Betriebssysteme

Prof. Dr. P. P. Spies WS 2002/2003

Stefan Kugele
Technische Universität München
email: kugele@cs.tum.edu

7. Februar 2003

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Betriebssysteme

Rechensysteme sind *technische Systeme* mit Fähigkeiten zur *Speicherung* und *Verarbeitung* von Daten. Das Management eines Rechensystems und das geeignete Einsetzen von Hilfsmitteln (*Ressourcen Management*) ist Aufgabe eines *Betriebssystems*.
Dies wird durch geeignetes Einsetzen von Ressourcen erreicht. Wir unterscheiden zwischen passiven *Datenobjekte* und aktiven, sequentiellen Prozessen, den sogenannten *Rechenobjekte*.
Ziele sind die Planung, Steuerung, Kontrolle und Koordination von Berechnungen, die sich folgendermaßen charakterisieren lassen:

- sequentiell: Ausführen von sequentiellen Programmen
- Kombination davon:
  - unabhängig, also ohne Wechselwirkung ⇒ nebenläufige Systeme
  - zulässige Abhängigkeiten: Hierbei werden Wechselwirkungen toleriert und gezielt genutzt ⇒ Kooperation von Prozessen

1.2 Abriss der Geschichte von Rechen- und Betriebssystemen

Die geschichtliche Entwicklung von Rechen- und Betriebssystemen soll nun im Folgenden an Hand von fünf Phasen geschehen: Von den Anfängen der programmgesteuerten Rechensystemen in den 40er Jahren des vergangenen Jahrhunderts bis zu heutigen Systemen.

1. Phase: Die Anfänge: 1940 - 1955
Anfänge von programmgesteuerten Rechensysteme

   - ein Benutzer
   - keine Software
   - sehr leistungsschwach
   - Nutzung durch Spezialisten für die jeweilige Anwendung, Programmierung und Bedienung der Maschine

   Zum Beispiel, die *Z22*, eine Maschine mit analytischem Befehlscode. Dies sind Elementaroperationen mit sehr vielen Kombinationen, falls diese konfliktfrei sind. Durch Interpretation der Bitstellen kommt man zu den gewünschten Effekten.
   Dieses Vorgehen war sehr mühsam, ist aber als Grundlage sehr wichtig.

2. Phase: 1955 - 1965

   - Fortschritte der Hardware-Technik
   - Anfänge von Programmiersprachen (Hauptprogramme, Unterprogramme), Übersetzer, Binder, Lader, Automatisierung der Handhabung für elementaren Stapelbetrieb (Lochkartenstapel) mit Druckerlisten als Ausgabe ⇒ Monitor als elementare Betriebssysteme

3. Phase: 1965 - 1980
Leistungssteigerung durch:

   - verbesserte Technik (Hardware und Software)
   - verbesserte Organisation

   Gravierender Mangel: Diskrepanz der Verarbeitungszeiten für interne Operationen und Ein-/Ausgabe-Operationen (EA-Operationen)
   Lösung durch Mehrprogrammbetrieb und dann Nutzung. Die Benutzer arbeiten wieder direkt an den Rechnern
   ⇒ ausgereifter Mehrprogrammbetrieb einschließlich interaktiver Arbeit der Nenutzer.
   Dies geschieht durch verbesserte Techniken und insbesondere Langzeitspeicher (≡ Dateisystem).
   Verbesserte organisation führt zu Betriebssysteme, die effizienten Mehrprogrammbetrieb realisieren.
   Systeme:

- CTSS (computing time-sharing system) mit relativ einfachen Techniken
- MULTICS, wesentlich für alle Teilnehmersysteme
- UNIX-Basis
- Vernetzung mit elementaren Mitteln ARPA-Netz, Anfänge des Internets mit rudimentären technischen Mitteln

4. Phase: 1980 - heute
Wesentliche Fortschritte im Bereich

- der Hardware-Technik:

  Mainframe (Großrechner)
  $\Downarrow$
  PC's und Workstations

  wesentliche Verbesserung im Preis-Leistungs-Verhältnis
- Bereich der Nachrichtennetze $\Rightarrow$ Internet
- Software-Techniken
  * Programmier- und Programmentwicklung mit entsprechenden Hilfsmitteln
  * Datenbanken und Fatenbank-Systeme

  Durch Kombination kommen wir zu verteilten Systemen, das sind räumlich und geographisch verteilte, kooperative Informationsverarbeitungs-Systeme als universelle Hilfsmittel. Bis jetzt konnten verteilte Systeme erreicht werden, doch fehlt es an der notwendigen integration.

Kapitel 2

Grundlagen

2.1 Einführung der Hardware-Konfigurationen der Programm- und Datenstrukturen

2.1.1 Die Grundstruktur

Grundlegende Eigenschaften - von der Technik abgesehen - in der 1. Phase.
Von Neumann Konzept[1]

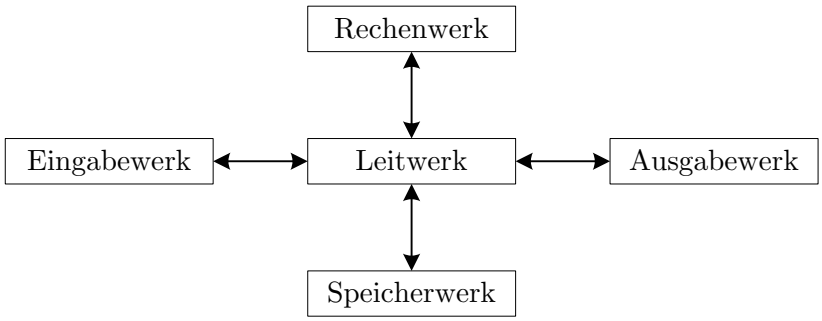


Abbildung 2.1: von Neumann Konzept

- Programmgesteuerte Berechnung mit den erforderlichen Hilfsmitteln
- Zentral: das Leitwerk steuert und kontrolliert alles
- Sequentielle Berechnungen

[1] Neumann, John von (1903-1957), eigentlich János von Neumann, amerikanischer Mathematiker ungarischer Herkunft.

Einfache Erweiterungen insbesondere mit Modularisierung und Funktionszuordnung
Dies war die Grundlage für Erweiterungen für die nächsten 30 Jahre.

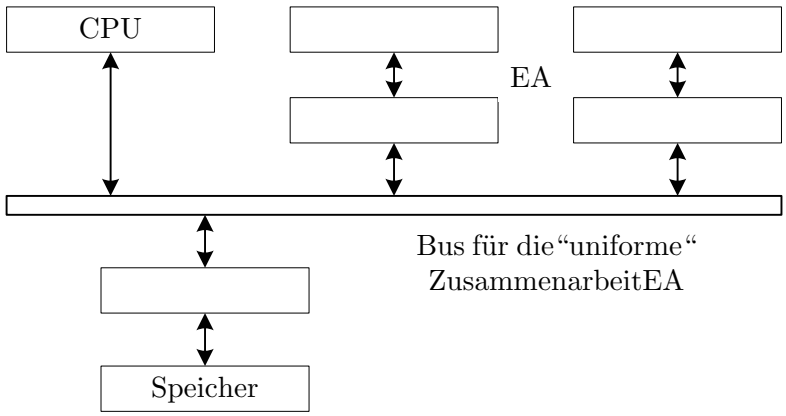


Abbildung 2.2: Systembus

Zu den Erweiterungen zählen:

(a) Speicherfähigkeit
Ausstattung einer Konfiguration mit einer Speicherhierarchie

→ CPU mit Registern: Erweiterung durch einen Zwischenspeicher/-Cache, der durch Hardware verwaltet wird

→ Arbeits-/ Hauptspeicher: Arbeitsspeicher mit wachsender Kapazität und abnehmender Lese-/ Schreibzeit
für die CPU direkt verfügbar

→ Hintergrundspeicher: wachsende Kapazität, wachsende Zugriffs-Lese-/ Schreibzeiten

→ Zugreifbare Einheiten:

– für Haupt- und Zwischenspeicher: Bytes und Wörter mit einigen Bytes

– für Hintergrundspeicher: Blöcke ($\frac{1}{2}$ k bis 8 k)

⇒ Entsprechende Zugriffs-/ Lese-/ Schreibzeiten

In dieser Speicherhierarchie gibt es zusätzlich Langzeit- und Archivspeicher, die durch notorische Speicher realisiert werden. ⇒ lange Ausführungszeit

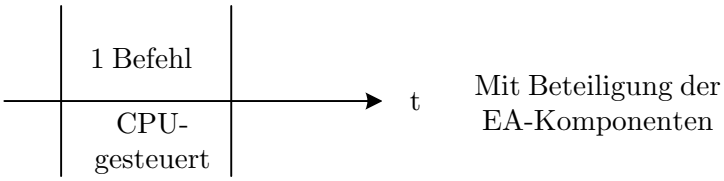


Abbildung 2.3: Ablauf-Zeitdiagramm

(b) E/A-Fähigkeiten

Vielfälltigere leistungsfähigere Nutzung insbesondere mit Grafik-Fähigkeiten.

(c) Rechen- und Verarbeitungsfähigkeiten:
→ Bessere Organisation

2.1.2 Funktions- und Leistungserweiterung

Erweiterungen, die wesentliche, unterschiedlichen Techniken nutzen:

- Interne Rechenoperationen: Elektronisch realisiert ⇒ kurze Ausführungszeiten
- Ein- und Ausgabeoperationen: elektromechanische realisiert ⇒ lange Ausführungszeiten

2.1.3 Ausführung der E/A-Operationen

- 1. Art der Ausführung:
  Programmiertes E/A: vollständig gesteuert und kontrolliert von der CPU
  mit einer Aufteilung in:
  - CPU-Anteile: Hierbei ist das *CPU-Warten* (busy waiting) westentlich.
  - EA-Werk-Anteil
- 2. Art der Ausführung:
  Aufteilung von Steuerung und Kontrolle auf CPU und EA
  unterbrechungsgesteuerte EA

Ausführungsschema

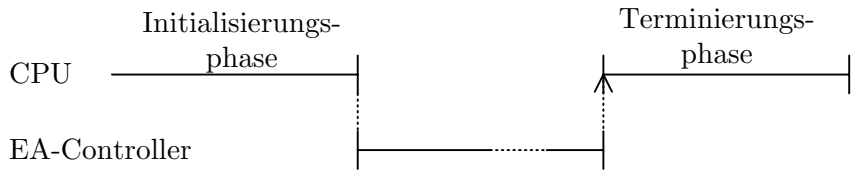


Abbildung 2.4: Ablauf-Zeitdiagramm

- Beauftragung des EA-Controllers einschließlich der Übergabe der Parameter: EA-Gerät, EA-Adresse, [Nachrichten]
- Beitrag des EA-Controllers
  - Steuerung der EA-Schnittstelle mit dem EA-Gerät
  - Erzeugung und Übergabe der Fertigmeldung an die CPU als Unterbrechungssignal an die CPU
- Initialbeitrag
- Überprüfung des Ausführungsergebnisses: Terminalbeitrag

⇒ Entlastung der CPU ⇒ die CPU kann in der Entlastungsphase weitere Berechnungen ausführen.

Wesentlich für die Effizienz:
Anzahl der mit einem EA-Befehl transportierten Elementarnachrichten / Bytes

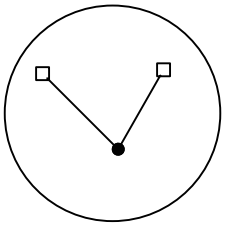
- Ein Byte (einige wenige)
- Byte-Folgen mit jeweils festlegbarer Länge
  dazu: Nachrichten im Arbeitsspeicher
  ⇒ Konkurenzkonflikte bei Zugriffen zum Arbeisspeicher. Dies wird dadurch aufgelöst, indem der EA-Controller Priorität vor der CPU hat.

Erstes, einfaches Beispiel für Nebenläufigkeit
hier: zur Effizienzsteigerung

$$\text{einfache Verhältnisse} \equiv \text{klar abgegrenzte Bereiche}$$
$$\equiv \text{keine Intereferenzen}$$

Verallgemeinerungen erfordern:

- genaue Analysen der Interferenzbereiche
- systematische Grenzkontrollen



Momentaufnahme des Zustandsraumes eines Prozesses

Verallgemeinerung auf Nebenläufigkeit

Abbildung 2.5: Momentaufnahme des Zustandsraumes

2.1.4 Von zentralen zu verteilten Hardware-Konfigurationen

- zentrale Hardware Konfiguration (physikalisch/räumlich)
- sequentiell ≡ lineare Folge von Berechnungsschritten
  Konzentration auf die Berechnungswirkung für einen Zeitpunkt: Zustand
  Konzentration auf die Berechnungswirkung für eine Folge: Zustandsfolge

Verallgemeinerung für Nebenläufigkeit Bei diesem Übergang sind zwei Schritte notwendig:

- Verteilung der EA-Geräte / Endgeräte ≡ Endgeräte bei den Benuterarbeitsplätzen ⇒ Nachrichtenübertragung mit geeigneten Techniken. Das System bleibt bezüglich Berechnungen im Wesentlichen *zentral*
- Verteilung der Rechenfähigkeiten zu den Benutzerarbeitsplätzen
  Nachrichtenübertragungstechniken:

  ⇒ vernetzte Systeme

  ⇒ Basis für verteilte Berechnungen
  Es sind jedoch noch die Integration und Nutzung für verteilte kooperative Problemlösungen zu leisten.

