Vorlesung Softwaretechnik
- Regelbasierte und zustandsorientierte Sicht -

Prof. Klaus-Peter Fähnrich

Wintersemester 2008/2009

Überblick LE 10

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

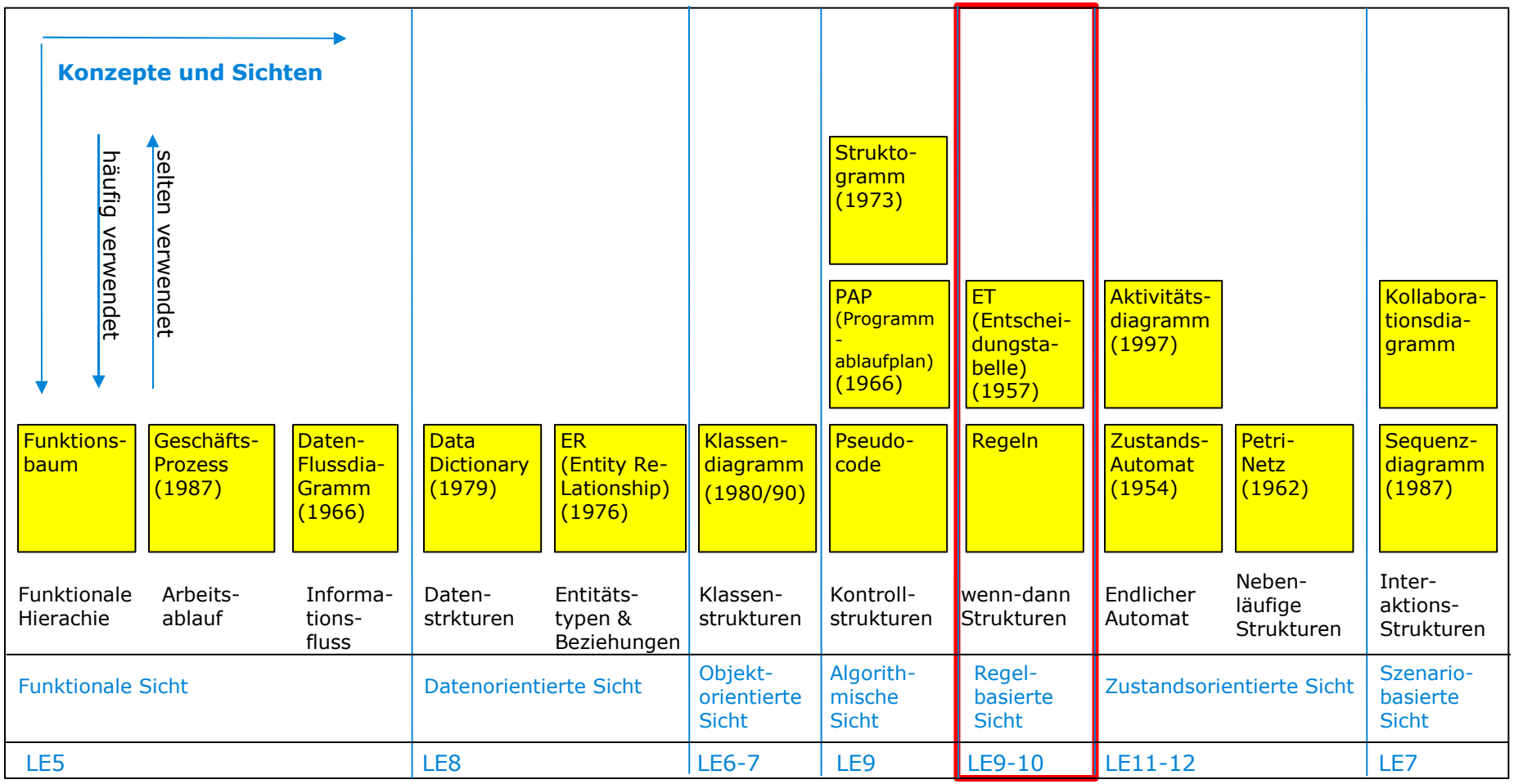
LE 10: Regelbasierte Sicht

- Regeln
  - Vorwärtsverkettung
  - Rückwärtsverkettung
  - Vorwärtsverkettung vs. Rückwärtsverkettung
  - Tiefensuche vs. Breitensuche
  - Komplexität der Vorbedingungen
  - Strukturierung von Regeln
- Zusammenfassung

LE 11 und 12: Zustandsorientierte Sicht

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Überblick



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Lernziele

1. Aufbau einer Regel
2. Komponenten eines regelbasierten Expertensystems
3. Plan und Planung
4. Die Strategien Vorwärtsverkettung, Rückwärtsverkettung, Tiefensuche und Breitensuche
5. Erläutern von Konfliktmenge, Konfliktlösungsstrategie, Meta-Regel, Regelinterpreter, Agenda und Feuern anhand von Beispielen.
6. Aufstellung von Regeln bei gegebener Problemstellung
7. Ausführen von Regelmengen nach den Strategien Vorwärtsverkettung, Rückwärtsverkettung, Tiefensuche und Breitensuche.
8. Planungsvarianten am Beispiel der dargestellten Block-Welt
9. Beurteilung von Konfliktlösungsstrategien

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Regeln

Eine Regel besteht aus einer Vorbedingung und einer Aktion

- wenn Vorbedingung dann Aktion
- Vorbedingung beschreibt eine Situation, in der die Aktion ausgeführt werden soll.

