Modellierung

Prof. Dr. Uwe Kastens

WS 2000/2001

Begründung der Vorlesung

- Das **Modellieren** ist eine für das Fach **Informatik typische Arbeitsmethode**.
- Mit der Modellierung einer **Aufgabe** zeigt man, ob und wie sie **verstanden** wurde.
- Ein zutreffendes Modell ist **Voraussetzung** und Maßstab **für eine systematische Lösung.**
- Als **Ausdrucksmittel** muss man **passende Kalküle und Notationen** anwenden können.

Vorlesung Modellierung WS 2001/2002 / Folie 101

Ziele:

Hinweis auf die Bedeutung der Modellierung

in der Vorlesung:

Kurze Erläuterung der Aussagen.

Ziele

Die Teilnehmer sollen

- einen Überblick über **grundlegende Modellierungsmethoden und -kalküle** bekommen,
- den **konzeptionellen Kern der Kalküle** beherrschen,
- erste Erfahrungen an **typischen Beispielen** sammeln und
- die für die Methoden **typischen Techniken** erlernen.

Insgesamt sollen sie lernen,

- sich bei der Analyse von Problemen **präzise auszudrücken**,
- die **Scheu vor formalen Kalkülen verlieren** und
- den **praktischen Wert von präzisen Beschreibungen** erkennen.

Vorlesung Modellierung WS 2001/2002 / Folie 102

Ziele:
Ziele der Vorlesung verstehen

in der Vorlesung:
Begründung der Ziele

Durchführung

Zu jedem **Modellierungskalkül** soll(en)

- mit einigen typischen kleinen **Beispielen motivierend eingeführt** werden,
- der **konzeptionelle Kern** des Kalküls vorgestellt werden,
- **Anwendungstechniken und Einsatzgebiete** an Beispielen gezeigt und in den Übungen erfahren werden,
- auf **weiterführende Aspekte** des Kalküls, seine Rolle in Informatikgebieten und -vorlesungen sowie auf algorithmische Lösungsverfahren **nur verwiesen** werden.

Vorlesung Modellierung WS 2001/2002 / Folie 103

Ziele:

Ausrichtung der Vorlesung

in der Vorlesung:

- Hier: Einführung und Anwendung der Kalküle,
- in anderen Vorlesungen werden sie vertieft.

Inhalt

Thema	Semesterwoche	Abschnitte im Buch
1. Einführung	1	
2. Grundlegende Strukturen		
Wertebereiche	2	Anhang A
Terme, Algebren	3, 4	3.1 - 3.8
3. Logik		
Aussagenlogik	5	4.1
Prädikatenlogik	6	4.2
4. Graphen Verbindung, Zuordnung, Anordnung, Fluss	7, 8, 9	2.2
5. Modellierung von Strukturen Typisierung, Klassifikation, KFG, Entity-Relationship	9,10,11	
6. Modellierung von Abläufen endl. Automaten, Petri-Netze	12, 13, 14	2.4, 2.5
7. Zusammenfassung	15	

Vorlesung Modellierung WS 2001/2002 / Folie 104

Ziele:

Überblick über den Inhalt bekommen

in der Vorlesung:

Die Struktur wird erläutert.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt Inhaltsverzeichnis

Verständnisfragen:

- Welche der Begriffe sind Ihnen schon begegnet?
- Was stellen sie sich darunter vor?

Literaturhinweise

elektronisches Skript (entsteht zusammen mit der Vorlesung):

- **U. Kastens: Vorlesung Modellierung WS 2000 / 2001**
  http://www.uni-paderborn.de/cs/ag-kastens/model

zum Nachlernen und Nachschlagen:

- **G. Goos: Vorlesungen über Informatik, Band 1, 3. Auflage, Springer-Lehrbuch, 2000**
- **T. Scheurer: Foundations of Computing, System Development with Set Theory and Logic, Addison-Wesley, 1994**

elektronisches Skript der vorigen Jahrgänge:

- **H. Kleine Büning: Vorlesung Modellierung WS 1999 / 2000**
  http://www.uni-paderborn.de/cs/ag-klbue/courses/ws99/modellierung

Weitere Hinweise im Material zur Vorlesung im WWW

Ziele:

Literatur zur Vorlesung kennenlernen

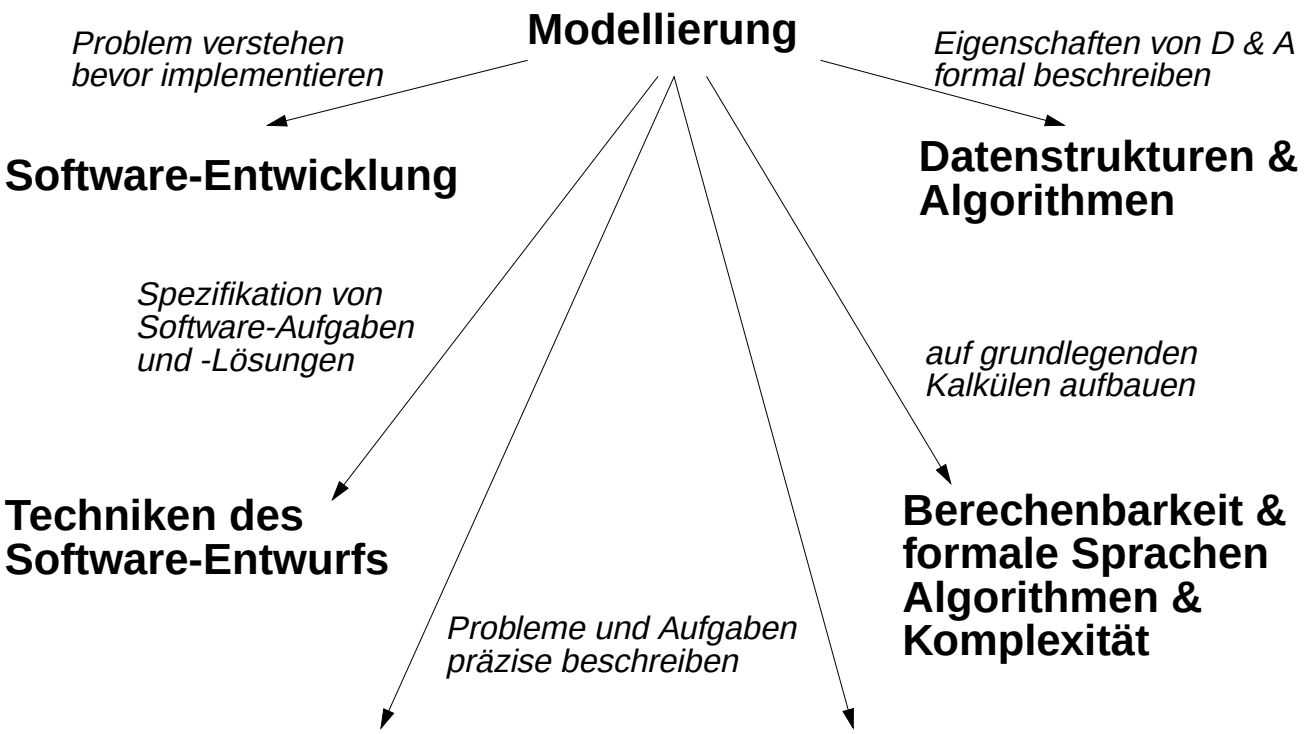
in der Vorlesung:

Erläuterungen dazu.

Verständnisfragen:

- Suchen Sie das Buch von G. Goos im Semesterapparat.
- Verfolgen Sie die URLs.

Bezüge zu anderen Vorlesungen



Vorlesung Modellierung WS 2001/2002 / Folie 106

Ziele:

Einordnung der Vorlesung

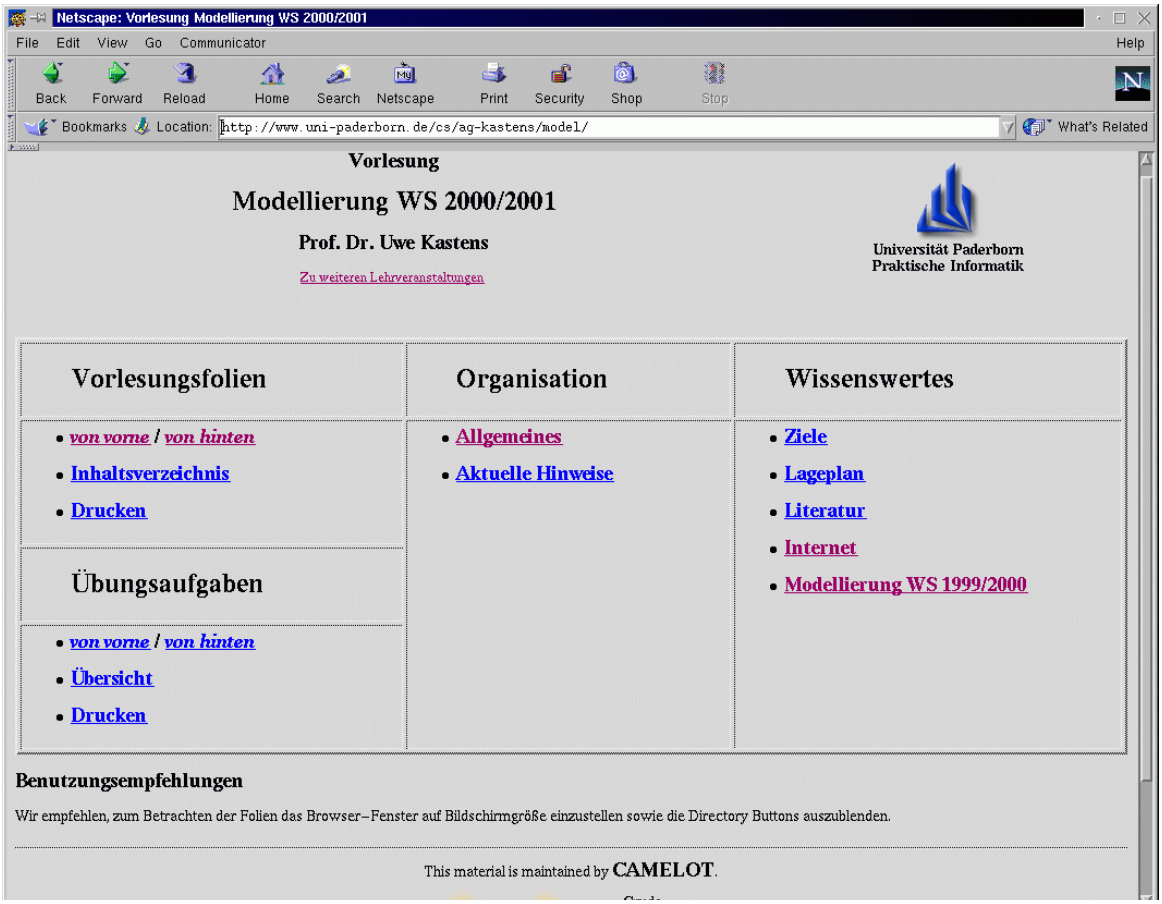
in der Vorlesung:

- Vorlesungen im Studienplan zeigen.
- Bezüge erläutern.

Verständnisfragen:

- Finden Sie den Studienplan im WWW.
- Können Sie die Bezüge an den Inhaltsbeschreibungen der Vorlesungen nachvollziehen?

Elektronisches Skript: Startseite



Vorlesung Modellierung WS 2001/2002 / Folie 107

Ziele:

Das elektronische Skript kennenlernen.

in der Vorlesung:

Erläuterungen dazu

Verständnisfragen:

Suchen Sie das Skript im WWW.

Elektronisches Skript: Organisation

Sprechstunde Prof. Dr. Uwe Kastens:

- Die 11.00 - 12.00 F2.308
- Mi 15.00 - 16.00 F2.308

Übungsbetreuer:

- Kim von Grawert (kim@upb.de)
- Jochen Kreimer (jotte@upb.de)
- Mark Langer (grey@upb.de)
- Dinh Khoi Le (le@upb.de)
- Dr. Theodor Lettmann (lettmann@upb.de)
- Thomas Lücking (luck@upb.de)
- Carsten Schmidt (cschmidt@upb.de)
- Michael Suermann (michel@upb.de)

Termine

Vorlesung:

- Mo 11 00 - 12.30 AudiMax
- Fr 09 00 - 10.30 AudiMax

Beginn: Freitag, 20.10.2000

Zentralübung:

- Mi 13 00 - 13.45 AudiMax

Vorlesung Modellierung WS 2001/2002 / Folie 108

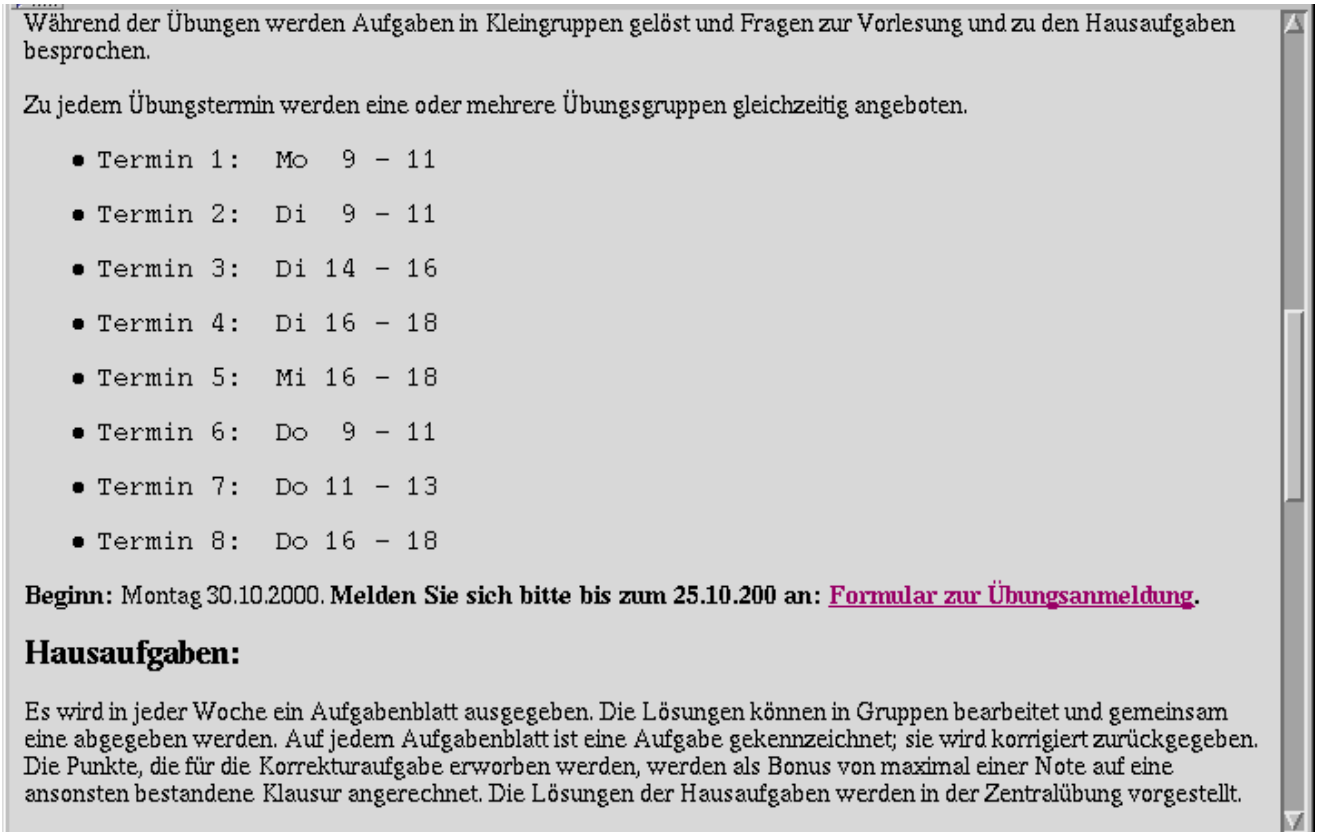
Ziele:

Organisation der Vorlesung kennenlernen

in der Vorlesung:

Organisatorisches erläutern

Elektronisches Skript: Übungstermine



Während der Übungen werden Aufgaben in Kleingruppen gelöst und Fragen zur Vorlesung und zu den Hausaufgaben besprochen.

Zu jedem Übungstermin werden eine oder mehrere Übungsgruppen gleichzeitig angeboten.

- Termin 1: Mo 9 - 11
- Termin 2: Di 9 - 11
- Termin 3: Di 14 - 16
- Termin 4: Di 16 - 18
- Termin 5: Mi 16 - 18
- Termin 6: Do 9 - 11
- Termin 7: Do 11 - 13
- Termin 8: Do 16 - 18

Beginn: Montag 30.10.2000. **Melden Sie sich bitte bis zum 25.10.200 an:** Formular zur Übungsanmeldung.

Hausaufgaben:

Es wird in jeder Woche ein Aufgabenblatt ausgegeben. Die Lösungen können in Gruppen bearbeitet und gemeinsam eine abgegeben werden. Auf jedem Aufgabenblatt ist eine Aufgabe gekennzeichnet; sie wird korrigiert zurückgegeben. Die Punkte, die für die Korrekturaufgabe erworben werden, werden als Bonus von maximal einer Note auf eine ansonsten bestandene Klausur angerechnet. Die Lösungen der Hausaufgaben werden in der Zentralübung vorgestellt.

Vorlesung Modellierung WS 2001/2002 / Folie 109

Ziele:

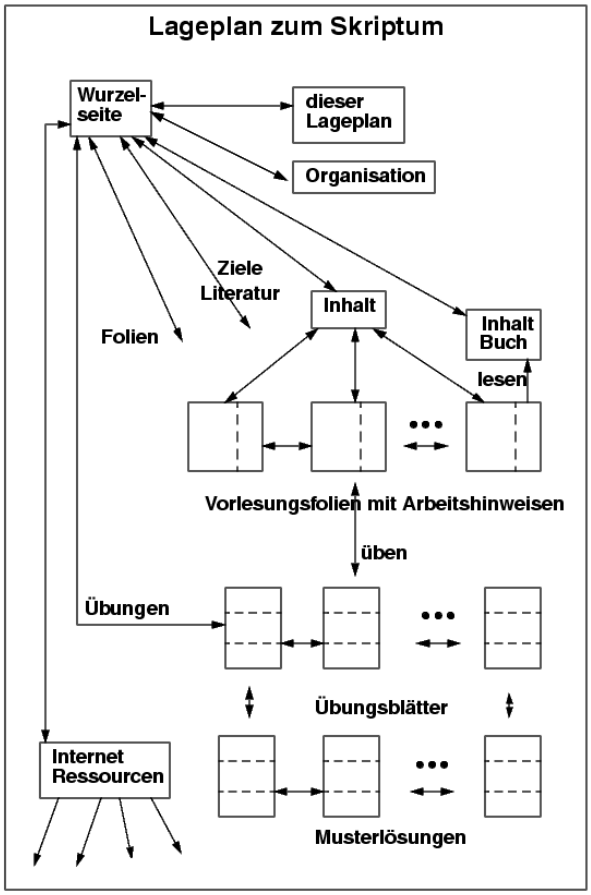
Organisation der Vorlesung kennenlernen

in der Vorlesung:

Organisatorisches erläutern:

- Anmeldung zu Übungen,
- Bearbeitung der Hausaufgaben,
- Klausuren

Elektronisches Skript: Lageplan



Vorlesung Modellierung WS 2001/2002 / Folie 110

Ziele:

Struktur des elektronischen Skriptes überblicken

in der Vorlesung:

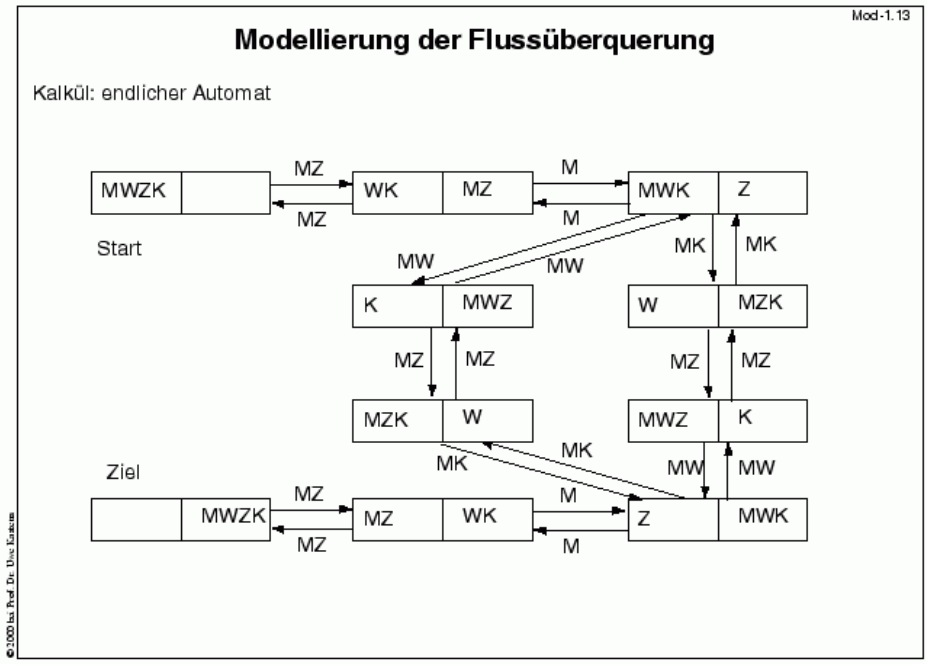
Erläuterungen dazu

Verständnisfragen:

Navigieren Sie im Skript.

