhalten bleiben, deren Schalten in eine andere Markierung im Residuum führt. Jede andere Markierung wird durch eine Anzahl neuer Transitionen ersetzt.

Sinn: Vermeidung von Verklemmungen

▷ **Wann ist eine Markierung lebendig?**

▷ **Wann ist ein Netz lebendig?**

▷ **Wann ist ein Netz fair?**

Markierung ist lebendig 226

- Jede Transition ist in der Markierung lebendig
- Es kann von m jede Transistion t erreicht werden.
- Alle von m aus erreichbaren Markierungen sind T-forsetzbar.

Netz ist lebendig 226

- $\forall t \in T, m \in R(N), u \in T^* \quad \exists v \in T^* : \quad m \xrightarrow{u} \Longrightarrow m \xrightarrow{uvt}$
- Nach jeder Schaltfolge kann jede Transition wieder zum schalten gebracht werden.
- $\forall t \in T, m \in R(N) \quad \underline{\exists} w \in T^\omega : \quad m \xrightarrow{w} \wedge |w|_t = \infty$
- In jeder unendlichen Schaltfolge kann jede Transition unendlich oft vorkommen.
- Die Anfangsmarkierung m_0 ist lebendig.
- Es kann von m_0 jede Transistion t erreicht werden.

▷ nicht abgeschlossen: neue Marken können zu partiellen Verklemmungen führen

▷ entscheidbar: es gibt aber noch kein algorithmisches Verfahren

Netz fair 230

- $\forall t \in T, m \in R(N) \quad \underline{\forall} w \in T^\omega : \quad m \xrightarrow{w} \wedge |w|_t = \infty$
- In jeder unendlichen Schaltfolge muß jede Transition unendlich oft vorkommen.
  (es könnten Transitionen unfairerweise übergangen werden)

▷ **Wann ist eine Markierung fortsetzbar?**

▷ **Wann ist eine Markierung T-fortsetzbar?**

▷ **Wann ist eine Markierung E-fortsetzbar?**

▷ **Was ist eine partielle Verklemmung?**

Markierung ist fortsetzbar 196/226

- $\exists w \in T^*, m_E \in M_E : \quad m \xrightarrow{w} m_E$
- Es gibt eine Schaltfolge, die zu einer Endmarkierung führt.

Markierung ist T-fortsetzbar 226

Es gibt eine unendliche Schaltfolge, daß jede Transition unendlich oft schalten kann.

▷ abgeschlossen: ...

Eine Markierung ist lebendig, wenn alle von ihr aus erreichbaren Markierungen T-fortsetzbar sind.

Markierung ist E-fortsetzbar 226

- $\exists w \in T^\omega : \quad m \xrightarrow{w}$ und alle $t \in E$ kommen in der Schaltfolge w unendlich oft vor.
- In jeder unendlichen Schaltfolge schaltet jede Transitionen aus E unendlich oft.

▷ ω-fortsetzbar/ω-sicher=E-fortsetzbar

partielle Verklemmung ($\neq$ lebendig) 226

- Ein Teil eines Netzes kann nie wieder zum Schalten gebracht werden.

▷ Wann schaltet ein Netz nach der verschleppungsfreien Schaltregel?

▷ Wann schaltet ein Netz nach der fairen Schaltregel?

verschleppungsfreie Schaltregel 232

- $\forall t \in T, m \in M(R) \quad \neg\exists w \in T^\omega :$
  $m \xrightarrow{w} \wedge\, t$ ist bei w permanent aktiviert $\wedge\, |w|_t = 0$
- Es gibt keine Schaltfolge, bei der eine Transition permanent (bei jeder Markierung) aktiviert ist, aber nie schaltet.
- Wenn eine Transition aktiviert ist, muß sie auch schalten.

faire Schaltregel 232

- $\forall t \in T, m \in M(R) \quad \neg\exists w \in T^\omega :$
  $m \xrightarrow{w} \wedge\, t$ ist bei w unendlich oft aktiviert $\wedge\, |w|_t = 0$
- Es gibt keine Schaltfolge, bei der eine Transition unendlich oft aktiviert ist, aber nie schaltet.
- Wenn eine Transition immer wieder aktiviert ist, schaltet sie auch irgendwann.

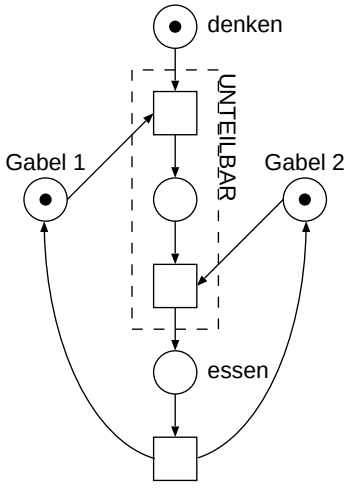
▷ faire Schaltregel impliziert nicht faires Verhalten!

▷ Was ist das Philosophenproblem?

Philosophenproblem 229

Beispiel zur Problematik der Fairneß. Fünf Philosophen sitzen an einem runden Tisch, in dessen Mitte eine Schüssel Spaghetti steht. Jeder Philosoph kann denken oder essen. Zum Essen stehen insgesamt fünf Gabeln (jeweils zwischen den Tellern) zur Verfügung. Will ein Philosoph essen, so nimmt er erst die rechte Gabel und dann die linke Gabel.

Lösung: das Aufnehmen von rechter und linker Gabel muß unteilbar gemacht werden. Damit ist das Netz lebendig — aber noch nicht fair, denn zwei Philosophen könnten ewig essen und die anderen würden verhungern.



▷ verfeinert: nicht lebendig (jeder könnte rechte Gabel nehmen)

▷ vergröbert: nicht fair (zwei Philosophen könnten ewig essen)

▷ **Ausfall, Störung, Behebung?**

▷ **Ausfallzeit?**

Ausfall 293

Überschreiten eines Fehlerkriteriums (Folgen eines permanenten Fehlers oder Häufung von transienten Fehlern)

Störung 294

Fehlerhafte Ausführung eines Auftrags aufgrund einer fehlerhaften Funktionseinheit

Behebung 293

Beendigung eines Ausfalls durch

- manuellen Eingriff
- automatische Rekonfiguration
- überschreiben mit korrekten Daten

Ausfallzeit 294

Zeitraum zwischen Ausfall und Behebung

▷ **Wie kennzeichnet man die Zuverlässigkeit einer Funktionseinheit?**

▷ **Zufallsdichte für das Auftreten einer Störung einer Funktionseinheit?**

▷ **Zufallsverteilung für das Auftreten einer Störung einer Funktionseinheit?**

Zuverlässigkeit einer Funktionseinheit 296

- MTBM (mean time between malfunctions) = mittlerer Störungsabstand — $\frac{1}{\text{MTBM}} =$ Störungsfrequenz
- MTBF (mean time between failures) = mittlerer Ausfallabstand
- Lebensdauer = Zeit von Beansprungsbeginn bis Ausfall (Behebung dann meist nicht mehr möglich)

(Zufalls-)Dichte 300

Ableitung der Verteilung: $f_T(t) = F_T(t)'$

„Wahrscheinlichkeit, daß exakt zum Zeitpunkt t eine Störung auftritt."

Und natürlich ist $\int_0^\infty f_T(t)\ dt = 1$

(Zufalls-)Verteilung 299

$F_T(t) = P[T \leq t]$

„Wahrscheinlichkeit, daß bis zum Zeitpunkt t eine Störung aufgetreten ist."

▷ **Erwartungswert für das Auftreten eines Ereignisses?**

▷ **Ereignisrate bei einer Funktionseinheit?**

▷ **Was ist eine Markov-Kette?**

▷ **Was ist die Markov-Eigenschaft?**

Erwartungswert für T 300

$$E[T] = \int_0^\infty t \cdot \underbrace{f_T(t)}_{\text{Störungswsk.}}\ dt = \int_0^\infty \underbrace{1 - F_T(t)}_{\text{Überlebenswsk.}}\ dt$$

Ausfallrate/Störungsrate/Enderate 302

$$\lambda(t) = \frac{f_T(t)}{1 - F_T(t)} \underbrace{= \lambda}_{\text{konstante Rate}}$$

Markov-Kette

Stochastischer Prozeß (Zustandsfolge), bei dem die Übergangswahrscheinlichkeiten p_{ij} eindeutig durch die Zustände beschrieben werden.

Markov-Eigenschaft (Gedächtnislosigkeit)

$$P[S(t+\Delta t) = j \mid S(t) = i \ \wedge \ \underbrace{S(t-a) = k}_{\text{Vorgeschichte}}]$$

$$= \quad P[S(t+\Delta t) = j \mid S(t) = i]$$

$$= \quad p_{ij}(\Delta t) \quad (\text{unabhängig von } t)$$

- $S(t) = i$ heißt: System ist im Zustand i zur Zeit t
- $P[S(t) = i]$ heißt: Wahrscheinlichkeit für Zustand i zur Zeit t
- die Übergangswahrscheinlichkeiten sind unabhängig von der Vorgeschichte

▷ **Dichte/Verteilung bei kontanter Ereignisrate?**

▷ **Was ist ein Rekurrenzzeitpunkt?**

▷ **Was ist ein Poisson-Prozeß?**

Dichte bei konstanter Ereignisrate? 304

$f_T(t) = \theta \cdot e^{-\theta t}$

Verteilung bei konstanter Ereignisrate? 304

$F_T(t) = 1 - e^{-\theta t}$

Rekurrenzzeitpunkt 306

Ereignisse eines Prozesses (z.B. Ausfälle, Ankünfte, Bedienungen), von denen an ein Vorgang wieder derselben Wahrscheinlichkeitsgesetzmäßigkeit folgt. Beim Poisson-Prozeß ist jeder Zeitpunkt Rekurrenzzeitpunkt (Paradoxon).

Poisson-Prozeß 308

- jedes Ereignis tritt unabhängig von allen anderen ein
- die Wahrscheinlichkeit für das Eintreten zwischen t und $t + \Delta t$ ist:

$$\lim_{\Delta t \to 0} \frac{P(t, \Delta t)}{\Delta t} = \rho \quad \text{(konstant für kurze Zeiträume)}$$

$P(t, \Delta t)$ ist die Wahrscheinlichkeit, daß das Ereignis zwischen t und $t + \Delta t$ eintritt.

▷ Was ist die Lebensdauer?

Lebensdauer 313

Die Lebensdauer L beschreibt das Zuverlässigkeitsverhalten eines Systems.

Überlebenswahrscheinlichkeit

$$r(t) = P[L > t] = 1 - \underbrace{F_L(t)}_{\text{Ausfallwsk.}}$$

mittlere Lebensdauer

$$E[L] = \int_0^\infty r(t)dt$$

Ausfallwahrscheinlichkeit

$$a(t) = P[L \leq t] = F_L(t)$$

Überleben (binär)

$$z(t) = \begin{cases} 0, & \text{wenn } L \leq t \quad \text{(kein Überleben)} \\ 1, & \text{wenn } L > t \quad \text{(Überleben)} \end{cases}$$

Überlebenszustand der n Elemente des Systems

$$\vec{z}(t) = (z_1(t), \ldots, z_n(t))$$

Überlebensfunktion des Systems

$$B_S : \{0,1\}^n \to \{0,1\}$$

Überleben des Systems

$$z_S(t) = B_S(\vec{z}(t))$$

Rechensysteme Stand: 12. März 2011

▷ Was ist eine Serienstruktur?

Serienstruktur 314

Das System überlebt, wenn alle Elemente überleben.

Formal: $z_S(t) = \bigwedge_{i=1}^{n} z_i(t)$

Überlebenswahrscheinlichkeit des Systems: $r_S(t) = \prod_{i=1}^{n} r_i(t)$

Parallelstruktur 315

Das System überlebt, wenn mindestens ein Element überlebt.

