

# Systemprogrammierung

## Prozesssynchronisation: Nichtsequentialität

Wolfgang Schröder-Preikschat

Lehrstuhl Informatik 4

24. Mai 2011

# Gliederung

- 1 Kausalitätsprinzip
  - Parallelisierbarkeit
  - Kausalordnung
- 2 Konkurrenz und Koordination
  - Koordinierung
  - Konkurrenz
- 3 Verfahrensweisen
  - Einordnung
  - Kritischer Abschnitt
  - Lebendigkeit
- 4 Zusammenfassung

# Nebenläufige Programmabschnitte

Sequentielles  $\mapsto$  Nichtsequentielles Programm

**Nebenläufigkeit** (engl. *concurrency*) bezeichnet das Verhältnis von nicht kausal abhängigen Ereignissen, die sich also nicht beeinflussen

- Aktionen können nebenläufig ausgeführt werden, wenn keine das Resultat des anderen benötigt

```
1:   foo = 4711;  
2:   bar = 42;  
3:  foobar = foo + bar;  
4:  barfoo = bar + foo;  
5:   hal = foobar + barfoo;
```

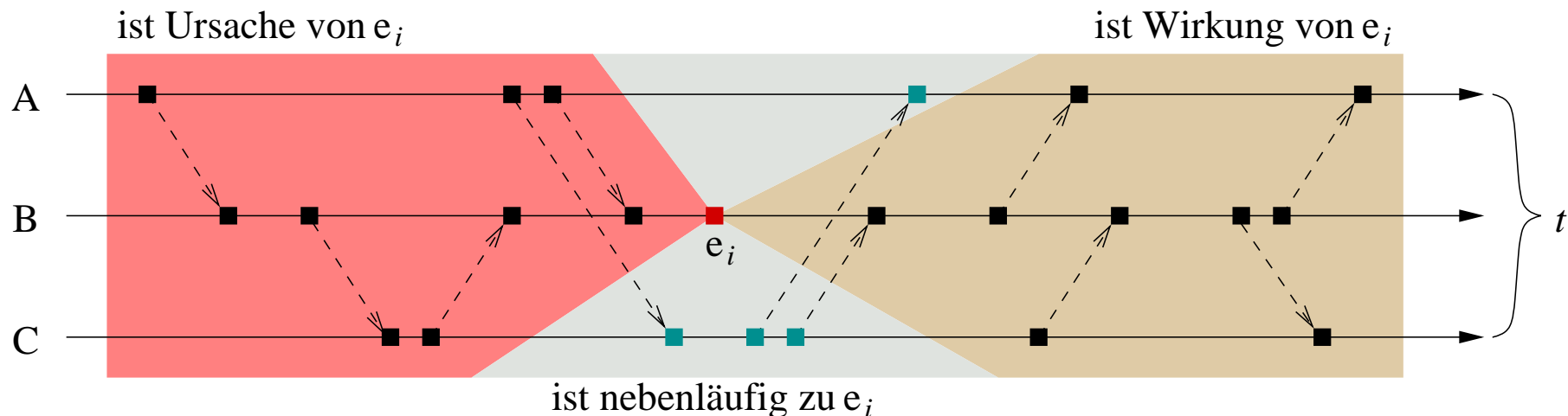
- Zeile 1 kann nebenläufig zu Zeile 2 ausgeführt werden
- Zeile 3 kann nebenläufig zu Zeile 4 ausgeführt werden

**Kausalität** (lat. *causa*: Ursache) ist die Beziehung zwischen **Ursache** und **Wirkung**, d.h., die ursächliche Verbindung zweier Ereignisse

- Ereignisse sind nebenläufig, wenn keines Ursache des anderen ist

# Ursache und Wirkung

Nebenläufigkeit als relativistischer Begriff von Gleichzeitigkeit



- ein Ereignis **ist nebenläufig zu** einem anderen ( $e_i$ ), wenn es im **Anderswo** des anderen Ereignisses ( $e_i$ ) liegt
  - d.h., weder in der Zukunft noch in der Vergangenheit des anderen
- das Ereignis ist nicht Ursache/Wirkung des anderen Ereignisses ( $e_i$ )
  - ggf. aber Ursache/Wirkung anderer (von  $e_i$  verschiedener) Ereignisse

# Rangfolge aus Gründen von Daten- und Zeitabhängigkeit

Aktionen können — um selbst ein korrektes Ergebnis zu produzieren — nebenläufig stattfinden, sofern:

**allgemein** keine das Resultat der anderen benötigt (S. 3)

- **Datenabhängigkeiten** gleichzeitiger Prozesse beachten

**speziell** (zusätzlich im Echtzeitbetrieb) keine die **Zeitbedingungen** der anderen verletzt

- Zeitpunkte dürfen nicht/nur selten verpasst werden
- Zeitintervalle dürfen nicht/nur begrenzt gedehnt werden

## Umgang mit Ereignissen bzw. Aktionen gleichzeitiger Prozesse

„ist Ursache von“  
„ist Wirkung von“ }  $\rightsquigarrow$  **Koordinierung** (vor/zur Laufzeit)  
„ist nebenläufig zu“  $\models$  **Parallelität** (implizit)

# Gliederung

- 1 Kausalitätsprinzip
  - Parallelisierbarkeit
  - Kausalordnung
- 2 Konkurrenz und Koordination
  - Koordinierung
  - Konkurrenz
- 3 Verfahrensweisen
  - Einordnung
  - Kritischer Abschnitt
  - Lebendigkeit
- 4 Zusammenfassung

# Reihenschaltung gleichzeitiger Aktivitäten

Maßnahme zum korrekten Zugriff gleichzeitiger Prozesse auf gemeinsame aber unteilbare Betriebsmittel, mit sehr unterschiedlicher Auswirkung:

- blockierend** ● pessimistische Annahme einer Überlappung ☹️
- nichtblockierend** ● optimistische Annahme keiner Überlappung 😊

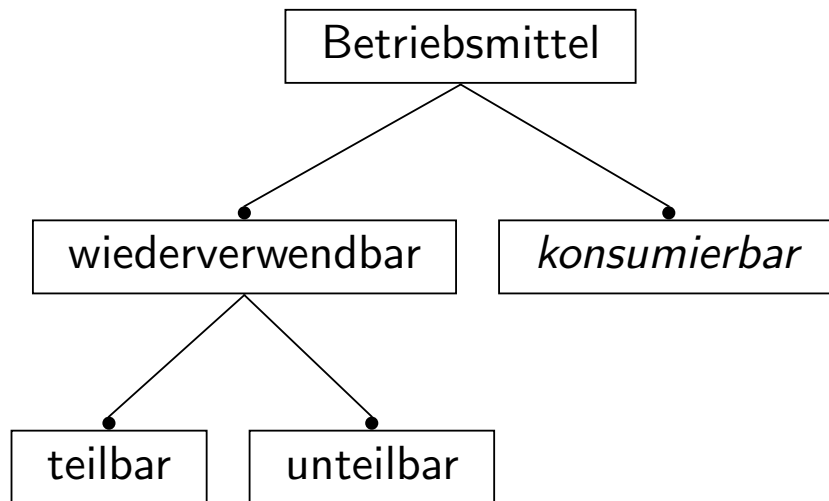
**Synchronisation** (gr. *syn*: zusammen, *chrónos*: Zeit) bezeichnet allg. das Herstellen von Gleichzeitigkeit

- (a) Koordination der Kooperation und Konkurrenz zwischen Prozessen ✕
- (b) Abgleich von Echtzeituhren (oder Daten) in verteilten Systemen
- (c) Sequentialisierung von Ereignissen entlang einer Kausalordnung

## Herangehensweisen

- analytisch** ● Prozesseinplanung: implizite Synchronisation
- konstruktiv** ● Programmieretechniken: **explizite Synchronisation** ✕

# Betriebsmittel und Betriebsmittelarten



## Hardware

- CPU, Speicher
- Geräte (Peripherie)
- *Signale*

## Software

- Dateien, E/A-Puffer
- Seitenrahmen
- Deskriptoren, ...
- *Signale, Nachrichten*

## Wettbewerb um Betriebsmittel (engl. *resource contention*)

Anzahl und Art der Betriebsmittel bedingen **problemspezifische Verfahren**:

- |                             |                                    |
|-----------------------------|------------------------------------|
| einseitige Synchronisation  | ● konsumierbare Betriebsmittel     |
| mehrseitige Synchronisation | ● wiederverwendbare Betriebsmittel |



# Konfliktfall bei unteilbaren Betriebsmitteln

Prozesse befinden sich untereinander im **Konflikt**, wenn:

- 1 nur eine begrenzte Anzahl gemeinsamer Betriebsmitteln vorrätig ist
- 2 diese Betriebsmittel unteilbar und von derselben Art sind
- 3 ein Zugriff darauf gleichzeitig geschieht: **gleichzeitige Prozesse** [1]