Elektronisches Skript: Kommentierte Folien

Vorlesung Modellierung WS 2000/2001 – Folie Nr. 113



Ziele:
Prozess der Modellierung am Beispiel erkennen

in der Vorlesung:
Erläuterungen dazu (siehe auch nächste Folie):

- Bedeutung der Graphik und der Symbole
- Zustände und Übergänge eines endlichen Automaten (siehe Kap. 8),
- Darstellung als Graph mit Knoten und Kanten (siehe Kap. 6)
- Wertebereiche der Information zu Zuständen (siehe Kap. 2)

Verständnisfragen:

- Prüfen Sie, ob das Modell die Aufgabe korrekt und vollständig beschreibt.

Vorlesung Modellierung WS 2001/2002 / Folie 111

Ziele:

Arbeitshinweise zu den Folien kennenlernen

in der Vorlesung:

Erläuterungen dazu

Beispiel: Die Flussüberquerung

Aufgabe:

> Ein Mann steht mit einem Wolf, einer Ziege und einem Kohlkopf am linken Ufer eines Flusses, den er überqueren will. Er hat ein Boot, das groß genug ist, ihn und ein weiteres Objekt zu transportieren, so dass er immer nur eins der drei mit sich hinübernehmen kann.
>
> Falls der Mann allerdings den Wolf und die Ziege oder die Ziege und den Kohlkopf unbewacht an einem Ufer zurücklässt, so wird einer gefressen werden.
>
> Ist es möglich, den Fluss zu überqueren, ohne dass die Ziege oder der Kohlkopf gefressen werden?

Quelle: Hopcroft, Ullman: Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie, S. 14, 15

Vorlesung Modellierung WS 2001/2002 / Folie 112

Ziele:

Modellierung am Beispiel kennenlernen

in der Vorlesung:

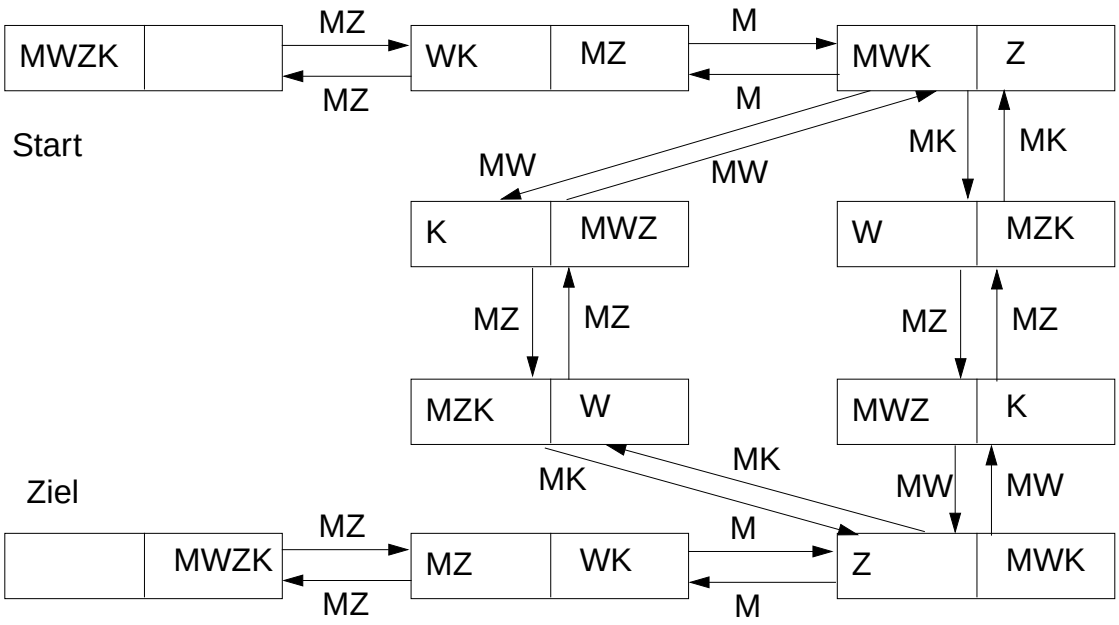
Erläuterungen zu der Aufgabe, Skizze.

Verständnisfragen:

- Welche Aspekte der Aufgabe sind für die Lösung wichtig?
- Welche sind unwichtig?
- Wie können wir die wichtigen Aspekte präzise beschreiben?

Modellierung der Flussüberquerung

Kalkül: endlicher Automat



Vorlesung Modellierung WS 2001/2002 / Folie 113

Ziele:

Prozess der Modellierung am Beispiel erkennen

in der Vorlesung:

Erläuterungen dazu (siehe auch nächste Folie):

- Bedeutung der Graphik und der Symbole,
- Zustände und Übergänge eines endlichen Automaten (siehe Kap. 8),
- Darstellung als Graph mit Knoten und Kanten (siehe Kap. 6)
- Wertebereiche der Information zu Zuständen (siehe Kap. 2)

Verständnisfragen:

- Prüfen Sie, ob das Modell die Aufgabe korrekt und vollständig beschreibt.

Diskussion des Modellierungsbeispiels

- Modellierung von **Abläufen**, Folgen von Schritten: Kalkül endlicher Automat
- **Abstraktion**: nur die Zustände und Übergänge interessieren
- **relevante Objekte benannt**: M, W, Z, K
- jeder **Zustand** wird charakterisiert durch ein **Paar von Mengen** der Objekte, (linkes Ufer, rechtes Ufer); jedes Objekt kommt genau einmal vor
- zulässige und **unzulässige Zustände** (unzulässige sind nicht angegeben)
- **Übergänge** werden mit den transportierten Objekten beschriftet

Hier liefert die Modellierung schon die Lösung: Es gibt Wege vom Start zum Ziel.
Das ist i. a. nicht so.

Besonders wichtig ist, was **nicht modelliert** wurde, da es **für die Aufgabe irrelevant** ist!
z. B. die Länge des Bootes, die Breite und Tiefe des Flusses, usw.

Vorlesung Modellierung WS 2001/2002 / Folie 114

Ziele:

Prozess der Modellierung am Beispiel erkennen

in der Vorlesung:

Erläuterungen dazu (siehe auch vorige Folie):

- In jedem Kalkül: Namen und Wertebereich für die relevanten Dinge, Irrelevantes weglassen.
- Es gibt auch ernsthafte Aufgaben nach diesem Muster: Finden Zulässiger Folgen von Zustandsübergängen!

Verständnisfragen:

- Wie würden sie eine ähnliche Aufgabe modellieren, in der die beiden Tiere nicht hungrig sind aber das Boot nur begrenzte Tragfähigkeit hat?

Allgemeiner Modellbegriff

- **Abbild** eines vorhandenen Originals (z. B. Schiffsmodell)
- **Vorbild** für ein herzustellendes Original (Gebäude in kleinem Maßstab; Vorbild in der Kunst)
- **konkretes** oder **abstraktes Modell** (Schiffsmodell, Rentenmodell)
- konkretes oder abstraktes **Original** (Schiff, Bevölkerungsentwicklung)

davon abweichende Bedeutungen:

- Fotomodell: führt Mode (oder sich) vor
- Automodell: Typreihe
- in der Logik: Eine Struktur S ist ein Modell der Formeln F, wenn alle F für S gelten.

hier in der Informatik:

- **abstraktes Abbild oder Vorbild zu abstrakten oder konkreten Originalen**

Vorlesung Modellierung WS 2001/2002 / Folie 115

Ziele:

Begriff fixieren

in der Vorlesung:

Erläuterungen und weitere Beispiele dazu

Verständnisfragen:

- Geben Sie weitere Beispiele zu den Aspekten.
- Schlagen Sie den Begriff Modell in allgemeinen Lexika und in Lexika der Informatik nach.

Zweck des Modells

Der **Verwendungszweck** bestimmt die Art des Modells! z. B.

- Gebäudemodell: optischer Eindruck
- Grundriss: Einteilung des Grundstückes und der Räume
- Kostenplan: Finanzierung
- Gewerkeplan: Bauabwicklung

Nur was **für den Zweck relevant** ist, wird modelliert!

Vollständige Modellierung des Originals ist nicht sinnvoll.

Für den Zweck die jeweils passende Modellierungsmethode (Kalkül) verwenden!

Vorlesung Modellierung WS 2001/2002 / Folie 116

Ziele:

Bedeutung des Verwendungszweckes erkennen

in der Vorlesung:

Erläuterungen zu den Zwecken und Modellierungsmethoden.

Verständnisfragen:

- Was sind die Modellierungsmethoden in den angegebenen Beispielen?
- Geben Sie weitere Kalküle an und Zwecke für die sie sich eignen.

Arbeiten mit dem Modell

- **Operationen, die man am Original nicht durchführen kann**
  z. B. neue Flügelform im Windkanal oder in der Computer-Simulation erproben
- Bestimmte Aspekte eines **komplexen Gebildes untersuchen und verstehen**,
  z. B. Geschäftsabläufe in einer Firma
- **Verständigung zwischen Auftraggeber und Hersteller** des Originals,
  z. B. Hausbau, Software-Konstruktion
- Fixieren von **Anforderungen für die Herstellung** des Originals,
  Software: Requirements, Spezifikation

Modell validieren:

Nachweisen, dass die
relevanten Eigenschaften des Originals korrekt und vollständig im Modell erfasst sind und darüber Einvernehmen herstellen.

Vorlesung Modellierung WS 2001/2002 / Folie 117

Ziele:

Modellieren heißt verstehen!

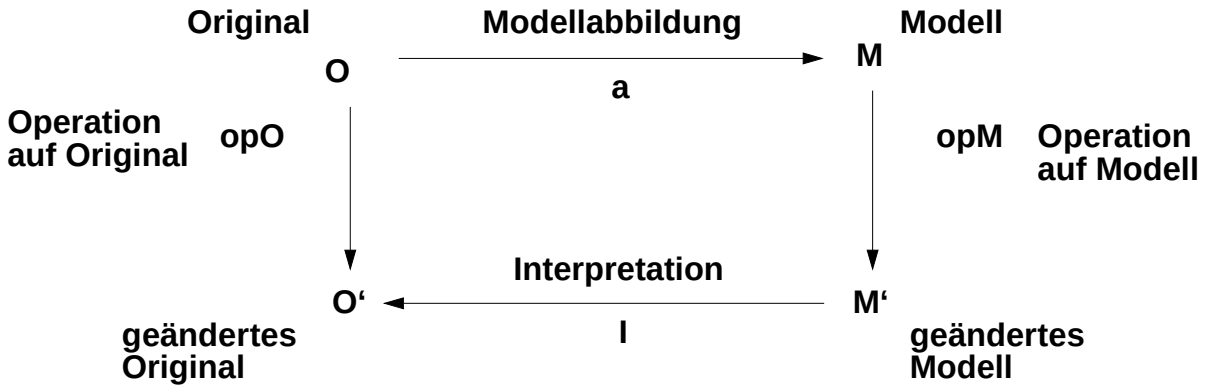
in der Vorlesung:

- Modellieren zum eigenen Verständnis,
- Modell zur Abstimmung mit anderen
- Modell untersuchen, wenn Original nicht verfügbar.
- Beispiele zur Validierung

Verständnisfragen:

- Geben Sie weitere Beispiele zur Validierung von Modellen.

Bezug zwischen Original und Modell



Für alle relevanten Operationen muss das Diagramm kommutieren, d. h

opO (O) = I (opM (a (O)))

Die Operation auf dem Original entspricht der Interpretation der Operation auf dem Modell.

Vorlesung Modellierung WS 2001/2002 / Folie 118

Ziele:

Formale Anforderung an das Modell

in der Vorlesung:

Erläuterungen dazu

Modellierte Aspekte

Ein Modell beschreibt nur bestimmte Aspekte des Originals und seiner Teile:

- **Struktur**, Zusammensetzung des Originals (z. B. Organisationsschema einer Firma)
- **Eigenschaften** von Teilen des Originals (z. B. Farbe und Wert einer Spielkarte)
- **Beziehungen** zwischen Teilen des Originals
  (z. B. Abhängigkeiten der Gewerke beim Hausbau)
- **Verhalten** des Originals unter Operationen (z. B. Zugfolge bei der Flussüberquerung)

Zur Modellierung bestimmter Aspekte eignen sich bestimmte Methoden und Kalküle:

- **Struktur**: Wertebereiche, Entity-Relationship, KFG, Klassifikation, Typen
- **Eigenschaften**: Logik, Relationen
- **Beziehungen**: Graphen, Relationen, Logik, Entity-Relationship
- **Verhalten**: endliche Automaten, Petri-Netze, Algebren, Graphen

Vorlesung Modellierung WS 2001/2002 / Folie 119

Ziele:

Einteilung der modellierten Aspekte

in der Vorlesung:

- Verschiedene Sichten auf das Original.
- Der Zweck bestimmt die passende Sicht.
- Zuordnung zu den Kalkülen der Vorlesung.

Verständnisfragen:

- Diskutieren Sie die Sichten am Beispiel **eines** Originals.

Deklarative oder Operationale Beschreibung

Deklarative Beschreibung des Modells
macht Aussagen über Aspekte des Originals.

Operationale Beschreibung des Modells
gibt an, wie sich das Original unter bestimmten Operationen verhält.

Beispiel Balkenwaage:

deklarativ:
Die Waage ist im Gleichgewicht, wenn sich die Gewichte umgekehrt proportional zu den Längen der Balken verhalten: x * a = y * b.

operational:
Wenn ich auf den Balken der Länge a ein Gewicht x auflege, muss ich auf den Balken der Länge b ein Gewicht y = x * a / b auflegen, damit die Waage im Gleichgewicht ist.

deklarativ:
Aussagen meist allgemein gültig,
auf die Aufgabe bezogen,
ohne redundante Abläufe

operational:
häufig nur Beispiele, unvollständig,
legt eine Lösung nahe (fest),
erzwingt Nachvollziehen von Abläufen

Vorlesung Modellierung WS 2001/2002 / Folie 120

Ziele:

Möglichst deklarativ beschreiben.

in der Vorlesung:

Diskussion von Beispielen.

2. Grundlegende Strukturen
2.1 Wertebereiche beschrieben durch Mengen

In der Modellierung von Systemen, Aufgaben, Lösungen kommen
Objekte unterschiedlicher Art und Zusammensetzung vor.

Die Beschreibung des Modells wird präzisiert durch **Angabe der Wertebereiche**,
aus denen die Objekte, Konstanten, Variable stammen.

Wertebereich nennen wir eine **Menge gleichartiger Objekte eines Modells**.

Zur Angabe der Wertebereiche verwenden wir Mengen und Strukturen darüber.

Beispiel:
MünzWährung als Menge von Werten, für die es Geldstücke gibt:

MünzWährung $\subseteq \mathbb{N}_0$
brdMünzen := {1, 2, 5, 10, 50, 100, 200, 500}

Ein Sack voll Hartgeld wird beschrieben durch eine Funktion Geldsack mit dem
Wertebereich MünzWährung -> $\mathbb{N}_0$

Geldsack (10) = 4 bedeutet, dass vier 10-Pfennigstücke in diesem Geldsack sind.

Vorlesung Modellierung WS 2001/2002 / Folie 201

Ziele:

Beschreibung von Wertebereichen Motivieren

in der Vorlesung:

Erläuterungen dazu

- präzise Angabe von Wertebereichen,
- Informationsgehalt untersuchen

Hinweis: In der Menge der natürlichen Zahlen schließen wir die 0 ein (wie von Peano axiomatisch definiert). Um Verwirrungen mit Mathe I zu vermeiden schreiben wir meist dafür N0 (abweichend vom Buch, wo N nicht mit 0 indiziert wird).

Verständnisfragen:

Beschreiben Sie in Worten den Informationsgehalt eines Sackes voller Münzen.

Vorschau auf Begriffe

Grundlegender Kalkül: **Mengenlehre** (halbformal)

wichtigste Begriffe für die Modellierung:

- Mengen und Mengenoperationen
- Potenzmengen
- Kartesisches Produkt, Tupel
- Folgen
- Relation
- Funktion
- disjunkte Vereinigung

Verwendung des Kalküls:

Modellierung von Strukturen und Zusammenhängen

Grundlage für alle anderen formalen Kalküle

abstrakte Grundlage für Typen in Programmiersprachen

Vorlesung Modellierung WS 2001/2002 / Folie 202

Ziele:

Übersicht zu diesem Abschnitt

in der Vorlesung:

Hinweise auf Bezüge zu

- anderen Kalkülen,
- Datentypen in Programmiersprachen

Einführendes Beispiel 2.1

Internationale Arbeitsgruppen

Bei der UNO sollen Arbeitsgruppen aus Delegierten der drei Nationen A, B und C gebildet werden. Jede Nation hat vier Delegierte. Jede Gruppe besteht aus drei Personen, eine aus jeder Nation. Die Sprachen der drei Nationen sind verschieden; wir nennen sie auch A, B, C. Die Mitglieder jeder Arbeitsgruppe sollen eine gemeinsame Sprache sprechen.

aus [T. Scheurer S. 155]

Vorlesung Modellierung WS 2001/2002 / Folie 203

Ziele:

Definition von Wertebereichen motivieren

in der Vorlesung:

- Erläuterung der Aufgabe
- Präzise Modellierung interessiert hier - nicht Verfahren, um Lösungen zu finden.
- Modellierung auf der nächsten Folie

Beispiel 2.1 Modellierung

Menge der Nationen	Nationen := {A, B, C}
Indexmenge zur Unterscheidung der Delegierten	Ind := {1, 2, 3, 4}
ein Delegierter modelliert durch ein **Paar**	(a, i) mit a ∈ Nationen, i ∈ Ind
Wertebereich der Delegierten	Delegierte := Nationen × Ind
Wertebereich für **Teilmengen** von Sprachen	SprachMengen := Pow (Nationen)
	Pow (M) ist die **Potenzmenge** von M
Eine **Funktion** Sp gibt an, welche Sprachen ein Delegierter spricht:	Sp ∈ DSpricht
Wertebereich solcher Funktionen	DSpricht := Delegierte -> SprachMengen
Wertebereich der Arbeitsgruppen **3-Tupel, kartesischesProdukt**	AGn := {(A, i) | i ∈ Ind} × {(B, j) | j ∈ Ind} × {(C, k) | k ∈ Ind}
Gemeinsame Sprachen einer AG Wertebereich	AGSpricht := AGn -> SprachMengen
Funktion daraus	GemSp ∈ AGSpricht

Das Zeichen := wird hier mit der Bedeutung verwendet „ist definiert als“.

Vorlesung Modellierung WS 2001/2002 / Folie 204

Ziele:

Beispiele im Zusammenhang sehen

in der Vorlesung:

- Vorschau auf Anwendung des Kalüls,
- informelle Erläuterungen,
- Werte aus den Wertebereichen angeben

Verständnisfragen:

- Geben Sie zu jedem der Wertebereiche einen konkreten Wert als Beispiel an.

Mengen

Menge: Zusammenfassung M von verschiedenen Objekten, den Elementen der Menge.
a ist Element aus M wird notiert $a \in M$.

Die Existenz von atomaren Objekten des jeweiligen Modellierungsbereiches wird vorausgesetzt, z. B. die natürlichen Zahlen.

Definition von Mengen:

- durch **Aufzählen der Elemente (extensional)**: M := {1, 4, 9, 16, 25}
- durch **Angabe einer Bedingung (intensional)**:
  $M := \{a \mid a \in \mathbb{N} \text{ und } a \text{ ist Quadratzahl und } a \leq 30\}$

Um Anomalien auszuschließen soll in der Bedingung angegeben werden, aus welchem größeren, schon definierten Wertebereich die Elemente stammen; hier „$a \in \mathbb{N}$“.