Formal: $z_S(t) = \bigvee_{i=1}^{n} z_i(t)$

Überlebenswahrscheinlichkeit des Systems:

$$r_S(t) = 1 - \prod_{i=1}^{n} a_i(t) = 1 - \prod_{i_1}^{n}(1 - r_i(t))$$

k-aus-n Struktur 315

Das System überlebt, wenn mindestens k der n Elemente überleben.

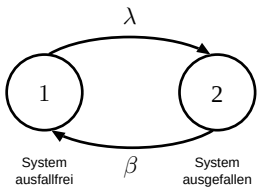
Die Überlebenswahrscheinlichkeit des Systems ist binomial verteilt.

Rechensysteme Stand: 12. März 2011

Rechensysteme Stand: 12. März 2011

▷ Charakterisierung eines Systems mit Fehlerbehebung?

System mit Fehlerbehebung 322



- Ausfallrate λ wird als konstant angenommen
- Ausfallzeit (Aufenthalt in Zustand 2) ist negativ-exponentiell verteilt: $F_{T2}(t) = 1 - e^{-\beta t}$
- Mittlere Ausfallzeit: $\frac{1}{\beta}$
- Behebungsrate β wird als konstant angenommen
- ausfallfreie Zeit (Aufenthalt in Zustand 1) ist negativ-exponentiell verteilt: $F_{T1}(t) = 1 - e^{-\lambda t}$
- Mittlere ausfallfreie Zeit: $\frac{1}{\lambda}$

▷ Was ist ein homogener markovscher Prozeß in endlicher Zeit?

▷ Wann ist ein Zustand erreichbar?

Homogener markovscher Prozeß 326

Prozeß $S[t]$ ist ein zustandsdiskreter Prozeß, denn...

- das System befindet sich immer in einem Zustand $s \in S \subseteq N$
- der Zustand $s(t)$ sei durch die Zufallsvariable S beschrieben, die einer zeitabhngigen Verteilung gehorcht.

Prozeß $S[t]$ ist ein Markov-Prozeß (Markov-Kette), denn...

- die Übergangswahrscheinlichkeit $P[S(t+\delta t) = j | S(t) = j]$ im Intervall $[t, t+\delta t]$ von i nach j überzugehen ist unabhängig davon, in welchem Verlauf der Zustand i eingenommen wurde.

  $\Rightarrow P[S(t+\delta t) = j \mid S(t) = i \ \wedge \ S(t-a) = k]$

  $= P[S(t+\delta t) = j \mid S(t) = i]$ (Markov-Eigenschaft)

Prozeß $S[t]$ ist homogen, denn...

- die Übergangswahrscheinlichkeiten hängen nicht von der Zeit ab:

  $P[S(t+\delta t) = j \mid S(t) = i] \quad = \quad P_{ij}(\delta t)$

Prozeß $S[t]$ in kontinuierlicher Zeit, denn...

- Zustandsübergangsrate soll zeitkonstant sein

erreichbar

Ein Zustand ist von einem anderen aus erreichbar, wenn die Wahrscheinlichkeit, daß er irgendwann eingenommen wird, größer Null ist.

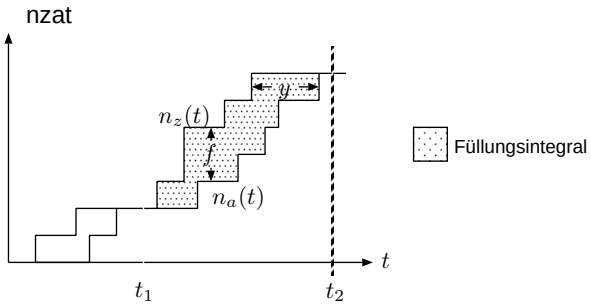
▷ Was ist die Littlesche Formel?

Littlesche Formel 342

operational: $f = d \cdot y$

stochastisch: $E[F] = E[D] \cdot E[Y]$

Zustandsgraph der Füllung im Wartesystem



$n_z(t) =$ angefangene Aufträge bis zum Zeitpunkt t
$n_a(t) =$ beendete Aufträge bis zum Zeitpunkt t

▷ Herleitung der Littleschen Formel?

Herleitung: Littlesche Formel 342

Füllung: $f(t) = n_z(t) - n_a(t)$

Zugang: $z(t_1, t_2) = \frac{n_z(t_2) - n_z(t_1)}{t_2 - t_1}$

Durchsatz: $d(t_1, t_2) = \frac{n_a(t_2) - n_a(t_1)}{t_2 - t_1}$

Zu den Zeiten t_1 und t_2 sind keine Aufträge in der Funktionseinheit. Es werden im Intervall alle zugegangenen Aufträge erledigt (Flußgleichgewicht). Also: $z(t_1, t_2) = d(t_1, t_2)$

Die Bedienzeit jedes Auftrags leistet einen Beitrag zum Füllungsintegral.

$$
\begin{aligned}
\overline{y} &= \frac{1}{n} \cdot \sum y \\
&= \frac{1}{n_z(t_2) - n_z(t_1)} \cdot \sum_{i=n_z(t_1)+1}^{n_z(t_2)} y_i \\
&= \frac{1}{n_z(t_2) - n_z(t_1)} \cdot \int_{t_1}^{t_2} f(t) \\
&= \frac{1}{n_z(t_2) - n_z(t_1)} \cdot \overline{f} \cdot (t_2 - t_1) \\
&= \frac{\overline{f}}{z} = \frac{\overline{f}}{d} \qquad \text{(wegen Flußgleichgewicht)} \\
\Longleftrightarrow \quad & \overline{f} = d \cdot \overline{y}
\end{aligned}
$$

▷ **Was ist ein Wartesystem?**

▷ **Was ist ein elementares Wartesystem?**

▷ **Was ist die Wartezeit?**

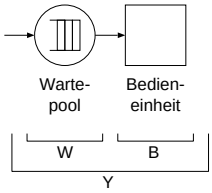
▷ **Was ist eine Bedienstation?**

Wartesystem ($\neq$ Verlustsystem) 348

System, in dem aktuell nicht ausführbare Aufträge warten, bis sie bearbeitet werden (kein Abbruch möglich).

Elementares Wartesystem 348

Nur eine Bedienstation — nur ein Wartekanal.



Wartezeit 348

Zeit, die ein Auftrag im Wartepool bis zur Bedienung verbringt (wegen Begrenzung der Kapazität der Funktionseinheit).

Bedienstation 348

Instanz, die aus einer oder mehreren Bedieneinheiten (entspr. Kapazität) besteht.

▷ **Was ist ein Auftragsankunftsprozeß?**

▷ **Was ist ein Bedienprozeß?**

▷ **Wie charakterisiert man elementare Wartesysteme?**

Auftragsankunftsprozeß 349

Der Auftragsankunftsprozeß gibt die Auftragsbeginnzeitpunkte an und ist eine Folge $(t_{z,i}) \quad i = [1 \ldots n]$

- In stochastischen Modellen wird meist angenommen, daß jede Auftragsankunft ein Erneuerungszeitpunkt des Prozesses ist.
- Der Prozeß wird durch die Verteilung der Zwischenankunftszeiten A beschrieben.
- Diese Verteilung A ist zeitunabhängig.

Fälle des Prozesses A

D (deterministisch):	A ist konstant
M (Markov):	A ist negativ-exponentiell verteilt $F_A(t) = 1 - e^{-\lambda t}$ (λ heißt Ankunftsrate)
G (general):	A ist eine beliebige Verteilung

Bedienprozeß 351

Die Folge der Bedienende-Zeitpunkte bildet den Bedienprozeß.

Charakterisierung elem. Wartesys. 353

Ankunftsprozeß / Bedienprozeß / Zahl der Bedieneinheiten / Kapazität des Systems (Wartepool+Bedienstation) / maximale Aufträge

Die letzten beiden Parameter entfallen, wenn sie ∞ sind.

▷ z.B. M/M/3/8

- **Was ist ein offenes System?**
- **Was ist ein geschlossenes System?**
- **Was ist ein Verkehrsengpaß?**
- **Was ist die Sättigungsfüllung?**
- **Was ist das Grenzdurchsatzgesetz?**
- **Was ist die Sättigungsfüllung?**

offenes System 359

System mit veränderlicher Füllung

geschlossenes System 359

System mit konstanter Füllung

Verkehrsengpaß 363

Funktionseinheiten mit der größten Auslastung

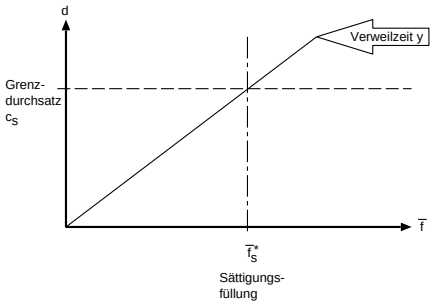
Sättigungsfüllung 386

$\overline{f_S}^* = c_S \cdot \overline{b_S}$

Grenzdurchsatzgesetz 385

$c_S = \frac{c_{VE}}{\overline{v}_{VE}}$

Sättigungsfüllung 386



- **Was ist ein Wartenetz?**

Wartenetz 392

Ein Wartenetz besteht aus

- einer Menge M von elementaren Wartesystemen $(1..|M|)$ (Knoten des Wartenetzes) mit je
  - m_i gleichen Bedieneinheiten
  - Kapazität k_i
  - Bedienstrategie BS_i
- seriell ausgeführten Auftragssystemen mit Teilaufträgen, die zu einer Klasse $r \in K$ gehören
  - Bedienzeit b_{ir} am Knoten i
  - Folgegesetzmäßigkeit $(i, r) \rightarrow (j, s) \quad (i, j \in M; r, s \in K)$

▷ Was ist ein elementares M/M/m/k-Wartesystem?

M/M/m/k-Wartesystem 393

- negativ-exponentiell verteilte Zwischenankunftszeiten A $F_A(t) = 1 - e^{-\lambda t}$
- negativ-exponentiell verteilte Bedienzeiten B $F_B(t) = 1 - e^{-\mu t}$
- m gleiche Bedieneinheiten
- Kapazität k

Es wird $m < k$ vorausgesetzt und die Bedienstrategie sei produktiv und restzeitunabhängig.

Die Füllung F bildet einen Markov-Prozeß, da der Ankunfts- und Bedienprozeß unabhängig sind.

mittlere Füllung

$$E[F] = \sum_{l=0}^{k} l \cdot p_l$$

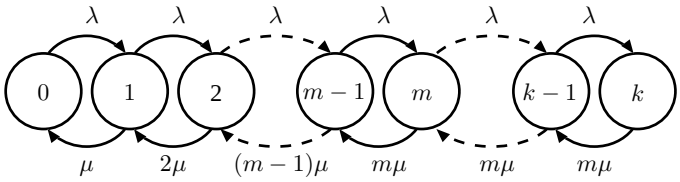
. . . denn ein Erwartungswert berechnet sich aus dem Ereignis (hier die Füllung l) multipliziert mit der Wahrscheinlichkeit, daß das Ereignis eintritt (hier $p_l = P[F = l]$)

▷ M/M/m/k ist eigentlich kein Wartesystem, weil Aufträge verloren gehen, wenn das System belegt ist.

▷ Wieso gilt das Flußgleichgewicht beim M/M/m/k-Wartesystem?

Flußgleichgewicht bei M/M/m/k 394

Zustandsgraph der Füllung im Wartesystem



Flußgleichgewicht bedeutet (S. 335):

$$\sum_{j=1,\ j\neq i}^{k} p_j \cdot \lambda_{ji} \quad = \quad \sum_{j=1,\ j\neq i}^{k} p_i \cdot \lambda_{ij}$$

Für die einzelnen Zustände gilt:

$$\begin{aligned} p_0 \cdot \lambda &= p_1 \cdot \mu \\ &\vdots \\ p_{k-1} \cdot \lambda &= p_k \cdot m \cdot \mu \end{aligned}$$

D.h. die Summe aller Flüsse von j nach i ist genauso groß wie die Summe aller Flüsse von i in alle Zustände j.