2.2 Von Programmen zu Prozessen und Datenobjekten

Komponenten nebenläufiger Systeme:

- Prozesse ≡ rechenfähige Objekte

- Datenobjekte ≡ speicherfähige Objekte
- Prozess ≡ Programm in Ausführung

Ein sequentielles Programm mit den zugehörenden Datenobjekten als Basis für die Einführung von Prozessen.
Für Programme:

- Programm-Module und Kompositionen
- Programmiersprachen

Für Erklärungen von Prozessen:

Programm-Module in Zwischen-/ Bindersprachen, Kompositionen zu vollständigen Programmen für Prozesse ≡ Beschreibung der Berechnungen, die ein prozess ausführen muss
Module ≡ Bausteine ≡ unvollständige Lösungen von Teilproblemen. Diese werden durch Auflösen der offenen Bezüge zwischen den Bausteinen aufgelöst.
Zwischensprache ≡ Pseujdo-Maschinensprache
⇒ Sei M die Menge von Modulen: $m \in M$:

Es muss zwischen den beiden Teilen unterschieden werden:

- Befehlsteil $b(m)$ definiert durch die Bytefolge $Nb(m)$
- Datenteil $d(m)$ definiert durch die Bytefolge $Nd(m)$

Symbolische Bezeichner für Elemente: $S(m)$. $s \in S(m) \equiv$ Bezeichner für die Objekte, die m definieren und die für m formal sind.

2.2.1 Programm-Module und Programme

Module sind *maschinennahe* also speicher- und fast ausführbare, *unvollständige* und *offene* Programm-Bausteine. Da diese nicht ausführfähig sind, bedarf es einer komponierenden Vervollständigung.

Im Folgenden werden nun die sechs Bestandteile eines Moduls genauer beschrieben:

- $desk(m)$: Deskriptor:
  Beschreibung des entsprechenden Datenobjekts
- $b(m)$: Befehlsteil:
  Der Befehlsteil ist ein Folge der mit und durch m definierten Befehle. Er ist als eine Bytefolge $Nb(m) = \{0, 1, \ldots, |b(m)| - 1\}$ codiert.

- $d(m)$: Datenteil:
  Der Datenteil ist ein Folge der mit und durch m definierten elementaren Datenobjekte. Er ist als eine Bytefolge $Nd(m) = \{0, 1, \ldots, |d(m)| - 1\}$ codiert.

- $ExT(m)$: External-Symbole bzw. Exporte von m
  Für m seien symboische Bezeichner definiert: $S(m)$
  $\Rightarrow \forall s \in S(m) \Rightarrow$ Element aus $Nb(m) \mid Nd(m)$ zugeordnet
  Eine Tabelle von Paaren $(s, n) : s \in S(m)$, n wird dem s zugeordnet
  s ist der numerische Bezeichner des Objekts: $s \in Nb(m)$
  Die Elemente von $S(m)$ sind eindeutig.

- $ImT(m)$: Tabelle der Importe von m
  Elemente (s, n) sind Paare mit:

  - s: symbolischer Bezeichner für ein Objekt, das m benutzt und nicht definiert $\Rightarrow$ s tritt frei in m auf
  - $n \in Nb(m)$ ist der numerische Bezeichner eines von und durch m definierten Objekts, in dem s (anwendend) auftritt. s kann in $b(m)$ mehrfach auftreten: hier mehrfach in $ImT(m)$

- $TralaT(m)$: Translationstabelle von m für die notwendigen Kompositionen
  hier: für die nicht lokal unterscheidbaren numerischen Bezeichner $\equiv$ Elemente aus $Nb(m)$ und $Nd(m) \Rightarrow$ Kennzeichnung dafür in $TralaT(m)$
  Die Translationstabelle ist für die *Kompositionen* wichtig.

Für jedes Modul m : $desk(m)$ und die weiteren Teile

Modulmenge M ist die Ausgangsbasis für die Kompositionen von Modulen zu „vollständigen“ Programmen. Hierbei meint vollständig *ausführar* Modul m in Bindersprache ist unvollständig und offen als Bausteine.

Für Modul m:

- *Bytefolgen* $\Rightarrow$ durchnumeriert, dies ist eine Basis für die interne Identifikation

- *Symbolische Bezeichner*: $S(m)$

- *Exporte*: $(symb, num)$

- *Importe*: $(symb, nummer)$
  nummer: hier: tritt symbolisch anwendend auf

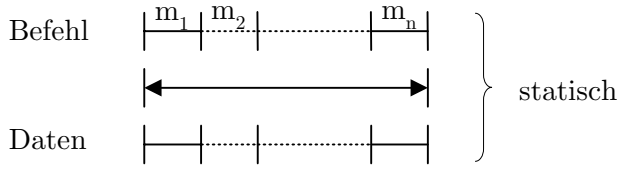


Abbildung 2.6: Befehlsabfolge

- *TralaT*: die anwendend auftretenden numerierten Bezeichner können mehrdeutig sein ⇒ hier wenden diese aufgelöst, und werden eindeutig

Modulmenge M: Basis für ausführbereite Programme
$\forall m \in M$: hier interessiert allein, dass alle anwendend auftretenden Bezeichner wohldefiniert sind. es wird gefordert, dass M vollständig ist.
In m symbolisch: in m muss es ein m' geben, mit $m' \neq m$, das das Symbol definiert.
Konstruktion *vollständiger*, ausführbereiter Programme jeweils mit M von $m_1 \in M$ ausgehend oder bei Bedarf konstruktion mit *zstatischem* Binden von M und $m_0 \in M$ ausgehend oder Konstruktion mit *dynamischen* Binden bei aktuellem Bedarf. Freie Bezeichner treten anwendend auf ⇒ Fortsetzen des Bindeprozesses

Mit diesen Modulen werden *ausführfähige* Programme konstruiert.

- sequentielle prtogramme für sequentielle prozesse
- Standard-Strukturten werden benötigt:
  Prozess-Programm in drei Teilen:
  - Befehlsbereich $b(m)$
  - Datenbereich $d(m)$: statisch definiert
  - Kellerbereich für die Berechnungsausführung

  ⇒: rechenfähige aktive Objekte, speicherfähige passive objekte

2.3 Einfache Systeme mit atomaren Prozessen

Nebenläufige Systeme mit sequentiellen Prozessen und Datenobjekten als Bausteine.

Hierbei ist folgendes wichtig:
Menge von Bausteinen
mögliche Abhängigkeiten (Wechselwirkungen, einfache, also nur mit wenig Wechselwirkungen). Satt technischen Systemen betrachen wir eir ein Modell bei dem wir uns auf die wichtigsten Eigenschaften beschränken um das Verhalten also die *Dynamik* zu beschreiben.
Die Dynamik, also die Veränderungen mit der zeit, ist bei diesen Systemen zur Informationsspeicherung und -verarbeitung von zentralener Bedeutung.

Zur Analyse der wesentlichen Eigenschaften dieser Systeme sind die folgenden Modelle zu nennen:

$\Rightarrow$ Gesetzmäßigkeiten / Regeln für die Konstruktion realer Systeme

$\Rightarrow$ Grundlage für die technischen Systeme

2.3.1 Uninterpretierte Systeme

Ein uninterpretiertes System mit atomaren Prozessen wird mit einem Quintupel $\mathfrak{A} = (D, A, \kappa, V, N)$ definiert, dabei gelten:

- D ist die nichtleere, endliche, linear geordnete Menge der *Datenobjekte*
- A ist die nichtleere, endliche, linear geordnete Menge der atomaren *Prozessen*
- κ ist die zwei-stellige *Koordinierungsrelation* auf $A \subseteq A \times A$
- $V, N: \ A \rightarrow \mathcal{P}\ (D) - \emptyset$
- V: ist die Vorbereichsfunktion für alle Prozesse
- N: ist die Nachbereichsfunktion für alle Prozesse

Für jeden Prozess $a \in A$ gilt: a führt eine Berechnung als unteilbare Einheit aus. Für die Berechnung von a ist $V(a)$ der Vorbereich und $N(a)$ der Nachbereich von a. Deise benutzen/benötigen Datenobjekte:
$d \in V(a)$: d ist Eingabe-Datenobjekt
$d' \in N(a)$: d' ist Ausgabe-Datenobjekt

atomar $\equiv$ unteilbar. Man unterscheidet zwischen den drei folgenden Arten von Atomarität:

- *w-atomar* $\equiv$ unteilbar in der Wirkung
- *t-atomar* $\equiv$ unteilbar in der Zeit

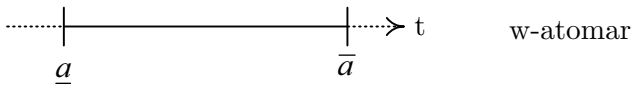


Abbildung 2.7: Zeitachse einer Berechnung

- *wt-atomar* ≡ unteilbar in Zeit und Wirkung

Beschreibung des Verhaltens eines Prozesses a bei der Ausführung seiner Berechnungen mit einem geordneten Paar von *wt-atomaren* Ereignissen:

$$a : [\underline{a},\ \overline{a}]$$

Hierbei ist $\underline{a}$ das *Initialisierungsereignis* und $\overline{a}$ das *Terminierungsereignis*

- Atomar in der Wikrkung
  funktionale Wirkungsbeschreibung ⇒ $\underbrace{\text{Eingabe} \longrightarrow \text{Ausgabe}}_{\text{funktionale Zuordnung}}$
  ⇒ atomar in der Wirkung - *w-atomar*, „Zusammenfassung“ von Eingabe, Transformation und Ausgabe

- Atomar in der Zeit
  Zeitintervall: $[t, t']$Zeitpunkt $t \leq t$
  ⇒ atomar in der Zeit: *t-atomar*

- Atomar in Wirkung und in der Zeit: *wt-atomar*

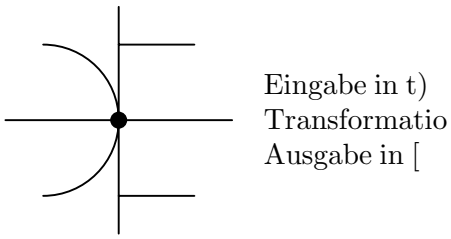


Abbildung 2.8: Eingabe in t), Transformation, Ausgabe in [

Prozess a
Ausführung einer Berechnung von a beschrieben durch ein geordnetes Paar von wt-atomaren Ereignissen

- *Initialisierungsereignis* $\underline{a}$ oder $\iota(a)$ ≡ die Ausführung der Berechnung beginnt

- *Terminierungsereignis* $\overline{a}$ oder $\tau(a) \equiv$ die Ausführung der Berechnung endet

$\Rightarrow$ $[\underline{a}, \overline{a}]$: geordnetes *Ereignispaar* oder Ereignisintervall
$[t(\underline{a}), t(\overline{a})]$: endlich, > 0, Extremfall: Länge $\to 0$
Für a: wt-atomare Ausführung von a

Für $\mathfrak{A} = (D, V, \kappa, V, N)$
Ausfürung einer Berechnung von $\mathfrak{A}$ wird durch Folgen/Spuren von Initialisierungs- und Terminierungsereignissen beschrieben

$$E(\mathfrak{A}) = \{\iota(a) | a \in A\} \cup \{\tau(a) | a \in A\}$$

Die Menge dieser Ereignis ist Basis für Beschreibungen des Verhaltens von $\mathfrak{A}$ bei der Ausführung von Berechnungen
$\iota(a) \equiv$ Lesen der Eingabe
$\tau(a) \equiv$ Schreiben der Ergebnisse

κ Koordinierungsrelation $S \subseteq A \times A$
Mit κ werden Regeln formuliert, mit denen erreicht wird, dass $\mathfrak{A}$ genau die *beabsichtigte* Berechnung durchführt.
Zunächst: $\kappa = \emptyset$; speziell $\kappa = \emptyset$: Beschreibung der Berechnungen $\mathfrak{A}$ mit den vollständigen Ereignisspuren von $\mathfrak{A}$.

Sei $\mathfrak{A} = (D, A, \kappa, V, N)$ ein *uninterpretiertes System* mit der Erignismenge $E(\mathfrak{A})$. Die Folge $\eta = \epsilon^1 \dots \epsilon^k$, $k \in \mathbb{N}_0$, mit Elementen aus $E(\mathfrak{A})$ ist eine *Ereignisspur* des Systems $\mathfrak{A} \Leftrightarrow$ es gelten:

- in η tritt jedes Element von $E(\mathfrak{A})$ höchstens einmal auf
- für $\epsilon^k = \overline{a}$ für $a \in A$ gibt es $j \in \{1, \dots, k-a\}$ mit $\epsilon' = \underline{a}$
- die Ereignisse treten in η verträglich mit κ auf
- $\eta = \epsilon^1 \dots \epsilon^k$ mit $k = |E(\mathfrak{A})| = 2|A|$ ist eine vollständige Ereignisspur von $\mathfrak{A}$

Verhaltensbeschreibungen mit Ereignisspuren
Für $\mathfrak{A}$ sei $H(\mathfrak{A})$ die Menge der *vollständigen Ereignisspuren.*
$H(\mathfrak{A})$ entspricht der großen Anzahl der Möglichkeiten für Berechnungen mit einem entsprechenden System.