Wichtige grundsätzliche Eigenschaften von Mengen:

- **Es gibt verschiedene Definitionen derselben Menge.**
- **Alle Elemente einer Menge sind verschieden.**
- **Die Elemente einer Menge sind nicht geordnet.**

Vorlesung Modellierung WS 2001/2002 / Folie 205

Ziele:

Definition von Mengen verstehen

in der Vorlesung:

- Beispiele zu den Eigenschaften.
- Hier informelle Definition des Mengenbegriffs; axiomatische Mengenlehre definiert ihn strenger.
- Russells Paradoxon: Sei P die Menge der Mengen, die sich nicht selbst als Element enthalten. Enthält P sich selbst als Element?

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt A.1

Verständnisfragen:

- Geben Sie Beispiele jeweils für unterschiedliche intensionale und für unterschiedliche extensionale Definitionen derselben Menge.
- Vergleichen Sie den Mengenbegriff mit dem aus Mathe I.

Mengenoperationen

die leere Menge	$\varnothing$	
Teilmenge von	$M \subseteq N$	aus $a \in M$ folgt $a \in N$
echte Teilmenge von	$M \subset N$	$M \subseteq N$ und $M \neq N$
Vereinigung	$M \cup N$	$:= \{x \mid x \in M$ oder $x \in N\}$
Durchschnitt	$M \cap N$	$:= \{x \mid x \in M$ und $x \in N\}$
Differenz	$M \setminus N$	$:= \{x \mid x \in M$ und $x \notin N\}$
relatives Komplement	$U \setminus N$	mit $N \subseteq U$ Trägermenge

M und N sind **disjunkt** genau dann, wenn gilt $M \cap N = \varnothing$

$|M|$ oder Card(M) ist die **Kardinalität** von M, d. H. die Anzahl der Elemente in M

Vorlesung Modellierung WS 2001/2002 / Folie 206

Ziele:

Vorstellung der Mengenoperatoren

in der Vorlesung:

- Hinweis auf algebraische Gesetze; werden hier nicht vertieft

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt A.1

Verständnisfragen:

- Schlagen Sie die algebraischen Gesetze der Mengenoperatoren nach.

Potenzmengen

Potenzmenge (powerset) einer Grundmenge U ist die **Menge aller Teilmengen** von U, geschrieben Pow (U) oder $\wp$ (U).
Pow (U) := {M | M $\subseteq$ U}

Beispiel: Grundmenge U := {a, b} Potenzmenge Pow (U) = {$\emptyset$, {a}, {b}, {a, b}}

Kardinalität: $|\text{Pow (U)}| = 2^{|U|} = 2^n$ wenn $|U| = n$

Anwendungen:

Ein Wertebereich ist Potenzmenge von U, wenn die Werte Teilmengen von U sind.

Beispiel 2.1: Wertebereich der Sprachen, die ein Deligierter spricht
SprachMengen := Pow (Nationen), {A, B} $\in$ SprachMengen

Manche Aufgaben haben nicht immer genau eine Lösung, sondern je nach Daten mehrere oder keine Lösung. Dann kann man nach der Menge aller Lösungen fragen. Der Wertebereich der Antwort ist die Potenzmenge des Wertebereiches der Lösungen.

Vergleiche auch Mengentyp in Pascal:

```
type Sprachen = set of {A, B, C};
var spricht: Sprachen;
spricht := {A, B};
```


Vorlesung Modellierung WS 2001/2002 / Folie 207

Ziele:

Potenzmenge als Wertebereich verstehen

in der Vorlesung:

- Kardinalität begründen
- Beispiele,
- Wert - Wertebereich; Menge - Potenzmenge
- Aufgabe mit Lösungsmenge

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt A.1

Verständnisfragen:

- Begründen sie die Kardinalitätsformel.

Kartesisches Produkt

Kartesisches Produkt der Mengen M und N:

Menge aller **geordneten Paare** mit erster Komponente aus M und zweiter aus N

$$M \times N := \{Z \mid Z = (x, y) \text{ und } x \in M \text{ und } y \in N\}$$

oder kürzer $M \times N := \{(x, y) \mid x \in M \text{ und } y \in N\}$

Zusammengesetzte Wertebereiche, z.B. Delegierte := Nation $\times$ Ind

Verallgemeinert zu (geordneten) **n-Tupeln**:

$M_1 \times M_2 \times \ldots \times M_n := \{(a_1, a_2, \ldots, a_n) \mid a_i \in M_i \text{ und } i \in I\}$
mit **Indexmenge** I := {1, … , n}

Beispiel:
Daten := Tage $\times$ Monate $\times$ Jahre

Notation bei **gleichen Mengen M_i** : $M \times M \times \ldots \times M = M^n$

Beispiel:
Wertebereich der Ergebnisse 3-maligen Würfelns: DreiWürfe := $\{1, 2, 3, 4, 5, 6\}^3$

Kardinalität: $| M_1 \times M_2 \times \ldots \times M_n | = \prod_{i \in I} |M_i|$ mit I = {1,..., n}

Vorlesung Modellierung WS 2001/2002 / Folie 208

Ziele:

Zusammengesetzte Wertebereiche verstehen

in der Vorlesung:

Erläuterungen dazu

- Ordnung der Komponenten ist wichtig.
- Beispiel für geschachtelte Tupel

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt A.2

Verständnisfragen:

- Was ist der Unterschied zwischen Nation x Ind x SprachMengen und (Nation x Ind) x SprachMengen?
- Welches ist besser im Sinne von Beispiel 2.1?

Folgen

Endliche Folgen von Elementen aus A: n-Tupel über A von beliebiger Länge n

$\mathbf{A^+} := \{ x \mid x \in A^i \text{ und } i \geq 1 \}$ d. h. $(a_1, ..., a_n) \in A^+$ für $n \geq 1$ und $a_i \in A$

Sei ε **oder () die leere Folge** und $A^0 := \{ \varepsilon \}$, dann ist

$\mathbf{A^* := A^+ \cup A^0}$

der Wertebereich von Folgen über A, der auch die leere Folge enthält.

Beispiele:

$(1, 1, 2, 5, 5, 10, 20) \in \mathbb{N}^+$

('m', 'o', 'd', 'e', 'l', 'l') $\in$ Buchstaben$^+$

neueAufträge = Auftrag*

Vorlesung Modellierung WS 2001/2002 / Folie 208a

Ziele:

Folgen gleichartiger Elemente

in der Vorlesung:

Erläuterungen dazu

- + und * Notation erläutern
- weitere Beispiele
- Verwendung der leeren Folge

Verständnisfragen:

- Aus einer Folge natürlicher Zahlen sollen die geraden Zahlen gestrichen werden. Aus welchem Wertebereich stammt das Ergebnis?

Relationen

Relationen sind Teilmengen aus kartesischen Produkten.

n-stellige Relation: $R \subseteq M_1 \times M_2 \times \ldots \times M_n$

Eine solche Relation R stammt also aus dem **Wertebereich Pow ($M_1 \times M_2 \times \ldots \times M_n$)**

Seine **Kardinalität** ist $|\mathbf{Pow}\ (M_1 \times M_2 \times \ldots \times M_n)| = 2^{\prod_{i \in I} |Mi|}$

Eine Relation R definiert eine **Aussage über Tupel**. Wir sagen auch:
„Eine Relation R gilt für die Tupel in R."

Beispiele:

NationenKleiner := {(A, C), (C, B), (A, B)} $\subseteq$ Nationen^2
GültigeDaten $\subseteq$ Daten = Tage $\times$ Monate $\times$ Jahre
(20, Oktober, 2000) $\in$ GültigeDaten, (30, Februar, 2000) $\notin$ GültigeDaten

Schreibweisen:
R (a) für **a** $\in$ **R**, z. B. GültigeDaten (20, Oktober, 2000)

bei **2-stelligen (binären) Relationen** auch als **Operator**:
x **R y** für **(x, y)** $\in$ **R** , z.B. $x \leq y$, $a \neq b$, $p \Rightarrow q$

Vorlesung Modellierung WS 2001/2002 / Folie 209

Ziele:

Allgemeine Relationen verstehen

in der Vorlesung:

- Erläuterungen dazu
- Zusammenhang zwischen Relation und Aussage über Tupel.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt A.2

Verständnisfragen:

- Geben Sie Beispiele für Relationen zwischen Werten aus unterschiedlichen Wertebereichen.

Wichtige Eigenschaften 2-stelliger Relationen

Für zweistellige Relationen $R \subseteq M \times M$ sind folgende Begriffe definiert:

- **reflexiv**, wenn für alle $x \in M$ gilt x R x;
- **irreflexiv**, wenn für kein $x \in M$ gilt x R x;
- **symmetrisch**, wenn für alle $x, y \in M$ gilt: aus x R y folgt y R x;
- **antisymmetrisch**, wenn für alle $x, y \in M$ gilt: aus x R y und y R x folgt $x = y$;
- **asymmetrisch**, wenn für alle $x, y \in M$ gilt: aus x R y folgt, y R x gilt nicht;
- **transitiv**, wenn für alle $x, y, z \in M$ gilt, aus x R y und y R z folgt x R z;
- **alternativ**, wenn für alle $x, y \in M$ gilt x R y oder y R x.
- **Äquivalenzrelation**, wenn sie reflexiv, symmetrisch und transitiv ist;
- **partielle Ordnung** oder **Halbordnung**, wenn sie reflexiv, antisymmetrisch und transitiv ist;
- **strenge Ordnung** oder **strenge Halbordnung**, wenn sie irreflexiv und transitiv ist;
- **Quasiordnung**, wenn sie reflexiv und transitiv ist;
- **totale** oder **lineare Ordnung**, wenn sie eine alternative Halbordnung ist.

Vorlesung Modellierung WS 2001/2002 / Folie 210

Ziele:

Definitionen einprägen

in der Vorlesung:

Erläuterungen und Beispiele zu einigen der Eigenschaften.

- Hinweis: Bei der Eigenschaft "alternativ" bedeutet das "oder", dass mindestens eines der Paare in der Relation ist.
- Hinweis: Die Formulierung "aus A folgt B" kann man evtl. leichter überprüfen in der logisch gleichbedeutenden Form "nicht A oder B".

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt A.2

Verständnisfragen:

- Beschreiben Sie den Unterschied zwischen Halbordnung und totaler Ordnung. Geben Sie ein Beispiel dazu an.

Hinweise zu den Definitionen von Eigenschaften von Relationen

- Die Begriffe sind **nur** für Relationen R aus dem Wertebereich **M** $\times$ **M** definiert.
- **„**für alle x $\in$ M gilt ...“: der **gesamte Wertebereich M** muss geprüft werden
- „für alle x, y $\in$ M gilt ...“: man kann **für x und y auch denselben Wert** wählen
- „**x R y**“ bedeutet „**(x, y)** $\in$ **R**“
- „A **oder** B“ ist wahr, wenn **mindestens eins von beiden wahr** ist
- „**aus A folgt B**“ ist gleichwertig zu „**(nicht A) oder B**“.

Vorlesung Modellierung WS 2001/2002 / Folie 210a

Ziele:
Definitionen aus Mod-2.10 verstehen

in der Vorlesung:
Erläuterungen dazu

nachlesen:
G. Goos: Vorl. über Informatik Bd.1, Abschnitt A.2

Funktionen

Eine Funktion f ist eine **2-stellige Relation f $\subseteq$ D $\times$ B** mit folgender Eigenschaft:
Aus $(x, y) \in f$ und $(x, z) \in f$ folgt $y = z$,
d. h. zu einem Urbild x gibt es nur ein Bild y.

Wertebereich, aus dem solche Funktionen stammen: **D -> B $\subseteq$ Pow (D $\times$ B),** also **f $\in$ D -> B**.

D ist der **Definitionsbereich** von f; B ist der **Bildbereich** von f

Man sagt auch **f hat die Signatur D -> B** oder kurz **f: D -> B**

Beispiel: Funktion Quadrat: $\mathbb{N}$ -> $\mathbb{N}$ mit Quadrat = {(a, b) | a $\in$ $\mathbb{N}$ und b = a*a}

Definitionsbereich und Bildbereich können beliebige,
auch zusammengesetzte Wertebereiche sein, z. B. ggt: $\mathbb{N} \times \mathbb{N}$ -> $\mathbb{N}$

Schreibweisen für $(x, y) \in f$ auch $y = f(x)$ oder $f(x) = y$ oder $x\ f\ y$

Die Menge der Paare $(x, y) \in f$ heißt **Graph von f**.

Beispiele:

geldBeutel: brdMünzen -> $\mathbb{N}_0$ mit
geldBeutel = {(1, 2), (2, 0), (5, 0), (10, 3), (50, 0), (100, 4), (200, 0), (500, 1)}

Sp: Deligierte -> SprachMengen

Vorlesung Modellierung WS 2001/2002 / Folie 211

Ziele:

Grundbegriffe von Funktionen verstehen

in der Vorlesung:

- Begriffe an Beispielen erläutern
- unterscheiden: Funktion, ihre Definition, der Wertebereich, aus dem sie stammt

Achtung! unterschiedliche Verwendung von Begriffen:

- hier: Der **Wertebereich D -> B** ist die Menge aller Funktionen, die von D auf B abbilden.
- hier: Die Funktion f: D -> B hat den **Bildbereich B**.
- Mathe I: Der **Wertebereich einer Funktion f: D -> B ist B**
- Goos: Der Bildbereich einer Funktion f: D -> B ist **Bild(f) = B** . (Werde ich hier nicht verwenden.)
- Mathe I: Das **Bild von f** ist die Menge der Werte, auf die f abbildet.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt A.2

Wichtige Eigenschaften von Funktionen

Eine Funktion **f : D -> B** heißt

- **total**, wenn es für jedes x ∈ D ein Paar (x, y) ∈ f gibt,
- **partiell**, wenn f für einige x ∈ D nicht definiert sein könnte,
- **surjektiv**, wenn es zu jedem y ∈ B ein Paar (x, y) ∈ f gibt,
- **injektiv**, wenn es zu jedem y ∈ B höchstens ein Paar (x, y) ∈ f gibt,
- **bijektiv**, wenn f zugleich surjektiv und injektiv ist.

n-stellige Funktion: f: D -> B; der Definitionsbereich D ist ein Wertebereich von n-Tupeln

0-stellige Funktion k: -> B; der Definitionsbereich ist der **leere Wertebereich**;
k ist eine **konstante Funktion**; b = k() ist ein fester Wert

Kardinalität des Wertebereiches, aus dem Funktionen stammen | D -> B | = | B $|^{|D|}$

Anzahl der Möglichkeiten für unterschiedliche Funktionen mit dieser Signatur

Vorlesung Modellierung WS 2001/2002 / Folie 212

Ziele:

Begriffe einprägen

in der Vorlesung:

Erläuterungen dazu

- Begriffe erläutern
- Graphiken dazu
- Kardinalität begründen

Verständnisfragen:

- Charakterisieren Sie die Eigenschaften graphisch.

Spezielle Funktionen

Identitätsfunktion id_M : M -> M mit $id_M(x) := x$ für alle $x \in M$

Charakteristische Funktion χ_M einer Menge $M \subseteq U$, mit der Trägermenge U:

χ_M : U -> Bool mit Bool = {true, false}, χ_M ist eine totale Funktion,
$\chi_M(x) :=$ true, falls $x \in M$, $\chi_M(x) :=$ false, falls $x \notin M$;
gibt an, welche Elemente der Trägermenge U in M enthalten sind.

Funktionen mit Bildbereich Bool heißen **Prädikate**.

Funktionen auf Indexmengen z. B. Ind = {1, ..., n}:

- modellieren **Auftreten von Werten in Folgen**
  z. B. ws: Ind -> Worte mit ws (i) := das Wort an der i-ten Postion in einem Satz

- modellieren **Zuordnungen zwischen Mengen**
  z. B. das Funktionenpaar
  Marke1: Ind -> Gepäckstücke; injektiv
  Marke2: Ind -> Eigentümer
  ordnet jedem Eigentümer sein Gepäck zu

Multimengen (bags) : Werte können mehrfach vorkommen
b: V -> $\mathbb{N}_0$ gibt für jeden Wert aus V die Häufigkeit in der Multimenge b an.
z.B. geldBeutel: brdMünzen -> $\mathbb{N}_0$ (siehe oben)

Vorlesung Modellierung WS 2001/2002 / Folie 213

Ziele:

Spezielle Anwendungsmuster

in der Vorlesung:

Erläuterungen dazu

- Zusammenhang zu Relationen erläutern.
- Beispiele für Funktionen mit Indexmengen.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt A.2

Verständnisfragen:

- Geben Sie weitere Anwendungen für Funktionen mit Indexmengen an.

Disjunkte Vereinigung

Die allgemeine **disjunkte Vereinigung** fasst n Wertebereiche (Mengen) A_1, A_2, ..., A_n zu einem **vereinigten Wertebereich V** zusammen, wobei I := {1, ..., n}.
Die Herkunft der Elemente aus A_i wird in den Paaren von V gekennzeichnet:

$$\mathbf{V := \{ (i, a_i) \mid a_i \in A_i \}}$$

Die erste Komponente der Paare ist eine **Kennzeichenkomponente** (tag field).
Die A_i brauchen nicht paarweise disjunkt zu sein.

Kardinalität: $|V| = \sum_{i \in I} |A_i|$

Anwendungsmuster:

V ist ein allgemeinerer Wertebereich, abstrahiert von den spezielleren A_i

Beispiele:

GeschäftsPartner = { (Kunde, k) | k ∈ Adresse } ∪ { (Lieferant, l) | l ∈ Adresse}
mit Ind = {Kunde, Lieferant}

Zeichen = { (1, b) | b ∈ Buchstabe} ∪ { (2, z) | z ∈ Ziffer} mit Ind = {1, 2}

Vorlesung Modellierung WS 2001/2002 / Folie 214

Ziele:

Kennzeichnung von Werten

in der Vorlesung:

- Rolle des Kennzeichenfeldes
- Klassifikation von Wertebereichen
- Vergleich mit Varianten-Records in Pascal

2.2 Terme

Übersicht:

Terme bestehen aus **Operationen**, **Operanden**, **Konstanten und Variablen**:

a + 5 blau ? gelb = grün ♥ > ♦

Terme werden nicht „ausgerechnet“; sie haben keinen „Wert“.
Nur ihre **Struktur** ist wichtig.

Notation von Termen in **Infix**-, **Postfix**-, **Präfix**-Form, als **Baum**.

Für **Variable** in Termen werden Terme **substituiert:**

in a + a = 2∗a substituiere 3∗b für a: 3∗b + 3∗b = 2∗3∗b

Abstrakte Algebra: (Abschnitt 2.3)
Terme mit Rechenregeln

Rechenregeln identifizieren Terme mit gleicher Bedeutung:

x + 0 ≡ x pop (push (k, t)) ≡ k

Eigenschaften veränderlicher **Datenstrukturen** werden durch abstrakte Algebren **spezifiziert**.

Konkrete Algebra:
definiert **konkrete Funktionen** zu den Operationen der Terme;

beschreibt **Implementierung** spezifizierter Datenstrukturen.

Vorlesung Modellierung WS 2001/2002 / Folie 215

Ziele:

Informelle Übersicht

in der Vorlesung:

Erläuterungen dazu

- Terme als Strukturen erklären.
- Terme umformen ohne zu "rechnen".

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.1

Sorten und Signatur

Terme werden zu einer **Signatur** gebildet.
Sie legt die verwendbaren Symbole und die Strukturierung der Terme fest.

Signatur $\sum$:= (S, F), S ist eine Menge von **Sorten**, F ist eine Menge von Operationen.

Sorte s $\in$ **S: Name** für einen Wertebereich; er braucht nicht weiter definiert zu sein.
z. B. STACK, T, BOOL

Operation f $\in$ **F**: Name einer Funktion (Operatorsymbol),
seine **Stelligkeit**, **Sorten seiner Operanden** und des **Ergebnisses**

Beispiele:	push:	STACK x T		-> STACK	2-stellig
	createStack:			-> STACK	0-stellig
	$\wedge$:	BOOL x	BOOL	-> BOOL	2-stellig
	$\neg$:		BOOL	-> BOOL	1-stellig
	T:			-> BOOL	0-stellig

0-stellige Operationen sind Konstante

Beispiel für eine Signatur: $\sum_{BOOL}$:= (S_{BOOL}, F_{BOOL}) mit
S_{BOOL} := { BOOL },
F_{BOOL} := { T: -> BOOL, F: -> BOOL, $\wedge$: BOOL x BOOL -> BOOL, $\neg$: BOOL-> BOOL}

Die **Menge der Elementaroperanden X** enthält die **Konstanten aus** $\sum$, ggf. weitere Symbole für Konstante bestimmter Sorten (z.B. Zahlen), sowie Namen für **Variable** bestimmter Sorten.