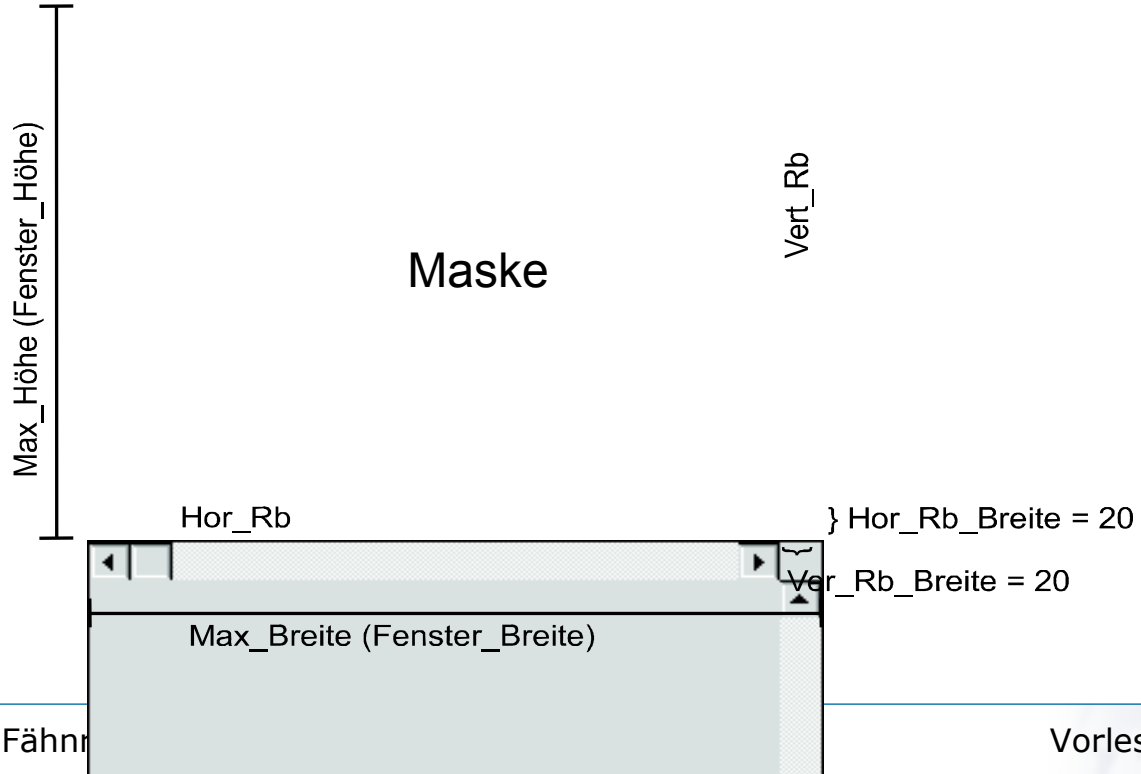
- Zwei Typen von Aktionen
  - **Implikationen** oder **Deduktionen**, mit denen der Wahrheitsgehalt einer Feststellung hergeleitet wird.
  - **Handlungen**, mit denen ein Zustand verändert wird.

- Regeln vergleichbar mit denen in Entscheidungstabellen
  - Regeln stehen hier jeweils für sich,
  - ET sind in einen festen Kontext eingebettet.
  - Bei den Aktionen handelt es sich um Handlungen.
  - Lassen sich Anforderungen an ein neues System noch nicht als Entscheidungstabellen oder Bäume strukturieren, dann als Menge einzelner Regeln formulieren.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Beispiel zu Regeln - Erfassungsmaske 1

- Erfassungsmaske soll durch ein Fenster eingerahmt werden
- Dieses Fenster hat eine maximale Größe
- Überschreitet die Maskenbreite die Fensterbreite, so ist ein horizontaler Rollbalken einzufügen
- Die Breite dieses Rollbalkens verringert wiederum die maximal verfügbare Fensterhöhe
- Bei zu hoher Maske wird ein vertikaler Rollbalken eingeführt.



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Beispiel zu Regeln - Erfassungsmaske 2

- Es soll ein Fenster ausgegeben werden, das die maximale Fenstergröße nicht überschreitet. Rollbalken sollen nur dann in das Fenster eingesetzt werden, wenn sie benötigt werden.
- Regeln zur Problembeschreibung:

(R1) //Ziel: Horizontale Rollbalken setzen

```
wenn Hor_Rb = JA dann Hor_Rb_Breite = 20
```

(R2) //Ziel: Horizontale Rollbalken löschen

```
wenn Hor_Rb = NEIN dann Hor_Rb_Breite = 0
```

(R3) //Ziel: Vertikale Rollbalken setzen

```
wenn Ver_Rb = JA dann Ver_Rb_Breite = 20
```

(R4) //Ziel: Vertikale Rollbalken löschen

```
wenn Ver_Rb = NEIN dann Ver_Rb_Breite = 0
```