Für $|A| = n$: $a \leq k \leq \dots 2^n - 1,\ k+1 \leq l \leq 2^n;\ a \in A$

$$\alpha = |H(\mathfrak{A})| :$$

$$\eta(n+1) = \epsilon^1 \dots \underbrace{\epsilon^k = \underline{a}}\, \epsilon^{k+1} \dots \underbrace{\epsilon^l = \overline{a}}\, \epsilon^{l+1} \dots \epsilon^{2n}$$

$$\eta(n-1) = \epsilon^1 \dots \epsilon^{k-1}\epsilon^{k+1} \dots \epsilon^{l-1}\epsilon^{l+1}\epsilon^{2n}$$

Vollständige Ereignisspur eines Systems mit $n-1$ Prozessen

$$\alpha(n) = \sum_{k=1}^{2n-1} (2n-k)$$

$$\alpha(n) = n(2n-1)\alpha(n-1) \text{ für } n \in \mathbb{N}, n > 1, \alpha(1) = 1$$

$$\alpha(n) = \frac{n(2n-1)!}{2^{n-1}}$$

$\Rightarrow$ Gezielte Nutzung dieser Möglichkeiten
$\Rightarrow$ Analyse der Verhaltensgesätzmäßigkeiten

$H(\mathfrak{A}) \longrightarrow$ um Wirkungsbeschreibungen zu ergänzen

2.3.2 Interpretierte Systeme

$\mathfrak{A} = (D, A, \kappa, N)$
D: Zustandsräume dür die $d \in D$
A: Rechenfunktionen für die $a \in A$

Ein Paar (Z, R) mit

- den Zustandsraum Z der Datenobjekte und
- der Familie $R = (R_a | a \in A)$ der Rechenfunktionen der Prozesse

ist eine *Interpretation* des uninterpretierten Systems $\mathfrak{A} = (D, A, \kappa, V, N)$.
Das Paar $(\mathfrak{A}, \mathcal{I})$ mit $\mathcal{I} = (Z, R)$ ist ein interpretiertes System mit atomaren Prozessen.
Für dieses System wird das Verhalten durch die für Anfangszustände $z^0 \in Z$ und Ereignisspuren $\eta \in E(\mathfrak{A})$ zugeordneten Zustandsspuren beschrieben.
$a \in A: \quad \eta = \epsilon^1 \dots \underbrace{\epsilon^i}_{\underline{a}} \dots \underbrace{\epsilon^j}_{\overline{a}} \quad \epsilon \in E(\mathfrak{A}) \; ; \quad \xi(z^0, \eta) = z^0 z^1 \dots^k$

- mit $\underline{a}$ ließt a die Eingabe für deine Berechnungen
- mit $\overline{a}$ schreibt a die Ausgabe für seine Berechnungen

Für die Definietion dieser Zustände sind technische Maßnahmen nötig: Projektionen und Einordnungen in die Zustandsräume.

Sei $\mathfrak{A} = (D, A, \kappa, N)$ ein uninterpretiertes System mit atomaren Prozessen und $\mathcal{I} = (Z, R)$ eine Interpretation von $\mathfrak{A}$, dann ist $(\mathfrak{A}, \mathcal{I})$ ein interpretiertes System mit atomaren Prozessen.
$\mathcal{I} = (Z, R)$

- Z: Für $d \in D$ sei $Z(d)$ ein nichtleerer, endlicher Zustandsraum von d $\Rightarrow Z \equiv \underset{d \in D}{\times} Z(d)$ der Zustandsraum der Datenobjekte

- R: Sei $a \in A$ mit $V(a), N(a)$ und R_a mit der Rechenfunktion von a

$$R_a : \underset{v \in V(a)}{\times} Z(v) \longrightarrow \underset{n \in N(a)}{\times} Z(n)$$

  linkstotale Abbildung. $R = (R_a | a \in A)$ ist die Familie der Rechenfunktionen.

Interpretiertes System $(\mathfrak{A}, \mathcal{I})$
Verhalten (Dynamik) von $\mathfrak{A}$ wird mit $\eta \in H(\mathfrak{A})$ beschrieben. Das Verhalten von $(\mathfrak{A}, \mathcal{I})$ wird dadurch beschrieben, indem den $\eta \in H(\mathfrak{A})$ *Zustandsspuren* zugeordnet werden: $\eta = \epsilon^1 \dots \epsilon^k$.
$\zeta(z^0, \eta) = z^0 z^1 \dots z^k \equiv$ die z^0 und η zugeordnete Zustandsspur. Auswahl ist

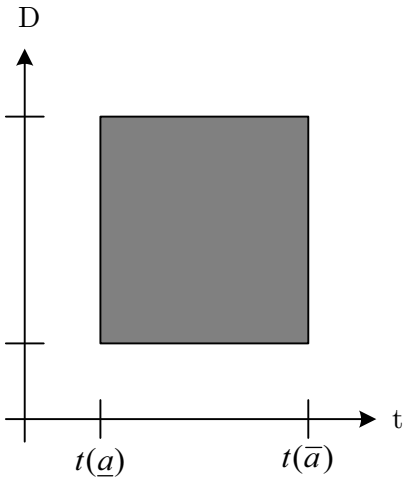


Abbildung 2.9: bezüglich D Auswahl und Einordnungen nötig

eine Projektion $\prod$, Mischoperator $\coprod$
Für η und $a \in A$: Seine $\epsilon^k = \underline{a}$ und $\epsilon^l = \overline{a}$. Für ein Initialisierungsereignis$\epsilon^k =$

$\underline{a}$ wird der zugehörige Zustand z^k der Semantik des Ereignisses entsprechend als Lesen des Zustands interpretiert:

$$z^k \equiv z^{k-1} \equiv z^k \text{ liefert die Eingabe für } R_a$$

Auswahl: $\prod^D_{V(a)}(z^k)$ als Eingabe
Für ein Terminierungsereignis $\epsilon^l = \overline{a}$ wird z^l dem von a berechneten Ergebnis entsprechen definiert:

- in $N(a)$ wird $R_a(\prod_{V(a)}(z^k))$ übernommen
- in $\overline{N(a)} \equiv D - N(a)$ wird das mit z^{l-1} erreichte übernommen
  $z^l \equiv \coprod^{(N(a),\overline{N(a)})}(R_a(\prod_{V(a)}(z^k)), \prod_{\overline{N(a)}}(z^{l-1}))$

Für den Anfangszustand $z^0 \in Z$ und der Ereignisspur $\eta = \epsilon^1 \dots \epsilon^k$ mit $k \in \mathbb{N}$ zugeordneten Zustandsspur $\zeta(z^0, \eta) = z^0 z^1 \dots z^k$ gelten:

- Für $k = 0$: $\zeta(z^0, \Lambda) \equiv z^0 - \Lambda$ [Zeichen für die leere Spur]
- Für $k \geq 1$:
  - Falls $\epsilon^k = \underline{a}$ mit $a \in A : z^k \equiv z^{k-1}$
  - Falls $\epsilon^k = \overline{a}$ mit $\epsilon^i = \underline{a}$ für $i < k$
    $z^k \equiv \coprod^{(N(a),\overline{N(a)})}(R_a(\prod_{V(a)}(z^i)), \prod_{\overline{N(a)}}(z^{k-1}))$

Das Verhalten des Systems $(\mathfrak{A}, \mathcal{I})$ wird beschrieben mit:

- der Menge der vollständigen Ereignisspuren $\eta \in H(\mathfrak{A})$ und
- der Menge der einem Anfangszustand $z^0 \in Z$ und $\eta \in H(\mathfrak{A})$ zugeordneten Zustandsspuren $\zeta(z^0, \eta)$ für alle $z^0 \in Z$.

Vollständige Verhaltensbeschreibungen erfordert Abstraktionen. Hierzu sind Verhaltenskenntnisse erforderlich: Verhaltensanalysen für Klassen von Systemen.

Die eingeführten vollständigen Verhaltensbeschreibungen enthalten Wiederholungen/Redundanzen, die eliminiert werden können ohne weitere Kenntnisse.
Für $(\mathfrak{A}, \mathcal{I})$ seien $\eta \in H(\mathfrak{A})$ und $z^0 \in Z$ und $\zeta(z^0, \eta)$ vollständige Beschreibungen

- Für $d \in D$ ist $\varrho(d, z^0, \eta)$ die z^0 und η zugeordnete Zustandsspur von d: $\varrho(d, z^0, \eta)$ ist die Folge der Zustände von d, die dem Anfangszustand bzw. den Terminierungsereignissen der $a \in A$ mit $d \in N(a)$ entspricht

- $\varrho(D, z^0, \eta)$ ist die Familie $(\varrho(d, z^0, \eta) | d \in D)$
- Seien $d \in D$ und $\varrho(d, z^0, \eta) = z^0(k) \dots z^l(k)$
  $\varphi(d, z^0, \eta) \equiv z^l(k)$ ist der Endzustan von d
- $\varphi(D, z^0, \eta) \equiv \underset{d \in D}{\times} \varphi(d, z^0, \eta)$ ist der Endzustand von D
  $z^0 \xrightarrow{\eta} \varphi(D, z^0, \eta)$

2.3.3 Stark determinierte Systeme

$(\mathfrak{a}, \mathcal{I})$ für $\mathfrak{a}$: $H(\mathfrak{a})$ und die zugehörigen Zustandsspuren
Für $d \in D, z^0 \in Z, \eta \in H(\mathfrak{a})$: $\varrho(d, z^0, \eta)$ vergleichen mit den entsprechenden Spuren für die $\eta' \in H(\mathfrak{a})$. Charakterisierung nach gleichen relevanten zugehörigen Zustandsspuren.
Strukturierung von $H(\mathfrak{a})$ durch Zerlegung bzw. *Äuivalenzrelationen.*

Die Spuren $\eta, \eta' \in H(\mathfrak{a})$ sind äquivalent genau dann wenn gilt: $\forall(d \in D, z^0 \in Z)\varrho(d, z^0, \eta) = \varrho(d, z^0, \eta'))$
Relation auf $H(\mathfrak{a})$ *reflexiv, symmetrisch, transitiv* $\Leftrightarrow$ *Äquivalenzrelation*
$\Rightarrow$ Zerlegung von $H(\mathfrak{a})$ nach dieser Äquivalenzrelation. Besonders einfach: Eine Äquivalenzklasse von H.

- Das System $(\mathfrak{a}, \mathcal{I})$ ist <u>*stark determiniert*</u> $\Leftrightarrow$ alle $\eta \in H(\mathfrak{a})$ sind äquivalent
- Das System $\mathfrak{a}$ ist <u>*stark determiniert*</u> $\Leftrightarrow$ für jede Interpretation $\mathcal{I}$ von $\mathfrak{a}$ ist $(\mathfrak{a}, \mathcal{I})$ stark determiniert.

$\mathfrak{a} = (D, A, \emptyset, V, N),\ H(\mathfrak{a})$
$\mathcal{I} = (Z, R)$
$(\mathfrak{a}, \mathcal{I})$: interpretiertes System

Äquivalenz von Ereignisspuren: $\eta, \eta' \in H(\mathfrak{a})$ für die $d \in D,\ z^0 \in Z,\ \varrho(d, z^0, \eta)$
$\eta \equiv \eta' \Leftrightarrow \forall d \in D, z^0 \in Z : \ (\varrho(d, z^0, \eta) = \varrho(d, z^0, \eta'))$

Das System $(\mathfrak{a}, \mathcal{I})$ ist *stark determiniert* $\Leftrightarrow$ für alle $d \in D,\ z^0 \in Z,\ \eta, \eta' \in H(\mathfrak{a})$ ist $\eta \equiv \eta'$
Die Spuren gemäß $H(\mathfrak{a})$ haben keinen *Einfluss* auf die Wirkungen der Berechnungen von $(\mathfrak{a}, \mathcal{I})$
Das System $\mathfrak{a}$ ist *stark determiniert* $\Leftrightarrow$ für jede Interpretation $\mathcal{I}$ von $\mathfrak{a}$ ist $(\mathfrak{a}, \mathcal{I})$ stark determiniert.
Bedingungen für starke Determiniertheit, Wirkungen der Berechnungen:
stark determinert: weniger restriktive Bedingungen sind sinnvoll

$\Rightarrow$ Unterscheidung zwischen uninterpretierten und interpretierten Systemen

Bedingungen für uninterpretierte Systeme:
$a, b \in A,\ a \neq b$
$V(a)$, sofern disjunkt zu $N(a)$: Eingabe für R_a ohne Veränderung
$N(a)$: Bereich für Veränderungen mit R_a

Die Prozessse $a, b \in A,\ a \neq b$, erfüllen die *Bernstein-Bedingung* $\mathcal{B}(a, b) \Leftrightarrow V(a) \cap N(b) = N(a) \cap V(b) = N(a) \cap N(b) = \emptyset$. Das System $\mathfrak{A}$ erfüllt die Bernstein-Bedingung $\mathcal{B}(\mathfrak{A}) \Leftrightarrow$ für alle $a, b \in A,\ a \neq b$, gilt $\mathcal{B}(a, b)$.
Charakterisierung stark determinierter Systeme:
Das System $\mathfrak{A}$ ist stark determiniert $\Leftrightarrow$ für $\mathfrak{A}$ gilt $\mathcal{B}(A)$

Beweis:

- $\Leftarrow$ Gilt offensichtlich, a keine Wechselwirkungen möglich sind
- $\Rightarrow$ Mit Widersprichsbeweisen für die drei Teile von $\mathcal{B}()$
  - $V(a) \cap N(b) \neq \emptyset\ nd \in V(a) \cap N(b)$
    $\Rightarrow$ Interferenzen; azu wesentlich: Nutzung der Uninterpretiertheit also Wahl einer „passenden“ Interpretation für d: Die Prozesse berechnen bei unterschiedlichen Eingaben Unterschiedliches!$\Rightarrow$ verschiedene relevante Spuren $\surd$

Für as interpretierte System $(\mathfrak{A}, \mathcal{I})$
Wenn für $\mathfrak{A}$ $\mathcal{B}(\mathfrak{A})$ gilt, dann ist $(\mathfrak{A}, \mathcal{I})$ stark determiniert.
$(\mathfrak{A}, \mathcal{I})$ stark determiniert und $\mathcal{B}(\mathfrak{A})$
Was berechnet $(\mathfrak{A}, \mathcal{I})$ mit $\mathcal{I} = (Z, R)$
$a \in A, R_a : \underset{v \in V(a)}{\times} Z(v) \longrightarrow \underset{n \in N(a)}{\times} Z(n)$
Für $\mathfrak{A}$ gilt $\mathcal{B}(\mathfrak{A})$: Keine Interferenzen!
System $\mathfrak{A}$ berechnet $R_a \equiv \underset{a \in A}{\times} R_a$ im Allgemeinen partiell auf Z.