Prozesse sind im **Streit** (engl. *contention*) um ein Betriebsmittel, wenn einer das Betriebsmittel anfordert, das ein anderer bereits besitzt

- der anfordernde Prozess blockiert und wartet auf die Freigabe des Betriebsmittels durch den Prozess, der das Betriebsmittel belegt
- der das Betriebsmittel belegende Prozess löst den auf die Freigabe des Betriebsmittels wartenden Prozess aus, deblockiert ihn wieder

## Wechselseitiger Ausschluss

- $n$ -fach mehrseitige, für  $n - 1$  blockierend wirkende Synchronisation
- eine Option zum Schutz eines kritischen Abschnitts

# Wechselseitiger Ausschluss

Reihenschaltung von Prozessen mit begrenzten/unteilbaren Betriebsmitteln

**Sequenzialisierung** der Zugriffe gleichzeitiger Prozesse per Protokoll:

**Vergabe**  $\mapsto$  das Betriebsmittel sperren und dem Prozess zuteilen

- beim Versuch, ein gesperrtes Betriebsmittel erneut zu belegen, wird der anfordernde Prozess blockiert
- der blockierende Prozess erwartet das Ereignis/Signal zur Freigabe des gesperrten Betriebsmittels, ihm wird die CPU entzogen

**Freigabe**  $\mapsto$  das Betriebsmittel dem Prozess wieder entziehen

- sollten Prozesse die Freigabe dieses Betriebsmittels erwarten, wird es sofort der **Wiedervergabe** zugeführt; das bedeutet:
  - (a) das Betriebsmittel entsperren und alle Prozesse deblockieren, die sich dann wiederholt um die Vergabe zu bemühen haben *oder*
  - (b) einen Prozess auswählen und ihm das Betriebsmittel zuteilen
- nur der das Betriebsmittel „besitzende“ Prozess kann es freigeben

# Reihenschaltung gleichzeitiger Prozesse: Koordinierung

Synchronisation  $\equiv$  Koordination der Kooperation und Konkurrenz zwischen Prozessen

**ko-or-di'nie-ren** beiordnen; in ein Gefüge einbauen; aufeinander abstimmen; nebeneinanderstellen; Termine  $\sim$ .

- sich überlappen könnende Aktivitäten der Reihe nach ausführen
  - gleichzeitige Prozesse im kritischen Abschnitt koordinieren
- „der Reihe nach“  $\rightsquigarrow$  **Verzögerung gleichzeitiger Prozesse erzwingen**
  - (a) potentiell überlappende Prozesse verzögern, der Konflikt ist ungewiss
  - (b) den überlappten Prozess verzögern, der Konflikt ist gewiss

## Synchronisationsverfahren...

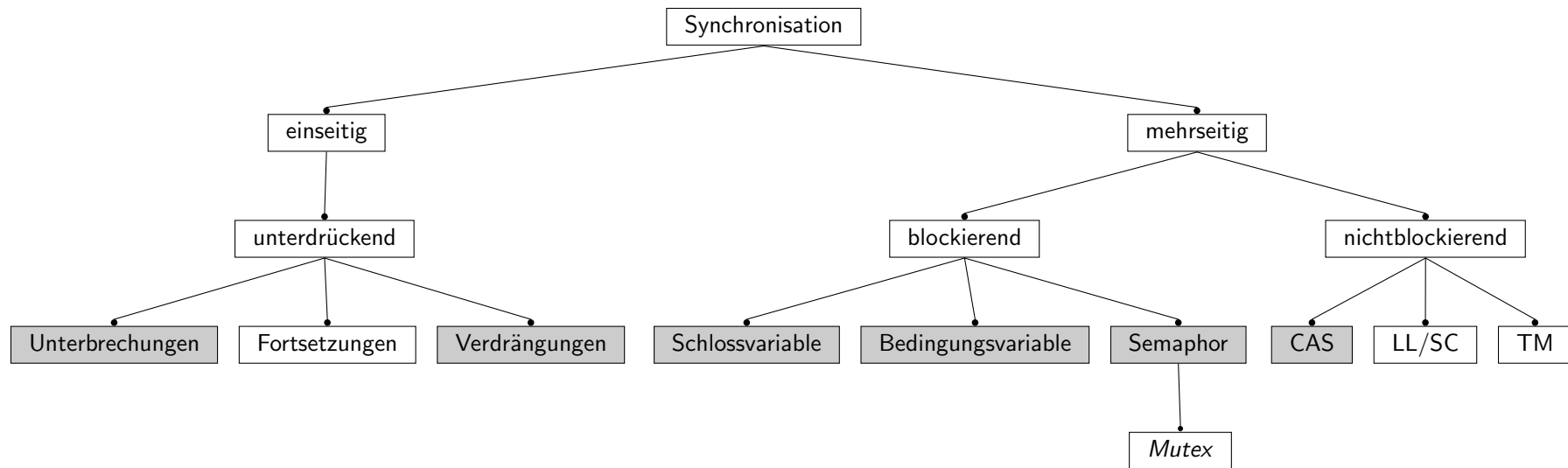
- wirken einseitig oder mehrseitig
  - unterdrückend, blockierend, nichtblockierend
    - behinderungs-, sperr-, wartefrei