(R5) //Ziel: Berechnung der Fensterhöhe

```
wenn (Masken_Höhe=?maske) und (Hor_Rb_Breite = ?rb)
dann Fenster_Höhe = ?maske + ?rb
```

(R6) //Ziel: Berechnung der Fensterbreite

```
wenn (Masken_Breite = ?maske) und(Ver_Rb_Breite = ?rb)
dann Fenster_Breite = ?maske + ?rb
```

(R7) //Beschränkung: Fensterbreite nicht überschreiten

```
wenn Fenster_Breite >= Max_Breite
dann //Einfügen horizon. Rollbalken; Hor_Rb = JA
       und Fenster_Breite=Max_Breite
```

(R8) //Fensterhöhe nicht überschreiten

```
wenn Fenster_Höhe >= Max_Höhe
dann //Einfügen vertikaler Rollbalken;Ver_Rb = JA
       und Fenster_Höhe = Max_Höhe;
```

Durch ein vorangestelltes Fragezeichen wird eine Variable gekennzeichnet, z.B. ?maske, ?rb.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Regelbasierte Expertensysteme

Zwei Strategien zur Abarbeitung von Regeln:

- **Vorwärtsverkettung** (forward chaining)
  (Datengesteuerte Systeme)
  - Ausgangspunkt: Vorhandene Datenbasis mit Regeln
  - Auswahl aller Regeln, deren **Vorbedingung** in der Datenbasis erfüllt ist
    - Selektion einer dieser Regeln
    - Ausführung des Aktionsteils
    - Die Regel »feuert«
    - Dadurch Änderung der Datenbasis
    - Wiederholung, bis keine Regel mehr anwendbar ist.
- **Rückwärtsverkettung** (backward chaining)
  (Zielgesteuerte Systeme)
  - Ausgehend von einem **Ziel** werden nur die Regeln überprüft, deren Aktionsteil das Ziel enthält.
  - Falls Parameter der Vorbedingung unbekannt sind, werden sie vom Benutzer erfragt oder mit anderen Regeln hergeleitet.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Vorwärtsverkettung

- Ein vorwärtsverketteter Regelinterpreter arbeitet nach folgendem Algorithmus:

```
(1)  DATEN:= Ausgangsdatenbasis
(2)  while Inhalt von DATEN erfüllt nicht das Endekriterium
     {
(3)     Wähle eine anwendbare Regel R, deren Bedingungsteil durch DATEN
        erfüllt ist
(4)     DATEN:= Ergebnis der Anwendung des Aktionsteils von R auf DATEN
     }
```

- Das Auswahlverfahren im Schritt (3) wird meist in zwei Teilschritten Vorgenommen:

  (3a) Vorauswahl:
  Bestimmung der Menge aller ausführbaren Regeln (Konfliktmenge)

  (3b) Auswahl:
  Auswahl einer Regel aus der Konfliktmenge entsprechend einer **Konfliktlösungsstrategie** (Auswahl nach **Reihenfolge**, Auswahl nach **syntaktischer Struktur** der Regeln, Auswahl mittels **Zusatzwissens**) oder einer Kombination aus mehreren.

- Diese Art des Schlussfolgerns wird als **»Erkennen-Handeln«-Zyklus** bezeichnet.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Konfliktlösungsstrategien

- Auswahl nach Reihenfolge z.B.:
  - Erste anwendbare Regel feuert (Trivialstrategie) (im Beispiel R2)
  - Aktuellste Regel feuert
    - Regel, deren Vorbedingung sich auf möglichst neue Einträge in der Datenbasis bezieht
    - Regel, die zuletzt in die Agenda aufgenommen wurde
- Auswahl nach syntaktischer Struktur der Regeln
  - Spezifischste Regel feuert
    - Regel, deren Vorbedingung die einer anderen Regel enthält und zusätzlich noch weitere Aussagen
  - Syntaktisch größte Regel feuert
    - Regel, die die meisten Aussagen enthält (im Beispiel R5 oder R6).
- Auswahl mittels Zusatzwissens, z.B.:
  - Regel mit der höchsten Priorität feuert
    - Dazu muss jede Regel eine Priorität, die z.B. als Zahl repräsentiert sein kann, zugeordnet werden.
  - Zusätzliche Regeln (sog. Meta-Regeln) steuern den Auswahlprozess
    - z.B. Alle Beschränkungsregeln haben Vorrang)
- Kombinationen zwischen verschiedenen Strategien sind möglich.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Beispiel zu Regeln - Erfassungsmaske 3

- Am Anfang werden durch den Benutzer folgende Fakten vorgegeben (Voreinstellungen):
  - (F1) Max_Breite = 400 (Pixel)
  - (F2) Max_Höhe = 400 (Pixel)
  - (F3) Hor_Rb = NEIN
  - (F4) Ver_Rb = NEIN
- Es soll ein Fenster für folgende Maskenwerte konfiguriert werden:
  - (F5) Masken_Breite = 390 (Pixel)
  - (F6) Masken_Höhe = 500 (Pixel)
- Diese Fakten bilden die Ausgangsdatenbasis.