Dabei ist zu beachten, daß für jede Teilmenge der Zustandsmenge das Flußgleichgewicht gilt, wenn es für jeden einzelnen Zustand gilt.

▷ Was ist ein elementares M/M/1-Wartesystem?

M/M/1-Wartesystem 396

- negativ-exponentiell verteilte Zwischenankunftszeiten A $F_A(t) = 1 - e^{-\lambda t}$
- negativ-exponentiell verteilte Bedienzeiten B $F_B(t) = 1 - e^{-\mu t}$
- eine Bedieneinheit

mittlerer Durchsatz

$E[D] = \lambda$

▷ Zugang=Durchsatz!

mittlere Bedienzeit

$E[B] = \frac{1}{\mu}$

▷ Es gibt ja nur eine Bedieneinheit.

mittlere Füllung

$E[F] = \frac{\rho}{1-\rho}$

Bei $\rho \to 1$ wird die Füllung sehr groß, denn die Aufträge kommen nicht regelmäßig und wegen der negativ-exponentiellen Verteilung sind kürzere Zwischenankunftszeiten wahrscheinlicher. Trotzdem bleibt wegen des Flußgleichgewichts die Füllung endlich.

▷ Was ist ein elementares M/M/m-Wartesystem?

M/M/m-Wartesystem 393

- negativ-exponentiell verteilte Zwischenankunftszeiten A $F_A(t) = 1 - e^{-\lambda t}$
- negativ-exponentiell verteilte Bedienzeiten B $F_B(t) = 1 - e^{-\mu t}$
- m gleiche Bedieneinheiten
- unbegrenzte Kapazität

Beobachtungen

- Der Zustandsraum ist unendlich (keine Kapazitätsbeschränkung).

▷ Was ist ein elementares M/M/∞-Wartesystem?

M/M/∞-Wartesystem 395

- negativ-exponentiell verteilte Zwischenankunftszeiten A $F_A(t) = 1 - e^{-\lambda t}$
- negativ-exponentiell verteilte Bedienzeiten B $F_B(t) = 1 - e^{-\mu t}$
- unendlich viele Bedieneinheiten
- unbegrenzte Kapazität

Beobachtungen

- Jeder Auftrag wird sofort in Bedienung genommen — es gibt keine Wartezeit.

mittlerer Durchsatz

$E[D] = \lambda$

▷ Zugang=Durchsatz!

mittlere Füllung

$E[F] = E[D] \cdot E[Y] = \frac{\lambda}{\mu} = \rho$

mittlere Wartezeit

$E[W] = 0$

▷ Jeder Auftrag wird sofort bedient.

mittlere Verweilzeit

$E[Y] = E[B] = \frac{1}{\mu}$

▷ Wofür steht ρ (rho)?

▷ Was bedeutet irreduzibel?

▷ Was ist ein Absorptionszustand?

▷ Was ist ein stationärer Prozeß?

ρ (rho) 394

ρ ist die Auslastung einer Funktionseinheit

$\rho = \frac{E[Z]}{c} = \frac{\lambda}{m \cdot \mu}$

irreduzibel 335

Ein Zustandsraum ist irreduzibel, wenn alle Zustände des Zustandsraums untereinander erreichbar sind. (Keine Absorptionszustände!)

Absorbtionszustand 335

Ein Zustand, von dem aus kein anderer Zustand erreichbar ist.

stationärer Prozeß 336

Ein Prozeß, dessen Verteilung nicht von der Zeit abhängt.

▷ Was ist ein Jackson-Netz?

Jackson-Netz 397

Offenes Wartenetz mit den besonderen Eigenschaften an jedem Knoten:

- $k = \infty$
- produktive, restzeitunabhängige Bedienstrategie
- nur eine Auftragsklasse
- Ankunftsprozeß: Poisson
- Bedienprozeß: Poisson
- die Folgegesetzmäßigkeiten sind durch Wegewahlwahrscheinlichkeiten
  $p_{ij} (i \in M, j \in M \cup \{0\}, M$ Knotenmenge, 0 ist Außenwelt$)$
  gegeben

▷ Was ist ein Gordon-Newell-Netz?

Gordon-Newell-Netz 401

Geschlossenes Wartenetz mit den besonderen Eigenschaften an jedem Knoten:

- $k_i \geq n$
- es sind immer n Aufträge im Netz
- produktive, restzeitunabhängige Bedienstrategie
- nur eine Auftragsklasse
- Bedienprozeß: Poisson
- die Folgegesetzmäßigkeiten sind durch Wegewahlwahrscheinlichkeiten
  $p_{ij} (i \in M, j \in M \cup \{0\}, M$ Knotenmenge, 0 ist Außenwelt$)$
  gegeben

▷ **Was ist ein BCMP-Netz?**

BCMP-Netz 409

Wartenetz mit den besonderen Eigenschaften:

- an jedem Knoten gehört ein Auftrag genau einer Klasse $k \in K$ an
- die Folgegesetzmäßigkeiten sind durch Wegewahlwahrscheinlichkeiten $p_{ir;js}(i, j \in M, r, s \in K, M$ Knotenmenge$)$ gegeben
- durch die klassenabhängigen Wegewahlwahrscheinlichkeiten können Subketten gebildet werden, in denen das Netz offen oder geschlossen ist

???

▷ **Was ist eine Bedienstrategie?**

▷ **Wann ist eine Bedienstrategie. . .**

▷ **. . . nicht-verdrängend?**

▷ **. . . verdrängend?**

▷ **. . . produktiv?**

▷ **. . . fair?**

▷ **. . . overhead-frei?**

Bedienstrategie 351

Die Bedienstrategie legt die Reihenfolge der Aufträge fest, die vom System bedient werden sollen.

nicht verdrängend

Jeder Auftrag, dessen Bedienung bereits begonnen hat, wird erledigt, bevor ein anderer Auftrag bedient wird.

verdrängend

Die Ausführung eines Auftrages kann unterbrochen werden, während ein anderer Auftrag bearbeitet wird. Die Bearbeitung wird später wieder aufgenommen.

Verdrängende Bedienstrategien sind in der Praxis wichtig, weil ein Rechensystem kurze Aufträge bevorzugen kann, ohne die Bedienzeiten im voraus zu kennen. Dafür ist durch die Verdrängung die Strategie nicht mehr overhead-frei.

produktiv

Solange die Füllung $f <$ Kapazität k ist, wird ein Auftrag bedient.

▷ in schwach gefüllten Systemen ist jede Strategie wirkungslos

fair (gegen Aufträge einer Klasse K)

Kein Auftrag mit endlicher Bedienzeit aus der Klasse K muß unendliche lange warten, wenn ein später ankommender Auftrag aus K nur endlich lange warten muß.

overhead-frei

Die Bedienung eines Auftrages trägt nicht zur Füllung bei.

▷ **Was ist die Last?**

▷ **Was ist die Freiphase / Beschäftigungsphase?**

▷ **In welchem Modell untersucht man Bedienstrategien?**

Last 448

Die Last $u(t)$ (in einem elementaren Wartesystem) ist die Summe der zur Zeit t von einer Bedieneinheit noch abzuarbeitenden Bedienzeiten.

Freiphase / Beschäftigungsphase 449

Freiphase: Zeitintervall, in dem die Last null ist.
Beschäftigungsphase: Zeitintervall, in dem die Last größer null ist.

Untersuchungsmodell für Bedienstrategien 449

Die Bedienstrategien werden hier für ein M/G/1-Wartesystem untersucht. Der Ankunftsprozeß verläuft gedächtnislos (negativ-exponentielle Verteilung), denn das ist eine gute Näherung an realistische Ankunftszeiten. Die Bedienung ist aber zu speziell, deshalb ‚G'.

▷ **Was für eine Bedienstrategie ist FCFS?**

▷ **Was für eine Bedienstrategie ist LCFS?**

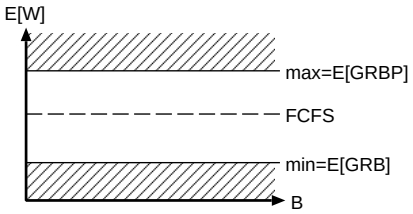
FCFS (FIFO) 455

- produktiv
- fair
- nicht verdrängend
- bedienzeitunabhängig
- die Aufträge werden in der Reihenfolge ihrer Ankunft bedient

Erwartungswerte in M/G/1

$E[W] = \frac{\lambda \cdot E[B^2]}{2 \cdot (1-\rho)}$ (Pollaczek-Khinchinsche Mittelwertformel)

Schranken



▷ Die mittlere Wartezeit ist direkt abhängig von der Bedienzeit.

LCFS 455

LCFS verhält sich wie FCFS und es gilt auch die Pollaczek-Khinchinsche Mittelwertformel. Allerdings ist LCFS nur für eine Auslastung $\rho < 1$ fair, weil ständig große Aufträge ins System kommen können, die einen Auftrag sehr lange warten lassen können.

▷ **Was für eine Bedienstrategie ist SPT?**

▷ **Was für eine Bedienstrategie ist LPT?**

SPT - shortest processing time first [468]

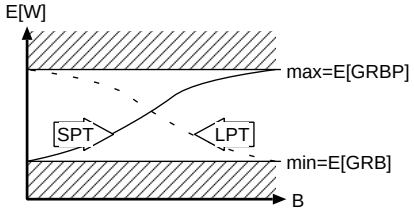
- nicht verdrängend
- fair für $\rho < 1$
- overheadfrei
- als nächster Auftrag wird der mit der kürzesten Bedienzeit ausgewählt

Bei $\rho \geq 1$ ist SPT nicht fair, denn es können sich ständig kürzere Aufträge vordrängeln.

LPT - longest processing time first [468]

- nicht verdrängend
- fair für $\rho < 1$
- overheadfrei
- als nächster Auftrag wird der mit der längsten Bedienzeit ausgewählt

Bei $\rho \geq 1$ ist LPT nicht fair, denn es können sich ständig längere Aufträge vordrängeln.



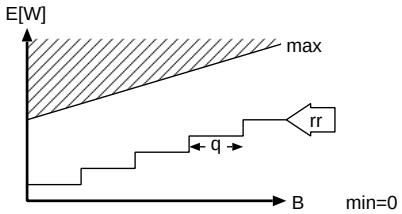
▷ **Was für eine Bedienstrategie ist Round Robin?**

Round Robin [490]

- verdrängend
- immer fair
- nicht overheadfrei
- Aufträge werden in der Reihenfolge der Ankunft in Bedienung genommen. Jeder unfertige Auftrag wird aber nach einer Zeitscheibe q verdrängt und der nächste unfertige Auftrag bedient.

$q \to 0$: Processor-Sharing.

$q \to \infty$: Das System arbeitet nach FCFS. Es findet keine Verdrängung statt.



▷ q wird beliebig klein gehalten (praktisch: 300 ms)

Die maximale Wartezeit ist eine wachsende Funktion, weil mit wachsender Bedienzeit auch die Wahrscheinlichkeit wächst, daß ein Auftrag in den Wartepool verdrängt wird.

▷ Was für eine Bedienstrategie ist SET?

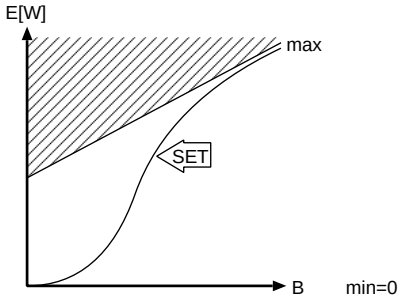
SET - shortest elapsed time next 490

- verdrängend
- immer fair
- nicht overheadfrei
- Das Strategie überprüft permament (in unendlich kleinen Zeitscheiben), welcher Auftrag die geringste bisherige Bedienzeit erreicht hat und führt ihn solange aus, bis ein anderer Auftrag eine geringere bisherige Bedienzeit hat.