Vorlesung Modellierung WS 2001/2002 / Folie 216

Ziele:

Begriff der Signatur verstehen

in der Vorlesung:

- Erläuterung der Begriffe
- Beispiele für Terme zu Signaturen
- Signatur zu STACK-Operationen
- Hinweis: Der Name Signatur wird 2-fach verwendet: wie hier definiert und als "Signatur einer Funktion (siehe Folie Mod-2.11)".
- Hinweis: Im Buch werden die Sorten von Operationen erst später unterschieden.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.1

Korrekte Terme

In **korrekten Termen** muss jeweils die Zahl der Operanden mit der **Stelligkeit** der Operation und die **Sorten** der Operandenterme mit den Operandensorten der Operation übereinstimmen.

Die **Menge τ der korrekten Terme der Sorte s** wird induktiv definiert:
Zu einer Signatur $\Sigma = (S, F)$ und Elementaroperanden X ist ein **korrekter Term der Sorte $s \in S$**:

- ein Elementaroperand $a \in X$ der Sorte s, oder
- die **Anwendung einer Operation** $f: s_1 \times s_2 \times ... \times s_n \rightarrow s \in \Sigma$: $f(t_1, t_2, .., t_n)$
  wobei jedes **t_i ein korrekter Term der Sorte s_i** ist.

Nur die nach diesen Regeln gebildeten Terme gehören zur Menge τ der korrekten Terme.

$f(t_1,..., t_n)$ ist ein **n-stelliger Term**; die t_i sind seine **Unterterme**.

Terme aus τ, die **keine Variablen** enthalten, heißen **Grundterme**.

Beispiel: korrekte Terme zur Signatur Σ_{BOOL}: F $\neg$ T T $\wedge$ x $\neg$(a $\wedge$ b) x $\wedge \neg$ y

nicht korrekt: a $\neg$ b $\neg$ ($\wedge$ b)

Vorlesung Modellierung WS 2001/2002 / Folie 217

Ziele:

Regeln zur Struktur von Termen

in der Vorlesung:

- Terme zu den Signaturen von Mod-2.16 konsturieren.
- Beispiele für falsche Terme.
- Vergleich mit Typregeln in Programmiersprachen.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.1

Verständnisfragen:

- Welche Terme kann man aus den Operationen 0: -> N0 und succ: N0 -> N0 bilden?
- Geben Sie einige Terme zu den Signaturen von Mod-2.16 an.

Notationen für Terme

Notation eines n-stelligen Terms mit Operation (Operator) f und Untertermen t_1, t_2, ..., t_n:

Bezeichnung	Notation	Beispiele
Funktionsform:	f (t_1, t_2, ..., t_n)	empty (push (createStack, c)) $\wedge$ (< (0, a), < (a, 10))
Präfixform:	f t_1 t_2 ... t_n	$\wedge$ < 0 a < a 10
Postfixform:	t_1 t_2 ... t_n f	0 a < a 10 < $\wedge$
Infixform (bei 2-stelligen Operationen)	t_1 f t_2	0 < a $\wedge$ a < 10

In der **Infixform** sind ggf. **Klammern** nötig, um Operanden an ihren Operator zu binden:
(x + 3) ***** y

Für die **Infixform** können Operatoren unterschiedliche **Bindungsstärken (Präzedenzen)** haben, z. B. bindet ***** seine Operanden vereinbarungsgemäß stärker an sich als +.

Deshalb sind x + 3 ***** y und x + (3 ***** y) und ((x) + ((3) ***** (y))) verschiedene Schreibweisen in Infixform für denselben Term.

In der Funktionsform, Präfixform, Postfixform werden weder Präzedenzen noch zusätzliche Klammern benötigt.

Vorlesung Modellierung WS 2001/2002 / Folie 218

Ziele:

Verschiedene Notationen für denselben Term

in der Vorlesung:

An weiteren Beispielen erläutern :

- Struktur der Notationen
- Beispiele für 1- und 3-stellige Operationen
- Klammerung nur in Infixform und Funktionsform
- Präzedenz

nachlesen:

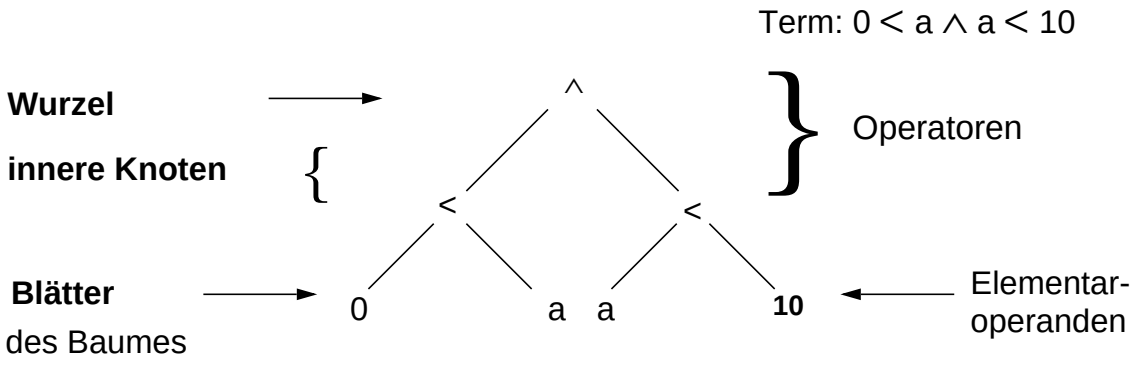
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.1

Verständnisfragen:

- Weshalb benötigen Präfix- und Postfixform keine Klammern?
- In welcher Reihenfolge stehen die Elementaroperanden in den 4 Formen?

Terme als Bäume

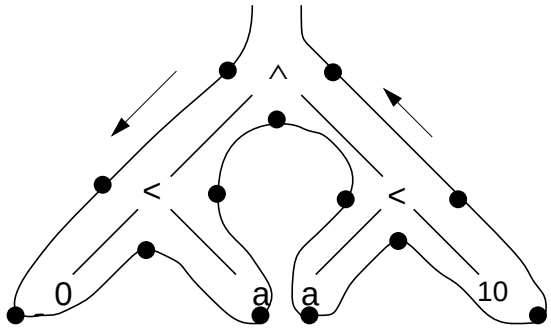
Terme kann man als Bäume darstellen (Kantorowitsch-Bäume):



Aus einem Durchlauf des Baumes in Pfeilrichtung erzeugt man

- **Präfixform**, wenn man beim **ersten Besuch**
- **Postfixform**, wenn man beim **letzten Besuch**
- **Infixform**, wenn man beim **vorletzten Besuch**

den Operator aufschreibt.



Vorlesung Modellierung WS 2001/2002 / Folie 219

Ziele:

Zusammenhang der Darstellungen verstehen

in der Vorlesung:

- Bäume erläutern
- Baumdurchläufe erläutern
- Rekursive Definition der Notationen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.1

Verständnisfragen:

- Warum braucht man in der Baumdarstellung keine Klammern?

Substitution

Eine **Substitution** beschreibt, wie in einem Term vorkommende **Variable durch Terme ersetzt** werden.

Eine **einfache Substitution** σ ist ein Paar aus einer Variablen v und einem Term t zur Signatur $\sum$, v und t haben die Sorte s.
Schreibweise: σ **= [t / v]** Beispiel: σ = [2*****b / x]

u σ ist die **Anwendung einer einfachen Substitution** σ = [t / v] auf einen Term u; ist definiert:

- $u\sigma = t$, falls u die zu ersetzende Variable v ist,
- $u\sigma = u$, falls $u \neq v$ und u ein Elementaroperand oder eine andere Variable ist,
- $u\sigma = f(u_1\sigma, u_2\sigma, ..., u_n\sigma)$, falls $u = f(u_1, u_2, ..., u_n)$

D. h. in u werden **alle Vorkommen der Variablen v gleichzeitig** durch den Term t ersetzt; kommt v auch in t vor, so wird es nicht nochmals ersetzt.
Beispiele: (x + 1) [2*****b / x] = (2*****b + 1) (x + y) [y*****y / y] = (x + y*****y)

Eine **Substitution** $\sigma = [t_1 / x_1, ..., t_n / x_n]$ mit paarweise verschiedenen Variablen x_i steht für die **gleichzeitige Substitution aller Vorkommen aller Variablen** x_i jeweils durch die Terme t_i.

Beispiel: $\sigma = [\ 2*b / x, 3 / y\ ]$

$(x + y)\ \sigma = (2*b + 3)$
$(y + a*y)\ \sigma = (3 + a*3)$
$(x * y)\ [\ y / x,\ y*y / y\ ] = (y * (y * y))$

Vorlesung Modellierung WS 2001/2002 / Folie 220

Ziele:

Formale Definition des Einsetzen für Variable

in der Vorlesung:

An Beispielen erlätern:

- konsistentes Ersetzten mehrerer Vorkommen
- gleichzeitiges Ersetzen
- Ersetzen wird nicht iteriert
- Variable können ungebunden bleiben

Hinweis: Wir haben hier die Notation aus dem Buch von Goos verwendet. In der Literatur gibt es auch andere Varianten der Notation von Substitutionen:

- Die Paare werden auch in umgekehrter Reihenfolge angegeben: Variable / Term oder (Variable, Term)
- Die Notation [s / x, t / y] wird auch als Hintereinanderausführung von [s / x][t / y] definiert. Das führt z. B. bei (x*y)[y+1 / x, a / y] zu einem anderen Ergebnis als mit unserer Definition.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.6

Verständnisfragen:

Geben Sie Beispiele für Substitutionen zu Termen der Signatur zu BOOL an.

Umfassende Terme

Rechenregeln werden mit **allgemeineren Termen** formuliert, die auf **speziellere Terme** angewandt werden,

z. B. Distributivgesetz:	$a * (b + c)$	=	$a * b + a * c$
angewandt auf	$2 * (3 + 4*x)$	=	$2 * 3 + 2 * 4*x$

Ein **Term s umfasst einen Term t**, wenn es eine Substitution σ gibt, die s in t umformt: $s\,\sigma = t$

s ist ein **allgemeineres** Muster; es umfasst den **spezielleren** Term t.

s umfasst t, ist eine **Quasiordnung**, d. h. die Relation **umfasst** ist

transitiv:	sei $r\,\sigma_1 = s$, $s\,\sigma_2 = t$, dann ist $(r\,\sigma_1)\,\sigma_2 = t$
reflexiv:	$t\,[\,] = t$, mit der leeren Substitution [ ]
nicht antisymmetrisch:	$(2*x)\,[\,y / x\,] = 2*y$ und $2*y\,[\,x / y\,] = 2*x$

Vorlesung Modellierung WS 2001/2002 / Folie 221

Ziele:

Terme als Muster verstehen

in der Vorlesung:

- Substitution als Anwendung einer Rechenregel
- Erläuterung der Relation "umfasst"
- weitere Beispiele

Verständnisfragen:

Warum ist es nicht völlig korrekt, zu sagen, dass t spezieller ist als s, wenn s t umfasst?

Unifikation

Bei der Relation „umfasst“ wird ein Term mit einer Substitution in einen anderen transformiert.

Bei der **Unifikation** werden **zwei Terme so substituiert, dass sie gleich werden**.

Zwei Terme s und t sind unifizierbar, wenn es eine Substitution σ gibt mit s σ = t σ.
σ heißt **Unifikator von s und t**.

Beispiel: Terme: s = (x + y) t = (2 + z)
Unifikatoren: σ_1 = [2 / x, z / y] σ_2 = [2 / x, y / z],
σ_3 = [2 / x, 1 / y, 1 / z] σ_4 = [2 / x, 2 / y, 2/ z] ...

Ist σ **ein Unifikator** von s und t und τ **eine Substitution**, dann ist auch die Hintereinanderausführung von erst σ dann τ, also $\sigma\,\tau = \sigma'$ auch ein Unifikator von s und t. Wir sagen σ **ist allgemeiner als** σ'**,** denn tσ umfasst tσ'.

Im Beispiel sind σ_1 und σ_2 allgemeiner als σ_3 und σ_4 .

Ein Unifikator σ heißt **allgemeinster Unifikator** der Terme s und t, wenn es zu allen anderen Unifikatoren σ' eine Substitution τ gibt mit $\sigma\,\tau = \sigma'$.

Im Beispiel sind σ_1 und σ_2 allgemeinste Unifikatoren, z. B. σ_1 [1 / z] = σ_3

Es kann **mehrere allgemeinste Unifikatoren** geben. Sie können durch **Umbenennen von Variablen** ineinander überführt werden, z. B. σ_1 [y / z] = σ_2

Vorlesung Modellierung WS 2001/2002 / Folie 222

Ziele:

Allgemeines Prinzip Unifikation verstehen

in der Vorlesung:

An Beispielen zeigen:

- Unifikatoren
- nicht unifizierbare Terme
- allgemeinste Unifikatoren machen keine unnötigen Festlegungen.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.6

Verständnisfragen:

- Wie müssen 2 Terme beschaffen sein, damit es Unifikatoren gibt, die verschieden sind von den allgemeinsten Unifikatoren? Hinweis: Nur wenn ein allgemeinster Unifikator noch Variablen offen lässt, kann es speziellere geben.

Unifikationsverfahren

Unifikation zweier Terme s und t nach Robinson:

Seien s und t Terme in **Funktionsschreibweise**.
Dann ist das **Abweichungspaar A(s, t) = (u, v)** das erste Paar unterschiedlicher, korrespondierender Unterterme u und v, das man beim Lesen von links nach rechts antrifft.

Algorithmus:

1. Setze σ = [ ] (leere Substitution)
2. Solange es ein Abweichungspaar A(s σ, t σ) = (u, v) gibt wiederhole:
   a. ist **u eine Variable x**, die in v nicht vorkommt, dann ersetze σ durch σ [v / x], oder
   b. ist **v eine Variable x**, die in u nicht vorkommt, dann ersetze σ durch σ [u / x],
   c. **sonst** sind die Terme s und t **nicht unifizierbar; Abbruch** des Algorithmus.
3. Bei Erfolg gilt s σ = t σ und σ **ist allgemeinster Unfikator**.

Beachte, dass bei jeder Iteration die bisherige Substitution auf die vollständigen Terme s, t angewandt wird.

Vorlesung Modellierung WS 2001/2002 / Folie 223

Ziele:

Terme systematisch unifizieren

in der Vorlesung:

- Verfahren an Beispielen zeigen.
- Begründen, weshalb die Substitution immer wieder aus s und t angewandt wird.
- Begründen, weshalb in 2a und 2b geprüft wird, ob die Variable x in dem Unterterm vorkommt.

Verständnisfragen:

- Zeigen sie an einem Beispiel, dass es nötig ist, in 2a und 2b zu prüfen ob die Variable x in dem Unterterm vorkommt.

2.3 Algebren
Abstrakte Algebra

Eine **abstrakte Algebra A = (τ, Σ, Q)** ist definiert durch die **Menge korrekter Terme** τ zur **Signatur** Σ mit Elementaroperanden X und einer **Menge von Axiomen Q**.

Ein **Axiom** hat die Form $t_1 \equiv t_2$ oder $t_1 \not\equiv t_2$, wobei t_1, t_2, **korrekte Terme gleicher Sorte** sind und **Variable** enthalten können.

Ein Axiom $t_1 \equiv t_2$ ist auf alle Paare von Termen r_1, r_2 **anwendbar**, für die es eine Substitution σ gibt, mit $t_1 \sigma = r_1$ und $t_2 \sigma = r_2$. Man kann dann r_1 in r_2 und r_2 in r_1 **umformen**; ebenso kann man größere Terme, die r_1 bzw. r_2 als Unterterme enthalten, mit dem Axiom umformen.

Die Algebra definiert alle Paare von Termen als **gleichbedeutend**, die man mit den Axiomen ineinander umformen kann.

Beispiel: **abstrakte Algebra Bool**: Signatur Σ = ({BOOL}, F)

Operationen F:

true:			-> BOOL
false:			-> BOOL
∧:	BOOL x	BOOL	-> BOOL
∨:	BOOL x	BOOL	-> BOOL
¬:		BOOL	-> BOOL

Axiome Q:

true	≢ false
¬ true	≡ false
¬ false	≡ true
true ∧ x	≡ x
false ∧ x	≡ false
x ∨ y	≡ ¬ (¬ x ∧ ¬ y)

Vorlesung Modellierung WS 2001/2002 / Folie 224

Ziele:

Axiome definieren Terme als gleichbedeutend

in der Vorlesung:

- Anwendungen von Axiomen auf Termpaare zeigen (Substitution)
- Algebra BOOL erläutern

Konkrete Algebra

Zu einer abstrakten Algebra $A = (\tau, \Sigma, Q)$, kann man **konkrete Algebren $\mathbf{A = (A_S, F_A, Q)}$** angeben, wobei

A_S eine **Menge von Wertebereichen**, je einer **für jede Sorte** aus S von Σ

F_A eine **Menge von Funktionen**, je eine **für jede Operation** aus F von Σ.

Die Definitions- und Bildbereiche der Funktionen müssen konsistent den Sorten der Operationen zugeordnet werden.

Die **Axiome Q** müssen mit den Funktionen F_A **für alle entsprechenden Terme aus τ gelten**. (Es können in der konkreten Algebra noch weitere Gesetze gelten.)

Beispiel: eine konkrete Algebra FSet zur abstrakten Algebra Bool:

A_S: $\{\emptyset, \{1\}\}$ für die Sorte BOOL,

F_A: $\{1\}$ für true, $\emptyset$ für false,
Mengendurchschnitt $\cap$ für $\wedge$, Mengenvereinigung $\cup$ für $\vee$,
Mengenkomplement bezüglich $\{1\}$ für $\neg$.

Q: Man kann zeigen, dass die Axiome für die entsprechenden Terme mit F_A gelten,
z. B. false$\wedge$ x $\equiv$ false in Bool entspricht in FSet $\emptyset \cap x \equiv \emptyset$

Die übliche boolsche Algebra ist natürlich auch eine konkrete Algebra zur abstrakten Algebra Bool.

Vorlesung Modellierung WS 2001/2002 / Folie 225

Ziele:

Zusammenhang zwischen konkreter und abstrakter Algebra verstehen

in der Vorlesung:

Am Beispiel erläutern

- Funktionstafeln der konkreten Funktionen angeben
- Gültigkeit von Axiomen zeigen
- Ebenso für die konkrete boolesche Algebra
- Hinweis: Eine konkrete Algebra zu einer abstrakten heißt auch Modell der abstrakten Algebra

Übungsaufgaben:

Tauschen Sie in FSet die Funktionen zu T und F. Gelten die Axiome noch?

Verständnisfragen:

- Zeigen Sie, dass alle Axiome Q von Bool in FSet gelten.

Beispiel 2.2: Datenstruktur Keller

Die Eigenschaften einer **Datenstruktur Keller** beschreiben wir zunächst informell.
Folgende **Operationen** kann man mit einem Keller ausführen:

create Stack:	liefert einen leeren Keller
push:	fügt ein Element in den Keller ein
pop:	entfernt das zuletzt eingefügte Element
top:	liefert das zuletzt eingefügte und nicht wieder entfernte Element
empty:	gibt an, ob der Keller leer ist.

Die Eigenschaften der Datenstruktur Keller sollen präzise durch eine abstrakte Algebra spezifiziert werden.

Vorlesung Modellierung WS 2001/2002 / Folie 226

Ziele:

Das Kellerprinzip informell verstehen

in der Vorlesung:

- Keller-Prinzip: Last-in-first-out (LIFO)
- Beispiel: Aufrufkeller
- Anwendung eines Kellers: Infix in Postfix umwandeln

Verständnisfragen:

- Erklären Sie das Kellerprinzip an der Abarbeitung von Funktions- bzw. Methodenaufrufen in Programmen.

Beispiel: Abstrakte Algebra spezifiziert Keller

Abstrakte Algebra Keller:

Signatur $\Sigma = (S, F)$, $S = \{Keller, Element, BOOL\}$, F:

createStack:		-> Keller
push:	Keller x Element	-> Keller
pop:	Keller	-> Keller
top:	Keller	-> Element
empty:	Keller	-> BOOL

Axiome Q:
für beliebige Terme t der Sorte Element und k der Sorte Keller gilt:

K1:	empty (createStack)	≡ true
K2:	empty (push (k, t)	≡ false
K3:	pop (push (k, t))	≡ k
K4:	top (push (k, t))	≡ t

Keller ist die Sorte, deren Terme Kellerinhalte modellieren.
Element und BOOL sind **Hilfssorten** der Algebra.