- Es wird eine Kombination zweier Konfliktlösungsstrategien angewandt:
  a) Die aktuellste Regel feuert
  b) Die spezifischste Regel feuert

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Beispiel zu Regeln – Erfassungsmaske 4

Maske

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Rückwärtsverkettung

- Startet mit einem vorgegebenen Ziel.
- Wenn das Ziel nicht in der Datenbasis bekannt ist, entscheidet der Regelinterpreter, ob es abgeleitet werden kann oder erfragt werden muss.
- Konfliktlösungsstrategien hierzu sind :
  - erst fragen, dann ableiten;
  - erst ableiten, dann fragen und
  - Mischformen mit weiteren Kriterien.
- Bei der Ableitung werden alle Regeln abgearbeitet, in deren Aktionsteil das Ziel enthalten ist. Diese lassen sich durch Indizierung der Regeln gemäß ihrer Aktionsteile effizient bestimmen.
- Ist bei der Überprüfung der Aussagen einer Regel ein Parameter unbekannt:
  - Ein Unterziel wird zur Bestimmung dieses Parameters generiert und die Prozedur Bestimme(Ziel) rekursiv angewendet.
  - Endergebnis:
    - Wert für das vorgegebene Ziel und für alle Unterziele
    - Evaluation der relevanten Regeln und das Stellen der notwendigen Fragen
  - Rückwärtsverkettung: enthält eine Dialogsteuerung
    - Reihenfolge der gestellten Fragen
      - Hängt von der Reihenfolge der Regeln zur Ableitung eines Parameters und von der Reihenfolge der Aussagen in der Vorbedingung einer Regel ab.
      - Erschwert eine Modularisierung des Regelsystems.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Rückwärtsverkettung

- Effizienz wird durch die Formulierung des Ziels bestimmt:
  - Je präziser das Ziel, desto kleiner der Suchbaum von zu überprüfenden Regeln und zu stellenden Fragen
  - Beispiel:
    - Wie heißt die Krankheit? vs.
    - Heißt die Krankheit x?

- Beispiele für rückwärtsverkettete Regelinterpreter
  - PROLOG (Programmiersprache)
  - EMYCIN (Expertensystem).

Unterschiede Vorwärts/Rückwärtsverkettung

Art des **Suchraums** bestimmt, ob

- Vorwärtsverkettung oder
- Rückwärtsverkettung effizienter ist.

Probleme im Zusammenhang mit **Steuerungsfragen**

- werden oft mit Hilfe von vorwärtsverketteten Inferenzmechanismen gelöst.
- Problemklasse hat oft eine große Zahl möglicher Ergebnisse.
- Das Ziel oder die Lösung wird konstruiert oder zusammengefügt.
  - Beispiel: Konstruktion von Bildschirm-Fenstern.
- Planung von Roboteroperationen:
  - Vorwärtsverkettung ungünstig
    - Im Prinzip müssen alle Operationen betrachtet werden, die von der Ausgangssituation aus ausführbar sind, auch wenn sie mit der Lösung des Problems nichts zu tun haben.
  - Ursache dieses Effekts ist es, dass eine Situation mehr Nachfolger als Vorgänger hat.
- Mögliche Ergebnisse bekannt, Anzahl in vernünftigen Grenzen
  - Rückwärtsverkettung sehr effizient
  - Automatische Zielzentriertheit.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Unterschiede Tiefen/Breitensuche

Zusätzliche Unterscheidung

- Tiefensuche (depth first search)
- Breitensuche (breadth first search)

Tiefensuche

- Nach Ermittlung des ersten Teilziels wird das erste Teilziel des Teilziels gesucht usw..
- Suche geht immer ins Detail.
  - Führt eine verfolgte Regelkette nicht zum Erfolg, dann wird nächstes Teilziel der vorhergehenden Ebene betrachtet und dort in die Tiefe gegangen
- Backtracking
  - Abbrechen einer verfolgten Regelkette wegen Misserfolgs (Sackgasse) und Zurücksetzen auf den letzten Alternativpunkt.

Breitensuche

- Alle Vorbedingungen einer Regel werden untersucht, bevor mehr ins Detail gegangen wird
- Dies ist effizienter, wenn eine der Regeln sich als zutreffend erweist und das Ziel damit ermittelt wird.
- Beispiel: Labyrinth
  - An jedem Verzweigungspunkt wird in beide Richtungen gegangen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Beispiele Tiefen/Breitensuche

- **Tiefensuche:**
  Es wird versucht so weit wie möglich „voran zugehen".
  In der Sackgasse zurück zum vorherigen Entscheidungspunkt.

Start

Ziel

- **Breitensuche:**
  An jedem Entscheidungspunkt wird in alle Richtungen gegangen.

Start

Ziel

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Komplexität der Vorbedingung

- Komplexitätstypen von Regeln

Aussagetyp	Art der Auswertung	Beispiel
Erfassung zunehmender komplizierter Zusammenhänge ↓	Nachschauen in der Datenbasis	Masken_Breite = 300?
	Nachschauen in der Datenbasis und Rechnen	Fenster_Breite = ?
	Mustervergleich (pattern matching) und Rechnen	Wird Fenster x angezeigt?

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Strukturierung von Regeln

Diagnosesysteme

- Aufteilung nach den Hypothesen, die sie überprüfen.