$b \to 0$: der Auftrag wird ohne Verdrängung fertig bearbeitet

$b \to \infty$: der Auftrag wird nur noch bedient, wenn er alleine im System ist

SET bietet gegenüber Round-Robin eine überlegene Bevorzugung kurzer Aufträge trotz Vorhandenseins vieler langer Aufträge. Lange Aufträge sind allerdings extrem benachteiligt.



▷ Was ist eine rekursive Funktion?

▷ Was ist eine Rekurrenzgleichung?

▷ Was ist eine Relation?

▷ Was ist die transitive Hülle einer Relation?

▷ Was ist eine Funktion?

rekursive Funktion 7

Eine Funktion ist durch Rückgriff auf sich selbst definiert.

Rekurrenzgleichung 7

Schreibweise einer rekursiven Funktion

Relation 56

Teilmenge des Kreuzprodukts zweier oder mehrere Mengen. Binäre Relationen sind darstellbar durch Boolesche Matrizen oder gerichtete Graphen.

transitive Hülle 58

$R^+ = \bigcup_{i=1}^{\infty} R^i$ (transitiv)

$R^* = \bigcup_{i=0}^{\infty} R^i$ (transitiv und reflexiv)

aRb heißt: $(a, b) \in R$

aR^*b heißt: es gibt einen Pfad von a nach b

Funktion 57

Injektive Abbildung einer Wertemenge auf eine Bildmenge

▷ **Was bedeutet für eine Relation...**

▷ **... reflexiv?**

▷ **... symmetrisch?**

▷ **... transitiv?**

▷ **Was ist eine Äquivalenzrelation?**

Eigenschaften von Relationen 57

reflexiv

$a \in A \Rightarrow (a, a) \in R$

symmetrisch

$(a, b) \in R \Rightarrow (b, a) \in R$

transitiv

$(a, b) \in R \wedge (b, c) \in R \Rightarrow (a, c) \in R$

Äquivalenzrelation 57

R ist reflexiv, symmetrisch und transitiv.
$[a]_R = \{b \in A | (a, b) \in R\}$ heißt Äquivalenzklasse

▷ **Welche Funktionsklassen für die asymptotische Analyse gibt es?**

▷ **Was ist die charakteristische Funktion?**

▷ **Was sind die Fibonacci-Zahlen?**

Funktionsklassen für asymptotische Analyse 44

$O(g(x)) = \{f(x) \mid \exists c : |f(x)| \leq c \cdot |g(x)|\}$
(„wächst nicht schneller als“)
▷ Das $\leq$ darf nur für endlich viele x nicht gelten.

charakteristische Funktion (einer Menge) 53

$\chi_A(x) = [x \in A]$ („chi“)

Fibonacci-Zahlen 16

Rekursiv definierte Zahlenfolge:
$F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2}$ (für $n > 1$)

▷ Was sind erzeugende Funktionen?

Erzeugende Funktionen 22

$A(z)$ ist eine erzeugende Funktion für die unendliche Folge $\langle a_0, a_1, a_2, a_3, \ldots \rangle$, wenn:

$A(z) = a_0 + a_1 \cdot z + a_2 \cdot z^2 + a_3 \cdot z^3 + \cdots = \sum_{k \geq 0} a_k \cdot z^k$

$A(z)$ ist eine Potenzreihe mit (komplexer) Variable z.

$[z^n]A(z)$ steht für den Koeffizienten von z^n (nämlich: a_n)

▷ Eine geschlossene Formel repräsentiert eine unendliche Folge.

Beispiel

Gesucht: erzeugende Funktion für $\langle 1, 1, 1, 1, \ldots \rangle$.

Lösung: $\sum_{k \geq 0} z^k$ (kürzer: $\frac{1}{1-z}$)

Begründung:

1. $\sum_{k \geq 0} z^k$ ist nach bekannter Summenformel auch $\frac{1}{1-z}$
2. Die (vor dem z^k stehenden) Koeffizienten der Summe sind $1, 1, 1, 1, \ldots$

Anwendungen

- Mit erzeugenden Funktionen kann man Rekurrenzgleichungen lösen. Die erzeugende Funktion für die Fibonacci-Zahlen lautet z.B.: $\frac{z}{1-z-z^2}$
- Erzeugende Funktionen sind geschlossene Formeln, so daß man mit unendlichen Folgen einfacher „rechnen“ kann.

▷ Was ist ein Graph?

▷ Was ist ein Baum?

▷ Was ist ein Gerüst?

▷ Was ist ein Homomorphismus?

▷ Wann ist ein Graph gewichtet?

Graph 61

$G = (V, E)$

- V sind die Knoten (vertices)
- E sind die Kanten (edges)

Baum 65

Ein zusammenhängender ungerichteter zyklenfreier Graph.

Der Grad eines Knotens ist die Anzahl der Kanten.

Ein Blatt ist ein Knoten mit Grad 1.

n-ärer Baum: jeder Knoten hat maximal n Söhne

Homomorphismus 69

Substitution $\rho : \Sigma_1^* \to \Sigma_2^*$

gewichteter Graph

Der Graph hat eine Kostenfunktion:

Kst: $K \to R^+$

also: $G = (E, K, \text{Kst})$

▷ **Was bedeutet für einen Graph...**

▷ **. . . ungerichtet?**

▷ **. . . endlich?**

▷ **. . . azyklisch?**

▷ **. . . planar?**

▷ **. . . isomorph?**

▷ **. . . Weg?**

▷ **. . . Kreis?**

▷ **. . . Abstand?**

▷ **. . . Durchmesser?**

Graph

ungerichtet

Es gibt keine Kantenrichtungen — $(a, b) \in E \Leftrightarrow (b, a) \in E$

endlich

V (Menge der Knoten) ist endlich

azyklisch

Graph hat keine Kreise

planar

Graph kann ohne Überschneidungen in eine Ebene projiziert werden

isomorph (strukturgleich)

Es gibt eine Bijektion zwischen den Knoten der zwei Graphen

Weg

Folge von Kanten $(v_0, v_1)(v_1, v_2)(v_2, v_3)\ldots$

Kreis

Anfangsknoten (v_0) und Endknoten (v_n) sind identisch

Abstand $d(v, w)$

Länge des kürzesten Weges zwischen zwei Knoten v und w

Durchmesser

Maximaler Abstand zwischen zwei Knoten

$max\{d(v, w)|v, w \in V\}$

▷ **Was ist ein Alphabet?**

▷ **Was ist ein Wort?**

Alphabet Σ 67

endliche Menge unterscheidbarer und geordneter Symbole

Wort 67

Folge von Elementen eines Alphabets

- λ ist das leere Wort.
- Σ^n ist die Menge aller Wörter der Länge n
- Σ^* ist die Menge aller endlichen Wörter
- Σ^ω ist die Menge aller unendlichen Wörter
- $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$
- Wort $X = \overbrace{x_1 \ldots x_m}^{\text{Präfix}} \ldots x_n$
  Wort $X = x_1 \ldots \underbrace{x_m \ldots x_n}_{\text{Suffix}}$
- $x \leq y \quad \Leftrightarrow \quad x$ ist Präfix von y

▷ Was ist eine (formale) Sprache?

formale Sprache [68]

Menge von Wörtern (Zeichenketten aus Symbolen) eines Alphabets:

Formal: $L \subset \Sigma^*$.

Komplexprodukt: $L_1 L_2 = \{xy \mid x \in L_1, y \in L_2\}$

$L^0 = \{\lambda\} \qquad L^* = \bigcup_{i=0}^{\infty} L^i \qquad L^+ = \bigcup_{i=1}^{\infty} L^i$

Sprachen werden von Automaten akzeptiert bzw. von Grammatiken erzeugt.

▷ Was ist eine Grammatik?

Grammatik

Eine Grammatik ist ein 4-Tupel $G = (V, \Sigma, P, S)$.

- V ist Menge von Variablen
- Σ ist Menge von Terminalsymbolen (Terminalalphabet)
  $V \cap \Sigma = \emptyset$
- P ist Menge von Produktionen
  $P \subset (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$
- S ist Startvariable

$u \Rightarrow_G v$ (ein Wort u geht unter G über in v)
(u wird durch eine Produktion zu v – dabei können sich auch nur Teile von u ändern)

Die von G erzeugte Sprache ist: $L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$

▷ Das Ableiten mittels $\Rightarrow$ ist nicht-deterministisch!

▷ Grammatiken erzeugen Sprachen

▷ Was ist die Chomsky-Hierarchie?

Chomsky-Hierarchie

Einteilung von Sprachen in Klassen.

- Typ 0 — aufzählbare Sprachen
  jede Grammatik
  $P : (V \cup \Sigma)^+ \to (V \cup \Sigma)^*$
  ▷ Turing-Maschinen, 2-Keller, 2-Zähler
- Typ 1 — kontextsensitive Sprachen
  für alle Produktionen $w_1 \to w_2$ gilt: $|w_1| \leq |w_2|$
  ▷ linear beschränkte Turing-Maschinen
  ▷ monoton
  ▷ $a^i b^i c^i$ (bräuchte PDA mit 2 Kellern)
- Typ 2 — kontextfreie Sprachen
  $P : V \to (V \cup \Sigma)^*$
  ▷ nichtdeterminische Kellerautomaten, Transducer
  ▷ $a^i b^i$
- Typ 3 — reguläre Sprachen
  $P : V \to (\Sigma \cup \Sigma V)^*$
  ▷ endliche Automaten
  ▷ ab^i

Typ-3 $\subset$ Typ-2 $\subset$ Typ-1 $\subset$ Typ-0

Eine Sprache ist vom Typ x, wenn es eine Typ x-Grammatik gibt, die sie erzeugt.

▷ Über den Chomsky-Typ-0-Sprachen stehen noch die abzählbaren Sprachen.

▷ Welche Eigenschaften haben die Sprachen (Chomsky...)?

▷ Welche Automaten sind äquivalent?

Sprach-Eigenschaften Schöning–82

Typ	abgeschlossen...					Entscheidbarkeit	
	$\cap$	$\cup$	$\neg$	Produkt	Stern	a-transducer	Wortproblem	Äquivalenz
0	j	j	n	j	j	j	n	n
1	j	j	j	j	j	n	exp (PSPACE)	n
2	n	j	n	j	j	j	$O(n^3)$ (P)	n
3	j	j	j	j	j	j	$O(n)$ (P)	j

Äquivalenz von Automaten Schöning–82

deterministisch	nichtdeterministisch	äquivalent?
DFA	NFA	PSPACE
DPDA	NPDA	n
DLBA	NLBA	?
DTM	NTM	j

- Typ 0-Sprachen können gegen Komplement nicht abgeschlossen sein, weil sie ja sonst auch entscheidbar wären

▷ **Was ist die BNF?**

▷ **Was ist das Wortproblem?**

▷ **Was ist die Chomsky-Normalform?**

▷ **Was ist die Greibach-Normalform?**

Backus-Naur-Form (BNF)

Formalismus für kontextfreie (Typ 2) Grammatiken:

- $A \to b_1|b_2|b_3$
  Metaregel (A geht in b_1, b_2 oder b_3 über)

Weitere Vereinfachung durch erweiterte BNF (EBNF):

- $A \to a[b]c$
  das b ist optional
- $A \to a\{b\}c$
  das b kommt beliebig oft oder gar nicht vor

Wortproblem 109

Das Wortproblem für einen Automaten (analog: Grammatik) ist die Frage, ob ein Wort von einem Automaten akzeptiert wird.

Chomsky-Normalform Schöning–52

Alle Produktionen einer kontextfreien Grammatik haben die Form:

- $A \to a \mid VV$

Greibach-Normalform Schöning–54

Alle Produktionen einer kontextfreien Grammatik haben die Form:

- $A \to a \mid aVVV \ldots$

▷ **Was ist das Pumping-Lemma für Typ 3-Sprachen?**

Pumping-Lemma Schöning–39

uvw-Theorem (für reguläre Sprachen)

Ist L eine reguläre Sprache, dann gibt es eine Zahl n, so daß sich alle Wörter, die mindestens n Zeichen lang sind, in uvw zerlegen lassen, daß gilt:

1. $|v| \geq 1$
2. $|uv| \leq n$
3. $uv^iw \in L \quad \forall i \geq 0$

▷ Hauptanwendung: gibt es keine passende Zahl n, so ist L nicht regulär.