# Gliederung

- 1 Kausalitätsprinzip
  - Parallelisierbarkeit
  - Kausalordnung
- 2 Konkurrenz und Koordination
  - Koordinierung
  - Konkurrenz
- 3 Verfahrensweisen
  - Einordnung
  - Kritischer Abschnitt
  - Lebendigkeit
- 4 Zusammenfassung

# Arten der Synchronisation



## SP: Untersuchte Verfahren

(Grau hinterlegte Techniken)

einseitig unterdrueckend: Ereignisse (Unterbrechungen, Verdraengungen)

mehrseitig blockierend bzw. nichtblockierend: Prozessinkarnationen

Ebene<sub>5</sub> Bedingungsvariable, Monitor

Ebene<sub>3</sub> Verdraengungen, Semaphor

Ebene<sub>2</sub> Unterbrechungen, Schlossvariable, CAS

# Einseitige Synchronisation

## Unilateral

Auswirkung haben die Verfahren nur auf einen der beteiligten Prozesse; zwei Begrifflichkeiten sind gebräuchlich:

### Bedingungssynchronisation

- der Ablauf des einen Prozesses ist abhängig von einer Bedingung
- der andere Prozess erfährt keine Verzögerung in seinem Ablauf

### logische Synchronisation

- die Maßnahme resultiert aus der logischen Abfolge der Aktivitäten
- vorgegeben durch das „Rollenspiel“ der beteiligten Prozesse

Beachte: Andere Prozesse sind jedoch nicht gänzlich unbeteiligt

- die Veränderung einer Bedingung, auf die ein Prozess wartet, ist z.B. von einem anderen Prozess herbeizuführen

# Mehrseitige Synchronisation

## Multilateral

Auswirkung haben die Verfahren ggf. auf alle beteiligten Prozesse:

- welche Prozesse verzögert werden, ist i.A. unvorhersehbar

### Beispiel: Wechselseitiger Ausschluss (engl. *mutual exclusion*)

- erzwungene sequentielle Ausführung von Anweisungsfolgen
- im Regelfall zeitlich begrenzte, exklusive Betriebsmittelvergabe

### Beispiel: Kritischer Abschnitt (engl. *critical section/region*)

wettlaufintolerant

- wechselseitiger Ausschluss
- Verzögerung ggf. folgender gleichzeitiger Prozesse

wettlaufftolerant

- nichtsequentielle Ausführung
- Verzögerung überlappter gleichzeitiger Prozesse

# Betriebsmittelart und Synchronisationstechnik

## Artenspezifische Verfahren

Notwendigkeit des Ausschlusses gleichzeitiger Prozesse ist immer nur in folgenden Fällen gegeben  $\rightsquigarrow$  **blockierende Synchronisation**:

- beim Zugriff auf unteilbare wiederverwendbare Betriebsmittel
- bei Inanspruchnahme eines konsumierbaren Betriebsmittels<sup>1</sup>

Schutz eines kritischen Abschnitts (KA) bedingt sich in anderer Weise und ist auf zwei verschiedenen Wegen möglich:

**pessimistisch**  $\hookrightarrow$  **wechselseitiger Ausschluss** (block. Synchronisation) ☹️

- KA implementiert als sequentielles Programm
- entsprechend der „herkömmlichen Sichtweise“ (S. 17)

**optimistisch**  $\hookrightarrow$  **nichtblockierende Synchronisation** 😊

- KA implementiert als *nichtsequentielles Programm*
- entsprechend der „alternativen Sichtweise“ (S. 17)

---

<sup>1</sup>Der Konsument erwartet die Betriebsmittelbereitstellung durch den Produzenten.