Konstruktionssysteme

- Den Zuständen werden Regelmengen zugeordnet,
  - die Regeln des aktuellen Zustandes untersuchen.

Strukturierungsmöglichkeit

- Syntaktische Unterscheidung zwischen verschiedenen Typen von Aussagen in der Vorbedingung einer Regel.

Bewertung

- Die einzelnen Wissenseinheiten werden in elementarer Form festgelegt.
- Das Wissen wird explizit repräsentiert.
- U.U. können die Regeln natürlichsprachlich dargestellt werden.
- Die Regeln können leicht geändert werden.
- Im Idealfall können die Regeln in beliebiger Reihenfolge aufgeschrieben werden.
- Der Lösungsweg wird von dem Inferenzmechanismus selbständig gesucht.
- Inbetriebnahme bzw. Test ist auch bei unvollständiger Regelmenge möglich.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Zusammenfassung

Regeln erlauben die Beschreibung von Anforderungen an ein neues System in der Wenn-Dann-Form. Werden Regeln exakt entsprechend einer gegebenen Syntax und Semantik formuliert, dann können sie auch maschinell durch regelbasierte Expertensysteme abgearbeitet werden. Ein Regelinterpreter liest die gespeicherten Regeln und prüft sie ausgehend vom vorgegebenem Ziel (**Rückwärtsverkettung, backward chaining**) oder ausgehend von den vorgegebenen Fakten (**Vorwärtsverkettung, forward chaining**).

Alle anwendbaren Regeln werden in einer Agenda abgelegt (**Konfliktmenge**). Entsprechend einer Konfliktlösungsstrategie (z. B. **Tiefensuche** bzw. **depth first search**, **Breitensuche** bzw. **breath first search**, **Meta-Regeln**) wird einen Regel aus der Agenda ausgewählt. Das Anwenden der ausgewählten Regeln bezeichnet man als Feuern der Regel. Gerät der Regelinterpreter beim Überprüfen von Regeln in einer Sackgasse, dann erfolgt ein **Backtracking**.

Probleme bei denen eine Operatorsequenz gesucht wird, um eine Ausgangssituation in eine Zielsituation zu transformieren, werden als **Planungsprobleme** bezeichnet. Das ermittelte Ergebnis ist ein **Plan**.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Literatur

- Balzert H., Lehrbuch der Softwaretechnik, 2. Auflage, Spektrum Akademischer Verlag,Heidelberg, 2000.

- Buchanan B. G., Shortliffe E. H., Rule-Based Expert Systems. The MYCIN Experiments of the Stanford Heuristic Programming Project, Addison-Wesley, 1985
- Crevier D., Ai. The Tumultuous History of the Search of Artificial Intelligence, Basic Books, New York, 1993
- Jüttner G., Feller H., Entscheidungstabellen und wissensbasierte Systeme, Oldenbourg Verlag, München, 1989
- Puppe F., Problemlösungsmethoden in Expertensystemen, Springer-Verlag, Berlin, 19990
- Puppe F., Einführung in Expertensysteme, Springer-Verlag, Berlin 1988