Analyse des Verhaltens von komplizierteren Systemen

- Dazu ntwenig: Abstraktionsprozesse bereits für Analysen
- Dazu notwendig Unterscheidung zwischen wichtigen und unwichtigen Eigenschaften. Kriterien hierfür: Dazu muss das System bekannt sein.

Beispiel: Die streng determinierten Systeme
Atomarität: $\left\{ \begin{array}{c} \text{Raum und Zeit} \\ \equiv \\ \text{Wirkungsbereiche} \end{array} \right\} \begin{array}{c} \text{Berechnungen} \\ \equiv \\ \text{beschreiben} \end{array}$

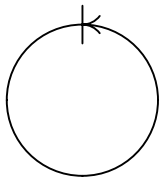


Abbildung 2.10: Kreisschluss wird durch Experimente und Beobachtungen aufgebrochen ⇒ iterativ!

t-atomar, w-atomar ⇒ wt-atomar

Prozessmenge $A, a \in A$ Prozess: $\underline{a}, \overline{a}, V(a), N(a), R_a$ 𝒶 mit A streng deter-

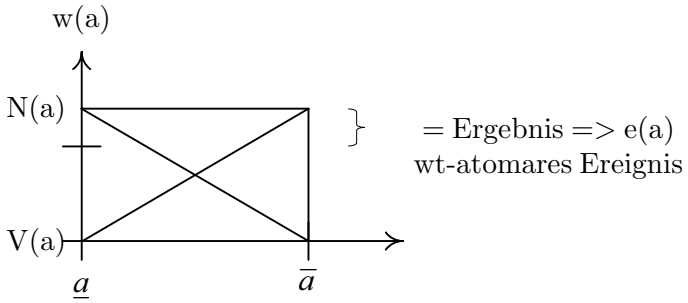


Abbildung 2.11: $wt - atomares$ Ereignis

ministisch
$a \in A \rightarrow e(a)$
$A \Rightarrow (e(a)|a \in A) \Rightarrow R_a = \underset{a \in A}{\times} R_a$
$\Downarrow$
$\boxed{e(\mathcal{a})}$

⇒ Spezielle Systeme mit wesentlichen Eigenschaften

Abschwächungen mit:

- weniger strengen Anforderungen an die Systeme
- Selektion bezüglich wesentlichen Eigenschaften

2.3.4 Serialisierbare Berechnungen

System $(\mathfrak{A}, \mathcal{I})$ mit atomaren Prozessen
Verhalten: $H(\mathfrak{A})$ und zugehörige Zustandsspuren $\rightarrow$ in $H(\mathfrak{A})$ gibt es Spuren mit seriellen Berechnungen und mit diesen äquivalenten Spuren [serialisierbar]

Begriffe und Eigenschaften:
System $\mathfrak{A}$ mit A, $(\mathfrak{A}, \mathcal{I})$, $H(\mathfrak{A})$
Sei $\eta = \epsilon^1 \ldots \epsilon^k$ eine Ereignisspur von $\mathfrak{A}$. Der Prozess $a \in A$ ist aktiv in $\epsilon^k \Leftrightarrow \eta$ enthält $\underline{a}$ und η enthält $\overline{a}$ nicht. η ist Ereignisspur einer *seriellen Berechnung* von $\mathfrak{A}$. $\Leftrightarrow$ für alle $i \in \{1, \ldots, k\}$ gilt: Höchstens ein Prozess ist aktiv in ϵ^i.

Seien $a, b \in A, a \neq b$: Die Prozesse a und b führen ihre Berechnungen in ϵ^k *nebenläufig* aus $\Leftrightarrow$ a und b sind aktiv in ϵ^k.
η ist Ereignisspur einer nebenläufigen Berechnung von $\mathfrak{A} \Leftrightarrow$ es gibt $i \in \{1, \ldots, k\}$ so dass mehrere Prozesse von $\mathfrak{A}$ in ϵ^i aktiv sind.
$H(\mathfrak{A})$: Sei $\Sigma(\mathfrak{A}) \subset H(\mathfrak{A})$ die Menge der vollständigen Ereignisspuren serieller Berechnungen con $\mathfrak{A}$.
$\sigma \in \Sigma(\mathfrak{A})$: beschreibbar mit einer Permutation Π von $\mathfrak{A}$: Für $\Pi = (\pi(1), \ldots, \pi(n))$ für $|A| = n \Rightarrow |\Sigma(\mathfrak{A})| = n!$
Für $(\mathfrak{A}, \mathcal{I})$ mit $\mathcal{I} = (Z, R) \Rightarrow \mathfrak{A}$ berechnet gemäß Π

$$\bigodot_{i=1}^{n} R_{\pi(1)} \circ R_{\pi(2)} \circ \ldots \circ R_{\pi(n)}$$

für die Wirkung der entsprechenden Berechnungen.

Auswahl der Berechnungen, die diesen einfachen Gesetzen genügen:
Für $(\mathfrak{A}, \mathcal{I})$ ist $\eta \in H(\mathfrak{A})$ Ereignisspur einer serialisierbaren Berechnung $\Leftrightarrow$ es gibt $\sigma \in \Sigma(\mathfrak{A})$ mit $\eta \equiv \sigma$.
Für $\mathfrak{A}$ ist $\eta \in H(\mathfrak{A})$ Ereignisspur einer serialisierbaren Berechnung von $\mathfrak{A} \Leftrightarrow$ für jede Interpretation $\mathcal{I}$ von $\mathfrak{A}$ ist η Ereignisspur für $(\mathfrak{A}, \mathcal{I})$.

Für $\mathfrak{A}$ bzw. $(\mathfrak{A}, \mathcal{I})$:

- $H(\mathfrak{A})$ sind die vollständigen Ereignisspuren
- $\Sigma(\mathfrak{A})$ sind die vollständigen seriellen Ereignisspuren
- $\Sigma^*(\mathfrak{A})$ sind die vollständigen serialisierbaren Ereignisspuren

$\rightarrow$ die $\sigma^* \in \Sigma(\mathfrak{A})$ sind die zulässigen serialisierbaren Ereignisspuren

$\Rightarrow$ Für Beschreibungen des Verhaltens von $\mathfrak{A}$ bzw. $(\mathfrak{A}, \mathcal{I})$ sind die $\sigma^* \in \Sigma(\mathfrak{A})$ bzw. die zugehörigen Zustandsspuren die zugelassen sind.
$\Rightarrow$ Entsprechende Beschränkungen

Beschränkung auf serialisierbare Spuren von $\mathfrak{A}$ bzw. $(\mathfrak{A}, \mathcal{I})$
Für Datenbanken üblich für Transaktionen (statt Prozessen)
Beschränkung: Vorgehen:

- konstruiere $\eta \in H(\mathfrak{A})$
- überprüfe η auf Linearisierbarkeit $\{\overset{\eta}{j}, \overset{\neg\eta}{n}\}$

Dazu erforderlich:

- Vorgehensweise
- Kriterien

$\eta_2 = \underline{a}|\underline{b}\ldots$
$\eta_3 = \underline{a}|\underline{b}\ldots$
ist nicht serialisierbar; mit $\underline{a}$ ist entschieden, dass allein mit $\overline{a}$ fortsetzbar, wenn Serialisierbarkeit verlangt ist.

$\Rightarrow$ zweckmäßig sind Verfahren, die serialisierbare Vervollständigungen sukzessive prüfen und gewährleisten.

Serialisierbarkeit:
$a \in A; V(a), N(a), R_a; \eta = \underset{\underline{\underline{a}}}{\epsilon^i} \ldots \underset{\overline{\overline{a}}}{\epsilon^j}$

a, also R_a berechnet das Ergebnis für a aus der Eingabe, also:
(ϵ^i, ϵ^j)
$(z^i, z^j) \Rightarrow \boxed{\prod_{N(a)}(z_j) = R_a(\prod_{V(a)}(z^i))}$

Diese Gesetzmäßigkeiten müssen bei Überführungen von (serialisierbaren) Spuren in serielle Spuren betrachtet werden.
Schrittweise Konstruktion von serialisierbar vervollständigbaren Ereignisspuren.

Start : $\eta' = \underline{a}$ mit $a \in A$
ist σ-zulässig für jedes $a \in A$ nach Definition von $\mathfrak{A}$ bzw. $(\mathfrak{A}, \mathcal{I})$

Schritt : $\eta' = \epsilon^1 \ldots \epsilon^k \boxed{\epsilon^{k+1}} \equiv$ unvollständig und σ-zulässig zu vervollständigen.

System $(\mathfrak{A}, \mathcal{I})$ mit atomaren Prozessen $H(\mathfrak{A})$ akzeptiert werden serialisierbare Spuren - äquivalent mit den seriellen Spuren

$$H(\mathfrak{A}) \begin{cases} \neg\sigma \\ \sigma! \end{cases} \quad \text{Kontrolle für die Elemente von } H(\mathfrak{A})+!$$

Schrittweise Konstruktion ≡ Entscheidung bei jedem Schritt → Erfolg $\eta' = \epsilon^1 \dots \epsilon^k \;\; k \geq 1 \;\; \boxed{\epsilon^{k+1}} \leftarrow \sigma$-zulässig
$\zeta(z^0, \eta') = z^0 \dots z^k \;\; \eta'$ kann serialisierbar vervollständigt werden.

Für η':

- $A(\eta') \subseteq A$: für die $a \in A(\eta')$ enthält η' $\underline{a}$ und $\overline{a}$
- $\mathcal{I}(\eta') \subseteq A$: für die $a \in \mathcal{I}(\eta')$ enthält η' $\underline{a}$ und nicht $\overline{a}$

Für ϵ^{k+1} ist σ-zulässig, wenn gilt:

- $e = \overline{a}$ für ein $a \;\; \in \mathcal{I}(\eta')$, weil η' nach Voraussetzung serilisierbar vervollständigt werden kann
- $e = \underline{a}$ für $a \in A - (\mathcal{I}(\eta') + A(\eta'))$, falls

$$V(a) \cap N(b) = \emptyset \text{ für alle b } \in \mathcal{I}(j) \tag{2.1}$$
$$N(a) \cap V(b) = \emptyset \text{ für alle b } \in \mathcal{I}(j) \tag{2.2}$$

  → $\underline{b} \dots \underline{a}$ falls (2.1) gilt ⇒ a, b serialisierbar
  falls (2.2) gilt ⇒ b, a serialisierbar

2.3.5 Wesentliche determinierte Systeme

$(\mathfrak{A}, \mathcal{I})$, $\mathfrak{A} = (D, A, \emptyset, V, N), \mathcal{I} = (Z, R)$
$a \in A$: Berechnung von a bestimmt von $V(a)$ und R_a und liefert Ergebnisse in $N(a)$ ← diese Ergebnisse müssen nach $\overline{a}$ erhalten bleiben!

Stattdessen: a berechnet wesentliche Ergebnisse und unwesentliche Ergebnisse ≡ $\boxed{\text{Hilfsergebnisse}}$ überflüssig nach $\overline{a}$
Für $(\mathfrak{A}, \mathcal{I})$ sei N^+ als Menge der Datenobjekte für die wesentlichen Ergebnisse d. Berechnungen der Prozesse
⇒ Aufteilung ⇒ wesentlich determiniert ≡ determiniert bezüglich der $d \in N^+$
⇒ Für $a \in A : \;\; N^+(a) \subseteq N(a); H(a) \equiv N(a) - N^+(a) : \;\; d \in H(a) \equiv$

Hilfsdatenobjekt für a
gefordert: $H(a) \cap V(a) = \emptyset$
a berechnet R_a: $V(a) \xrightarrow{R_a} N(a) \xrightarrow{\Pi} N^+(a)$ Für die Berechnung von R_a gefordert: Keine neunen Schritte

$$\boxed{R_a^+ \equiv \prod_{N^+(a)}^{N(a)} R_a}$$

$(\mathfrak{A}, \mathcal{I})$, N^+ Familie der wesentlichen Nachbereiche.