Beispiel: a^kb^k ist nicht regulär

Eine Zerlegung wäre: $x = \underbrace{\lambda}_{u}\underbrace{a^k}_{v^i}\underbrace{b^k}_{w}$ mit $n = 2 \cdot k$

Dann könnte man mit der uvw-Zerlegung aber auch das Wort a^kb^{k+1} erzeugen, die nicht in L ist.

▷ Was ist das Pumping-Lemma für Typ 2-Sprachen?

Pumping-Lemma Schöning–54

uvwxy-Theorem (für kontextfreie Sprachen)

Ist L eine kontextfreie Sprache, dann gibt es eine Zahl n, so daß sich alle Wörter, die mindestens n Zeichen lang sind, in $uvwxy$ zerlegen lassen, daß gilt:

- $|vx| \geq 1$
- $|vwx| \leq n$
- $uv^iwx^iy \in L \quad \forall i \geq 0$

▷ Hauptanwendung: gibt es keine passende Zahl n, so ist L nicht regulär.

▷ Welche Automaten gibt es?

Automaten und Maschinen (deterministische Versionen)

Maschine	Z, X, Y	Übergang / Ausgabe	Z_0	Z_{end}
DFA	Z, X	$\delta : Z \times X \to Z$	$\{q_0\}$	Z_{end}
2-DFA	Z, X	$\delta : Z \times (X \cup \{\$, c\}) \to Z \times \{L, H, R\}$	$\{q_0\}$	Z_{end}
h-2-DFA	Z, X	$\delta : Z \times X^h \to Z \times \{L, H, R\}^h$	$\{q_0\}$	Z_{end}
Moore	Z, X, Y	$\delta : Z \times X \to Z \ , \ \rho = Z \to Y$	$\{q_0\}$	–
Mealy	Z, X, Y	$\delta : Z \times X \to Z \ , \ \rho = Z \times X \to Y$	$\{q_0\}$	–
a-transducer	Z, X, Y	$K \subseteq Z \times X^* \times Y^* \times Z$	$\{q_0\}$	Z_{end}
gsm	Z, X, Y	$K \subseteq Z \times X \times Y^* \times Z$	$\{q_0\}$	Z_{end}
Büchi	Z, X	$K \subseteq Z \times X \times Z$	$\{q_0\}$	Z_{end}
Muller	Z, X	$\delta : Z \times X \to Z$	$\{q_0\}$	F
PDA	Z, X, Y	$\delta : Z \times \{X \cup \lambda\} \times Y \to Z \times Y^*$	z_0	
Rabin-PFA	Z, X	$\delta : Z \times X \to Z^*$	z_0	Z_{end}
LBA	Z, X, Y	$\delta : Z \times Y \to Y \times \{L, H, R\} \times Z$	$\{q_0\}$	Z_{end}
DTM	Z, X, Y	$\delta : Z \times Y \to Y \times \{L, H, R\} \times Z$	$\{q_0\}$	Z_{end}
NTM	Z, X, Y	$K \subseteq Z \times Y \to Y \times \{L, H, R\} \times Z$	$\{q_0\}$	Z_{end}

▷ **Was ist ein Akzeptor/Transduktor/Generator?**

▷ **Was ist ein Moore-Automat?**

▷ **Was ist ein Mealy-Automat?**

Akzeptor / Transduktor / Generator 72

- Akzeptor: endlicher Automat, der Worte akzeptieren kann (das Akzeptieren wird durch Endzustände angezeigt)
- Transduktor: endlicher Automat mit Ein- und Ausgabe (Endzustände spielen in der Regel keine Rolle)
- Generator: erzeugt nur eine Ausgabe (probabilistisch)

Moore-Automat 73

$M = (Z, X, Y, \delta, \rho, \{q_0\})$

- Z ist Zustandsmenge
- X ist Eingabealphabet
- Y ist Ausgabealphabet
- δ ist Übergangsrelation ($\delta : Z \times X \to Z$)
- ρ (Rho) ist Ausgabefunktion ($\rho : Z \to Y$) (Ausgabe auf den Zuständen)
- q_0 ist Startzustand

Mealy-Automat 74

$M = (Z, X, Y, \delta, \rho, \{q_0\})$

- Z ist Zustandsmenge
- X ist Eingabealphabet
- Y ist Ausgabealphabet
- δ ist Übergangsrelation ($\delta : Z \times X \to Z$)
- ρ (Rho) ist Ausgabefunktion ($\rho : Z \times X \to Y$) (Ausgabe bei den Zustandsübergängen)
- q_0 ist Startzustand

▷ **Was ist ein a-transducer?**

▷ **Was ist ein General State Machine (gsm)?**

▷ **Was ist die Nerode-Äquivalenz?**

▷ **Was ist die syntaktische Kongruenz?**

a-transducer 75

$M = (Z, X, Y, K, \{q_0\}, Z_{end})$

- Z ist Zustandsmenge
- X ist Eingabealphabet
- Y ist Ausgabealphabet
- K ist Kantenmenge ($K \subseteq Z \times X^* \times Y^* \times Z$)
- q_0 ist Startzustand
- Z_{end} ist Menge von Endzuständen

Ein a-transducer definiert eine partielle Funktion $A : X^* \to 2^{Y^*}$

▷ Nicht-deterministische Arbeitsweise!

General State Machine (gsm) 77

Deterministisches Modell eines buchstabierenden a-transducers mit der Kantenmenge: $K \subseteq Z \times X \times Y^* \times Z$

▷ Mealy-Automat der ganze Wörter ausgeben kann (Y^*)

▷ akzeptieren kontextfreie Sprachen

Nerode-Äquivalenz R_A 83

Zwei Eingabefolgen sind Nerode-äquivalent, wenn sie vom selben Zustand zu einem Endzustand führen.

$x \; R_A \; y \Leftrightarrow (z_0)^x \in Z_{end} \wedge (z_0)^y \in Z_{end}$

Syntaktische Kongruenz 83

$x \equiv_L y \Leftrightarrow uxv \in L \wedge uyv \in L \quad (\forall u, v, x, y \in \Sigma^*)$

▷ **Was ist die Dyck-Sprache?**

▷ **Wie lautet der Satz von Chomsky-Schützenberger?**

▷ **Was sagt der Satz von Myhill/Nerode aus?**

▷ **Was ist ein Monoid?**

Dyck-Sprache D_k 77

Die Dyck-Sprache D_k enthält alle richtig geklammerten Wörter mit verschiedenen Klammerpaaren.

Satz von Chomsky-Schützenberger 77

Zu jeder kontextfreien Sprache L gibt es eine reguläre Menge R_L und einen Homomorphismus h, so daß $L = h(D_k \cap R_L)$.

Satz von Myhill/Nerode Schöning–42

Eine Sprache ist regulär, wenn es nur endlich viele syntaktische Kongruenzen gibt.

Monoid 68

Monoid (A^*, K):

- K ist die Konkatenation
- Identität (leeres Wort)
- neutrales Element (λ)

Ist A ein Alphabet, dann ist es ein freies Monoid.

▷ **Was sind ω- und $\omega\omega$-Wörter?**

▷ **Was ist ein Büchi-Automat?**

▷ **Wann ist eine Sprache ω-regulär?**

ω-Wörter 89

- ω-Wörter: einseitig unendliche Wörter
- $\omega\omega$-Wörter: zweiseitig unendliche Wörter
- X^ω ist die Menge aller unendlichen Wörter
- $X^\infty := X^* \cup X^\omega$

Büchi-Automat 89

$A = (Z, X, K, \{q_0\}, Z_{end})$

- A ist λ-frei
- Z ist Zustandsmenge
- X ist Eingabealphabet
- K ist Kantenmenge ($K \subseteq Z \times X \times Z$)
- q_0 ist Startzustand
- Z_{end} ist Menge von Endzuständen

Ein ω-Wort $\in X^\omega$ wird akzeptiert, wenn mindestens ein Endzustand unendlich oft durchlaufen wird.
$L_\omega(A)$ ist die Menge der akzeptierten ω-Wörter.

ω-regulär 89

Die Sprache L ist ω-regulär, wenn...

- es gibt einen passenden Büchi- oder Muller-Automaten gibt
- $\overline{L}$ ist ω-regulär

▷ **Was ist ein Muller-Automat?**

▷ **Wie bildet man das Thue-ω-Wort?**

▷ **Wie bildet man das Fibonacci-ω-Wort?**

Muller-Automat 91

$A = (Z, X, \delta, q_0, \mathcal{F})$

- Z ist Zustandsmenge
- X ist Eingabealphabet
- δ ist Übergangsrelation
- q_0 ist Startzustand
- $\mathcal{F} \subseteq 2^Z$ ist Familie von Ankermengen

Der Automat akzeptiert ein ω-Wort, wenn alle unendlich oft durchlaufenen Zustände in einer der Ankermengen $F \in \mathcal{F}$ liegen.

Thue-ω-Wort 91

$$\begin{array}{lcllcl} x_0 & = & a & \varphi(a) & = & ab \\ x_i & = & \varphi(x_{i-1}) & \varphi(b) & = & ba \end{array}$$

Fibonacci-ω-Wort 92

$$\begin{array}{lcllcl} x_0 & = & a & \varphi(a) & = & ab \\ x_i & = & \varphi(x_{i-1}) & \varphi(b) & = & a \end{array}$$

▷ φ ist ein Homomorphismus ($\varphi : \Sigma^* \to \Sigma^*$)

▷ **Was ist Rabins Primzahltest?**

▷ **Was ist ein Minimalautomat?**

Rabins Primzahltest 93

Zur Berechnung von Primzahlen gibt es nur Algorithmen, die exponentiell viel Zeit verbrauchen. Rabins probabilistischer Primzahltest kann sehr schnell ausgeführt werden. Ist eine Zahl nach Rabins Test keine Primzahl, so ist das absolut richtig. Ansonsten ist die Wahrscheinlichkeit, daß die Zahl tatsächlich eine Primzahl ist, größer als $\frac{1}{2}$. Man kann den Algorithmus oft und schnell für eine Zahl ausführen, um den Fehler zu minimieren.

Minimalautomat Schöning–45

Minimale Version eines Automaten. Man formt Automaten in Minimalautomaten um, um sie leichter vergleichen zu können.

▷ Was ist ein (Rabinscher) probabilistischer endlicher Automat?

(Rabinscher) Probab. endlicher Automat 94

$A = (Z, X, \delta, z_0, Z_{end})$

- Z ist Zustandsmenge
- X ist Eingabealphabet
- δ ist Übergangsrelation ($\delta : Z \times X \to Z^*$)
  $\forall q \in Z, a \in X : \quad |\delta(q, a)| \leq 2$
- z_0 ist Startzustand
- Z_{end} ist Menge von Endzuständen

Überführung

$\delta(z_1, a) = \{z_2, z_3\}$ bedeutet: 50% Wahrscheinlichkeit für den Übergang zu z_2 oder z_3

$\delta(z_1, a) = \{z_2\}$ bedeutet: 100% Wahrscheinlichkeit für den Übergang zu z_2

Akzeptieren

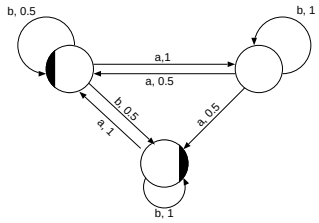
$p(w) = \sum_W p(W)$

W ist ein bewerteter Weg von q_0 zu einem Endzustand. $p(W)$ ist das Produkt der Kantenwahrscheinlichkeiten, die durchlaufen werden.

$L(A) = \{w \in X^* | p(w) > \frac{1}{2}\}$

▷ akzeptiert reguläre Mengen,

wenn Cutpoint = $\frac{1}{2}$



▷ Was ist ein deterministischer endlicher Zweiweg-Automat?