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Überblick LE 11 und LE 12

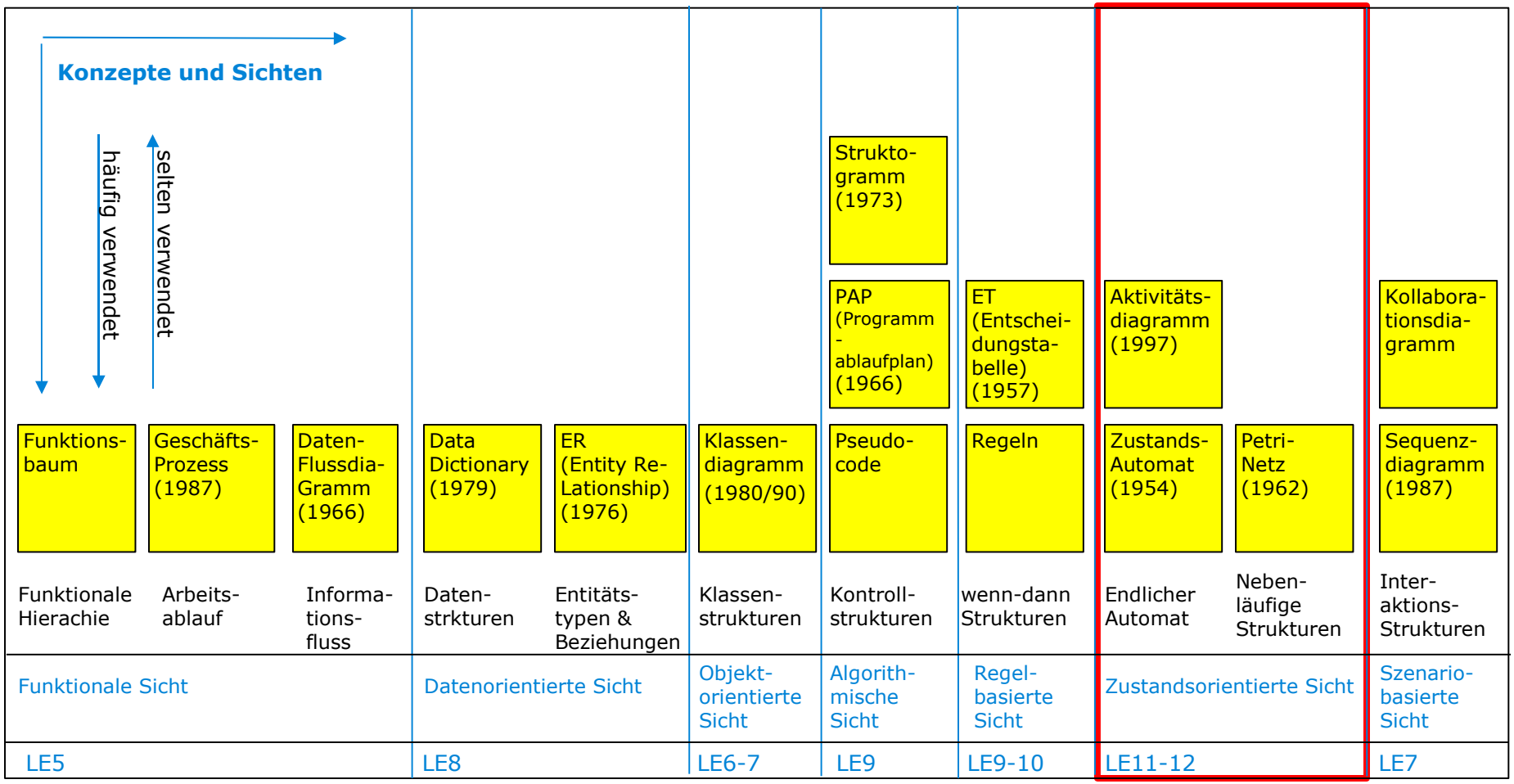
LE 10: Regelbasierte Sicht

LE 11 und 12: Zustandsorientierte Sicht

- Zustandsautomaten
- Petri-Netze

Überblick

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Lernziele

1. Mealy- and Moore-Automaten
2. Dynamik eines Zustandsautomaten
3. Die Weiterungen von Harel
4. Unterschied zwischen Mealy- und Moore-Automat
5. UML-Notation und –Semantik von Zustands- und Aktivitätsdiagrammen.
6. Erstellen eines Zustandsautomaten für gegebene Problemstellungen
7. Überführen der verschiedenen Darstellungsweisen eines Zustandsautomaten ineinander.
8. Modellierung des Lebenszyklus eines Objekts als UML-Zustandsautomaten
9. Konstruktion von UML-Aktivitätsdiagrammen für gegebene Problemstellungen
10. Erstellen von Zustands- und Aktivitätsdiagrammen mit CASE-Werkzeugen

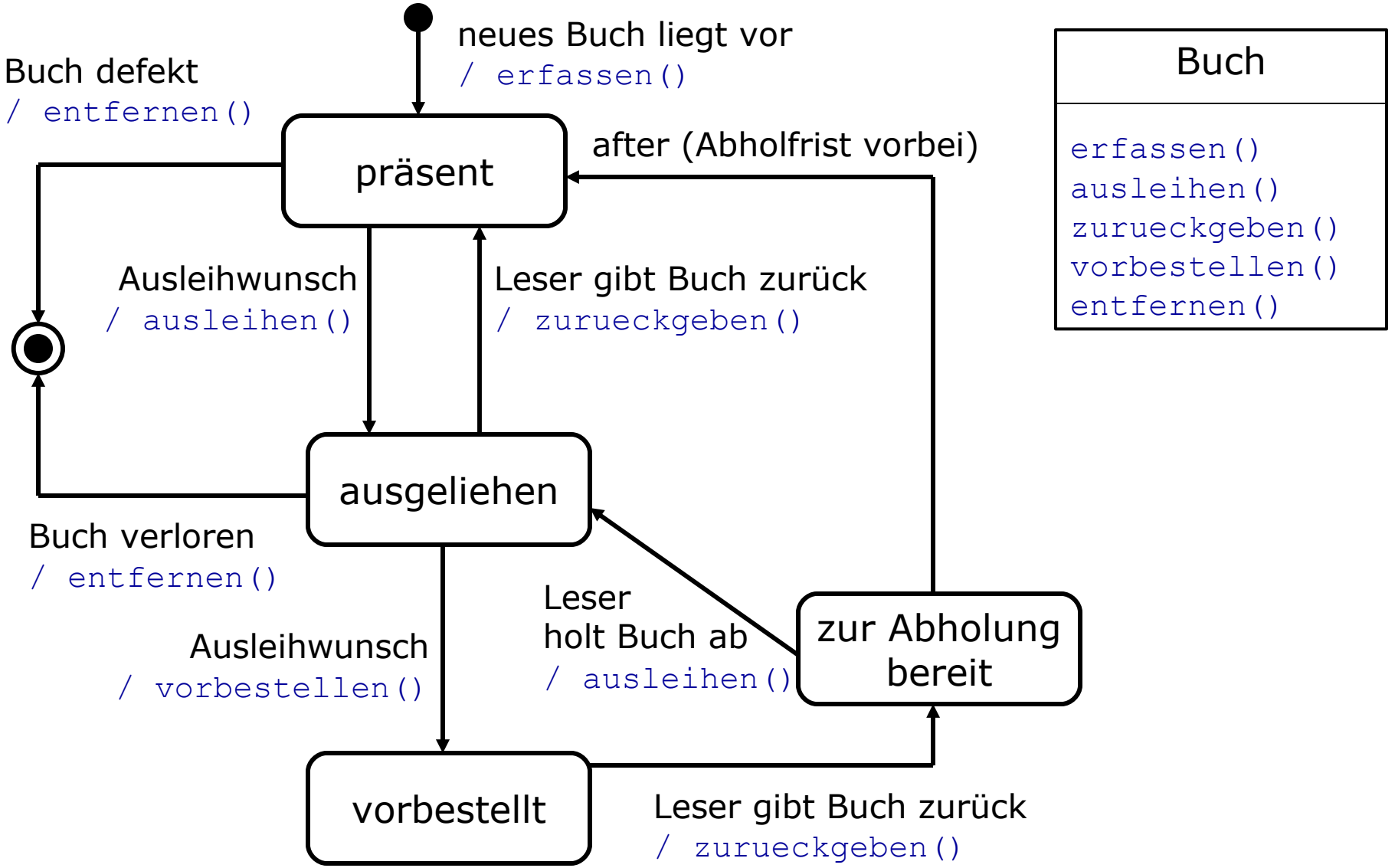
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Zustandsautomaten