$\equiv$ Beibehaltung der wesentlichen Ergebnisse
Für die Hilfsergebnisse

- benötigt für die Aussführung
- nach $\overline{a}$ nicht mehr interessant

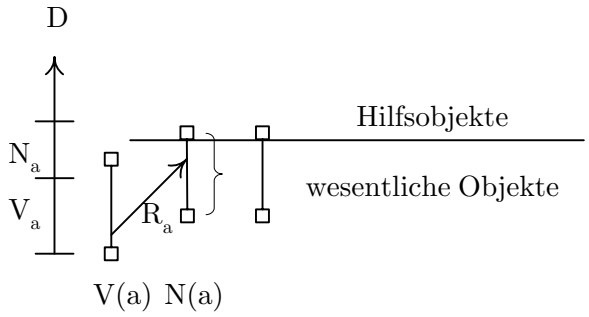


Abbildung 2.12: Modifikation des Systems

Modifikation der Forderungen
Modifiziertes System determiniert bezüglich der Ergebnisse in den $N^+(a) \forall a \in A$
Dafür:
Abgeschwächte Bernstein-Bedingung: beschränkt auf $V(a)$ und $N^+(a) \forall a \in A$

$\Rightarrow$ Beschreibung des Verhaltens bezüglich der Datenobjekte, die benötigt werden: $N(a) \Rightarrow N^+(a) : R_a \Rightarrow R_a^+ \equiv \prod_{N^+(a)}^{N(a)}(R_a)$ Achtung: Interferenzen mit den übrigen Prozessen bezüglich der Hilfsdatenobjekte $H(a)$ dazu Reihenfolgerestriktionen.
Dazu $\otimes \subseteq A \times A$ mit $(a, b) \in \otimes \equiv H(a) \cap H(b) \neq \emptyset$ für alle $a, b \in A, a \neq b$

Für das System $\mathfrak{A}^+$, das von $(\mathfrak{A}, \mathcal{I})$ mit N^+ ausgehend mit

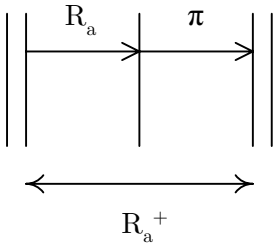


Abbildung 2.13: R_a^+

- der Familie der wesentlichen Nachbereiche N^+ und
- der Relation $\otimes$ auf $A \times A$

definiert ist, gilt:

> $\mathfrak{A}^+$ ist streng determiniert bezüglich $N^+ \Leftrightarrow$ erfüllt bezüglich V und N^+ die modifizierte Bernsteinbedingung.

$(\mathfrak{A}, \mathcal{I})$ Klärung der Anforderungen N^+

$(\mathfrak{A}, \mathcal{I})$ $\Rightarrow \mathfrak{A}$ hat das gewünschte Verhalten
$\Rightarrow$ Anforderungenreduziert
$\Rightarrow$ mehr Flexibilität für die Berechnungen bezüglich der $((H(a)|a \in A)$

2.4 Systeme mit sequentiellen Prozessen

Allgemeiner Prozesse:
Orientierung:

- Systeme mit atomaren Prozessen
- Sequentielle Programme

Für diese Systeme:

- Sequentielle Programme mit entsprechenden allgemeineren Fähigkeiten
  $\Rightarrow$ Verhaltensbeschreibungen und Analysen
- Bisher: *wenige Wechselwirkungen* zwischen Prozessen
  Jetzt: Mögliche Wechselwirkungen analysieren und klassifizieren; gezielt zulassen

Für die entsprechenden Systeme:

- Eigenschaften gezielt festlegen
- Zu gewährleisten, dass die beabsichtigten Eigenschaftenerreichbar sind und zulässig!
- Verfahren für die Steuerung zum Erfolg

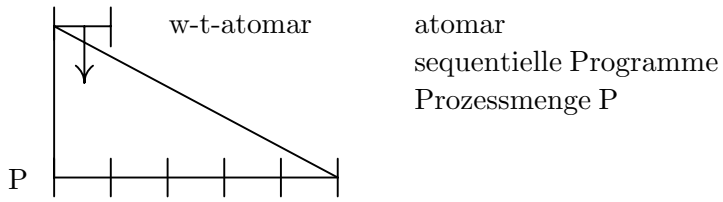


Abbildung 2.14: 2.4 Systeme mit sequ Prozessen

System: Verhalten → Abhängigkeiten ⇓
Gesetze und Reduktion der Abhängigkeiten

⇒ Für sequentielle Prozesse: Viele Möglichkeiten für Abhängigkeiten

- Niútzung von Abhängigkeiten
- Analyse des Verhaltens und Klassenbildung

2.4.1 Von Systemen mit atomaren Prozessen zu Systemen mit sequentiellen Prozessen

Uninterpretierte Systeme:

atomare Prozesse ⇒ atomare Berechnungen; Folge ≡ Sequentieller Prozess

Ein *uninterpretiertes* System wird definiert als ein 6-Tupel $\mathcal{R} = (D, P, \alpha, \kappa, V, N)$ mit:

- D: Menge der Datenobjekte: nichtleer, endlich, linear geordnet
- P: Menge der Prozesse: nichtleer, endlich, linear geordnet

- α: $P \longrightarrow \mathbb{N}$ linkstotal: Phasenanzahlfunktion
  $p \in P$; $\alpha(p)$ Anzahl der phasen, linear geordnet
  $A(p) = \{p_1, \ldots, p_{\alpha(p)}\}$
  $\Rightarrow A \equiv \bigcup\limits_{p \in P}$: Menge der Phasen der Berechnung von $\mathcal{R}$

- $\kappa \subseteq A \times A$: 2-stellige Koordinierungsrelation

- $V, N: \ A \longrightarrow \mathcal{P}(D) - \{\emptyset\}$: Vorbereiche und Nachbereiche der (Rechenfunktionen) der Phasen

- $\mathcal{R} = (D, P, \alpha, \kappa, V, N)$

- (A, κ): Phasenbeschreinbung: Basis
  [κ : Strukturbedingungen: Struktur der Prozesse, Struktur über den Prozessen]
  Für jeden Prozess $p \in P: \ A(p)$ sequentiell $\sigma(p)$ lineare Ordnung auf $A(p)$
  κ enthält insbesondere die $sigma(p)$ der $p \in P$: Krundlage für die koordinierung

Für $\mathcal{R}$ mit (A, κ) Verhaltensbeschreibungen
$a \in A$: beschrieben mit $(\underline{a}, \overline{a})$

Beschreibung von $\mathcal{R}$ durch Ereignismenge mit Struktur:
Für $\mathcal{R}$ mit (A, κ) sei $E \equiv \{\underline{a} \mid a \in A\} \ \cup \{\overline{a} \mid a \in A\}$;
$(E, x) \equiv$ Beschreibungsbasis mit Ereignissen
Die Koordinierungsrelationκ auf A ist zu übertragen auf E

κ_E auf E ergibt sich aus κ_A auf A:

- $a \Rightarrow (\underline{a}, \overline{a})$ für jedes $a \in A$

- $(a, b) \in \kappa_A \Rightarrow (\overline{a}, \underline{b}) \in \kappa_E$ für alle Elemente von κ_A

$\Rightarrow$ Ereignissicht von $\mathcal{R}: \ (E, \kappa_E)$

3 Sichten:

- Ereignissicht von $\mathcal{R}: \ (E, \kappa_E)$

- Phasensicht von $\mathcal{R}: \ (A, \kappa_A)$

- $\mathcal{R} = (D, P, \alpha, \kappa, V, N)$

Beschreibung des Verhaltens von $\mathcal{R}$ mit (E, κ):

- $\eta = \epsilon^1 \dots \epsilon^k$ ist eine *Ereignisspur* von $\mathcal{R} \Leftrightarrow$ die Elemente von η sind Elemente von E. Jedes $e \in E$ tritt in η höchstens einmal auf, die Reihenfolge der Ereignisse in η ist verträglich mit κ
- Die Ereignisspur η von $\mathcal{R}$ ist vollständig $\Leftrightarrow$ alle Elemente von E sind Elemente von η
- Das Verhalten von $\mathcal{R}$ wird vollständig beschrieben mit den vollständigen Ereignisspuren von $\mathcal{R}$: $H(\mathcal{R})$

$\mathcal{R}$ mit (A, κ): Interpretierte Systeme mit sequentiellen Prozessen

- Ein Paar $\mathcal{I} = (Z, R)$ mit
  - dem Zustandsraum $Z = \underset{d \in D}{\times} Z(d)$ der Datenobjekte und
  - der Familie $R = (R_a \mid a \in A)$ der Rechenfunktionen der Phasen ist eine Interpretation des Systems $\mathcal{R}$
- Das Paar $(\mathcal{R}, \mathcal{I})$ ist ein interpretiertes System mit sequentierllen Prozessen

Das Verhalten des interpretierten Systems mit sequentiellen Prozessen $(\mathcal{R}, \mathcal{I})$ und der Menge $H(\mathcal{R})$ der vollständigen Ereignisspuren von $\mathcal{R}$ wird beschrieben mit der Menge der einem Anfangszustand $z^0 \in Z$ und einem $\eta \in H(\mathcal{R})$ zugeordneten Zustandsspuren $\zeta(z^0, \eta)$.
$z^0 \in Z,\ \eta = \epsilon^1 \dots \epsilon^k,\ k \geq 1$, Ereignisspur von $\mathcal{R}$
$\zeta(z^0, 1) \equiv z^0 \quad 1 =$ leere Ereignisspur
$\zeta(z^0, \eta) \ = \ \zeta(z^0, \eta') z^k$ mit $\eta = \eta' \epsilon^k$ und $\zeta(z^0, \eta')$
mit:

- $z^k \equiv z^{k-1}$, falls $\epsilon^k = \underline{a}$ für eine Phase a von $\mathcal{R}$
- $z^k \equiv \coprod^{(N(a), \overline{N(a)})} (R_a(\prod_{V(a)}(z^j)), \prod_{\overline{N(a)}}(z^{k-1}))$ falls $\epsilon^k = \overline{a}$ und $\epsilon^j = \underline{a}$ mit $j < k$

$\mathcal{R}, H(\mathcal{R}), Z;\ \{\zeta(z^0, \eta \mid \dots\}$

Wichtige Faktoren zur Ordnung und Auswahl: Klassen von Abhängigkeiten in $\mathcal{R}$

$\mathcal{R}:\ D, P$

- $P \leftrightarrow D$

- $P \leftrightarrow D' \equiv$ gezielt gewählte D-Eigenschaften
- $P \leftrightarrow P$

2.4.2 Systeme mit FX-BM: Deadlock-Probleme

System mit Prozessen P und Datenobjekten D

D: Ein- und Ausgabe für Berechnungen
$\Rightarrow$ Speicherung für mehrere Phasen
$\Rightarrow$ Störungen einschränken D vergröbert zu *Betriebsmitteln BM*
BM $\Rightarrow$ Eigenschaften-Klassen
Die BM, die betrachtet werden, sind

- wiederholt benutzbar

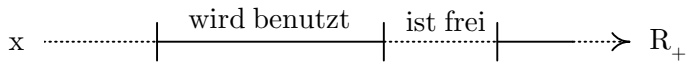


Abbildung 2.15: wiederholt benutzbare Betriebsmittel

- frei *exklusiv* benutzbar (oder wechselseitig ausgeschlossen)
  x ist jeweils *von einem Prozess* benutzbar ($\rightarrow$ exklusiv)
  x ist frei ohne spezifische Eigenschaften

$x:\ Z(x) \neq \emptyset$ frei: angefordert in einem $z \in Z(x)$

Für ein System:

- betrachet werden allein FX-BM
- davon existieren mehrere Arten
- für jede Art gibt es mehrere Exemplare, die als äquivalent vorausgesetzt sind

System mit FX-BM
Sequentielle Prozesse benötigen für ihre Berechnungen die BM $\Rightarrow$ Restriktionen bezüglich der Ausführbarkeit der Phasen
$\Rightarrow$ Koordinierung

- mit möglichem Erfolg

- aber zu Lasten schärferer Kontrollen!

Ein System mit sequentiellen Prozessen und FX-BM ist definiert durch ein 5-Tupel: $\mathcal{R} = (w, P, \alpha, X, Y)$ mit

- $w = \{w_1, \ldots, w_l\}$ mit $l \in \mathbb{N}$ und $w_i \in \mathbb{N}$ für alle $i \in \{1, \ldots, l\}$ ist ein BM-Vektor
  L: Anzahl der Arten
  w_i: Anzahl der Exemplare der Art i
- P Prozesse; α: Phasenanzahlfunktion
- $X = (X_{(p)} \mid p \in P)$: Familie der BM-Anforderungen
- $Y = (Y_{(p)} \mid p \in P)$: Familie der BM-Freigaben

$p \in P$: $x_{(p)}$: $\alpha(p) \times l$ Matrix mit $x(i,p) = (x_1(i,p), \ldots, x_l(i,p))$ für $i \in \{1, \ldots, \alpha(p)\}$
$x_j(i,p)$ = Anzahl der BM der Art j, die für das Initialisierungsereignis $\underline{p}_i$ zusätzlich benötigt werden.

$p \in P$: $y_{(p)}$: $\alpha(p) \times l$ Matrix mit $y(i,p) = (y_1(i,p), \ldots, y_l(i,p))$ für $i \in \{1, \ldots, \alpha(p)\}$
$y_j(i,p)$ = Anzahl der Exemplare der Art j, die mit dem Terminierungsereignis $\overline{p}_i$ freigegeben wird.

$\Rightarrow p$ mit Phasenstruktur $(p_1, \ldots, p_{\alpha(p)})$ und BM-Nutzung für Phasen-Teilfolgen für FX-BM

$\mathcal{R} = (w, P, \alpha, X, Y)$

Berechnungen: Aufgaben sollen lösbar sein!