▷ Was ist eine Kreuzungsfolge?

Deterministischer endlicher Zweiweg-Automat (2DFA) 100

$A = (Z, X, \delta, \{z_0\}, Z_{end})$

- Z ist Zustandsmenge
- X ist Eingabealphabet
- δ ist Übergangsrelation $\delta : Z \times (X \cup \{\$, c\}) \to Z \times \{L, H, R\}$
- z_0 ist Startzustand
- Z_{end} ist Menge von Endzuständen

$\delta(z_1, x) = (z_2, L)$ bedeutet:
wenn der Automat sich im Zustand z_1 befindet und x liest, geht er in den Zustand z_2 und bewegt seinen Kopf nach links (L).

\$ und c sind die linken und rechten Bandbegrenzungen. Unbeschriebene Felder sind mit # gekennzeichnet.

Der Automat akzeptiert, wenn sich der Lesekopf über ein Bandende bewegt und in einem Endzustand landet.

2DFA akzeptieren reguläre Sprachen.

Kreuzungsfolge 102

Folge der Konfigurationen (Rechnung) eines 2DFA.

Kreuzungsfolgen von akzeptierenden Rechnungen haben:

- eine ungerade Länge
- keinen Zustand zweimal an unterschiedlichen (un)geraden Positionen (sonst unendliche Rechnung)

▷ Was ist ein Deterministischer endlicher Mehrkopf-Zweiweg-Automat?

▷ Was ist das Postsche Korrespondenzproblem?

h-Kopf-Zweiweg-Automat (h-2DFA) 106

$A = (Z, X, \delta, \{z_0\}, Z_{end})$

- Z ist Zustandsmenge
- X ist Eingabealphabet
- δ ist Übergangsrelation $\delta : Z \times X^h \to Z \times \{L, H, R\}^h$
- z_0 ist Startzustand
- Z_{end} ist Menge von Endzuständen

Die Begrenzungszeichen \$ und c sind hier Elemente von X.

h-2DFA haben keine λ-Übergänge.

Alle Köpfe starten auf dem ersten Symbol des Eingabewortes. Der Automat akzeptiert das Wort, wenn mindestens einer der Köpfe ein Bandbegrenzungszeichen überschreiten.

Postsches Korrespondenzproblem (PCP) 110

Für eine endliche Menge von Wortpaaren $(x_i, y_i)(i = [1 \dots k]$ und $x_i, y_i \in \{a, b\}^*)$ ist die Frage, ob es Indizes gibt, so daß $x_{i_1} x_{i_2} x_{i_3} \cdots = y_{i_1} y_{i_2} y_{i_3} \cdots$

Das PCP ist entscheidbar, wenn das Alphabet nur ein Symbol hat oder höchstens 2 Wortpaare vorliegen.

Variation

Das PCP für das Wort $x_{i_1} x_{i_2} x_{i_3} \cdots = y_{j_1} y_{j_2} y_{j_3} \cdots$ (verschiedene Indizes) ist entscheidbar, denn die beiden Wörter sind reguläre Ausdrücke und es reicht, die Äquivalenz zweier endlicher Automaten zu zeigen.

▷ Wie ist ein Algorithmus definiert?

▷ Was ist eine Turing-Maschine?

Algorithmus 117

Präzise, endliche Beschreibung eines allgemeinen Verfahrens durch elementare ausführbare Verarbeitungsschritte.

Turing-Maschine 118

Die Turing-Maschine ist ein formales Modell zur Berechenbarkeit von Algorithmen.

$A = (Z, X, Y, \delta, q_0, Z_{end})$

- Z ist Zustandsmenge
- X ist Eingabealphabet ($X \subset Y$)
- Y ist Bandalphabet
- δ ist Übergangsrelation $\delta : (Z \times Y) \to (Y \times \{L, H, R\} \times Z)$
- q_0 ist Startzustand
- Z_{end} ist Menge von Endzuständen

Turing-Tafel

δ	#	a	b
q_0	$\#Rq_1$	aLq_2	$\#Hq_0$
q_1	bRq_3	bHq_1	–
q_2	#–	$\#Rq_0$	aHq_3
q_3	#–	–	–

$L(A) = \{w \in Y^* \mid \exists u, v \in Y^*, q_e \in Z_{end} : \quad q_0 w \vdash^* u q_e v\}$

▷ Was bedeutet abzählbar?

▷ Was bedeutet rekursiv aufzählbar?

abzählbar

Eine Menge/Sprache $M \subseteq X^*$ ist abzählbar, wenn

- sie endlich ist
- gleichmächtig zu N ist (es gibt eine Surjektion $g : N \to M$)
  ▷ g muß nicht berechenbar sein
- es eine totale Funktion $f : N \to X^*$ gibt, so daß
  $M = \{f(0), f(1), f(2) \ldots\}$ („f zählt M auf“)

rekursiv aufzählbar 126

Eine Menge/Sprache $M \subseteq X^*$ ist rekursiv aufzählbar, wenn

- eine TM sie akzeptiert (DTM oder NTM)
  (aber nicht hält, wenn Wörter nicht in L(TM) sind)
- es eine totale und berechenbare Funktion $f : N \to X^*$ gibt, die M aufzählt:
  $M = \{f(0), f(1), f(2) \ldots\}$
  (es gibt eine TM, die jedes Wort genau einmal auf das Band schreibt)
- sie berechenbar ist

Unterschied aufzählbar $\leftrightarrow$ entscheidbar

- Eine TM hält immer auf einer entscheidbaren Menge. Sie landet in einem Endzustand, wenn ein Wort in $L(A)$ ist – ansonsten in einem anderen Zustand.
- Eine TM akzeptiert zwar eine aufzählbare Menge. Aber ihr Verhalten für Wörter, die nicht in $L(A)$ sind, ist nicht definiert. (Sie könnte unendlich lange laufen und nie halten.)

▷ Was bedeutet entscheidbar?

entscheidbar 126

Eine Menge $M \subseteq X^*$ ist entscheidbar, wenn

- die charakteristische Funktion ($\chi_M : X^* \to \{0, 1\}$) berechenbar ist.
- es eine Turingmaschine gibt, die auf jedem Element/Wort von M anhält (aber nicht im Endzustand, wenn $w \notin L(A)$)
- M und $\overline{M}$ aufzählbar (bzw. semi-entscheidbar) sind.

Nicht-entscheidbare Probleme

- busy beaver
- Halteproblem
- Komplement des Halteproblems
- nicht-triviale Eigenschaften nach Rice

semi-entscheidbar

Eine Menge $M \subseteq X^*$ ist semi-entscheidbar, wenn

- die „halbe“ charakteristische Funktion berechenbar ist:
  $$\chi'_M(w) = \begin{cases} 1, & w \in M \\ \text{undefiniert}, & w \notin M \end{cases}$$

▷ **Was ist eine Konfiguration?**

▷ **Was ist die Schrittrelation?**

Konfiguration 120

- Momentaufnahme einer Turing-Maschine
- formal: $w \in Y^* \times Z \times Y^*$
- Menge aller Konfigurationen: $KONF_M$

z.B. bedeutet $w = xzy$, daß die TM sich im Zustand z befindet, das Wort xy auf dem Band steht und das erste Zeichen von y unter dem Lesekopf steht.

Falls $y = \lambda$, dann steht der Lesekopf über einem Blank (#).

Schrittrelation 121

$\vdash_A \subseteq KONF_A \times KONF_A$

Die Schrittrelation beschreibt Übergänge zwischen Konfigurationen einer Turing-Maschine.

▷ **Was ist eine Rechnung?**

▷ **Was bedeutet (turing-) berechenbar?**

▷ **Wann sind zwei Turingmaschinen äquivalent?**

▷ **Was ist eine Offline-Turingmaschine?**

Rechnung 122

Folge von Konfigurationen.

(turing-) berechenbar 122

Eine Wortfunktion (Abbildung $f : X^* \to X^*$) ist berechenbar, wenn es eine TM gibt, so daß:

- $\forall w \in \Sigma^* : \; q_0 w \vdash_A^* q_e v$
- $f(w) = v$

▷ berechenbar = aufzählbar (Typ-0)

Äquivalenz von TMs 123

Zwei TMs sind äquivalent, wenn sie die gleiche Sprache akzeptieren.

▷ nicht entscheidbar

Offline-TM 124

Eine k-Band Offline-Turingmaschine hat:

- k unbeschränkte Arbeitsbänder (jeweils mit eigenem Lesekopf)
- ein Eingabeband (in beide Richtungen beweglich)
- ein Ausgabeband (nur nach rechts beweglich)

Zu jeder Offline-TM gibt es eine äquivalente TM mit nur einem Band.

▷ Was ist eine nichtdeterministische Turingmaschine?

▷ Was ist eine Gödelisierung?

Nichtdeterministische Turingmaschine 125

$A = (Z, X, Y, K, q_0, Z_{end})$

- Z ist Zustandsmenge
- X ist Eingabealphabet ($X \subset Y$)
- Y ist Bandalphabet
- K ist Übergangsrelation
  $K \subseteq Z \times Y \times Y \times \{L, H, R\} \times Z$
- q_0 ist Startzustand
- Z_{end} ist Menge von Endzuständen

▷ NTMs und DTMs sind äquivalent.

Gödelisierung 128

Eine totale, injektive, berechenbare Funktion i, die eine Menge auf $I\!N$ abbildet. i^{-1} ist berechenbar und $i(x)$ (die Bildmenge) ist entscheidbar.

Man kann z.B. Turingmaschinen über eine Gödelisierung aufzählen:

Turingmaschine A $\xrightarrow{i} 1$

Turingmaschine B $\xrightarrow{i} 2$

Turingmaschine C $\xrightarrow{i} 3$

▷ Was ist eine universelle Turingmaschine?

▷ Was ist eine zeitbeschränkte TM?

Universelle Turingmaschine 130

DTM, wobei

- Anfangskonfiguration ist $q_0\langle A\rangle\langle k_0\rangle$
- $\langle A\rangle$ ist die Kodierung der TM A über dem Alphabet $G = \{0, 1\}$.

Die UTM erhält als Eingabe die Kodierung (Binärzahl) einer Turingmaschine und die Kodierung einer Konfiguration (Binärzahl) und berechnet daraus alle Folgekonfigurationen (Binärzahlen). Die Kodierung muß natürlich vorher vereinbart worden sein.

Die TM selbst ist nur ein Tupel. Die UTM bringt die TM zum funktionieren und simuliert erst dessen Funktionsweise.

▷ Universelle TMs sind Modelle für Interpreter

Zeitbeschränkung bei TM

Die Turingmaschine akzeptiert eine Eingabe der Länge $|x|$ nur, wenn höchstens $f(x)$ Schritte für die Berechnung benötigt werden. Bei einer nichtdeterministischen Turingmaschine (NTM) gelten die benötigten Schritte der kürzesten Rechnung (es kann sonst auch unendliche Rechnungen geben).

Ein Wort wird also nur akzeptiert, wenn:

$\text{time}_M(x) \leq f(|x|)$

▷ **Was ist das Halteproblem?**

▷ **Warum ist das Halteproblem aufzählbar?**

▷ **Was ist die Number of Wisdom?**

Halteproblem 132

$H = \{\langle A\rangle\langle w\rangle |$ die TM A hält bei Eingabe von w an $\}$

w und H sind binär kodiert, also:

- $H \subseteq \{0,1\}^*$
- $w \in \{0,1\}^*$

▷ nicht entscheibar (Satz von Rice für Halteproblem)
▷ aufzählbar (s.u.)

Warum ist H aufzählbar? 132

Aufzählbar bedeutet ‚akzeptierbar durch eine TM'. Eine UTM könnte also alle Wörter "$< A >< w >$" akzeptieren, indem sie jeweils eine TM $< A >$ simuliert, die das Wort $< w >$ akzeptieren soll. Und das ist aufgrund der Definition des Halteproblems immer so.