# Kritischer Abschnitt (KA)

Kennzeichnend für mehrseitige Synchronisation

Aspekte eines KA [4, S. 137], die die herkömmliche Sichtweise darlegen:

- sich gegenseitig ausschließende Aktivitäten
  - werden nie parallel ausgeführt
  - verhalten sich zueinander, als seien sie unteilbar, weil keine Aktivität die andere unterbricht
- Anweisungen, deren Ausführung wechselseitigen Ausschluss erfordert

Eine Frage der Abstraktionsebene  $\leftrightarrow$  alternative Sichtweise

- ein wechselseitiger Ausschluss ist nicht zwingend, um einen kritischen Abschnitt zu schützen
- vielmehr gilt es sicherzustellen, dass die Ausführung eines solchen Abschnitts jederzeit ein **konsistentes Ergebnis** liefert

- prozedurale Abstraktion eines KA als **Elementaroperation (ELOP)**

# Verhaltensregeln zum Durchlaufen kritischer Abschnitte

Betreten (engl. *enter*) und Verlassen (engl. *leave*) kritischer Abschnitte steuern **problemspezifische Protokolle**

**Eintrittsprotokoll** (engl. *entry protocol*)

- regelt die Belegung des kritischen Abschnitts durch einen Prozess
  - erteilt einem Prozess die Zugangsberechtigung
- bei bereits belegtem Abschnitt: **Wettstreitigkeit** (engl. *contention*)
  - blockierend** • den eintreffenden Prozess am Eintritt hindern
  - nichtblockierend** • den Abschnitt belegenden Prozess überlappen dürfen

**Austrittsprotokoll** (engl. *exit protocol*)

- regelt die Freigabe des kritischen Abschnitts durch einen Prozess
  - blockierend** • am Eintritt ggf. gehinderte(n) Prozess(e) freistellen
  - nichtblockierend** • den überlappten Prozess zurücksetzen/wiederholen
- Prozesse können den kritischen Abschnitt (wieder) belegen

- die Vorgehensweise variiert mit dem jew. Synchronisationsverfahren

## Bedingter kritischer Abschnitt<sup>2</sup>

Betreten des kritischen Abschnitts ist von einer Wartebedingung abhängig, die nicht erfüllt sein darf, um den Prozess fortzusetzen

- die Bedingung ist als **Prädikat** über die im kritischen Abschnitt enthaltenen bzw. verwendeten Daten definiert
- typischerweise technisch realisiert durch eine **Bedingungsvariable**

Auswertung der Wartebedingung muss im kritischen Abschnitt erfolgen

- bei Nichterfüllung der Bedingung wird der Prozess auf Eintritt eines zur Wartebedingung korrespondierenden Ereignisses blockiert
  - damit das Ereignis später signalisiert werden kann, muss der kritische Abschnitt beim Schlafenlegen jedoch freigegeben werden
- bei (genauer: nach) Erfüllung/Signalisierung der Bedingung versucht der Prozess den kritischen Abschnitt wieder zu belegen
  - ggf. muss ein deblockierter Prozess die Bedingung neu auswerten

---

<sup>2</sup>engl. *conditional critical section* resp. *region*, [5].

# Sprachunterstützung: Hoare [5]

Koordinierungsbefehle automatisch abgesetzt durch einen Kompilierer

## Kritischer Abschnitt

```
resource r;  
with r do  
  begin  
    ⋮  
  end
```

- with**
- Selektionsanweisung
  - r ist „beliebige“ Variable
  - assoziiert Daten mit KA
- do**
- Anweisung(en) des KA

## Bedingter kritischer Abschnitt

```
resource r; boolean b;  
with r when b do  
  begin  
    ⋮  
  end
```

- when**
- Fortführungsbedingung
  - *true*  $\leadsto$  fortfahren
  - *false*  $\leadsto$  warten

## Wechselseitiger Ausschluss

- für die gesamte **do**-Anweisung

## Signalisierung und Freistellung

- implizit bei Zustandswechsel

# Fortschrittsgarantien

Aussagen zur Lebendigkeit (engl. *liveliness*) nichtsequentieller Programme [2, 3]

## behinderungsfrei (engl. *obstruction-free*)

- ein einzelner, in Isolation ablaufender Faden wird seine Operation in begrenzter Anzahl von Schritten beenden
- ein Faden läuft in Isolation ab, wenn alle ihn behindern könnenden anderen Fäden in der Ausführung zurückgestellt sind