Eine **Implementierung** der abstrakten Algebra Keller kann durch eine **konkrete Algebra dazu** beschrieben werden.

Vorlesung Modellierung WS 2001/2002 / Folie 227

Ziele:

Abstrakte Algebra als Spezifikation verstehen

in der Vorlesung:

- Terme umformen
- K3 mit LIFO begründen
- Anschauliche Darstellungen und Implementierungen von Kellerinhalten entsprechen konkreten Algebren

Verständnisfragen:

- Geben sie verschiedene Terme an, die zu top (push (createStack, 1)) und zu push (push (createStack, 1), 2) gleichbedeutend sind.

Strikte Funktionen

Im Beispiel 2.2 ist der Term top (createStack) **nicht definiert**.

Dies kann durch ein weiteres Axiom explizit gemacht werden:

K5: top (createStack) ≡ ⊥

⊥ ist das Symbol für **undefiniert**.

Ebenso können wir unerwünschtes Verhalten explizit spezifizieren:

K6: pop (createStack) ≡ ⊥

Eine **Funktion heißt strikt**, wenn ihr Ergebnis ⊥ ist, falls ein Argument ⊥ ist.

Wir fordern, dass die **Funktionen konkreter Algebren strikt** sind.

Terme, die zu ⊥ equivalent sind, erklären wir als undefiniert.

In **Implementierungen** charakterisiert ⊥ **Fehlersituationen**.
Die Implementierung sollte darauf durch Auslösen einer Ausnahme und/oder Ausgabe einer Fehlermeldung reagieren.

Vorlesung Modellierung WS 2001/2002 / Folie 228

Ziele:

Modellierung von Fehlersituationen

in der Vorlesung:

Beispiele für strikte und nicht-strikte Funktionen

Verständnisfragen:

Was bedeutet es, wenn wir das neue Axiom K6 durch pop(createStack) = createStack ersetzen würden?

Klassifikation von Operationen

In der abstrakten Algebra Keller kann man mit dem Axiom K3 Vorkommen der Operation pop aus Termen entfernen.

Wir könne **gleichbedeutende Terme der Sorte Keller** nach der Anzahl der Vorkommen von Operationen darin vergleichen.
Wir sagen, dass die Terme mit den wenigsten Operationsvorkommen in **Normalform** sind:

$$\text{push}(\ldots.\ \text{push}(\text{createStack}, n_1), \ldots), n_m), \qquad \text{mit } m \geq 0$$

Alle **Operationen, die in der Normalform** vorkommen, heißen **Konstruktoren**.
(hier: push, createStack).

Andere Operationen mit der in der Algebra **definierten Sorte als Ergebnissorte** (hier: Keller) heißen **Hilfskonstruktoren**. (hier: pop).

Operationen, die eine **andere Ergebnissorte** haben, heißen **Projektionen**.
(hier: top, empty).

Vorlesung Modellierung WS 2001/2002 / Folie 229

Ziele:

Einteilung der Operationen

in der Vorlesung:

- Klassifikation erläutern
- Reduzierbarkeit auf Normalform durch strukturelle Induktion beweisen
- In der Normalform hat der Term m Vorkommen der push-Operation. Für m = 0 kommt push nicht vor, der Term lautet createStack.

Anwendungen algebraischer Spezifikationen: Eigenschaften aus den Axiomen erkennen

Beispiel: Keller

1. K3: pop (push (k, t)) ≡ k

Keller-Prinzip: zuletzt eingefügtes Element wird als erstes wieder entfernt (last-in-first-out, LIFO)

2. top: Keller -> Element
   K4: top (push (k, t)) ≡ t

top ist die einzige Operation, die Keller-Elemente liefert:
Nur auf das zuletzt eingefügte, nicht wieder entfernte Element kann **zugegriffen** werden.

3. push (.... push (createStack, n_1) , ...), n_m), mit m≥ 0
   K3: pop (push (k, t)) ≡ k

Die **Anzahl der Elemente im Keller** ist
die Anzahl der push-Operationen abzüglich der Anzahl der pop-Operationen im Term.
Begründung: Rückführung auf Normalform, eine push-Operation für jedes Element im Keller.

Vorlesung Modellierung WS 2001/2002 / Folie 230

Ziele:

Axiome spezifizieren Eigenschaften

in der Vorlesung:

- Die drei Beispiele erläutern
- Über Keller nachdenken, ohne sie zu implementieren

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.3

Spezifikation um Operationen erweitern

Erweitere die Keller-Spezifikation um eine **Operation size**.
Sie soll die **Anzahl der Elemente im Keller** liefern.

1. Operation **size in die Signatur einfügen:**
   size: Keller -> N

2. Ergebnis-Sorte **N zu den Sorten zufügen**:
   S = {Keller, Element, BOOL, N}

3. **Axiome zufügen**, so dass size für jeden Keller-Wert definiert ist:
   K7: size (createStack) ≡ 0
   K8: size (push (k, t)) ≡ size (k) + 1

Weil in der **Normalform** nur createStack und push vorkommen, braucht size nur für solche Terme definiert zu werden.

Es wird vorausgesetzt, dass für die Sorte N die Konstanten 0 und 1 sowie die Operation + definiert sind.

Ziele:

Operationen und Axiome entwerfen

in der Vorlesung:

Schritte zur Erweiterung der Spezifikation zeigen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.3

Verständnisfragen:

Erweitern Sie die Spezifikation um eine Operation, die 2 Elemente einfügt.

Realisierung der Spezifikation durch eine konkrete Algebra

Beispiel: eine Realisierung von Kellern durch **Funktionen auf Folgen** von natürlichen Zahlen:

Zuordnung der Sorten:	konkret	abstrakt
	Bool	BOOL
	$\mathbb{N}_0$	Element
	N-Folge = $\mathbb{N}_0^*$	Keller

Signatur und **Zuordnung von Funktionen**

konkret			abstrakt
newFolge:		-> N-Folge	createStack
append:	N-Folge x $\mathbb{N}_0$	-> N-Folge	push
remove:	N-Folge	-> N-Folge	pop
last:	N-Folge	-> $\mathbb{N}_0$	top
noElem:	N-Folge	-> Bool	empty

Definition der Funktionen

newFolge( )	-> ()
append ((a_1, ..., a_n), x)	-> (a_1, ..., a_n, x)
remove ((a_1, ..., a_{n-1}, a_n))	-> (a_1, ..., a_{n-1})
last ((a_1, ..., a_n))	-> a_n
noElem (f)	-> f = ()

Gültigkeit der Axiome zeigen

Vorlesung Modellierung WS 2001/2002 / Folie 232

Ziele:

Beispiel für eine Realisierung

in der Vorlesung:

Funktionen erläutern

- Zuordnung erläutern
- Gültigkeit der Axiome zeigen

Übungsaufgaben:

Mit der Klasse Vector aus java.lang kann man Keller implementieren. Schlagen sie ihre Dokumentation nach und begründen Sie, dass sich die Methoden addElement, removeElemenAt, lastElement, und size dafür eignen.

Keller in Algorithmen einsetzen

Aufgabe: Terme aus **Infix-Form in Postfix-Form** umwandeln
gegeben: Term t in Infix-Form, mit 2-stelligen Operatoren unterschiedlicher Präzedenz, zunächst ohne Klammern
gesucht: Term t in Postfix-Form

Eigenschaften der Aufgabe und der Lösung:

1. **Reihenfolge der Elementaroperanden bleibt unverändert**
2. **Elementaroperanden werden vor ihrem Operator ausgegeben**, also sofort
3. In der Infix-Form aufeinander folgende **Operatoren echt steigender Präzedenz** stehen in der Postfix-Form **in umgekehrter Reihenfolge**; also kellern.
4. **Operatorkeller enthält Operatoren echt steigender Präzedenz**.
   Es gilt die **Kellerinvariante KI**:
   Sei push (... push (CreateStack, opr_1), opr_2), ...) dann gilt
   Präzedenz (opr_i) < Präzedenz (opr_{i+1})

Vorlesung Modellierung WS 2001/2002 / Folie 233

Ziele:

Kelleranwendung verstehen

in der Vorlesung:

- Erläuterung der Eigenschaften
- Eigenschaften der Aufgabe verstehen, bevor sie gelöst wird
- Kellerinvariante: Eine Aussage, die für die Benutzung des Kellers immer gelten muss.

Verständnisfragen:

Wie werden bei diesen Regeln aufeinander folgende Operatoren gleicher Präzedenz behandelt?

Algorithmus: Infix- in Postfix-Form wandeln

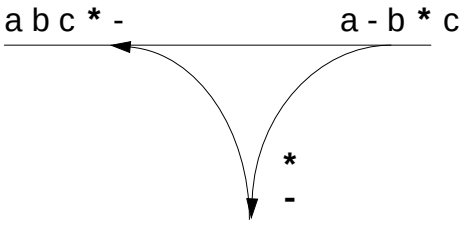
Die Eingabe enthält einen Term in Infix-Form;
die Ausgabe soll den Term in Postfix-Form enthalten

```
Variable: keller ∈ Keller; symbol ∈ Operator ∪ ElementarOperand

keller = createStack();
solange Eingabe nicht leer wiederhole                    {KI}
        lies symbol
        falls symbol ∈ ElementarOperand
                gib symbol aus
        falls symbol ∈ Operator                          {KI}
                solange  not empty (keller) ∧
                         Präzedenz (top (keller)) ≥ Präzedenz (symbol)
                wiederhole                               {KI}
                         gib top (keller) aus;
                         keller = pop (keller);
                keller = push(keller, symbol);           {KI}

solange not empty (keller) wiederhole
        gib top(keller) aus;
        keller = pop(keller);
```

An den Stellen {KI} gilt die Kellerinvariante.



Vorlesung Modellierung WS 2001/2002 / Folie 234

Ziele:

Benutzung der Kelleroperationen verstehen

in der Vorlesung:

- Algorithmus erläutern
- Graphik erläutern
- Gültigkeit der Kellerinvariante zeigen

3. Logik
3.1 Aussagenlogik

Kalkül zum **logischen Schließen**. Grundlagen: Aristoteles 384 - 322 v. Chr.

Aussagen: Sätze, die prinzipiell als wahr oder falsch angesehen werden können.
z. B.: „Es regnet.“, „Die Straße ist nass.“
aber „Dieser Satz ist falsch.“ ist in sich widersprüchlich, ist keine Aussage.

Junktoren verknüpfen Aussagen: „Es regnet nicht, **oder** die Straße ist nass.“

Aussagenlogische Formeln als Sätze einer formale Sprache:

z. B.	regen	$\rightarrow$	straßeNass	$\equiv$	$\neg$ regen	$\vee$	straßeNass	
Belegung der Aussagen mit **Wahrheitswerten**:	f		w		f		w	
Interpretation der Formel liefert Wahrheitswert:		w		w		w		w

Formales Schließen im Gegensatz zur empirischen Beurteilung, z. B. ob „die Straße nass ist.“

Aus „Wenn es regnet, ist die Straße nass.“ **und** „Es regnet.“ **folgt** „Die Straße ist nass.“

Formales Schließen über Algorithmen und Programmen, Hoaresche Logik

Aussagen in der **Spezifikation**, in der **Modellierung** von Aufgaben

Ziele:

Aussagen verknüpfen

in der Vorlesung:

Begriffe erläutern

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.1.1

Vorschau auf Begriffe

- **Aussagenlogische Formeln** definiert durch **Signatur der boolschen Algebra**
- **Belegung von Variablen** mit Wahrheitswerten
- **Interpretation** aussagenlogischer Formeln
- **Gesetze der boolschen Algebra** zur Umformung von Formeln
- **erfüllbare** und **allgemeingültige** Formeln
- **logisches Schließen**

- **Aussagen über Algorithmen**
- **Schlussregeln für Algorithmenelemente, Hoaresche Logik**

Vorlesung Modellierung WS 2001/2002 / Folie 302

Ziele:
Übersicht

in der Vorlesung:
Zusammenhang der Begriffe zeigen

Beispiel: Aussagenlogik in der Spezifikation

Unfall durch fehlerhafte Spezifikation:

Airbus A320, Warschau (1993). Der zuständige Rechner blockiert bei der Landung die Aktivierung von Schubumkehr und Störklappen, wodurch das Flugzeug über das Landebahnende hinausschießt. Es herrschen starker Wind von schräg hinten und Aquaplaning auf der Landebahn.

Beabsichtigte Spezifikation der Störklappenfreigabe:

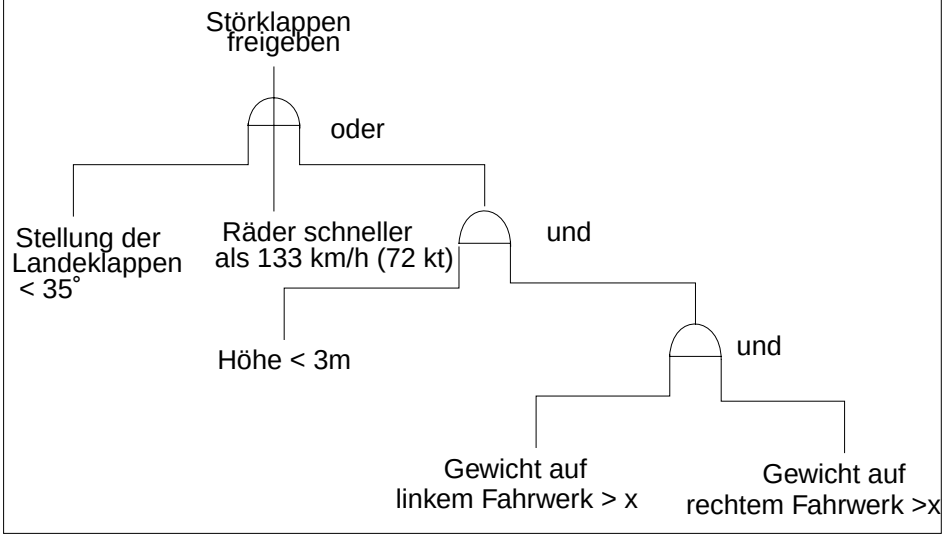
Die Störklappen dürfen benutzt werden

- im Reise- und Sinkflug (Bremswirkung)
- nach der Landung (Vernichtung des Auftriebes und Bremswirkung)

Sie dürfen nicht benutzt werden

- im Endanflug (gefährlicher Auftriebsverlust)

Tatsächliche Spezifikation der Störklappenfreigabe:



Vorlesung Modellierung WS 2001/2002 / Folie 303

Ziele:

Einfaches Beispiel für verknüpfte Aussagen

in der Vorlesung:

- Begründung der Spezifikation
- Erl&äuml;uterung der Unfallursache

Verständnisfragen:

Schlagen Sie eine Korrektur der Spezifikation vor.

Aussagenlogische Formeln

Aussagenlogische Formeln sind korrekte Terme mit Variablen zur Signatur der boolschen Algebra:

O:		-> Bool	falsch
L:		-> Bool	wahr
$\wedge$:	Bool x Bool	-> Bool	**Konjunktion**
$\vee$:	Bool x Bool	-> Bool	**Disjunktion**
$\neg$:	Bool	-> Bool	**Negation**

Erweiterung:

$\rightarrow$:	Bool x Bool	-> Bool	**Implikation** $p \rightarrow q$ für $\neg p \vee q$
$\equiv$:	Bool x Bool	-> Bool	**Äquivalenz** $p \equiv q$ für $(p \rightarrow q) \wedge (q \rightarrow p)$

Operatoren (**Junktoren**) in **fallender Präzedenz**: $\neg \wedge \vee \rightarrow \equiv$

Variable, sowie O und L sind **atomare Aussagen** (Elementaroperanden), die übrigen Formeln sind **zusammengesetzt**.

Für **Variable** schreiben wir meist kleine Buchstaben p, q, ... für **allgemeine Formeln** große Buchstaben F, G, H, ...

Die Definition der **Struktur** der Formeln heißt **Syntax der Aussagenlogik**.

Vorlesung Modellierung WS 2001/2002 / Folie 304

Ziele:

Syntax der Aussagenlogik

in der Vorlesung:

- Term: nur Struktur; Formel: Term mit Bedeutung; an Hand von Funktionen ausrechnen
- Signatur bestimmt Struktur der Formeln
- Beispiele
- Präzedenz und Klammerung
- Vorsicht beim Übertragen von Umgangssprache in Formeln, insbesondere bei Klammerung und Implikation.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.1.1

Interpretation aussagenlogischer Formeln

Eine **passende Belegung** ordnet allen Variablen, die in einer Menge von Formeln F vorkommen, jeweils einen Wahrheitswert w oder f für wahr oder falsch zu.
Die Belegung kann als Substitution angegeben werden, z.B. σ = [w/p, f/q].

Eine **Interpretation** $\Im_\sigma$ einer aussagenlogischen Formel F bildet F auf einen Wahrheitswert ab.
Für Variable ist die Interpretation $\Im_\sigma$ durch die Belegung σ definiert.
Für zusammengesetzte Formeln wird sie durch folgende Wahrheitstafeln erweitert:

$\Im(O) = f$
$\Im(L) = w$

$\Im(F)$	$\Im(\neg F)$
w	f
f	w

$\Im(F)$	$\Im(G)$	$\Im(F \wedge G)$
w	w	w
w	f	f
f	w	f
f	f	f

$\Im(F)$	$\Im(G)$	$\Im(F \vee G)$
w	w	w
w	f	w
f	w	w
f	f	f

Eine Interpretation $\Im_\sigma$ mit einer Belegung σ für eine Formel F bestimmt einen **Wahrheitswert der Formel F:** $\Im_\sigma(F)$

Wenn $\Im_\sigma(F)$ = w gilt, heißt $\Im_\sigma$ auch ein **Modell der Formel F.**

Vorlesung Modellierung WS 2001/2002 / Folie 305

Ziele:

Wahrheitswerte zu aussagenlogischen Formeln

in der Vorlesung:

- Belegung erläutern
- Wir gehen davon aus, dass wir passende Belegungen zu der Menge der Formeln wählen, die wir gerade untersuchen.
- logische Verknüpfungen zeigen
- Interpretation: Belegung plus Verknüpfungen
- Beispiele dazu
- Bei n Variablen 2 hoch n verschiedene Belegungen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.1.1

Vorsicht beim Formalisieren umgangssprachlicher Aussagen

Vorsicht bei **Implikationen**; mit Belegungen prüfen, was gemeint ist:

Wenn es regnet, benutze ich den Schirm.

Ich benutze den Schirm, wenn es regnet.

Ich benutze den Schirm, nur wenn es regnet.

„Oder“ kann fast immer in das **nicht-ausschließende** $\vee$ übersetzt werden:

Hast Du ein Markstück oder zwei Fünfziger?

Morgen fahre ich mit dem Zug oder mit dem Auto nach Berlin.

x ist kleiner y oder x ist gleich y.

Der Händler gibt Rabatt oder ein kostenloses Autoradio.

Aussagen sind häufig **kontext-abhängig**:

Weil ich die SWE-Klausur nicht bestanden habe, nehme ich am zweiten Termin teil.

Weil ich die Modellierungs-Klausur bestanden habe, nehme ich am zweiten Termin nicht teil.

Klammern sind meist nur aus dem Kontext erkennbar:

Sie wollten nicht verlieren oder unentschieden spielen.

Vorlesung Modellierung WS 2001/2002 / Folie 305a

Ziele:

Sorgfältig formalisieren

in der Vorlesung:

Erläuterungen dazu

- Aussagenlogische Formeln zu den Sätzen entwickeln
- Begründungen dazu.

Erfüllbarkeit von Formeln

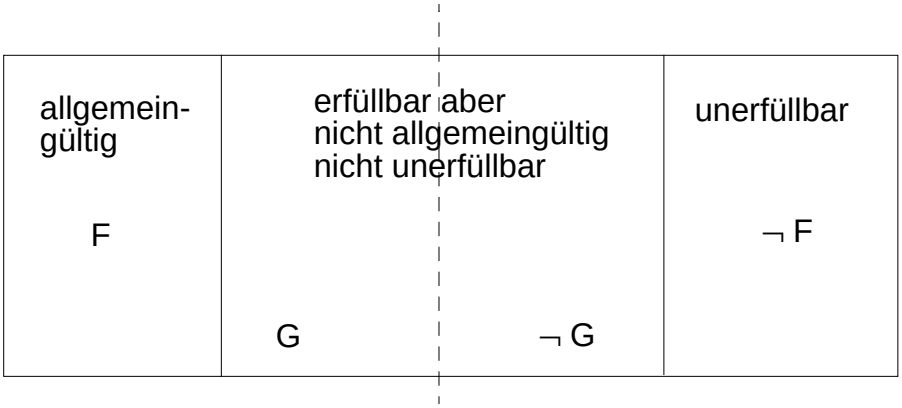
Eine Formel F heißt **erfüllbar**, wenn es
eine Interpretation $\Im_\sigma$ mit einer Belegung σ gibt, so dass gilt $\Im_\sigma$ (F) = w,
sonst ist sie **unerfüllbar** (**widerspruchsvoll**).

z. B. $p \wedge q$ ist erfüllbar; $p \wedge \neg p$ ist unerfüllbar.