Number of Wisdom 135

- $$\Omega = \sum_{U \text{ hält auf } p \in P} 2^{-|p|}$$
- P ist die Menge der präfixfreien Programme (Worte) einer UTM U
- Jedes Bit der binären Darstellung von Ω zeigt an, ob U auf dem entsprechenden Programm hält.

▷ **Was ist eine Eigenschaft der aufzählbaren Sprachen?**

▷ **Was ist der Satz von Rice?**

▷ **Was ist busy beaver?**

Eigenschaft 136

Teilmenge S von rekursiv aufzählbaren Sprachen. Die Eigenschaft S ist trivial, wenn $S = \emptyset$ oder S alle aufzählbaren Sprachen enthält.

Nur für nicht-triviale Eigenschaften gibt es Sprachen, die sie erfüllen und Sprachen, die sie nicht erfüllen.

Satz von Rice 136

Jede nichttriviale Eigenschaft S der aufzählbaren Sprachen ist unentscheidbar.

z.B.: Ist die Sprache L einer TM...

- endlich?
- unendlich?
- leer? ($S_{leer} = \{\emptyset\}$ hat ein Element!)
- regulär?
- kontextfrei?
- entscheidbar?
- eine Menge der Mächtigkeit 1?

busy beaver 132

$BB : N \to N$ beschreibt die Länge des längsten Blocks von Einsen, den eine TM mit n-Zuständen auf das Band schreibt und hält.

▷ nicht berechenbar

▷ **Was ist ein Keller-Automat?**

▷ **Was ist ein Zähler-Automat?**

Keller-Automat 138

$M = (Z, X, Y, \delta, q_0)$

- Z ist Zustandsmenge
- X ist Eingabealphabet ($\epsilon \in X$)
- Y ist Kelleralphabet
- δ ist Übergangsrelation ($\delta : Z \times X \times Y \to Z \times Y$)

▷ akzeptiert mit leerem Keller

Zähler-Automat 139

$M = (Z, X, Y, \delta, q_0)$

- Z ist Zustandsmenge
- X ist Eingabealphabet ($\epsilon \in X$)
- $Y = \{*\}$ ist Kelleralphabet
- δ ist Übergangsrelation ($\delta : Z \times X \times Y \to Z \times Y$)

Der Keller besteht nur aus $\$*^n$.

Bei einem 3-Zähler-Automaten bedeutet der Übergang $(z_1, x, +1, -1, 0, z_2)$: liest der Automat im Zustand z_1 das Symbol x, dann wechselt er in den Zustand z_2, und die drei Zähler werden um eins erhöht, um eins erniedrigt (sofern möglich) bzw. nicht verändert.

▷ **Was ist die Kolmogorov-Komplexität?**

▷ **Warum ist L_d nicht aufzählbar?**

Kolmogorov-Komplexität 140

Kolmogorov-Komplexität $K_S(x)$ eines Wortes x ist:

$$K_S(x) = \begin{cases} min\{|p|, S(p) = x\} \\ \infty, \text{wenn es kein } p \text{ gibt} \end{cases}$$

Die Kolmogorov-Komplexität $K_S(x)$ für eine Beschreibungsmethode S ist die Größe (Anzahl Bits) des kleinsten Programms p für eine UTM, die x ausgibt.

▷ minimale Information, die man braucht, um x zu definieren

Beschreibungsmethode

$S : \{0,1\}^* \to \Sigma^*$ ist eine TM-berechenbare Beschreibungsmethode

L_d nicht aufzählbar

Da H aufzählbar ist, wäre H auch entscheidbar, wenn $\overline{H}$ auch aufzählbar wäre. Nach Rice ist H aber unentscheidbar – und somit kann L_d nicht aufzählbar sein.

▷ Wann ist eine Funktion partiell rekursiv?

partiell rekursiv 143

Dies sind partiell rekursive Funktionen:

- die *Nullfunktion* ▷ $f(x) = 0$
- die *Nachfolgerfunktion* ▷ $f(x) = x + 1$
- *Projektionsfunktionen* ▷ $f(x_1, x_2, x_3) = x$
- Funktionen, die durch *Komposition* (Einsetzen) anderer primitiv rekursiver Funktionen enstehen (abgeschlossen unter Komposition)
- Funktionen, die durch *primitive Rekursion* entstehen. f muß die folgende Form haben:

$$\begin{aligned} f(0, \dots) &= g(\dots) \\ f(n+1, \dots) &= h(f(n, \dots), \dots) \end{aligned}$$

- Funktionen der Art $g(x_1, \dots, x_n) = \min\{y | f(y, x_1, \dots, x_n) = 0\}$ (abgeschlossen unter *Minimalisation*) Wichtig: es geht um das Minimum der Argumente, nicht der Funktionswerte!

▷ totale Funktion → rekursiv / nicht totale Funktion → partiell rekursiv

▷ partiell rekursiv = (turing-) berechenbar

▷ Wann ist eine Funktion primitiv rekursiv?

▷ Was ist die Ackermannfunktion?

primitiv rekursiv 143

Die Klasse der primitiv-rekursiven Funktionen ist ein formales Modell der Berechenbarkeit.

Dies sind primitiv rekursive Funktionen:

- die *Nullfunktion* ▷ $f(x) = 0$
- die *Nachfolgerfunktion* ▷ $f(x) = x + 1$
- *Projektionsfunktionen* ▷ $f(x_1, x_2, x_3) = x$
- Funktionen, die durch *Komposition* (Einsetzen) anderer primitiv rekursiver Funktionen enstehen (abgeschlossen unter Komposition)
- Funktionen, die durch *primitive Rekursion* entstehen. f muß die folgende Form haben:

$$\begin{aligned} f(0, \dots) &= g(\dots) \\ f(n+1, \dots) &= h(f(n, \dots), \dots) \end{aligned}$$

▷ primitiv rekursive Funktionen sind immer total

Ackermannfunktion 145

Eine partiell-rekursive totale Funktion, die nicht primitiv-rekursiv ist. Die Ackermannfunktion terminiert und wächst schneller als jede primitiv rekursive Funktion.

$$\begin{aligned} A(0, y) &= y + 1 \\ A(x, 0) &= A(x - 1, 1) \\ A(x, y) &= A(x - 1, A(x, y - 1)) \qquad x, y \in N \end{aligned}$$

▷ Unterschied TM – RAM?

▷ Was ist eine RAM?

Unterschied TM – RAM 153

TM

- Modell zur Untersuchung von Berechenbarkeit und Komplexität

RAM

- Modell zur Untersuchung von Entwurf und Analyse von Algorithmen und der Komplexität algebraischer und kombinatorischer Optimierungsprobleme
- realistischeres Modell eines realen sequentiellen Computers

RAM 153

Modell eines realen sequentiellen Computers (nicht von-Neumann, da das Programm nicht im Speicher steht!)

- hat Speicher (abzählbare Folge von Registern für Symbole oder beliebig lange Zahlen — initialisiert mit 0)
- Index der Register heißt Adresse
- Register R_0 heißt Akkumulator
- hat Programm (endliche numerierte Folge von Befehlen aus einer Befehlsmenge)
- Konfiguration läßt sich aus Datenspeichern und Befehlszähler IC bestimmen
- hat Eingabeband und Ausgabeband

▷ Was leistet eine RAM?

RAM leistet... 155

Berechnung von Funktionen

Das RAM-Programm P liest vom Eingabeband n Zahlen, berechnet daraus m Zahlen und schreibt sie auf das Ausgabeband. Dann ist P eine Funktion $P : N^n \to N^m$.

RAM-Programme berechnen genau die partiell rekursiven Funktionen.

Akzeptieren von Sprachen

Die Symbole des Eingabebandes werden durch natürliche Zahlen kodiert. Das Ende des Eingabewortes wird durch 0 markiert. Das RAM-Programm P akzeptiert das Eingabewort, wenn es eine 1 in die erste Stelle des Ausgabebandes schreibt.

RAM-Programme akzeptieren genau die rekursiv aufzählbaren Sprachen.

▷ Welche Komplexitätsmaße gibt es für RAMs?

▷ Wieviele Register braucht man bei einer Registermaschine, um alles berechnen zu können?

RAM-Komplexitätsmaße 160

Uniforme Komplexitätsmaße

- Uniformes Zeitmaß: 1 Schritt
- Uniformes Platzmaß: 1 Register
- ungeeignet bei Verarbeitung längerer Zahlen

Logarithmische Komplexitätsmaße

- Logarithmisches Zeitmaß: Zeit zum Speichern von Zahlen
- Logarithmisches Platzmaß: Länge der Binärdarstellung

Zeitkomplexität

Die Zeitkomplexität eines RAM-Programms ist die Summe der (logarithmischen/uniformen) Kosten aller Schritte, die das Programm durchläuft.

Platzkomplexität

Die Platzkomplexität eines RAM-Programms ist die Summe der (logarithmischen/uniformen) Kosten aller Register, die das Programm benutzt.

Benötigte Register für Berechenbarkeit

Zwei! Eine TM ist durch zwei Kellerautomaten beschreibbar (je ein Keller für linke und rechte Seite vom Lesekopf). Ein Kellerautomat ist durch einen 2-Zählerautomaten simulierbar. Und ein 4-Zählerautomat ist durch einen 2-Zählerautomaten simulierbar.

▷ Was ist eine RASP?

▷ Was sind unverzweigte Programme?

RASP (Random Access Stored Program machine) 162

Eine RASP ist eine RAM, dessen Programm im Speicher steht und während der Laufzeit modifizierbar ist. Das Programm mit r Befehlen wird in den Registern R_1 bis R_{2r} gespeichert. Jeder Befehl belegt zwei Register (Kodierung von Operation und Operand).

Bei der logarithmischen Zeitkomplexität eines RASP-Programms muß zur Länge der Operation auch die Länge der Adresse addiert werden!

▷ RASP-Modell entspricht von-Neumann-Architektur

Unverzweigte Programme 165

Modifikation von RAM-Programmen, so daß sie keine Verzweigungsbefehle enthalten.

Ein unverzweigtes Programm ist eine Folge von Zuweisungen $z_i \leftarrow x_i$ op y_i op $\in \{+, -, \times, /\}$ und x_i, y_i sind Konstante oder Eingabevariablen.

Wichtige Komplexitätsmaße

- Anzahl der Zuweisungen
- Anzahl der Zuweisungen mit der Operationen $\times$ und $/$

▷ **Kann eine RAM eine RASP simulieren?**

▷ **Kann eine RAM eine TM simulieren?**

Simulation RAM – RASP 163

RAM und RASP können sich gegenseitig simulieren, denn zu jedem RAM-Programm der Zeitkomplexität $T(n)$ gibt es ein äquivalentes RASP-Programm der Zeitkomplexität $c \cdot T(n)$ (und umgekehrt).

Simulation RAM – TM 168

Eine $T(n)$-zeitbeschränkte Offline-TM kann durch eine $O(T(n))$-zeitbeschränkte RAM_+ (ohne die Operationen Multiplikation und Division) simuliert werden.

▷ **Was ist eine PRAM?**

PRAM (Parallel-RAM) 173

PRAMs sind das Hauptmodell paralleler Rechner für Entwurf und Analyse paralleler Algorithmen.

Eine parallele Registermaschine (PRAM) ist eine (endlose) Folge von Prozessoren $P_1, P_2 \ldots$ Der Index von P dient der Identifikation und wird PID (Prozessor-ID) genannt. Jeder Prozessor ist eine RAM. Jeder Prozessor besitzt einen lokalen Speicher R_i (Register $R_{i,0}, R_{i,1} \ldots$) und einen globalen Speicher M (Register $M_0, M_1 \ldots$). Die Prozessoren kommunizieren lediglich über den globalen Speicher.

Alle Prozessoren führen synchron die Befehle ihrer Programme aus. Die Berechnung endet, wenn P_1 auf eine HALT-Operation trifft.