- Erstellung eines Zustandsautomaten → *siehe Prof. Kebschull, GDTI I&II*
- Mealy- vs. Moore-Automaten → *siehe Prof. Kebschull, GDTI I&II*
- Zustandsautomat nach Harel
  - Erweiterung der bisherigen Konzepte
  - Modellierung komplexer Zusammenhänge
  - Neue Konzepte des Harel-Automaten
    - Hybride Zustandsautomaten (Kombination von Mealy- und Moore-Automaten)
    - Bedingte Zustandsübergänge
    - Hierarchische Zustandsautomaten
    - Zustände mit Gedächtnis
    - Nebenläufige Zustände.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Beispiel eines Zustandsautomaten



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Definitionen

Viele Systeme und Geräte zeigen ein Verhalten, das von der bis dahin durchlaufenen Historie abhängt. Das aktuelle Verhalten wird durch den internen **Zustand** bestimmt, der durch vorausgegangene Eingaben oder **Ereignisse** erreicht worden ist. Zur Modellierung solcher Systeme eignen sich **Zustandsautomaten (finite state machine)**, auch **endliche Automaten (finite automaton, sequential machine)** genannt.

Gegenüber einem allgemeinen Automaten besitzen endlich Automaten nur eine endliche Zahl von Zuständen. Bei deterministischen endlichen Automaten gibt es zu einer Eingabe von einem Zustand aus höchstens einen **Zustandsübergang (transition)** in einen anderen Zustand, während bei nichtdeterministischen endlichen Automaten mehrere Übergänge für dieselbe Eingabe möglich sind.

Es gibt deterministische endliche Automaten mit und ohne Ausgabe bzw. **Aktionen**. In der Software-Technik benötigt man im allgemeinen Automaten mit Aktionen. Beim **Mealy-Automaten** ist die Aktion an den Zustandsübergang, beim **Moore-Automaten** an den Zustand gekoppelt. Man spricht dann von **Aktivitäten**.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Definitionen

Der **Harel-Automat (statechart)** erlaubt die Kombination von Mealy- und Moore-Automat (hybride Automaten). Außerdem können Zustände miteinander geschachtelt werden, was zu **hierarchischen Zustands-automaten** führt. Diese unterstützen eine top-down sowie bottom-up Entwicklung von Automaten. Außerdem kann Nebenläufigkeit beschrieben werden.

In der OO-Welt kann der Lebenszyklus von Objekten durch Zustands-automaten beschrieben werden.

Zustandsautomaten können als **Zustandsdiagramm**, **Zustands-tabelle** oder **Zustandsmatrix** dargestellt werden.

Ziel beim Aufstellen eines Zustandsautomaten ist es, mit möglichst wenig Zuständen auszukommen.

Zustandsautomaten können für viele Anwendungsbereiche eingesetzt werden.

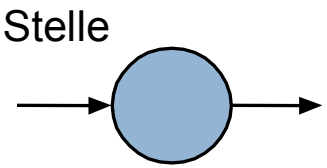
Das **Aktivitätsdiagramm** ist ein Sonderfall des Zustandsdiagramms. Es wird in der UML zur Beschreibung von Geschäftsprozessen und von Operationen verwendet.