BM-Zulässigkeit:

$\mathcal{R}$ ist BM-zulässig $\Leftrightarrow$ für jeden Prozess $p \in P$ und alle $i \in \{1, \ldots, \alpha(p)\}$ gelten

$$0 \leq \sum_{j=1}^{i} x(j,p) - \sum_{j=1}^{i-1} y(j,p) \leq w$$

$$\sum_{j=1}^{\alpha(p)} x(j,p) - \sum_{j=1}^{\alpha(p)} y(j,p) = 0$$

Im Weiteren sei $\mathcal{R}$ ein BM-zulässiges System. Beschreibung von Berechnungsausführungen mit Ereignisspuren und den zugehörigen zustandsspuren

Für die Ereignisspuren: Gesetze der Berechnungsabläufe
zusätzlich: besondere Eigenschaften der Betriebsmittel $\Rightarrow$ Zwischenschicht: BM-Zulässigkeit bei jedem Schritt

Beschreibung mit Zustandsspuren mit *Belegungs-Anforderungs-Zuständen*
$\Rightarrow z = (B, A)$ mit Belegung B und Anforderung A
$B = (b(p) \mid p \in P)$ und $b(p) = (b_1(p), \ldots, b_l(p))$:
$b_j(p) \equiv$ Anzahl der Exemplare der BM der Art $j \in \{1, \ldots, l\}$, die p belegt.
$A = (a(p) \mid p \in P)$ und $a(p) = (a_1(p), \ldots, a_l(p))$:
$a_j(p) \equiv$ Anzahl der Exemplare der BM der Art $j \in \{1, \ldots, l\}$, die p zusätzlich anfordert.

Für $\mathcal{R}$ sind zu definieren:

- die vollständigen, BM-zulässigen Ereignisspuren η und
- die z^0 und η zugeordnete (BM-zulässige) Zustandsspuren
- Λ ist die BM-zulässige leere Ereignisspur von $\mathcal{R}$
- z^0 mit $z^0 = (B^0, A^0)$ und $b^0(p) = 0\ \forall p \in P$ und $a^0(p) = x(1, p)\ \forall p \in P$ und $\zeta(z^0, \Lambda) = z^0$ ist die z^0 und Λ zugeordnete BM-zulässige Zustandsspur

Seien $\eta = \epsilon^1 \ldots \epsilon^k$ mit $k \geq 1$ eine BM-zulässige Ereignisspur von $\mathcal{R}$ und $\zeta(z^0, \eta) = z^0 z^1 \ldots z^k$ die BM-zulässigen Zustandsspuren mit:

- ϵ^k ist Ereignis der j-ten Phase des Prozesses $p \in P,\ 1 \leq j \leq \alpha(p)$
  ϵ^k ist BM-zulässig:
  - falls $\epsilon^k = \underline{p}_j$: $\sum_{q \in P} b(q) + a^{k-1}(p) \leq w$
  - falls $\epsilon^k = \overline{p}_j$: ist BM-zulässig
  - falls $\epsilon^k = \underline{p}_j$: $b^k(p) = b^{k-1}(p) + a^{k-1}(p),\ a^k(p) = 0$
    $b^k(q) = b^{k-1}(q),\ a^k(q) = a^{k-1}(q)\}; \forall q \in P - \{p\}$
  - falls $\epsilon^k = \overline{p}_j$: $b^k(p) = b^{k-1}(p) - y(j, p)$,
    $a^k(p) = \begin{cases} x(j+1, 0), & \text{falls } j \leq \alpha(p) \\ 0 & \text{sonst} \end{cases}$ $b^k(q) = b^{k-1}(q)\ \forall q \in P - \{p\}$

$$\eta = \underbrace{\epsilon_1 \dots \epsilon^{k-1}}_{\text{nach Voraussetzung BM-zulässig}} \quad \underbrace{\epsilon^k \dots}_{\text{BM-Zulässigkeit geprüft und erfüllt}}$$

$\zeta(z^0, \eta) = z^0 z^1 \dots z^{k-1} z^k \Rightarrow$ BM-zulässig

Fazit: für das BM-zulässige System $\mathcal{R}$ gesucht sind vollständige BM-zulässige Ereignissputen

Weil $\mathcal{R}$ nach Voraussetzung BM-zulässig ist, gibt es vollständige BM-zulässige Ereignissputen:
für jede Permutation π von P die Spur der sequentiellen Ausführung aller p demäß π,
$\Rightarrow$ von Interesse: Ereignisspuren, in denen mehrere Prozesse gleichzeitig aktiv sind.
Für $\mathcal{R}$: Sei $\eta = z^1 \dots z^k \dots$ $\zeta(z^0, \eta) = z^0 z^1 \dots z^k$ BM zulässig

- Der Prozess $p \in P$ ist in z^k blockiert $\Leftrightarrow$ p stellt in z^k BM-Anforderungen, die in z^k nicht erfüllt werden können
  $p: \epsilon^k$
  - p wartet und setzt die Berechnung später fort
  - das ist nicht notwendig der Fall
- z^k ist Deadlock-Zustand $\Leftrightarrow$ $\exists p \in P$ so, dass in z^k und in allen BM-zulässigen Folgezuständen von z^k blockiert ist
- z^k ist Deadlock-sicher $\Leftrightarrow$ es gibt eine vollständige BM-zulässige Ereignisspur η' mit dem Präfix η

z^k sei Deadlock-Zustand:

- p ist in z^k und in der Folge blockiert
  $\Rightarrow$ p stellt Anforderungen in $z^k: a^k(p) \neq 0$
- wegen der BM-Zulässigkeit von $\mathcal{R}$ gibt es weitere, entsprechende Prozesse

Für einen Deadlock-Zustand z^k gilt:

- es gibt $Q \subseteq P$ mit $|Q| \geq 2$ so, dass sich die $q \in Q$ wechselseitig blockieren
- jeder Prozess $q \in Q$ fordert Betriebsmittel an und belegt Betriebsmittel in z^k

2.5 Realisierung der einfachen nebenläufigen Systeme mit zentraler Hardwarekonfiguration

Hardwarekonfiguration einer Zentrale / eines Prozessors
Verallgemeinerung auf räumlich verteilte Hardwarekonfigurationen

System mit:

- Komponenten mit Speicherfähigkeiten ⇒ Datenobjekte und Datenobjektmanagement
- Prozesse und Prozess-Management
  Realisierung der Rechenfähigkeiten und Koordinierung
  Prozess realisieren
  Betriebssystem-Funktionen / Betriebssystem-Kern
- Interaktionen mit der Systemumgebung
  - Prozesse integrieren
  - Prozesse integrieren mit Umgebungskomponenten
  - Syseme integrieren $\equiv \underset{\text{Hardware-Konfiguration}}{\text{zentrale}} \longrightarrow \underset{\text{Hardware-Konfiguration}}{\text{räumlich verteilt / vernetzt}}$

Kapitel 3

Datenobjekte und Datenobjekt-Management

Datenobjekte ≡ Komponenten mit Speicherfähigkeit
Mehrere Ebenen / Schichten erfordert eine geeignete Vorgehensweise!

Eigenschaften:

- Sichten der Banutzer
- Sichten der automatischen Berechnungsausführung
- Sichten der technischen Realisierung

Forderungen:

- Effizienz ≡ Aufwand klein!
- Konsistenz der Objekte

Nutzung durch die Prozesse für die Lebenszeit der Prozesse
⇒ Zu verallgemeinern auf die Systemlebenszeit

- bezüglich der Prozess-Lebenszeiten
- permanent / persistent

Systeme mit Datenobjekten und Prozesse

Datenobjekte *angemessen* technisch realisieren: Geeignet für Zwecke mit angemessenen technischen Mitteln.

Verallgemeinerung:

- Lebenszeiten der Prozesse $\underline{p} \rightarrow \overline{p}$
- Lebenszeiten der Datenobjekte (Erzeugung, ..., Auflösung)
  Überdauern die Lebenszeiten der Prozesse möglicherweise: Persistente Datenobjekte

Nutzung durch die Prozessse und für die Prozesse; dafür angemessene technische Realisierungen mit Speichern:

- Kapazität Zugriffszeiten

Management: erzeugen, auflösen, realisieren und bereitstellen
⇒ Systematisches Vorgehen

Sichten auf Datenobjekte

- Anwendungssicht
- Prozesssicht
- Realisierungssicht

⇒ Geeignetes Vorgehen ⇒ Anwendungssicht: Datenobjekte und Anforderungen

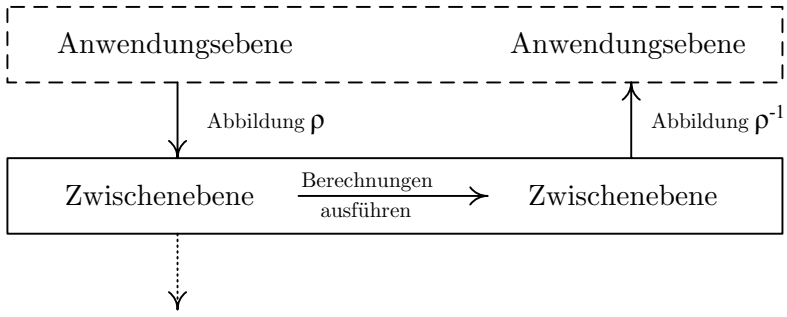


Abbildung 3.1: Abbildung ϱ von der Anwendungsebene in die Zwischenebene

3.1 Virtuelle Adress- und Speicherräume

3.1.1 Grundlagen

Elementare Objekte: Bytes, Byte-Zellen, Adressen für Bytezellen
Verallgemeinerung durch Komposition: Struktur: lineare Ordnung

Segment:	Virtueller Speicher	Identifikation der Komposition
1-dimensionales Feld	(VS)	nicht negative Ganzzahlen
		max. Anzahhl: 2^32
		$AS = \{0, 1, \ldots, 2^{32} - 1\}$

Segmente:

- Identifikator: S^+ und $S \subseteq S^+$
  S: existierendes Segment $s \in S$
  Länge: $l(s)$
  Indexmenge: $I(s) = \{0, 1, \ldots, l(s) - 1\}$
  $i \in I(s)$ Indentifikation der elementaren Komponenten
- Operationen auf $s \in S$
  - lesen / schreiben von elementaren Komponenten (Bytes)
  - Verallgemeinerung auf komponierte Objekte
- Operationen auf der jeweiligen Segmentmenge
  - erzeugen: $s' \in S^+ - S \Rightarrow S := S \cup \{s'\}$
  - auflösen: $s \in S \Rightarrow s' := S - \{s\}$

Segmente sind Zwischenschicht-Datenobjekte! Realisierungen:

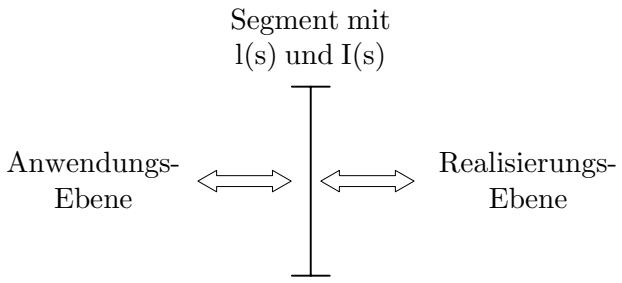


Abbildung 3.2: Segmente als Zwischenschicht-Datenobjekte

- AS - Arbeitsspeicher
  Zugriff von Prozessoren
  eingeschränkte Kapazität

- HS - Hintergrundspeicher
  hinreichende Kapazität
  „Rotationsspeicher“ $\Leftrightarrow$ EA-Operationen nötig

Realisierung der Segmente $s \in S$ mit Speichern AS und HS
Zahlreiche Verfahren: Effizienz, Konsistnz
davon: Seitenbasierte Verfahren zur Realisierung der Segmente!

3.1.2 Seitenbasierte Realisierung von Segmenten

Grundlagen

Datenobjekte $\equiv$ Segmente
Realisierungen für effiziente Nutzung: konstante Objekte und Beschreibungen

Realisierung mit Speichern der Hierarchie der Hardwarekonfiguration (HK)
Realisierung mit AS und HS
Nutzung der Segmente soll bei gemischten Realisierungen möglich sein
Nutzung für *Berechnungen* und Dauerhaftigkeit
dafür: Nutzung durch Rechenprozessoren

$\Rightarrow$ Prozesse und Datenobjekte

$\Rightarrow$ Prozesse und Datenobjekte Speicherfähigkeiten geeignet zur Nutzung durch Prozesse, die mit Rechenprozessoren realisiert sind, und mit ihren Zugriffsmöglichkeiten zur Speicherung

Prozesse und Datenobjekte
Prozessoren mit eingeschränkten Möglichkeiten für Zugriffe zur Speichern

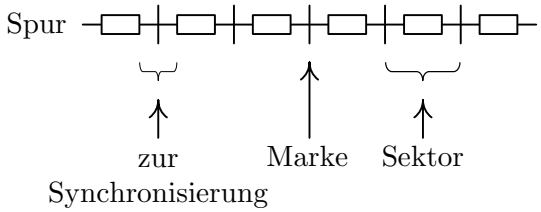
Realisierung der Segmente; geeignet für „diese Prozesse“
$\Rightarrow$ Segmente mit

- vollständige AS-Realisierung
- vollständige HS-Realisierung
- gemischte AS-HS-Realisierung

Ermöglicht mit Einführung von uniformen Kunst-Datenobjekte:
Seite: Folge von Bytes mit „kleiner“, fester Länge: Potenz von 2: $2^10, 2^11, 2^12$
Segment: s mit $l(s)$ und $I(s) \Rightarrow$
Seiteneinteilung S mit $I_p(s) = \{0, 1, \ldots, l_p(s) - 1\}$

Für AS mit A: Seitenbereiche (Kacheln) $\Rightarrow$ Kacheladressraum K
Für HS: Seiten-Blockbereiche mit B; realisiert z.B. als Platte mit entsprechender Struktur

Sektoren als Blockbereiche $\Rightarrow B$, Transformationen werden mit entsprechenden Umspeicherungen von Seiten realisiert. Beschreibungen der Reali-



Mit Prüfzeichen, nach dem Schreiben
Korrekturlesen

Abbildung 3.3: Abbild einer Spur

sierungen mit entsprechenden Speicherfunktionen:

- vollständige AS-Realisierung von s mit $I_p(s)$
  $\sigma_{AS}(s, \bullet) : \ I_p(s) \rightarrow K$ linkstotal und injektiv
- vollständige HS-Realisierung von s mit $I_p(s)$
  $\sigma_{HS}(s, \bullet) : \ I_p(s) \rightarrow B$ linkstotal und injektiv
- gemischte AS-HS-Realisierung von s mit $I_p(s)$
  - $\sigma_{AS}(s, \bullet) : \ I_p(s) \rightarrow K \cup \{\perp\}$
  - $\sigma_{HS}(s, \bullet) : \ I_p(s) \rightarrow B \cup \{\perp\}$

    mit injektiven σ_{AS} und σ_{HS} und
    $\forall i \in I_p(s) : \ \sigma_{AS}(s, i) \in K \vee \sigma_{HS}(s, i) \in B$

Bevorzugt werden gemischte AS-HS-Realisierungen
benötigt: $\Rightarrow$ partielle AS-Realisierungen
Seitenbasierte Verfahren ermöglichen echt partielle AS-Realisierungen für Segmente von Prozessen bei nutzung für Berechnungsausführungen; insbesondere für koexistierende, aktive Prozesse

System mit Prozess p mit zugeordnetem Segment s:
$I_p(s)$, partielle AS-Realisierung gefordert: effiziente Ausführung der Berechnungen

$\Rightarrow$ angepasste Verhaltensbeschreibung $\equiv$ Zugriffe zu s-Seiten über der Zeit
Beschreibung mit einer *Seiten-Referenzspur* $\varrho(s, \bullet) = i_1, i_2, \ldots$ endlich mit $i_j \in I_p(s)\ \forall j$

Bei Ausführung der Berechnungn von p auf der Basis von $\varrho(s, \bullet)$:
$\Rightarrow$ versuchter Zugriff zur Seite i_j, die

- AS-realisiert ist: direkt ausführbar $\equiv$ *Seitentreffer*
- HS-realisiert ist: *Seitenfehler* $\Rightarrow$ die Seite muss HS-realisiert werden

$\Rightarrow$ großer Aufwand:
$\equiv$ Effizienz gefordert
$\equiv$ die jeweils referenzierten Seiten sollen AS-realisiert sein

Einzelseiten-Anforderungsstrategien

Verfahren für p mit s und $\varrho(s, \bullet)$
Für s werden $k \in \mathbb{N}, k \leq |I_p(s)|$, Kacheln zur Verfügung gestellt. Jeweilige Realisierung wird mit entsprechend definierten Zuständen beschrieben: Zustand = menge der indizies der AS-realisierten Seiten von s
beschrieben mit Zustandsfolge $Z_0, Z_1, \ldots$ mit $Z_0 = \emptyset$
für jeden Zustand Z_j gilt: $|Z_j| \leq k,\ i_j \in Z_j$

Verfahren:Übergang von Z_j nach Z_{j+1} mit i_{j+1}

- falls $i_{j+1} \in Z_j \Rightarrow$ Seitentreffer: $Z_{j+1} \equiv Z_j$
- falls $i_{j+1} \notin Z_j \Rightarrow$
  - falls weiter $|Z_j| < k \Rightarrow i_{j+1}$ wird AS-realisiert, $Z_{j+1} := Z_{j+1} \cup \{i_{j+1}\}$
  - falls weiter $|Z_j| = k \Rightarrow$ Auswahl eines $i \in Z_j$, HS-Realisierung der Seite i
    i_{j+1} mit dieser Kachel realisieren $\Rightarrow Z_{j+1} \equiv (Z_j - \{j\} + \{i_{j+1}\}$

Entscheidungen:

- Nachlade-Disziplin
- Verdrängungs-Disziplin

} Strategie, hier ist Verdrängungs-Entscheidung ausschlaggebend

(*) *Mindestforderung:*
Wenn die Anzahl der verfügbaren Kacheln wächst, dann darf die Anzahl der Seitenfehler nicht auch wachsen.

$\Rightarrow$ gefordert: Zugelassen sind allein E-A-Strategien, die (*) erfüllen

Eine E-A-Strategie V ist eine *Keller-Strategie* $\Leftrightarrow$ für jedes ϱ und jedes $m \in \mathbb{N}$ gilt:

$$Z_n(\varrho, m) \subseteq Z_n(\varrho, m+1) \quad n \in \mathbb{N}$$

Es ist möglich, dass *Fotschritts-Stillstand* auftritt: *Seitenflattern*, Trathing

Quantifizierung:
$AS : t_{AS} \approx 10ns \;\; HS : t_{HS} \approx 10ms$ Dauer einer Aktion
Prob: $0 \leq \alpha \leq 1$
$t = (1-\alpha) \cdot t_{AS} + \alpha \cdot t_{HS}$

Working-Set-Strategien

Lokale Strategie: jeweils für isoliert betrachtete Prozesse, jeweils für einzelne Zeitpunkte / Zustände $\Downarrow$
Globale anpassungsfähige Strategien

Grundlage:
$\varrho(s, \bullet)$ zur Zeit m mit Fenstergröße $n \in \mathbb{N},\ m \in \{1, \ldots, T\},\ \ (m-n)-1 \geq 0$
ist $W(\varrho(s, \bullet), m, n) \equiv \{i_{m-n+1}, \ldots, i_m\}$

Nutzung:
Beobachtung der Vergangenheit als Schätzung des Verhaltens in der Zukunft.
Strategien:

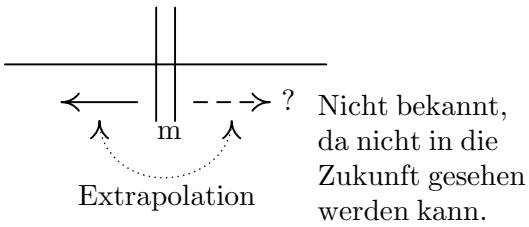


Abbildung 3.4: Extrapolation der Vergangenheit zur Schätzung der Zukunft

- die Working-Set sind fortlaufend zu beobachten!
- Grundregel:
  Für jeden aktiven Prozess soll jeweils die Working-Set *AS-realisiert* zur Verfügung stehen

- globale, anpassungsfähige Strategie!

$\Rightarrow$ Entscheidungen im Großen und im Kleinen zusammenwirken

Entscheidungen im Großen:

Menge der Prozesse P
Zerlegung: $P = P_a + P_b$ bei gegebener Kachelmenge

Entscheidung nach der Grundregel:

> Die jeweils aktiven Prozesse - $p \in P_a$ - sollen ihre Working-Sets AS-realisiert nutzen können.

Auswahl der $p \in P_a$
Algorithmus:

$P_a := \emptyset$;

while $\{P_a$ kann erweitert werden$\}$ **do**
  $p \in P$ — irgendein p aus P auswählen
  $P_a := P_a \cup \{p\}$ — wenn Kacheln zur Verfügung gestellt
  — werden können, dann wird p in P_a
  — aufgenommen
od

Nachbedingung:

- für alle $p \in P_a$ stehen Kacheln zur Verfügung
- P_a kann nicht erweitert werden
- alle $p \in P$, denen keine Kacheln zugewiesen werden können, stehen in P_b

$\Rightarrow$ $P := P_a + P_b$

$\Rightarrow$ die Working-Sets der $p \in P_a$ werden AS-realisiert: das dauert!

Sobald die Working-Set der $p \in P_a$ AS-realisiert zur Verfügung stehen:

- p führt seine Berechnungen aus
- AS-Realisierung der Working-Sets der $p' \in P_a - \{p\}$

Für die Prozesse $p \in P_a$, deren Working-Sets AS-realisiert zur Verfügung stehen, gilt:

- ihre Berechnungen werden ausgeführt
- bei auftretenden Seitenfehler folgen Entscheidungen im Kleinen

Entscheidungen im Kleinen:

AS-Realisierung der Seite, die den Seitenfehler verursacht hat

Im Gefolge von Entscheidungen im Kleinen können Entscheidungen im Großen veranlasst werden.
Übergänge erfolgen nach *Schwellwert-Kontrollen*

- *Anforderungsschwelle* ≡ mehr Kacheln sind nötig!
- *Freigabeschwelle* ≡ es können kacheln freigegeben werden

Schwellwerte sind festgelegt und orientieren sich an den Seitenfehlern.

Schwellwert-Kontrollen:

- Anforderungsschwelle wird überschritten
  ⇒ Kachelverteilung bei Beibehaltung der Prozess-Zerlegung korrigieren
  oder
  Änderung der Zerlegung ≡ Entscheidung im Großen
- Freigabeschwelle wird unterschritten
  ⇒ Kachelfreigabe für p möglich

Zusammenwirken der Entscheidungen im Kleinen:

> Freigabe von Kacheln von $p \in P$ für Erweiterung der Kachelmenge von Prozessen, die Bedarf haben.

Übergang von Entscheidungen im Kleinen zu Entscheidungen im Großen

> Wenn die Zusatzforderungen mit der aktuellen Zerlegung nicht erfüllbar sind.

Konsistenz der Segmente

Die Segmente sollen konsistente Datenobjekte sein
Segmente:

- per Definition festgelegte Eigenschaften
- für Berechnungen nutzbar realisiert
  ⇒ gemischte AS-HS-Realisierungen beschrieben mit *Desktiptor-Datenobjekten*

⇒ nutzbare Datenobjekte: wohldefiniert, vollständig beschrieben und widerspruchsfrei
Probleme entstehen durch:

Seite $p \left\langle \begin{array}{l} \text{AS-Realisierung } [desk, daten] \\ \text{HS-Realisierung } [desk, daten] \end{array} \right\rangle$ identische Replikate
nützlich und unproblematisch

Wertänderung für p: Änderung der AS-Realisierung

Zur Gewährleistung der Konsistenz

Veränderung der AS-Realisierung von p mit Veränderung:

- des *desk* der HS-Realisierung
- und der HS-Realisierung

Als atomare Aktion:

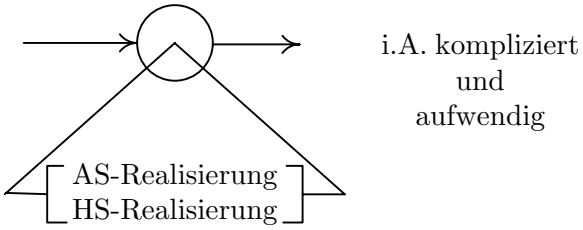


Abbildung 3.5: Veränderung der AS-Realisierung von p

3.2 Dateien und Dateisysteme

Anforderungen: langfristig existiere
⇒ große Vierlfalt

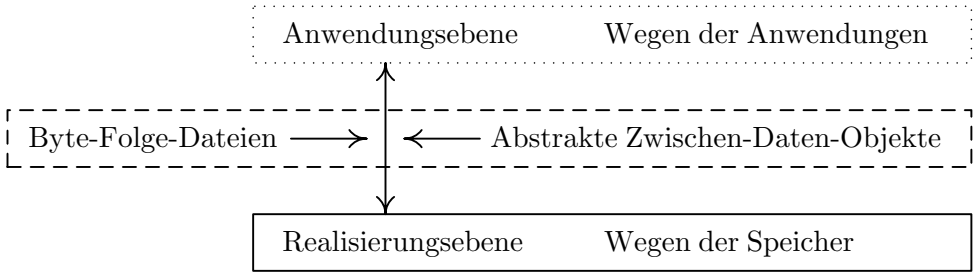


Abbildung 3.6: Byte-Folge-Dateien als abstrakte Zwischen-Datenobjekte

3.2.1 Byte-Folge-Dateien als persistente Datenobjekte

Datei: Objekt mit endlicher Lebenszeit, das wie folgt definiert ist:

- einem Identifikator / Bezeichner
- einer Folge von Byte-Datenobjekten mit einer Länge $l : 0 \leq l \leq 2^{32}$

und für das die Operationen

- erzeugen / auflösen
- lesen / schreiben von Bytefolgen
  Folger-erhaltend!

gelten.
Datei d als realisierte / benutzbare Datenobjekte:
$d \Rightarrow (Deskriptor(d),\ Körper(d))$
Deskriptor: Beschreibung der Eigenschaften von d

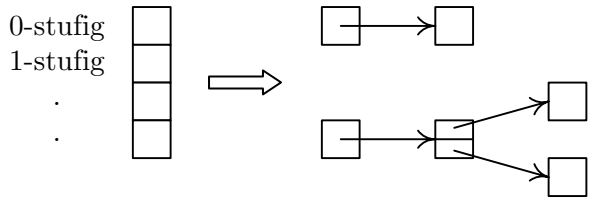
Attribute von *Deskriptor(d)*:

- Objekt-Attribute: Bzeichner, Typ, ...
- Nutzungs-Attribute: Eigner, weitere Nutzer, Rechte
- Realisierungs-Attribute: maximale und aktuelle Länge, Speicherfunktionen

Analogie zu Segmenten:

⇒ Seiten-basierte Realisierung
block-basiert: mehrere Blockformate, transformierbar
zusätzlich: Anpassung an variable Längen

⇒ gefordert: fähig zur Anpassung an die variablen Längen während der langen Lebenszeiten
entsprechende Speicherfunktionen:
⇒ Blockbasierte Speicherfunktionen: mehrstufig mit variierender Stufenzahl



Folger erhalten!