## sperrfrei (engl. *lock-free*), umfasst Behinderungsfreiheit

- jeder bei Ablauf eines Fadens auszuführende Schritt trägt dazu bei, dass das nichtsequentielle Programm insgesamt voranschreitet
- garantiert systemweiten Durchsatz, erlaubt jedoch **Aushungerung** (engl. *starvation*) eines einzelnen Fadens

## wartefrei (engl. *wait-free*), umfasst Sperrfreiheit

- die Anzahl der zur Beendigung einer Operation bei Fadenabläufen auszuführenden Schritte ist begrenzt
- garantiert systemweiten Durchsatz und ist frei von Aushungerung

# Gliederung

- 1 Kausalitätsprinzip
  - Parallelisierbarkeit
  - Kausalordnung
- 2 Konkurrenz und Koordination
  - Koordinierung
  - Konkurrenz
- 3 Verfahrensweisen
  - Einordnung
  - Kritischer Abschnitt
  - Lebendigkeit
- 4 Zusammenfassung

# Dualität von Koordinierungstechniken

Theorie vs. Praxis

## Problem:

- Erhöhung von Parallelität
- wechselseitiger Ausschluss
- explizite Prozesssteuerung
- bedingte Verzögerung
- Austausch von Zeitsignalen
- Austausch von Daten

## Methode:

- nichtblockierende Synchronisation
- Schlossvariable
- Bedingungsvariable
- bedingter kritischer Abschnitt
- Semaphor
- Nachrichtenpuffer

**logisch betrachtet** sind alle Methoden äquivalent, da jede von ihnen hilft, ein beliebiges Steuerungsproblem zu lösen

**praktisch betrachtet** sind die Methoden nicht äquivalent, da einige von ihnen für ein gegebenes Problem zu komplexen und ineffizienten Lösungen führen

# Resümee

- **Nebenläufigkeit** setzt voneinander unabhängige Prozesse voraus
  - bezeichnet das Verhältnis von nicht kausal abhängigen Ereignissen
  - schränkt sich ein aus Gründen von Daten- oder Zeitabhängigkeit
- gleichzeitige abhängige Prozesse implizieren **Koordinierung**
  - nämlich der Kooperation und Konkurrenz zwischen Prozessen
  - durch analytische (implizite) oder konstruktive (explizite) Techniken
- **Synchronisation** zeigt einen großen Facettenreichtum
  - klassifiziert nach der jeweiligen Auswirkung auf beteiligte Prozesse:
    - einseitig oder mehrseitig
    - unterdrückend, blockierend oder nicht-blockierend
    - behinderungs-, sperr- oder wartefrei
  - verschiedenen Abstraktionsebenen eines Rechensystems zugeordnet:
    - **Hochsprachenebene** Bedingungsvariable, Monitor
    - **Maschinenprogrammenebene** Verdrängungssteuerung, Semaphore
    - **Befehlssatzebene** Schlossvariable, Spezialbefehle (CPU)
- Aussagen zur „Lebendigkeit“ nichtsequentieller Programme leiten sich aus den **Fortschrittsgarantien** der Synchronisationsverfahren ab



# Literaturverzeichnis

- [1] HANSEN, P. B.:  
*Operating System Principles.*  
Prentice Hall International, 1973
- [2] HERLIHY, M. :  
Wait-Free Synchronization.  
In: *ACM Transactions on Programming Languages and Systems* 11 (1991), Jan., Nr. 1, S. 124–149
- [3] HERLIHY, M. ; LUCHANGCO, V. ; MOIR, M. :  
Obstruction-Free Synchronization: Double-Ended Queues as an Example.  
In: *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS 2003), May 19–22, 2003, Providence, Rhode Island, USA*, IEEE Computer Society, 2003, S. 522–529
- [4] HERRTWICH, R. G. ; HOMMEL, G. :  
*Kooperation und Konkurrenz — Nebenläufige, verteilte und echtzeitabhängige Programmsysteme.*  
Springer-Verlag, 1989. –  
ISBN 3–540–51701–4

# Literaturverzeichnis (Forts.)

- [5] HOARE, C. A. R.:  
Towards a Theory of Parallel Programming.  
In: HOARE, C. A. R. (Hrsg.) ; PERROT, R. H. (Hrsg.): *Operating System Techniques*.  
New York, NY : Academic Press, Inc., Aug. – Sept. 1971 (Proceedings of a Seminar at  
Queen's University, Belfast, Northern Ireland), S. 61–71