Eine Formel F heißt **allgemeingültig** oder **Tautologie**, wenn
für alle ihre Interpretationen $\Im_\sigma$ (F) = w gilt.

z. B. $p \vee \neg p$.

Eine Formel F ist genau dann allgemeingültig, wenn $\neg$ F unerfüllbar ist.



Vorlesung Modellierung WS 2001/2002 / Folie 306

Ziele:

Begriffe zur Erfüllbarkeit verstehen

in der Vorlesung:

- Weitere Beispiele dazu
- Schematische Einteilung der Formelmengen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.1.1

Gesetze der booleschen Algebra

Zwei Formeln F, G sind **logisch äquivalent**,
wenn sie für alle Interpretationen $\mathfrak{I}$ dasselbe Ergebnis haben: $\mathfrak{I}(F) = \mathfrak{I}(G)$

Für alle aussagenlogischen Fomeln X, Y, Z gelten folgende **logische Äquivalenzen**:

$(X \wedge Y) \wedge Z \equiv X \wedge (Y \wedge Z)$	$(X \vee Y) \vee Z \equiv X \vee (Y \vee Z)$	Assoziativität
$X \wedge Y \equiv Y \wedge X$	$X \vee Y \equiv Y \vee X$	Kommutativität
$X \wedge X \equiv X$	$X \vee X \equiv X$	Idempotenz
$X \vee (Y \wedge Z) \equiv (X \vee Y) \wedge (X \vee Z)$	$X \wedge (Y \vee Z) \equiv (X \wedge Y) \vee (X \wedge Z)$	Distributivität
$X \vee (X \wedge Y) \equiv X$	$X \wedge (X \vee Y) \equiv X$	Absorption
$X \wedge O \equiv O$ $X \wedge L \equiv X$	$X \vee O \equiv X$ $X \vee L \equiv L$	Neutrale Elemente
$X \wedge \neg X \equiv O$	$X \vee \neg X \equiv L$	Komplement
$\neg \neg X \equiv X$		Involution
$\neg (X \wedge Y) \equiv \neg X \vee \neg Y$	$\neg (X \vee Y) \equiv \neg X \wedge \neg Y$	De Morgan

Vorlesung Modellierung WS 2001/2002 / Folie 307

Ziele:

Übersicht zu Rechenregeln

in der Vorlesung:

- Überprüfung von Gesetzen durch Wahrheitstafeln
- Anwenden von Gesetzen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 3.2

Logischer Schluss

Sei $\mathbb{A}$ eine Menge von Formeln und F eine Formel.
Wenn für **alle Interpretationen** $\Im$, die alle Formeln in $\mathbb{A}$ erfüllen, auch $\Im$ (F) gilt, dann sagen wir

„F folgt semantisch aus $\mathbb{A}$“ $\quad \mathbb{A} \models F$

$\mathbb{A} \models F$ heißt auch **logischer Schluss**,
$\mathbb{A}$ **Annahme** oder Antezedent, F **Folgerung** oder Konsequenz.

Die **Korrektheit eines logischen Schlusses** $\mathbb{A} \models F$ mit $\mathbb{A} = \{A_1, ..., A_n\}$ kann man prüfen:

$A_1 \wedge ... \wedge A_n \wedge \neg F$ muss unerfüllbar sein.

Beweise werden aus logischen Schlüssen aufgebaut.

Ein **logischer Kalkül gibt Regelschemata für logische Schlüsse** an.

Ein Schema $\{F, G\} \models F \wedge G$ notiert man auch:

$$\frac{F \quad G}{F \wedge G}$$

Es bedeutet: Aus der Gültigkeit von F und G kann man schließen, dass $F \wedge G$ gilt.

Vorlesung Modellierung WS 2001/2002 / Folie 308

Ziele:

Grundbegriff des logischen Schlusses verstehen

in der Vorlesung:

- Beispiele für logische Schlüsse zeigen
- Unterscheiden den log. Schluss A |= F von der aussagenlog. Formel A -> F
- Beispiele für einfache Schlussregeln

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.1.2

Hoaresche Logik

Hoaresche Logik: Kalkül zum Beweisen von **Aussagen über Algorithmen und Programme**, Programm-**Verifikation**, [C.A.R. Hoare, 1969].

Statische Aussagen über Zustände des Algorithmus, über Werte von Variablen nachweisen, ohne den Algorithmus auszuführen. Aussagen **gelten für alle Ausführungen** des Algorithmus.

Im Gegensatz zum **dynamischen Testen**: Ausführen des Algorithmus für bestimmte Eingaben.

Durch logische Schlüsse zeigen, dass

- ein Algorithmus **aus jeder zulässigen Eingabe die geforderte Ausgabe** berechnet, d. h. die **Spezifikation erfüllt**
  {gegeben} Algorithmus {gesucht}
  {a, b $\in \mathbb{N}$} Euklidischer Algorithmus {x = ggt (a, b)}
- eine Aussage an jeder Stelle des Algorithmus gilt, **Invariante**,
  z. B. Kellerinvariante in Mod-2.34
- an einer bestimmten **Stelle im Algorithmus eine bestimmte Aussage gilt**
  z. B. { $\neg$ empty (keller) } keller = pop (keller);
- **Terminierung von Schleifen** beweisen.

Ein **Algorithmus und die Aussagen dazu sollen zusammen konstruiert** werden.

Vorlesung Modellierung WS 2001/2002 / Folie 309

Ziele:

Bedeutung der Verifikation erkennen

in der Vorlesung:

- Ziele der Verifikation erläutern
- Man wählt die Aussagen, die man über den Algorithmus beweisen will, z.B. seine Spezifikation
- Nicht: "Der Algorithmus wird bewiesen."

Verifikation eines Algorithmus zum Potenzieren

Vorbedingung: $x \in \mathbb{R}$ und $n \in \mathbb{N}_0$

Nachbedingung: $q = x^n$

Konstruktionsidee:

Invariante INV: $x^n = q * a^i \wedge i \geq 0$

Zielbedingung: $i \leq 0$

falls i gerade: $x^n = q * (a^2)^{i/2}$

falls i ungerade: $x^n = q * a * (a^2)^{i/2}$

Algorithmus:

```
a := x; q := 1; i := n;
{ INV }
solange i > 0 wiederhole
   { INV ∧ i > 0 }
   falls i ungerade:
        { x^n = q * a * (a^2)^(i/2) ∧ i > 0 } q := q * a; { x^n = q * (a^2)^(i/2) ∧ i > 0 }
        { INV ∧ i > 0 ∧ i gerade } → { x^n = q * (a^2)^(i/2) ∧ i > 0 }
   { x^n = q * (a^2)^(i/2) ∧ i > 0 }
   a := a * a;
   { x^n = q * a^(i/2) ∧ i > 0 } → { x^n = q * a^(i/2) ∧ i/2 ≥ 0 }
   i := i / 2
   { x^n = q * a^i ∧ i ≥ 0 } ≡ { INV }
{ INV ∧ i ≤ 0 } → { q = x^n }
```

Terminierung der Schleife: i fällt monoton und $i \geq 0$ ist invariant.

Vorlesung Modellierung WS 2001/2002 / Folie 309a

Ziele:

Beispiel für eine Algorithmenverifikation

in der Vorlesung:

Hier nur ersten Eindruck vermitteln. Später Erläuterungen

- zur Konstruktionsidee,
- zur Rolle der Schleifeninvarianten,
- zur Anwendung der Alternativenregel,
- zu Anwendungen der Zuweisungsregel
- zum Terminierungsnachweis

Notation

Aussagen über den Algorithmenzustand, über Werte von Variablen werden in den Algorithmus eingefügt:

{P} S {Q}
{ Vorbedingung } S { Nachbedingung }

Wenn vor der Ausführung des Schrittes S die Aussage P gilt, dann gilt Q nach der Ausführung von S, **falls S terminiert**.

Beispiel: .... { i + 1 ≥ 0} i = i + 1; {i ≥ 0} a [i] = k; ...

Die Aussagen werden über die **Struktur von S verfeinert**.
Anwendung von **Schlussregeln** für die Algorithmenelemente.

Eine **Spezifikation liefert Vorbedingung und Nachbedingung** für den gesamten Algorithmus:

gegeben:		gesucht:
Aussagen über die Eingabe		Aussagen über Zusammenhang zwischen Ein- und Ausgabe
{ Vorbedingung }	Algorithmus	{ Nachbedingung }

Vorlesung Modellierung WS 2001/2002 / Folie 310

Ziele:

Notation von Aussagen im Algorithmus

in der Vorlesung:

- Terminierung muss separat gezeigt werden
- Aussagen haben keinen Einfluss auf den Algorithmus

Schlussregeln für Algorithmenelemente

Hoarescher Kalkül definiert **für jedes Algorithmenelement** ein **Regelschema.**

Algorithmenelemente:

Sequenz	S_1; S_2
Zuweisung	x := e
Alternativen	falls B: S_1 sonst S_2 oder falls B: S_1
Aufruf eines Unteralgorithmus	UA
Schleife	solange B wiederhole S

Sequenzregel:

$$\frac{\{P\}\ S_1\ \{Q\} \qquad \{Q\}\ S_2;\ \{R\}}{\{P\}\ S_1; S_2\ \{R\}}$$

Bedeutung:

Wenn $\{P\}\ S_1\ \{Q\}$ und $\{Q\}\ S_2\ \{R\}$ korrekte Schlüsse sind, dann ist auch $\{P\}\ S_1; S_2\ \{R\}$ ein korrekter Schluss

Beispiel:

$$\frac{\{\ x = y*(1+q) + (r-y)\ \} \quad r := r - y \quad \{\ x = y*(1+q) + r\ \} \qquad \{\ x = y*(1+q) + r\ \} \quad q := 1 + q \quad \{\ x = y*q + r\ \}}{\{\ x = y*(1+q) + (r-y)\ \} \quad r := r - y;\ q := 1 + q \quad \{\ x = y*q + r\ \}}$$

Vorlesung Modellierung WS 2001/2002 / Folie 311

Ziele:

Notation von Schlussregeln

in der Vorlesung:

- Prinzip an der Sequenzregel erläutern
- Beispiel erläutern

Konsequenzregeln

Abschwächung der Nachbedingung

$$\frac{\{P\}\ S\ \{R\} \qquad R \rightarrow Q}{\{P\}\ S\ \{Q\}}$$

Verschärfung der Vorbedingung

$$\frac{P \rightarrow R \qquad \{R\}\ S\ \{Q\}}{\{P\}\ S\ \{Q\}}$$

Beispiel:

$$\frac{\{(x > 0) \wedge (y > 0)\}\ S\ \{(z + u * y = x * y) \wedge (u = 0)\} \qquad (z + u * y = x * y) \wedge (u = 0) \rightarrow (z = x * y)}{\{(x > 0) \wedge (y > 0)\}\ S\ \{(z = x * y)\}}$$

Vorlesung Modellierung WS 2001/2002 / Folie 312

Ziele:

Vor- und Nachbedingung anpassen

in der Vorlesung:

Anwendung beim Zusammensetzen von Algorithmenschritten zeigen

Verständnisfragen:

Zuweisungsregel

Die Zuweisung x := e wertet den Ausdruck e aus und weist das Ergebnis der Variablen x zu.

$$\{ P \begin{smallmatrix} x \\ e \end{smallmatrix} \} \; x := e \; \{P\}$$

$P \begin{smallmatrix} x \\ e \end{smallmatrix}$ steht für die **Substitution von x durch e in P**, d. h. P [e/x].

Beispiele: $\{a > 0\}$ $x := a$ $\{x > 0\}$
$\{i + 1 > 0\}$ $i := i + 1$ $\{i > 0\}$

Wenn man zeigen will, dass **nach der Zuweisung eine Aussage P für x** gilt,
muss man zeigen, dass **vor der Zuweisung dieselbe Aussage P für e** gilt.

Vorlesung Modellierung WS 2001/2002 / Folie 313

Ziele:

Zuweisungsregel verstehen

in der Vorlesung:

- Substitution erläutern
- Vorbedingung aus der Nachbedingung rückwärts konstruieren: Was will ich zeigen?
- Beispiele von Folie Mod-313a erläutern

Beispiele für Zuweisungsregel

$\{ P^{x}_{e} \}$ x := e { P }
in P x durch e ersetzt

1.	{ a > 0 }	x := a	{ x > 0 }	
2.	{ a > 0 ∧ a > 0 }	x := a	{ x > 0 ∧ a > 0 }	x durch a ersetzen - nicht umgekehrt
3.	{ a > 0 ∧ x = 7 }	x := a	{ x > 0 ∧ x = 7 }	**falscher Schluss**! **alle** x durch a ersetzen!
4.	{ a > 0 ∧ z > 0 }	x := a	{ x > 0 ∧ z > 0 }	z > 0 ist nicht betroffen
5.	{ i + 1 > 0 }	i := i + 1	{ i > 0 }	
6.	{ i ≥ 0} ≡ { i + 1 > 0 }	i := i + 1	{ i > 0 }	passend umformen
7.	{ i = 2 } ≡ { i + 1 = 3 }	i := i + 1	{ i = 3 }	passend umformen
8.	{ wahr } ≡ { 1 = 1 }	x := 1	{ x = 1 }	passend umformen
9.	{ z = 5 } ≡ { z = 5 ∧ 1 = 1 }	x := 1	{ z = 5 ∧ x = 1 }	passend umformen

Vorlesung Modellierung WS 2001/2002 / Folie 313a

Ziele:

Gebrauch der Zuweisungsregel üben

in der Vorlesung:

- Erläuterung und Begründung der Beispiele;
- Aus der gewünschten Nachbedingung die Vorbedingung herstellen.
- Dann zeigen, dass die Vorbedingung wirklich gilt.

Aufrufregel

Aufruf eines Unteralgorithmus UA
mit der Spezifikation: Vorbedingung P, Nachbedingung Q:

{ P } UA { Q }

Diese Regel gilt nur für Unteralgorithmen **ohne Parameter** und **ohne Funktionsergebnis**.
Sie lesen oder schreiben globale Variable. Sie können in P und Q vorkommen.

Vorlesung Modellierung WS 2001/2002 / Folie 313b

Ziele:

Einfache Aufrufregel

in der Vorlesung:

Erläuterungen und Beispiel dazu

Regeln für Alternativen

2-seitige Alternative

$$\frac{\{P \wedge B\}\ S_1\ \{Q\} \qquad \{P \wedge \neg B\}\ S_2\ \{Q\}}{\{P\}\ \text{falls } B\text{: } S_1 \text{ sonst } S_2\ \{Q\}}$$

Bedingter Schritt

$$\frac{\{P \wedge B\}\ S\ \{Q\} \qquad P \wedge \neg B \rightarrow Q}{\{P\}\ \text{falls } B\text{: } S\ \{Q\}}$$

Beispiele:

$$\frac{\{a > 0\}\ b := a\ \{b > 0\} \rightarrow \{\ b \geq 0\ \} \qquad \{\neg\ (a > 0)\ \} \equiv \{\ -a \geq 0\ \}\ b := -\ a\ \{\ b \geq 0\ \}}{\{true\}\ \text{falls } a > 0\text{: } b := a \ \text{ sonst } \ b := -\ a\ \{b \geq 0\}}$$

$$\frac{\{a < 0\} \rightarrow \{\ -a > 0\ \} \qquad a := -\ a;\ \{a > 0\} \rightarrow \{\ a \geq 0\ \} \qquad \neg\ (a < 0) \ \rightarrow \ a \geq 0}{\{true\}\ \text{falls } a < 0\text{: } a := -a\ \{\ a \geq 0\ \}}$$

Entwurfsparadigma Alternative:
Verschiedene Vorbedingungen führen zur gleichen Nachbedingung.

Vorlesung Modellierung WS 2001/2002 / Folie 314

Ziele:

Alternativenregeln verstehen

in der Vorlesung:

- Beide Zweige müssen auf dieselbe Aussage führen
- Beispiele zeigen
- Konsequenzregeln mitverwendet

Verständnisfragen:

Schleifenregel

Wiederholung, Schleife:

$$\frac{\{P \wedge B\}\ S\ \{P\}}{\{P\}\ \text{solange B wiederhole S}\ \{P \wedge \neg B\}}$$

Schleifeninvariante: Die **Aussage P** bleibt über die Ausführung der Schleife invariant.
Die Schleifeninvariante gilt **vor der Schleife**,
nach jeder Ausführung von S und
nach der Schleife.

Beispiel: a := x; b := y; z := 1;

$\{z \cdot a^b = x^y \wedge b \geq 0\}$

solange $b > 0$ wiederhole

$\{z \cdot a^b = x^y \wedge b > 0\} \equiv \{z \cdot a \cdot a^{b-1} = x^y \wedge (b-1) \geq 0\}$

b := b - 1;

$\{z \cdot a \cdot a^b = x^y \wedge b \geq 0\}$

z := z · a

$\{z \cdot a^b = x^y \wedge b \geq 0\}$

$\{z \cdot a^b = x^y \wedge b \geq 0 \wedge b \leq 0\} \equiv \{z \cdot a^b = x^y \wedge b = 0\} \rightarrow \{z = x^y\}$

Vorlesung Modellierung WS 2001/2002 / Folie 315

Ziele:

Schleifeninvariante verstehen

in der Vorlesung:

Erläuterungen dazu

- Das Prinzip Invariante erläutern.
- Schlussregel erläutern.
- Am Beispiel mehrere Invariante zeigen.

Schleifen aus Invarianten konstruieren

Entwurfsparadigma Invariante:

Die gewünschte Nachbedingung über Variablen a, b, ... ist zerlegbar in

(Invariante P über a, b, ...) $\wedge \neg$ **(Bedingung B über a, b, ...)**

Konstruiere eine Schleife mit Invariante P und Bedingung B.
Der Rumpf ändert die Variablen und lässt die Invariante gültig.

Beispiel:

Mit den Variablen a, b und z soll $z = x^y$ berechnet werden.

umformen in $(z * a^b = x^y \wedge b \geq 0)$ $\wedge (b \leq 0)$
P $\wedge \neg$ B

Initialisierung:

```
a := x; b := y; z := 1; { P }
```

Schleife mit Invariante P und Bedingung B:

```
solange b > 0 wiederhole
        b := b - 1;
        z := z * a
```


Vorlesung Modellierung WS 2001/2002 / Folie 316

Ziele:

Invariante als Konstruktionsprinzip

in der Vorlesung:

Erläuterungen dazu

- Schleife systematisch aus Invariante konstruieren.
- Invarianten nachträglich suchen: Nach der Schleife zu erreichende Zielaussage umformen in P und nicht B. Prüfen, ob P eine geeignete Invariante ist.
- Beispiele zeigen.

Terminierung von Schleifen

Die **Terminierung einer Schleife muss separat nachgewiesen werden**.

solange B wiederhole S

1. Gib einen **ganzzahligen Ausdruck E** an über Variablen, die in der Schleife vorkommen, und zeige, dass E bei jeder Iteration durch S **verkleinert** wird.
2. Zeige, dass **E nach unten begrenzt** ist, d. h. $0 \leq E$ ist eine Invariante der Schleife.

Es kann auch eine andere Grenze als 0 gewählt werden.

E kann auch monoton **vergrößert werden und nach oben begrenzt** sein.

Nichtterminierung wird bewiesen, indem man zeigt,
dass $R \wedge B$ eine Invariante der Schleife ist;
wobei R die Vorbedingung P der Schleife impliziert: $R \rightarrow P$.
R kann einen speziellen Zustand charakterisieren, in dem die Schleife nicht anhält.

Es gibt Schleifen, für die man **nicht entscheiden** kann, ob sie immer terminieren.

Vorlesung Modellierung WS 2001/2002 / Folie 317

Ziele:

Terminierungsnachweis verstehen

in der Vorlesung:

- Erläuterungen zu den Schritten.
- Der Ausdruck E braucht selbst nicht in der Schleife vorzukommen.
- Beispiele dazu.