Abbildung 3.7: Mehrstufige, blockbasierte Speicherfunktionen

$d \equiv (desk(d),\ körper(d))$

3.2.2 Dateisysteme

Datei $d \Rightarrow$ Dateienmenge D:

- Grundlage der Nutzung: Ordnung
- strukturierte Menge

Realisierung eines Dateisystems mit

- Dateienmenge D
- Struktur

  ⇒ geeignete Struktur: Bauem

  ⇒ dafür geeignete Datenobjekte: Zwei Arten von Dateien:

  * D-Dateien (Daten-Dateien)
  * V-Dateien ≡ Struktur (Verzeichnis-Dateien)

Dateimenge mit Struktur
Dateibaum ≡ D mit Baumstruktur
$D \equiv$ Knoten eines Graphen
V-Dateien als Strukturierungsmittel

D-Dateien als spezielle Knoten des Baumes

Die Dateien sind die Zusammenfassungen von Datei-Kopf-Elementen
V-Datei v : {die Köpfe der $d \in D(v) \subset D$}
$\equiv$ Folge von $kopf(d) \mid d \in D(v)$
repräsentiert als Byte-Folge

Regeln für die Struktur:

- Zerlegung der primären Menge der D-Dateien
- Erweiterung um die benötigten V-Dateien

$\Rightarrow$ Graph mit Knotenmenge K und darauf definierte Kantenmenge
$\boxed{x \longrightarrow y}$ x enthält den Kopf/Deskriptor für y
Baum mit ausgezeichneter Wurzel

Strukturierte Dateimenge

- Basis für Übersicht
- Basis für komplexe Objektstrukturen

3.2.3 Konsistenz der Dateien

Konsistenz bezüglich $d \in D$ mit $d \equiv (kopf(d), körper(d))$ für $kopf(d)$ als Beschreibung der wesentlichen Eigenschaften don d.
Für $körper(d)$: Konsistenz bei Langlebigkeit und Variabilität.

Wegen der mehrstufigen Speicherfunktionen
$\Rightarrow$ relativ komplizierte Änderungs-Operationen; atomar!

Dateien sind die wesentlichen D-Objekte eines nebenläufigen Systems
koexistierende und konkurrierende Prozesse
$\Rightarrow$ Konsistenz! $\Rightarrow$ Deadlock-Vermeidung; komplexe Operationen

Kapitel 4

Prozesse und Prozessmanagement

Systemkomponente: Prozesse realisieren
Koordinierung realisieren
⇒ Konsequenzen für die System-Strukturen

Koordinierungsmechanismen und -verfahren: systematisch zu realisieren!

4.1 Prozesse als aktive Systemkomponenten

Prozess ≡ Komponente mit Speicherfähigkeiten und Rechenfähigkeiten

Speicherfähigkeit ⟶ Segmente ⟶

Rechenfähigkeit ⟶ Spezifikation ⟶ Programme und zug. Daten

Berechnungen durchführen: Rechenprozessoren zur Ausführung der Berechnungen einsetzen

⇒ Einsatz der verfügbaren Prozessoren zur Ausführung der Berechnungen der koxistierenden Prozesse

Programme und Daten ≡ R-Prozessor-Programme ⇒ mit entsprechend vorbereiteten Segmenten bereitstellen
Nutzung der Registerspeicher der R-Prozessoren

⇓

Segmente mit ausführbaren Programmen
⇒ Spezielle Maßnahmen: *Dispatching, Scheduling*

Dispatching: gefordert
Wirkungen bezüglich der Registerobjekte können unterdrückt / ausgeblen-

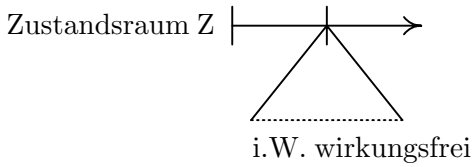


Abbildung 4.1: Dispatching

det werden
von den Register-Wirkungen abstrahieren duch Zwischenspeicherung und Wiederherstellung

Zum Dispatching:

- Maßnahmen zur Abstraktion von den Registerwirkungen
- Entzug der Neuzuteilung des R-Prozessors

Maßnahmen zum:

- Entbinden / Lösen und
- Binden

Dispatching-Maßnahmen:

Schduling-Strategien:

Die Prozesse $p \in P$ konkurrieren um einen R-Prozessor

Anforderungen der $p \in A$:

charakterisiert mit *Ankunftszeit* a_p und mit *Bedienzeit* b_p

Grundlage:

- $\mathbb{R}_+$ für die Zeit
- für $p \in P: \ a_p \in \mathbb{R}_+, \ b_p \in \mathbb{R}_+$

$\Rightarrow$ für $P : (a, \ b)$ Verfahren, welche die R-Prozessor-Zuteilung an die $p \in P$ festlegen

Festlegung dafür:

Strategie $S \Rightarrow$ Abläufe sind geregelt mit: $a,\ b,\ S$:
$\Rightarrow c(a,\ b,\ S) \equiv$ Zeitpunkt der Fertigstellung der Bedienung

Gesucht sind *geeignete* Strategien

$\Downarrow$

Verweilzeit minimal!

Prozess P, R-Prozessor
Scheduling mit einem Prozessor: Ankunftszeit a, Bedienzeit b
Strategien für a, b; Strategie S $c(S)$
Kriterium: Verweilzeit eines Auftrags
$\overline{V}(a, b, S)$ mittlere Verweilzeit
Ziel: S Verweilzeit-optimal

$$\overline{V}(a, b, S) = \frac{1}{n} \sum_{i=1}^{n} v_i(a, b, S)$$

N Aufträge, $a_1 = a_2 = \ldots = a_n = 0,\ b = (b_1, \ldots, b_n)$
Permutations-Strateie: $\pi(1, \ldots, n)$

$$\overline{V}(b, \pi) = \frac{1}{n} \sum_{i=1}^{n} (n + 1 - i) b_i$$

$\prod(n)$
Menge von Permutationen
Verweilzeit Minimale Permutation, optimal für *antithesisches* Skalarprodukt
$\pi_{opt} = ((1), (2), \ldots, (n)) : b_{(1)} \leq b_{(2)} \leq \ldots \leq b_{(n)}$

Verallgemeinert: Bedienzeiten, für die Wachstumsordnung möglich ist.
Verallgemeinerung: Übersicht über alle Permutationsstrategien: $\pi \in \prod(n)$

$\pi \in \prod(n)$: die zu π antithetische Permutation
$\pi^* \in \prod(n)$:
$\left.\begin{array}{l} \pi = ((1), \ldots, (n)) \\ \pi^* = ((n), \ldots, (1)) \end{array}\right\rangle \overline{V}(b, \pi) + \overline{V}(b, \pi^*) = 2 \cdot \mathbb{E}[\overline{V}(b, Random)]$

Für alle $\pi \in \prod(n)$

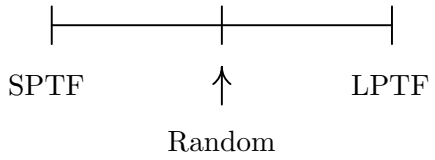


Abbildung 4.2: SPTF und LPTF

Kann verallgemeintert werden auf eine Folge von Auftrögen mit

- $a_1, a_2, \ldots$ $(a_i \mid i \in \mathbb{N})$
- $b_1, b_2, \ldots$ $(b_i \mid i \in \mathbb{N})$

Die Verweilzeit-optimale Strategie hier für ist $SRPTF$: die k+rzeste Restbedienzeit Strategien:

- nicht unterbrechende
- unterbrechende
  $\Downarrow$
  Zeitscheiben-Strategien
  n Aufträge, $a_1 = \ldots = a_n = 0,\ b = (b_1, \ldots, b_n)$ mit Quantum $q > 0,\ q \to 0$
  Permutation $((1), \ldots, (n))$; Alle Aufträge gemäß π, jedoch höchstens für die Dauer von Quantum q
  $\overline{W}$: Wartezeit: $\overline{W}(b, S_{q\to 0}) = 2 \cdot \overline{W}(b, SPTF)$

4.2 Prozess- und Kernberechnungen

System, Prozesse P, Koordinierung
Prozess-Management: Koordinierung und Realisierung
$\hookrightarrow$ Basis für die Prozesse, Datenobjekte $\Rightarrow$ Realisierung auf der gegebenen, zentralen Hardware-Konfiguration (HK)

Kern $\equiv$ Betriebssystem-Kern:
Realisierungs- und Koordinierungsaufgaben
$\Rightarrow$ Kern-Berechnungen und Kern-Dienste mit zugeordneten Daten-Objekten

$\Rightarrow$ Erweiterungen únd Abgrenzungen der entsprechenden Mittel

Basis: Adressräume, Speicherräume, Objekte, ...

Aufteilung: Virtueller-Adressraum (VA): $VA = \{0, \ldots, 2^{32} - 1\}$
Für die Prozesse: Segment für Code, zugeordnete Daten (statisch, dynamisch: Halde, Keller)

Privater Prozess-Wirkungsbereich $\begin{cases} cdh - Segment: \ 0 \ldots 2^{30} - 1 \\ k - Segment: \ 2^{30} \ldots 2^{31} - 1 \end{cases}$
Kern-Wirkungsbereich: $2^{31} \ldots 2^{31} + 2^{30} - 1$
ungenutzt: $2^{31} + 2^{30} \ldots 2^{32} - 1 \Rightarrow$ Grenz-kontrollen notwendig

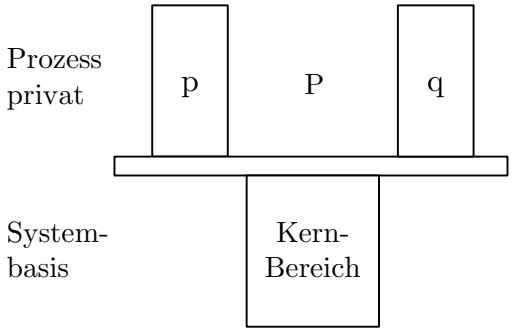


Abbildung 4.3: Prozess- und Kernbereich

Wirksame Grenzkontrollen
$\equiv$ Überschreibung $\equiv$ Inkonsistenz und Fehler
$\Rightarrow$ abgestuftes System von Maßnahmen
Basis hierfür: Privilegien-System

Prozessoren-Betriebs-Modi:

- Kern-Modus
- Prozess-/ Benutzer-Modus

} Alle R-Prozessor-Befehle

- privilegiert
- nicht privilegiert

Modi als Anker
Nutzung mit Wirkungsbereichs-Festlegungen und Wirkungsbereichs-Grenzkontrollen:

- Einhaltung
- kontrollierte Übergänge

Die entsprechenden Bereiche: Adressraumbereiche mit:

- Prozess-Bereiche
- Kern-Bereiche

Modi:
Kontrollierte Übergänge

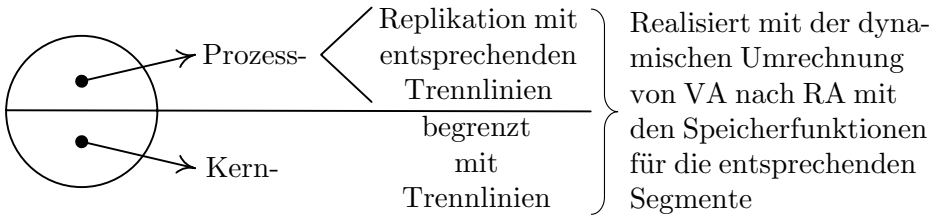


Abbildung 4.4: Prozessor- und Betriebs-Modi

Übergänge zwischen „Räumen“ erfordert Speicherfunktions-Übergänge
⇒ zu realisieren mit einer Folge von abgestimmten Schritten

- implizit (Exception-Handling, Unterbrechungs-Erkennung, -Handling)
- explizit ≡ Unterprogramm-Aufruf

Kern-Dienst-Aufrufe
als spezielle Prozedur-Aufrufe

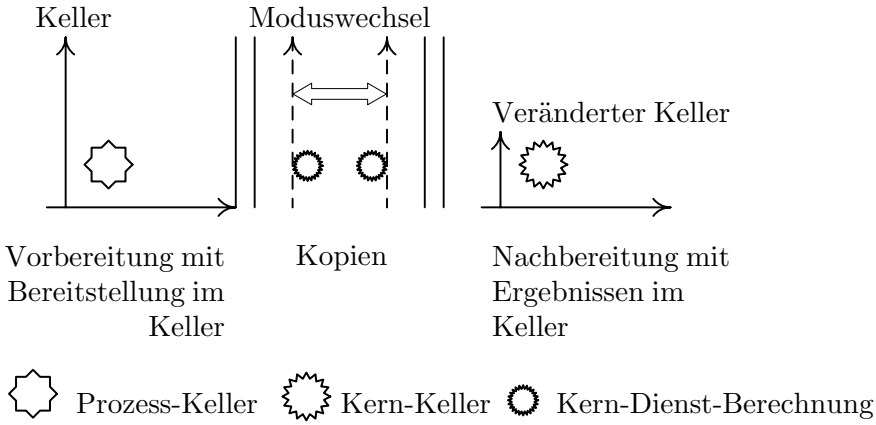


Abbildung 4.5: Prozess- und Kernbereich

- Dienste für Koordinierungs- und Kontrollaufgaben in nebenläufigem System mit implizierten Aufrufen
- weitere Dienste: Z.B. die Dienste des Dateisystems von UNIX

Frage: zuordnung der Objekte *Prozesse* $\leftrightarrow$ *Datenobjekte*

- einfach: statisch bzw. auf Lebenszeit der Prozesse
- sonst: Lösung der Koordinierungsprobleme
  Dazu: Differenzierung der Datenobjekte
  dynamische Zuordnung: $\Rightarrow$ entsprechende Hilfsmittel, entsprechend kostruierte/definierte Datenobjekte

Zentrale Systeme erweitern auf verteilte Systeme:

Hardware-Konfiguration

$\Downarrow$

bezüglich der Berechnungen