Petri-Netze

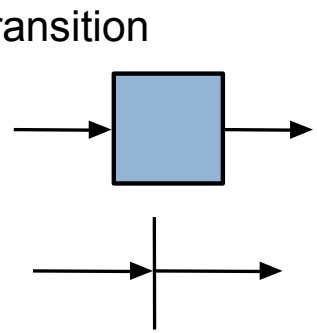
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Gerichteter bipartiter Graph, der aus zwei verschiedenen Sorten von Knoten besteht: Stellen und Transitionen

Stellen (Plätze, Zustände) entspricht einer Zwischenablage von Informationen



Transitionen (Hürden, Zustands-übergänge) beschreiben die Verarbeitung von Informationen

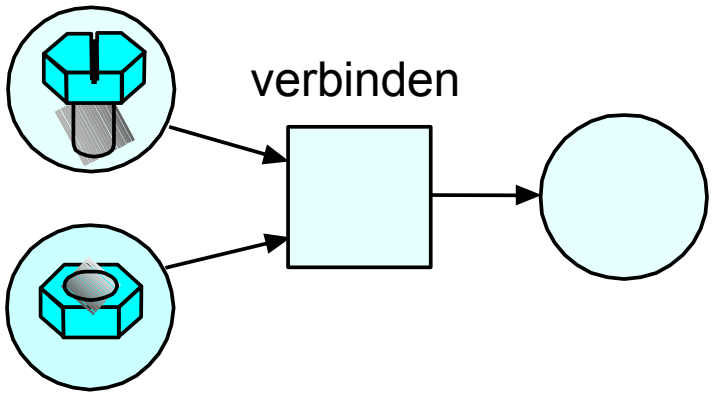


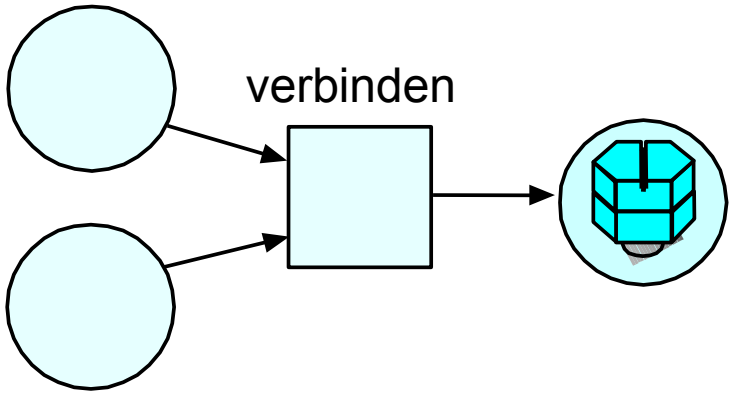
- Objekte werden als Marken bezeichnet
- **Schaltregel**:
  a. Eine Transition t kann »feuern«, wenn jede Eingabestelle von t mindestens 1 Marke enthält
  b. Schaltet eine Transition, dann wird aus jeder Eingabestelle eine Marke entfernt und zu jeder Ausgabestelle eine Marke hinzugefügt

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Schaltregel (Transitionen)

- Anschauliche Vorstellung der Schaltregel





- Abhängig von der Art der Objekte:
  - Bedingungs/Ereignis-Netze (B/E-Netz)
  - Stellen/Transitions-Netze (S/T-Netz)
  - Höhere Petri-Netze.

Zusammenfassung I

Petri-Netze ermöglichen die Modellierung, Analyse und Simulation nebenläufiger Systeme. In Abhängigkeit von dem zu modellierenden System können verschieden mächtige Klassen von Petri-netzen eingesetzt werden. Es werden einfache Petri-Netze **(Bedingungs/Ereignis-Netze, Stellen/Transitions-Netze)** und höhere Petri-Netze **(Prädikat/Transistions-Netze, zeitbehaftete Petri-Netze)** unterschieden.

Stellen und **Transitionen** werden je nach Netztyp unterschiedlich bezeichnet und interpretiert. Die Stellen unterscheiden sich im wesentlichen dadurch, ob sie eine oder mehrere Marken enthalten können und ob der Zeitintervall angegeben ist, das festlegt, wie lange die **Marken** auf der Stellen verharren müssen.

Für **B/E** und **S/T-Netze** gibt es jeweils eine allgemeine Schaltregel für Transitionen. Bei Pr/T-Netze kann jede Transition mit einer Schaltbedingung und einer Schaltwirkung versehen werden. In zeitbehafteten Petri-Netzen können Zeitintervalle an Transitionen notiert werden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Zusammenfassung II

Die Pfeile, die Stellen und Transitionen miteinander verbinden, können ungewichtet, gewichtet und mit Konstanten oder Variablen beschriftet sein. Marken können uniform oder individuell sein.

Jeden Netztyp kann man als **hierarchisches Petri-Netz** strukturieren. Die oberen Netzebenen heißen dann auch **Kanal-Instanzen-Netze**.

Ein Petri-Netz kann auf verschieden Eigenschaften hin analysiert werden. Wichtig ist die **Lebendigkeit** und **Verklemmungsfreiheit** eines Netzes.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Literatur

- Balzert H., Lehrbuch der Softwaretechnik, 2. Auflage, Spektrum Akademischer Verlag,Heidelberg, 2000.

- Harel D., Statechards: A Visual Formalism for complex Systems, in: Science of Computer Programming, Elsevier Science Publishers, 1987, S. 231-274
- Hopcroft J. E., Ullman J. D., Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie, Addison Wesley, Berlin, 1988
- Reisig W., Systementwurf mit Netzen, Springer Verlag, Berlin-Heidelberg, 1985
- Reisig W., Petri-Netze – eine Einführung, Springer Verlag, Berlin-Heidelberg, 1982