Beispiele zur Terminierung

1. { $a > 0 \wedge b > 0$}
solange $a \neq b$ wiederhole
    solange $a > b$ wiederhole
        a := a - b;
    solange $a < b$ wiederhole
        b := b - a

terminiert

2. { $a > 0 \wedge b > 0$}
solange $a \neq b$ wiederhole
    solange $a \geq b$ wiederhole
        a := a - b;
    solange $a < b$ wiederhole
        b := b - a

terminiert nicht immer

3. { $n \in \mathbb{N} \wedge n > 1$ }
solange $n > 1$ wiederhole
    falls n gerade:
        n := n / 2
    sonst n := 3 * n + 1

Terminierung ist unbewiesen

Vorlesung Modellierung WS 2001/2002 / Folie 317a

Ziele:

Terminierungsnachweis üben

in der Vorlesung:

- 1 und 2 durch geeignete Invariante begründen
- 3 wird nicht versucht, zu entscheiden

Denksportaufgabe zu Invarianten

In einem Topf seien s schwarze und w weiße Kugeln, s + w > 0

solange mindestens 2 Kugeln im Topf sind

nimm 2 beliebige Kugeln heraus

falls sie gleiche Farbe haben:

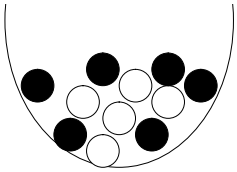
wirf beide weg und

lege eine neue schwarze Kugel in den Topf

falls sie verschiedene Farben haben:

lege die weiße Kugel zurück in den Topf und

wirf die schwarze Kugel weg



Welche Farbe hat die letzte Kugel?

Finden Sie Invarianten, die die Frage beantworten.

Vorlesung Modellierung WS 2001/2002 / Folie 318

Ziele:

Passende Invarianten finden

in der Vorlesung:

Lösung erfragen.

Schrittweise Konstruktion und Verifikation

Vorbedingung: $x \in \mathbb{R}$ und $n \in \mathbb{N}_0$

Nachbedingung: $q = x^n$

Konstruktionsidee:

Invariante INV: $x^n = q * a^i \wedge i \geq 0$

Zielbedingung: $i \leq 0$

falls i gerade: $x^n = q * (a^2)^{i/2}$

falls i ungerade: $x^n = q * a * (a^2)^{i/2}$

Algorithmus:

$\{ n \geq 0 \} \rightarrow \{ n = n \wedge n \geq 0 \wedge x = x \wedge 1 = 1 \}$

```
a := x; q := 1; i := n;
```

$\{ i = n \wedge i \geq 0 \wedge a = x \wedge q = 1 \} \rightarrow \{ INV \}$

```
solange i > 0 wiederhole
   { INV ∧ i > 0 }
   falls i ungerade: { INV ∧ i > 0 ∧ i ungerade } →
         { x^n = q * a * (a^2)^(i/2) ∧ i > 0 } q := q * a; { x^n = q * (a^2)^(i/2) ∧ i > 0 }
         leere Alternative für i gerade:
         { INV ∧ i > 0 ∧ i gerade } → { x^n = q * (a^2)^(i/2) ∧ i > 0 }
   { x^n = q * (a^2)^(i/2) ∧ i > 0 }
   a := a * a;
   { x^n = q * a^(i/2) ∧ i > 0 } → { x^n = q * a^(i/2) ∧ i/2 ≥ 0 }
   i := i / 2
   { x^n = q * a^i ∧ i ≥ 0 } ≡ { INV }
```

$\{ INV \wedge i \leq 0 \} \rightarrow \{ q = x^n \}$

Terminierung der Schleife: i fällt monoton und $i \geq 0$ ist invariant.

Schritte:

1. Vor-, Nachbedingung
2. Schleifeninvariante
3. Schleife mit INV
4. Initialisierung
5. Idee für Schleifenrumpf
6. Alternative
7. Schleife komplett
8. Terminierung

Vorlesung Modellierung WS 2001/2002 / Folie 319

Ziele:

Entwicklung eines vollständigen Beispiels

in der Vorlesung:

Erläuterung der einzelnen Schritte.

In der Datei verifikation.ps findet man die schrittweise Entwicklung des Inhaltes der Folie.

Prädikatenlogik

Die **Prädikatenlogik umfasst die Aussagenlogik**.
Esgibt nicht nur **atomare Aussagen**, die wahr oder falsch sind, wie „Es ist Nacht“.
Prädikatenlogik hat folgende **zusätzliche Konzepte**:

- parametrisierte Aussagen, wie „x ist eine Katze“,
  das ist ein 1-stelliges Prädikat, istKatze(x),
  es ist wahr oder falsch, je nach dem, wie x aus einem Individuenbereich gewählt wird.
  n-stellige Prädikate, z. B. teilt(a,b), größterGemeinsamerTeiler(a, b, g)
- **Quantoren** „**für alle x gilt P(x)**“ und „**es gibt ein x, so dass P(x) gilt**“
  in Symbolen: $\forall x P(x)$ bzw. $\exists x P(x)$
  Beispiel: $\forall x$ (esIstNacht $\wedge$ istKatze(x) $\rightarrow$ istGrau(x));
  in Worten: „Nachts sind alle Katzen grau.“
- **Terme als Parameter von Prädikaten**: $\forall x \forall y$ (gleichwertig(x + y, y + x))

Schon **zur Modellierung** einfacher Aufgaben reicht die Aussagenlogik nicht aus,
werden die **Erweiterungen der Prädikatenlogik gebraucht**:

gegeben: $a \in \mathbb{N}$, $b \in \mathbb{N}$

gesucht: größter gemeinsamer Teiler g von a und b, d. h.
teilt(g, a) $\wedge$ teilt(g, b) $\wedge$ ($\forall$h (teilt(h, a) $\wedge$ teilt(h, b) $\rightarrow$ h $\leq$ g))

Vorlesung Modellierung WS 2001/2002 / Folie 320

Ziele:

Motivation der Prädikatenlogik

in der Vorlesung:

- Parametrisierung von Aussagen verdeutlichen
- Prädikate und Relationen
- Quantoren intuitiv
- Einsatz in der Modellierung

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.2

Verständnisfragen:

Warum kann man einfache Aufgaben für algorithmische Berechnungen nicht mit Aussagenlogik modellieren?

Vorschau auf Begriffe

Ähnliche Folge von Begriffen wie in der Aussagenlogik:

- **prädikatenlogische Formeln** als Sprache der Prädikatenlogik
  Syntax: Terme, Prädikate, logische Junktoren, Quantoren
- **gebundene** und **freie Variable**
- **Individuenbereich**
- **Belegung** von Variablen mit Werten aus dem Individuenbereich
- **Interpretation**: Variablenbelegung und Definition der Funktionen und Prädikate
- **erfüllbar, allgemeingültig, unerfüllbar**
- **logischer Schluss**
- **Gesetze zum Umformen** von Formeln mit Quantoren

Ziele:
Übersicht

in der Vorlesung:
Vergleich mit Aussagenlogik

Prädikatenlogische Formeln

Prädikatenlogische Formeln werden induktiv wie folgt definiert:

1. **atomare Formeln** haben die Form **$P(t_1, ..., t_n)$**
   Dabei ist **P ein n-stelliges Prädikatsymbol** und
   die **t_i sind Terme zu einer Signatur Σ mit Variablen** aus der Menge X
   0-stellige Prädikatsymbole entsprechen den atomaren Aussagen der Aussagenlogik.
2. Sind F und G prädikatenlogische Formeln, so bilden auch **logischen Junktoren** prädikatenlogische Formeln: $\neg F$ $F \wedge G$ $F \vee G$
   Die Abkürzungen $F \rightarrow G$ und $F \equiv G$ werden übernommen.
3. Ist F eine prädikatenlogische Formel und x eine Variable aus V, so bilden auch der **Allquantor** $\forall$ und der **Existenzquantor** $\exists$ prädikatenlogische Formeln: $\forall x\ F$ und $\exists x\ F$

Nur nach (1. - 3.) gebildete Formeln sind **syntaktisch korrekte prädikatenlogische Formeln**.

Bemerkungen:

- Meist wird die **Signatur Σ nicht explizit angegeben**, sondern aus den Operationen angenommen, die in den Termen verwendet werden.
- Ebenso werden die **Prädikate häufig aus den verwendeten Symbolen** angenommen.
- Dies ist **Prädikatenlogik erster Stufe**: **Variable** sind nur als Operanden in Termen erlaubt, aber **nicht für Funktionen oder für Prädkate**.

Vorlesung Modellierung WS 2001/2002 / Folie 322

Ziele:

Notation und Struktur

in der Vorlesung:

- Beispiele
- Struktur an Bäumen erläutern
- Signatur explizit machen
- PL erster Stufe abgrenzen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.2

Verständnisfragen:

- Zeigen Sie, dass die Formeln auf Folie Mod-3.20 syntaktisch korrekt sind.
- Zu welchen Signaturen gehören ihre Terme?

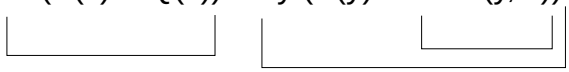
Gebundene und freie Variable

Ein Quantor ∀x F oder ∃x F **bindet alle Vorkommen des Variablennamens x** in der Formel F, außer den Vorkommen von x die durch einen weiteren Quantor innerhalb von F gebunden sind.

x ist der **Name der Variablen des Quantors**.

F ist der der **Wirkungsbereich des Quantors**.

Beispiel: $\forall x\,(P(x) \wedge Q(x)) \vee \exists y\,(P(y) \wedge \forall z\,R(y, z))$



Quantoren mit ihrenWirkungsbereichen

Ein Vorkommen eines Variablennamens y heißt **frei in der Formel F**, wenn es nicht im Wirkungsbereich eines Quantors für y liegt.
y ist dann der Name einer **freien Variablen** der Formel F.

Beispiel: Formel F

$x < 0 \wedge \exists y\,(P(y, x) \vee Q(y, z))$

freie Vorkommen

F hat 2 freie Variable. Sie haben die Namen x und z.

Vorlesung Modellierung WS 2001/2002 / Folie 323

Ziele:

Bindung von Namen verstehen

in der Vorlesung:

- Wirkungsbereiche erläutern, und an Bäumen zeigen
- Variablendefinition und ihre Anwendungen
- Unterschied: Variable, ihr Name, dessen Vorkommen in einer Formel
- Vergleich mit Variablendeklarationen in Programmen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.2

Umbenennung von Variablen

In einer Formel können mehrere Vorkommen von Quantoren **verschiedene Variable mit gleichem Namen** einführen und in ihrem Wirkungsbereich binden:

Beispiele:

$\forall y\ (\exists x\ R(x, y) \wedge \exists x\ Q(x, y))$ $\qquad$ $\forall x\ \forall y\ (P(x, y) \wedge \exists x\ R(x, y))$

Umbenennung: In einer Formel kann man **alle Vorkommen des Namens x einer Variablen durch einen neuen Namen z ersetzen**, der sonst nicht in der Formel vorkommt. Die Bedeutung der Formel ändert sich dadurch nicht.

Beispiele von oben:

$\forall y\ (\exists x\ R(x, y) \wedge \exists z\ Q(z, y))$ $\qquad$ $\forall x\ \forall y\ (P(x, y) \wedge \exists z\ R(z, y))$

Damit kann man erreichen, dass **verschiedene Variable verschiedene Namen** haben.

Wir sagen dann: Die Variablen der Formel sind **konsistent umbenannt**.

Formeln, in denen alle Variablen verschiedene Namen haben sind meist besser lesbar.

Manche Definitionen sind einfacher für konsistent umbenannte Formeln.

Vorlesung Modellierung WS 2001/2002 / Folie 324

Ziele:

Konsistente Umbenennung verstehen

in der Vorlesung:

- Anwendungen ihren Definitionen zuordnen
- Verdeckung in geschachtelten Wirkungsbereichen
- Substitution beachtet Bindungen nicht
- Vergleich mit Programmiersprachen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.2

Verständnisfragen:

- Geben Sie ein Beispiel, wo eine unzulässige Umbenennung in einen schon verwendeten Variablennamen Bindungen verändert.

Interpretation prädikatenlogischer Formeln

Einer prädikatenlogischen Formel wird durch eine **Interpretation $\mathfrak{I}$ ein Wahrheitswert** wahr oder falsch (w oder f) zugeordnet. $\mathfrak{I}$ wird in drei Schritten definiert; hier der erste:

Eine **Interpretation** $\mathfrak{I}$ wird bestimmt durch

- einen **Individuenbereich U,** der nicht leer ist (auch Universum genannt),
- eine **Abbildung der Funktions- und Prädikatsymbole** einer Signatur Σ auf dazu passende konkrete Funktionen und Prädikate, notiert z. B. $\mathfrak{I}(f)$, $\mathfrak{I}(P)$
- eine **Belegung von Variablen mit Werten aus U**, notiert z. B. $\mathfrak{I}(x)$

Bemerkungen:

- Aus dem **Individuenbereich U** stammen die **Werte der Variablen und Terme**.
- In der Prädikatenlogik enthält der **Individuenbereich U alle Individuen** - **auch verschiedenartige** - die für die Interpretation benötigt werden.
  Er ist **nicht in Wertebereiche gleichartiger Individuen** strukturiert (wie in Abschnitt 2).
- **Jede Sorte aus** Σ wird deshalb **auf den ganzen Individuenbereich U** abgebildet.
- Eine **Interpretation** wird immer **passend zu einer Menge prädikatenlogischer Formeln** definiert. Nur darin vorkommende Funktionen, Prädikate und Variable interessieren.

Vorlesung Modellierung WS 2001/2002 / Folie 325

Ziele:

Interpretation schrittweise verstehen

in der Vorlesung:

- Beispiele für Individuenbereiche U
- Vergleich mit Wertebereichen aus Abschnitt 2
- Beispiel von Mod-3.26

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.2

Beispiel für eine Interpretation einer Formel

Zur Formel $F = \forall x\, P(x, f(x)) \wedge Q(g(a, z))$ ist folgendes $\Im$ eine passende Interpretation:

U	:= $\mathbb{N}_0$	
$\Im(P)$	:= { (m, n) | m, n $\in$ U und m < n }	
$\Im(Q)$	:= { n | n $\in$ U und n ist Primzahl }	
$\Im(f)$	ist die Nachfolgerfunktion auf U, also $\Im(f)(n) = n + 1$	
$\Im(g)$	ist die Additionsfunktion auf U also $\Im(g)(m, n) = m + n$	
$\Im(a)$	:= 2	(a ist eine Konstante)
$\Im(z)$	:= 3	(z ist eine Variable)

Bemerkungen:

- Hier brauchen nur den freien Variablen der Formel Werte zugeordnet zu werden. Der nächste Schritt der Definition von $\Im$ zeigt, wie die Variablen der Quantoren Werte erhalten.
- Häufig wird die Interpretation von Funktions- und Prädikatssymbolen nicht explizit angegeben, sondern die „übliche Bedeutung der Symbole“ angenommen.

Das Beispiel stammt aus
U. Schöning: Logik für Informatiker, Spektrum Akademischer Verlag, 4. Aufl., 1995, S. 55

Vorlesung Modellierung WS 2001/2002 / Folie 326

Ziele:

Konkretes Beispiel verstehen

in der Vorlesung:

- Beispiel erläutern
- Andere Funktionen einsetzen

Verständnisfragen:

Variieren Sie das Beispiel mit anderen Funktionen und Prädikaten.

Interpretation von Termen

Zweiter Schritt: Erweiterung der **Definition der Interpretation auf Terme**:

Sei F eine Prädikatenlogische Formel und $\Im$ eine dazu passende Interpretation.
Sei t ein **Term, der aus Funktionssymbolen und Variablen** gebildet ist, die in F vorkommen.

Dann bildet $\Im$ den Term t auf einen **Wert $\Im$(t) aus dem Individuenbereich U** ab:

- falls t eine **Variable** x ist: $\Im(t) = \Im(x)$
  d. h. der Wert des Terms ist die Belegung der Variablen x
- falls t die Form hat f (t_1, ..., t_n): $\Im(t) = \Im(f)\ (\Im(t_1), ..., \Im(t_n))$
  d. h. $\Im$ bestimmt die Funktion zu f,
  sie wird auf die Werte der Argumentterme angewandt,
  das Ergebnis bestimmt den Wert von $\Im$(t)

  f ist **ein n-stelliges Funktionssymbol**, $n \geq 0$;
  0-stellige Funktionen sind **Konstante**

Beispiel für einen Term t = f(g(a, z))
mit der Interpretation von Folie Mod-3.26: $\Im(f(g(a, z))) = 6$

Bemerkung:

- Wie $\Im$ den Wert von freien Variablen bestimmt, ist schon definert;
  für Variable, die durch Quantoren gebunden sind folgt die Definition im dritten Schritt.

Vorlesung Modellierung WS 2001/2002 / Folie 327

Ziele:

Werte von Termen bestimmen

in der Vorlesung:

- Anwendung der Interpretation aus Mod-3.26 für Terme zeigen
- Verschiedene Interpretationen für dieselben Terme

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.2

Wahrheitswerte prädikatenlogischer Formeln

Dritter Schritt:
Erweiterung der Definition der Interpretation $\Im$ für **Prädikate, Junktoren und Quantoren**.

Sei F eine prädikatenlogische Formel. Dann bestimmt $\Im$**(F) einen Wahrheitswert** gemäß einer der folgenden Regeln, je nach der Form von F:

1. **n-stelliges Prädikat** P: $\Im(P(t_1, ..., t_n)) = w$ genau dann, wenn $\Im(P)\ (\Im(t_1), ..., \Im(t_n)) = w$

2. **logische Junktoren**: $\Im(\neg G) = w$ genau dann, wenn $\Im(G) = f$
   $\Im(G \wedge H) = w$ genau dann, wenn $\Im(G) = w$ und $\Im(H) = w$
   $\Im(G \vee H) = w$ genau dann, wenn $\Im(G) = w$ oder $\Im(H) = w$

3. **Quantoren**: $\Im(\forall x\ G) = w$ genau dann, wenn **für jeden Wert d** $\in$ **U** gilt $\Im_{[d/x]}(G) = w$

   $\Im(\exists x\ G) = w$ genau dann, wenn es **einen Wert d** $\in$ **U** gibt mit $\Im_{[d/x]}(G) = w$

Die Bedeutung der Quantoren wird in (3.) durch Bezug auf alle mit U möglichen Belegungen der gebundenen Variable definiert: $\Im_{[d/x]}$ bezeichnet eine „Variante der Interpretation $\Im$“:
$\Im_{[d/x]}$ ordnet der Variablen x den Wert d zu und stimmt sonst mit $\Im$ überein.

Wir nehmen an, dass alle Variablen in F verschiedene Namen haben (konsistent umbenannt). Dann betrifft $\Im_{[d/x]}$ nur die Variable des gerade betrachteten Quantors und keine andere.

Vorlesung Modellierung WS 2001/2002 / Folie 328

Ziele:

Interpretation vollständig definiert

in der Vorlesung:

- Die Regeln 1 und 2 erläutern
- Interpretation der Quantoren (3) erläutern; intuitiv und formal
- Vorsicht mit der Notation [d / x].

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.2

Beschränkung von Wertebereichen

In der **Modellierung** soll man die **Wertebereich**e der Variablen präzise angeben.
In der **Prädikatenlogik** können jedoch die **Variablen der Quantoren Werte** aus dem **gesamten Individuenbereich U** annehmen.
Deshalb müssen **Einschränkung explizit als Prädikat** formuliert werden.