Modelle paralleler Computer (mit globalem Speicher)

- MIMD (Multiple Instructions Multiple Data)
  Jeder Prozessor führt ein eigenes Programm aus
- SIMD (Single Instructions Multiple Data)
  In jedem Schritt führen die Prozessoren einheitliche Befehle aus

▷ Wie löst man Konflikte beim PRAM-Speicherzugriff?

PRAM-Speicherzugriff 176

Es kann zu Konflikten beim gleichzeitigen Zugriff auf den globalen Speicher kommen. Deshalb gibt es drei Modelle:

- EREW (Exclusive Read – Exclusive Write)
  Simultanes lesen und schreiben ist nicht erlaubt.
- CRCW (Concurrent Read – Concurrent Write)
  Simultanes lesen ist erlaubt. Simultanes Schreiben ist erlaubt.
  - CRCW^{com} (common)
    Gleichzeitiges Schreiben nur erlaubt, wenn alle Prozessoren denselben Wert schreiben wollen.
  - CRCW^{arb} (arbitrary)
    Nur irgendein Prozessor darf schreiben.
  - CRCW^{pri} (priority)
    Der Prozessor mit dem kleinsten Index (der höchsten Priorität) darf schreiben.
- CREW (Concurrent Read – Exclusive Write)
  Gleichzeitiges lesen erlaubt. Nur exklusives schreiben.

▷ Welche Komplexitätsmaße für PRAMs gibt es?

Komplexitätsmaße PRAM 177

Zeitkomplexität $time_M(x)$

Anzahl der synchronen Schritte bis die Berechnung terminiert.

Prozessorkomplexität $proc_M(x)$

Anzahl Prozessoren, die an Schreibzugriffen beteiligt sind. (Je mehr Prozessoren desto weniger benötigte Zeit.)

Gesamtkomplexität $pt_M(x)$

Summe der Schritte aller Prozessoren, die zur Berechnung nötig sind. $pt_M(x) = time_M(x) \cdot proc_M(x)$

▷ Was ist das primäre Ziel der Komplexitätstheorie?

▷ Was ist ein Problem?

▷ Was ist eine Orakel-TM?

Ziel der Komplexitätstheorie 230

Die Klassifizierung von Problemen nach Bedarf an Rechenzeit und Speicherplatz.

Problem 230

- Menge P von Paaren (I, A)
- I ist ein Wort über dem Alphabet Σ und heißt Probleminstanz
- A ist ein Wort über dem Alphabet Σ und heißt Antwort zu I

Wichtig: die Kodierung des Problems I kann die Komplexität des Problems beeinflussen!

Entscheidungsproblem

Problem, bei dem die möglichen Antworten „ja“ und „nein“ sind. $P = \{I \mid (I, \text{'ja'}) \in P\}$

Orakel-Turingmaschine 231

Eine Orakel-TM ist eine normale TM mit einem zusätzlichen Orakelband. Geht die Orakel-TM irgendwann in den Zustand $q_?$, dann wird in einem Schritt der Zustand q_j oder q_n eingenommen — je nachdem ob der Bandinhalt rechts vom Lesekopf im Orakel enthalten ist oder nicht.

$L(M^A)$ ist die von M^A akzeptierte Sprache.

Ist das Orakel eine rekursive Menge, dann kann die Orakel-TM durch eine TM ohne Orakel simuliert werden.

Eine Orakel-TM tut so, als ob das Wortproblem direkt lösbar wäre.

▷ Wieso können Probleme Sprachen sein?

Problem $\Leftrightarrow$ Sprache

Bei Fragen der Entscheidbarkeit können Sprachen auch als Probleme dargestellt werden.

Sprache

- Menge von Wörtern
- z.B. $\{I \mid (I, \text{'ja'}) \in P\}$

Problem

- Menge von Paaren (Instanz, Antwort)
- z.B. (a ist in M, 'ja') , (b ist in M, 'nein')

▷ Was bedeutet reduzierbar?

reduzierbar 232

Eine Sprache L_1 ($\subseteq \Sigma_1^*$) heißt reduzierbar auf eine Sprache L_2 ($\subseteq \Sigma_2^*$), falls es eine totale und berechenbare Funktion (genannt: Reduktion)

$f: \Sigma_1^* \to \Sigma_2^*$

gibt, so daß

$x \in L_1 \iff f(x) \in L_2 \quad \forall x \in \Sigma_1^*$

▷ Symbolisch: $L_1 \leq L_2$

turing-reduzierbar

Es gibt eine polynomialzeitbeschränkte Orakel-DTM M^{L_2}, die L_1 akzeptiert, dann ist L_1 turing-reduzierbar auf L_2.

▷ Symbolisch: $L_1 \leq_p^T L_2$

polynomialzeit-reduzierbar

Die Berechnung der Reduktion f hat polynomiale Zeitkomplexität. Die Reduktion wird durch eine TM dargestellt, die $f(x) = y$ berechnet.

▷ Symbolisch: $L_1 \leq_p L_2$

logspace-reduzierbar

Die Reduktion f kann durch eine logarithmisch-platzbeschränkte Offline-DTM berechnet werden.

Logspace-reduzierbar impliziert polynomialzeit-reduzierbar.

▷ Symbolisch: $L_1 \leq_{log} L_2$

▷ Was ist time_M?

▷ Was ist DTime(t)?

▷ Was ist NTime(t)?

▷ Was ist DSpace(s)?

▷ Was ist NSpace(s)?

time_M

$\text{time}_M : \Sigma^* \to N$ ist die Anzahl der benötigten Rechenschritte der Turingmaschine M bei der Eingabe des Wortes x.

Komplexitätsklassen 244

▷ $s, t : N \to R$ sind Funktionen

DTime(t)

$\{L \mid L = L(M)$ für eine t(x)-zeitbeschränkte DTM $M\}$

▷ Menge aller Sprachen, die eine Offline-DTM in höchstens $t(x)$ Schritten akzeptiert. x ist die Länge des Eingabewortes.

NTime(t)

$\{L \mid L = L(M)$ für eine t(x)-zeitbeschränkte NTM $M\}$

▷ Menge aller Sprachen, die eine Offline-NTM in höchstens $t(x)$ Schritten akzeptiert. x ist die Länge des Eingabewortes.

DSpace(s)

$\{L \mid L = L(M)$ für eine s(x)-platzbeschränkte DTM $M\}$

▷ Menge aller Sprachen, bei dessen Akzeptierung eine Offline-DTM höchstens $s(x)$ Felder besucht. x ist die Länge des Eingabewortes.

NSpace(s)

$\{L \mid L = L(M)$ für eine s(x)-platzbeschränkte NTM $M\}$

▷ Menge aller Sprachen, bei dessen Akzeptierung eine Offline-NTM höchstens $s(x)$ Felder besucht. x ist die Länge des Eingabewortes.

▷ Was ist L / NL / P / NP / PSPACE / NPSPACE?

Komplexitätsklassen 244

L, NL, P, NP, PSPACE, NPSPACE

$$\mathsf{L} = \bigcup_{i \geq 1} \mathsf{DSpace}(i \cdot \log n)$$
$$\mathsf{NL} = \bigcup_{i \geq 1} \mathsf{NSpace}(i \cdot \log n)$$
$$\mathsf{P} = \bigcup_{i \geq 1} \mathsf{DTime}(n^i)$$
$$\mathsf{NP} = \bigcup_{i \geq 1} \mathsf{NTime}(n^i) \qquad (\stackrel{?}{=} \mathsf{P})$$
$$\mathsf{PSPACE} = \bigcup_{i \geq 1} \mathsf{DSpace}(n^i)$$
$$\mathsf{NPSPACE} = \bigcup_{i \geq 1} \mathsf{NSpace}(n^i) \qquad (= \mathsf{PSPACE})$$

▷ P ist die Menge aller Sprachen, die eine polynomial-zeitbeschränkte DTM akzeptiert. . .

▷ NP $\subseteq$ PSPACE, da in jedem Schritt höchstens ein Feld besucht werden kann.

▷ NPSPACE = PSPACE, da sich durch das nicht-deterministische Ableiten nur die Zeitkomplexität erhöhen kann.

L $\subseteq$ NL $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE = NPSPACE

L $\subset$ PSPACE, aber welche Teilmengeninklusion echt ist, weiß man nicht.

▷ Was ist ein Polynom?

▷ Was bedeutet zeit-/platzkonstruierbar?

Polynom

Eine Funktion $p : N \to N$ der Form
$p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$

zeit-/platzkonstruierbar 245

Eine Funktion $F : N \to R$ ist zeitkonstruierbar (platzkonstruierbar), wenn eine DTM bei jeder Eingabe der Länge n genau $\lceil f(n) \rceil$ Schritte macht (bzw. genau $\lceil f(n) \rceil$ Felder des Bandes benutzt).

▷ **Was bedeutet hart?**

▷ **Was bedeutet vollständig?**

▷ **Was bedeutet P-vollständig?**

▷ **Was bedeutet NP-vollständig?**

▷ **Was bedeutet PSPACE-vollständig?**

hart 232

Eine Sprache L heißt hart (bzgl. einer Reduzierbarkeit) für eine Menge von Sprachen C, falls jede Sprache aus C auf L reduzierbar ist.

$\forall L' \in C: \quad L' \leq L \quad \Rightarrow L$ hart für C

vollständig 232

Eine Sprache L heißt vollständig (bzgl. einer Reduzierbarkeit) für eine Menge von Sprachen C, wenn sie hart für C und $L \in C$ ist.

$\forall L' \in C\ ,\ L \in C: \quad L' \leq L \quad \Rightarrow L$ vollständig für C

▷ ein Problem gehrt zu den schwierigsten in einer Sprachklasse

P-vollständig 233

vollständig für P bezüglich logspace-Reduzierbarkeit

NP-vollständig 233

vollständig für NP bezüglich Polynomzeit-Reduzierbarkeit

▷ Alle NP-vollständigen Probleme sind polynomialzeit-isomorph.

PSPACE-vollständig 233

vollständig für PSPACE bezüglich Polynomzeit-Reduzierbarkeit

▷ **Welche Probleme haben welche Komplexität?**

Komplexität von Problemen (Beispiele)

P

- virtuelle Serialisierbarkeit
- Kruskal-Algorithmus

NP-vollständig

- SAT (Erfüllbarkeitsproblem der Aussagenlogik)
  3-SAT (. . . mit höchstens 3 Literalen pro Klausel in KNF)
- HAMILTON-KREIS (durchlaufen aller Knoten (Zyklus) in einem Graphen)
- Invarianten einer Inzidenzmatrix berechnen

▷ Wann ist eine Boolesche Formel erfüllbar?

▷ Was ist SAT?

erfüllbar

Eine Boolesche Formel F ist erfüllbar, wenn es eine Belegung $a_1, \ldots, a_n \in \{0,1\}^n$ der Literale gibt, so daß $F(a_1, \ldots, a_n) = 1$.

SAT 234

SAT (satifiability) ist das Erfüllbarkeitsproblem der Aussagenlogik.

SAT$= \{$Boolesche Formeln $F \mid F$ ist eine erfüllbare Formel$\}$

Exkurs Aussagenlogik

$F = F(x_1, x_2, \ldots, x_n)$ ist eine Boolesche Formel. $x_1, \ldots, x_n$ sind die Variablen (Literale). F ist eine Formel in KNF (konjunktiver Normalform — z.B. $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$ ist in 3-KNF).

Beweis: SAT ist NP-vollständig

Durch naives nicht-deterministisches Ausprobieren der Belegungen für die Variablen benötigt man $2^{\text{Anzahl Variablen}}$ Schritte. Die Zeitkomplexität ist also exponentiell.

(Der Beweis für die Vollständigkeit ist leider viel komplizierter.)

▷ Thema Berechenbarkeit?

Berechenbarkeit

- Maschinenmodell: Berechenbarkeit definiert über Turing-Maschine
- Funktionsklassen: rekursive Funktionen (gleichmächtig TM)
- Registermaschinen: RAM, RASP, PRAM

▷ Wie hängen die Eigenschaften zusammen?

Zusammenhänge der Eigenschaften