Beschränkung des Wertebereiches bei Allquantoren durch Implikation $\rightarrow$:
„für alle $m \in M \subseteq U$ gilt P(m)“:

	ausführliche Notation:	abkürzende Notation:
allgemein:	$\forall m\ (m \in M \rightarrow P(m))$	$\forall m \in M: P(m)$
Beispiele:	$\forall i\ (i \in \{1, 2, 3, 4\} \rightarrow b_i = a_i^2)$	$\forall i \in \{1, 2, 3, 4\}: b_i = a_i^2$
	$\forall k\ (k \in \mathbb{N} \rightarrow a + k \geq a)$	$\forall k \in \mathbb{N}: a + k \geq a$

Beschränkung des Wertebereiches bei Existenzquantoren durch Konjunktion $\wedge$:
„es gibt ein $m \in M \subseteq U$, so dass P(m) gilt“:

allgemein:	$\exists m\ (m \in M \wedge P(m))$	$\exists m \in M: P(m)$
Beispiele:	$\exists k\ (k \in \mathbb{N} \wedge a * k = b)$	$\exists k \in \mathbb{N}: a * k = b$
	$\exists i\ (i \in \{1, 2, 3, 4\} \wedge a_i = x)$	$\exists i \in \{1, 2, 3, 4\}: a_i = x$

Ziele:

Prädikate präzisieren Wertebereiche

in der Vorlesung:

- Begründung der Implikation und der Konjunktion
- Erläuterung der Beispiele

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.2

Beispiele für prädikatenlogische Formeln

Wichtige Eigenschaften 2-stelliger Relationen $R \subseteq M \times M$: (siehe Mod-2.10)

- **reflexiv**: wenn für alle $x \in M$ gilt $x\ R\ x$, d. h. $(x, x) \in R$

  reflexiv(R): $R \subseteq M \times M \wedge \forall x\ (x \in M \rightarrow (x, x) \in R)$

  abgekürzt: $R \subseteq M \times M \wedge \forall x \in M: (x, x) \in R$

- **symmetrisch**, wenn für alle $x, y \in M$ gilt: aus $x\ R\ y$ folgt $y\ R\ x$

  symmetrisch(R): $R \subseteq M \times M \wedge \forall x\ (x \in M \rightarrow \forall y\ (y \in M \rightarrow ((x, y) \in R \rightarrow (y, x) \in R)))$

  abgekürzt: $R \subseteq M \times M \wedge \forall x \in M: \forall y \in M: (x, y) \in R \rightarrow (y, x) \in R$

Die Variablen in Axiomen abstrakter Algebren sind durch Allquantoren gebunden:

- Folie Mod-2.27: K3: $pop\ (push\ (k, t)) \equiv k$

  gebunden: K3: $\forall k\ (k \in Keller \rightarrow \forall t\ (t \in Element \rightarrow pop\ (push\ (k, t)) \equiv k))$

  abgekürzt: K3: $\forall k \in Keller: \forall t \in Element: pop\ (push\ (k, t)) \equiv k$

- Prädikat „kleinstes Element eine Menge“:
  istKleinstes(k, M): $k \in M \wedge \forall i\ (i \in M \rightarrow k \leq i)$

Vorlesung Modellierung WS 2001/2002 / Folie 330

Ziele:

Prädikatenlogik lesen lernen

in der Vorlesung:

- Erläuterung der Beispiele
- Formalisierung der Axiome

Beispiele für prädikatenlogische Formeln (2)

- Analysis:
  Eine Folge f(n) heißt konvergent mit dem Grenzwert a, wenn gilt

  $$\forall \varepsilon \in \mathbb{R}: \exists n_\varepsilon \in \mathbb{N}: \forall n \in \mathbb{N}: n > n_\varepsilon \rightarrow | f(n) - a | < \varepsilon$$

- Informatik:
  Eine Folge $a = (a_1, ..., a_n) \in \mathbb{N}^+$ heißt (nicht fallend) geordnet, wenn gilt

  $$\forall i \in \{1, ..., n\}: \forall j \in \{1, ..., n\}: i \leq j \rightarrow a_i \leq a_j$$

- Was bedeutet folgende Aufgabenstellung?

  gegeben: Eine Folge $a = (a_1, ..., a_n) \in \mathbb{N}^+$

  gesucht: $p \in Pos = \{1, ..., n\}$, so dass $\forall j \in Pos: a_p \leq a_j \wedge (a_p = a_j \rightarrow p \leq j)$

Vorlesung Modellierung WS 2001/2002 / Folie 330a

Ziele:

Weitere konkrete Beispiele

in der Vorlesung:

- Erläuterungen dazu

Verständnisfragen:

Gibt es zum letzten Beispiel

- immer eine Antwort,
- ist sie immer eindeutig bestimmt?

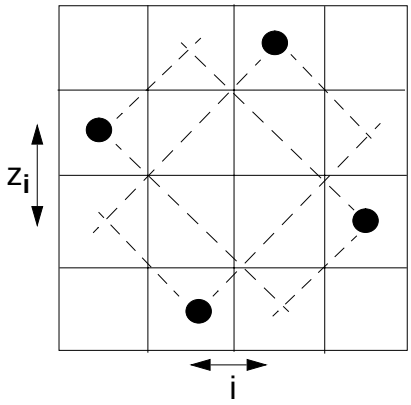
Beispiel: Spezifikation des n-Damen-Problems

gegeben:

Kantenlänge $n \in \mathbb{N}$ eines n * n Schachbrettes

gesucht:

Menge P zulässiger Platzierungen von jeweils n Damen auf dem Schachbrett, so dass keine Dame eine andere nach Schachregeln schlägt:



Sei Index := {1, ..., n}

$P := \{ p \mid p = (z_1, ..., z_n) \in Index^n \wedge zulässig(p) \}$

z_i gibt die Zeilennummer der Dame in Spalte i an.

Dabei bedeutet

zulässig (p): $\forall\, i \in Index: \forall\, j \in Index: i \neq j \rightarrow z_i \neq z_j \wedge |z_i - z_j| \neq |i - j|$

Vorlesung Modellierung WS 2001/2002 / Folie 330b

Ziele:

Spezifikation einer Aufgabe

in der Vorlesung:

Erläuterungen dazu

Begriffe zur Interpretation prädikatenlogischer Formeln

Die folgenden Begriffe sind in der Prädikatenlogik so definiert wie in der Aussagenlogik.
Aber: Interpretationen der Prädikatenlogik sind komplexe Strukturen.
Deshalb sind die Eigenschaften „**erfüllbar**“ und „**allgemeingültig**“ für prädikatenlogische Formeln **nicht allgemein entscheidbar**.

Wenn $\Im(F) = w$ gilt, heißt $\Im$ auch ein **Modell der Formel F.**

Eine Formel F heißt **erfüllbar**, wenn es eine Interpretation $\Im$ gibt, so dass gilt $\Im$ (F) = w, sonst ist sie **unerfüllbar** (**widerspruchsvoll**).

Eine Formel F heißt **allgemeingültig** oder **Tautologie**, wenn für alle Interpretationen von F $\Im$ (F) = w gilt.

Eine Formel F ist genau dann allgemeingültig, wenn $\neg$ F unerfüllbar ist.

Zwei Formeln F, G sind **logisch äquivalent**, wenn sie für alle Interpretationen $\Im$ dasselbe Ergebnis haben: $\Im(F) = \Im(G)$

Sei A eine Menge von Formeln und F eine Formel.
Wenn für **alle Interpretationen** $\Im$, die alle Formeln in A erfüllen, auch $\Im$ (F) gilt, dann sagen wir **„F folgt semantisch aus A“** bzw. **A |= F**;
A |= F heißt auch **logischer Schluss**.

Vorlesung Modellierung WS 2001/2002 / Folie 331

Ziele:

Analoge Begriffe wie in der Aussagenlogik

in der Vorlesung:

- gleiche Definition der Begriffe
- Interpretation unterscheidet sich von der in der Aussagenlogik
- Die Menge der Interpretationen kann i.a. nicht überprüft werden.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.2

Äquivalente Umformung prädikatenlogischer Formeln

Seien F und G beliebige prädikatenlogische Formel. Dann gelten folgende **Äquivalenzen**:

1. $\neg\forall x\ F \equiv \exists x\ \neg F$ $\qquad$ $\neg\exists x\ F \equiv \forall x\ \neg F$

2. Falls x in G nicht frei vorkommt, gilt

   $(\forall x\ F \wedge G) \equiv \forall x\ (F \wedge G)$ $\qquad$ $(\forall x\ F \vee G) \equiv \forall x\ (F \vee G)$

   $G \equiv \forall x\ G$

   $(\exists x\ F \wedge G) \equiv \exists x\ (F \wedge G)$ $\qquad$ $(\exists x\ F \vee G) \equiv \exists x\ (F \vee G)$

   $G \equiv \exists x\ G$

3. $(\forall x\ F \wedge \forall x\ G) \equiv \forall x\ (F \wedge G)$ $\qquad$ $(\exists x\ F \vee \exists x\ G) \equiv \exists x\ (F \vee G)$

4. $\forall x\ \forall y\ F \equiv \forall y\ \forall x\ F$ $\qquad$ $\exists x\ \exists y\ F \equiv \exists y\ \exists x\ F$

Im allgemeinen **nicht äquivalent** sind folgende Formelpaare:

$(\forall x\ F \vee \forall x\ G) \not\equiv \forall x\ (F \vee G)$ $\qquad$ $(\exists x\ F \wedge \exists x\ G) \not\equiv \exists x\ (F \wedge G)$

Vorlesung Modellierung WS 2001/2002 / Folie 332

Ziele:

Wichtige Äquivalenzen einprägen

in der Vorlesung:

- 1 - 4 begründen
- Eine Äquivalenz beweisen
- Beispiel für Umformungen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 4.2

Beispiel für Umformungen

Die folgende prädikatenlogische Formel wird so umgeformt, dass alle Quantoren vorne (außen) stehen:

	$(\neg(\exists x\ P(x, y) \vee \forall z\ Q(z)) \wedge \exists w\ P(f(a, w)))$	
$\equiv$	$((\neg\exists x\ P(x, y) \wedge \neg\forall z\ Q(z)) \wedge \exists w\ P(f(a, w)))$	DeMorgan
$\equiv$	$((\forall x\ \neg P(x, y) \wedge \exists z\ \neg Q(z)) \wedge \exists w\ P(f(a, w)))$	Negation von Quantorformeln (1.)
$\equiv$	$(\exists w\ P(f(a, w)) \wedge (\forall x\ \neg P(x, y) \wedge \exists z\ \neg Q(z)))$	Kommutativität
$\equiv$	$\exists w\ (P(f(a, w)) \wedge \forall x\ (\neg P(x, y) \wedge \exists z\ \neg Q(z)))$	Wirkungsbereiche ausweiten (2.)
$\equiv$	$\exists w\ (\forall x\ (\exists z\ \neg Q(z) \wedge \neg P(x, y)) \wedge P(f(a, w)))$	Kommutativität
$\equiv$	$\exists w\ (\forall x\ \exists z\ (\neg Q(z) \wedge \neg P(x, y)) \wedge P(f(a, w)))$	Wirkungsbereiche ausweiten (2.)
$\equiv$	$\exists w\ \forall x\ \exists z\ (\neg Q(z) \wedge \neg P(x, y) \wedge P(f(a, w)))$	Wirkungsbereiche ausweiten (2.)

In diesem Beispiel hätten die Quantoren auch in anderer Reihenfolge enden können, wenn in anderer Reihenfolge umgeformt worden wäre. Das ist nicht allgemein so.

Jede prädikatenlogische Formel kann durch Anwenden der Äquivalenzen und Umbenennung von Variablen in die Form gebracht werden $Q_1x_1\ Q_2x_2\ ...\ Q_nx_n\ F$ (**Pränexform**) wobei Q_i Quantoren sind und F keine Quantoren enthält.

Vorlesung Modellierung WS 2001/2002 / Folie 332a

Ziele:

Äquivalenzen anwenden

in der Vorlesung:

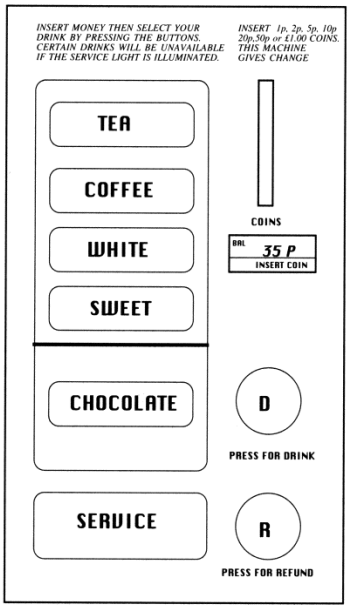
Erläuterungen dazu

Anwendungsstellen zeigen,

Ausschnitt aus einer Spezifikation in Z

Die **Spezifikationssprache Z** basiert auf typisierter Mengentheorie (Wertebereiche wie in Abschnitt 2) und verwendet Prädikatenlogik.

Ausschnitt aus der Fallstudie „A Drinks Dispensing Machine“ aus
Deri Sheppard: An Introduction to Formal Specification with Z and VDM, McGraw-Hill, 1994, S. 271ff



Get_Drink

$\Delta Abs\_State\_Machine$
$choice? : \mathbb{P} Selection\_buttons$
$d! : Drink$
$Change! : \text{bag } British\_coin$

$choice? \in Drink$
$Value\ Balance \geq Prices\ choice?$
$\forall i : Recipe\ choice? \bullet count\ Stock\ i > 0$
$Cups > 0$
$\exists b : bag\ British\_coins \bullet (b \sqsubseteq Takings \land Value\ Balance = Value\ b + Prices\ choice?)$
$Balance' = [\![\,]\!]$
$Stock' \uplus \{i : Recipe\ choice? \bullet i \mapsto 1\} = Stock$
$Cups' = Cups - 1$
$Change! \sqsubseteq Takings \land Value\ Balance = Value\ Change! + Prices\ choice?$
$Takings' \uplus Change! = Takings$
$Prices' = Prices$
$Service\_light' = Service\_light$
$Report\_display' = insert\ coin$
$d! = choice?$

Vorlesung Modellierung WS 2001/2002 / Folie 333

Ziele:

Ausschnitt aus einem größeren Beispiel

in der Vorlesung:

Erläuterungen zur

- Sprache Z,
- zur gestellten Aufgabe,
- zum Ausschnit aus der Spezifikation

4. Modellierung mit Graphen

Modellierung beschreibt **Objekte und Beziehungen** zwischen ihnen.

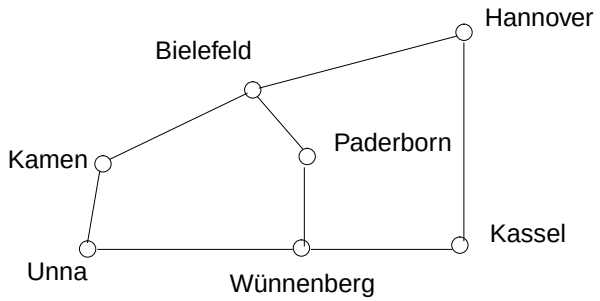
Graphen eignen sich zur Modellierung für ein **breites Aufgabenspektrum**.

Ein **Graph** ist eine Abstraktion aus Knoten und Kanten:

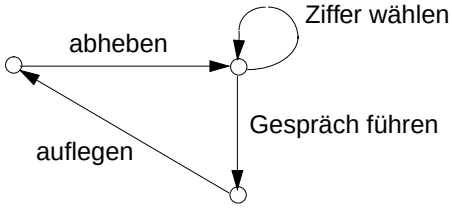
- **Knoten**: Eine Menge gleichartiger Objekte
- **Kanten**: Beziehung zwischen je zwei Objekten, 2-stellige Relation über Knoten

Je nach Aufgabenstellung werden **ungerichtete oder gerichtete** Graphen verwendet.

ungerichtet



gerichtet



Beschränkung auf **endliche Knotenmengen** und **2-stellige** Relation reicht hier aus.

Vorlesung Modellierung WS 2001/2002 / Folie 401

Ziele:

Intuitives Verständnis von Graphen

in der Vorlesung:

Erläuterung der Begriffe und Beispiele

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Verständnisfragen:

Geben Sie weitere Beispiele für Graphen an.

Themenübersicht

4.1 Grundlegende Definitionen
gerichteter, ungerichteter Graph, Graphdarstellungen, Teilgraphen, Grad, Markierungen

4.2 Wegeprobleme
Weg, Kreis, Rundwege, Zusammenhang

4.3 Verbindungsprobleme
Spannbaum

4.4 Modellierung mit Bäumen
gewurzelte Bäume, Entscheidungsbäume, Strukturbäume, Kantorowitsch-Bäume

4.5 Zuordnungsprobleme
konfliktfreie Markierung, bipartite Graphen

4.6 Abhängigkeitsprobleme
Anordnungen, Abfolgen

4.7 Flussprobleme
Potenzial, Fluss, Kapazität

Vorlesung Modellierung WS 2001/2002 / Folie 402

Ziele:

Vielfalt der Anwendungen von Graphen

in der Vorlesung:

- Eindruck der Aufgabenbereiche vermitteln,
- Beispiele skizzieren,
- gerichtete und ungerichtete Graphen zuordnen.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

4.1 Grundlegende Definitionen
Gerichteter Graph

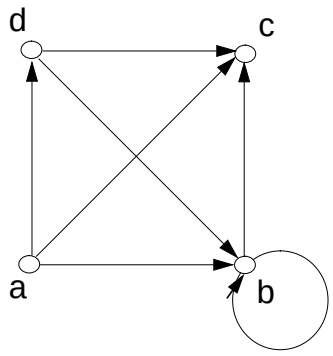
Ein **gerichteter Graph** G = (V, E) hat eine endliche **Menge V von Knoten** und eine **Menge E gerichteter Kanten**, mit $E \subseteq V \times V$.

Die Kantenmenge E ist eine **2-stellige Relation** über V.

Beispiel:

V = {a, b, c, d}

E = {(a, b), (a, c), (a, d), (b, b), (b, c), (d, b), (d, c)}



Eine Kante wird als (v, u) oder v -> u notiert.

Eine Kante (v, v) heißt **Schleife** oder Schlinge.

Die Definition schränkt eine auf

- endliche Graphen mit **endlichen Knotenmengen**,
- einfache Kanten: eine **Kante verbindet nicht mehr als 2 Knoten**, **zwischen zwei Knoten gibt es höchstens eine Kante**.

Vorlesung Modellierung WS 2001/2002 / Folie 403

Ziele:

Gerichteten Graph als Relation verstehen

in der Vorlesung:

Erläuterung der Begriffe.

Synonyme:

- Knoten: auch Ecke, engl. vertex
- Kante: engl. edge
- gerichtete Kante: engl. arc

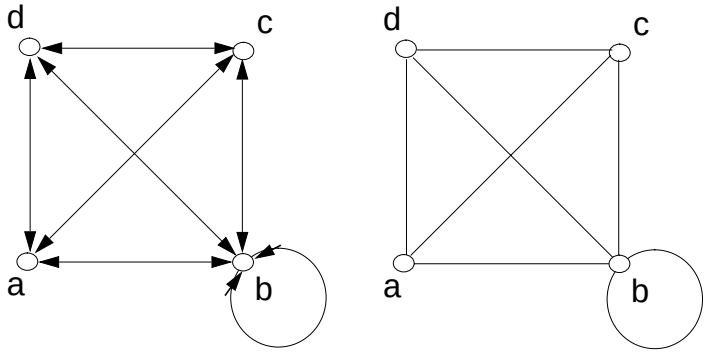
nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Ungerichteter Graph

Ist die **Kantenmenge** E eines gerichteten Graphen eine **symmetrische Relation**, so beschreibt er einen **ungerichteten Graphen**:
Zu jeder Kante x -> y aus E gibt es auch y -> x in E.

Wir fassen zwei Kanten x -> y, y -> x zu einer **ungerichteten Kante** zusammen:
{x, y} die Menge der Knoten, die die Kante verbindet.



Ungerichtete Graphen werden auch direkt definiert:

Ein **ungerichterter Graph** G = (V, E) hat eine **Menge V von Knoten** und eine **Menge E ungerichteter Kanten, mit E $\subseteq$ { {x, y} | x, y $\in$ V}**

Der abgebildete Graph mit ungerichteten Kanten:
V = {a, b, c, d} E = {{a, b}, {a, c}, {a, d}, { b }, {b, c}, {d, b}, {d, c} }

In dieser Notation ist eine Schleife eine 1-elementige Menge, z. B. { b }

Vorlesung Modellierung WS 2001/2002 / Folie 404

Ziele:

Zusammenhang zwischen gerichtetem und ungerichtetem Graph

in der Vorlesung:

- Zusammenhang erläutern,
- Definition gegenüberstellen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Darstellung von Graphen

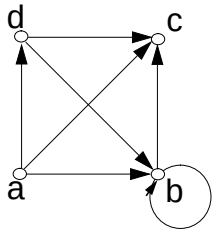
abstrakt: Knotenmenge, Kantenmenge

anschaulich: Graphik

Datenstruktur für
algorithmische Berechnungen:

V = {a, b, c, d}

E = {(a, b), (a, c), (a, d),
(b, b), (b, c),
(d, b), (d, c)}



Knotenmenge V als Indexmenge;
lineare Ordnung der Knoten definieren;
sei |V| = n

a, b, c, d

Darstellung der (gerichteten) Kanten:

a. **Adjazenzmatrix** AM, n * n Wahrheitswerte

AM(i, j) = (i, j) ∈ E

	a	b	c	d
a	f	w	w	w
b	f	w	w	f
c	f	f	f	f
d	f	w	w	f

b. **Adjazenzlisten**: zu jedem Knoten i eine Folge von Knoten, zu denen er eine Kante hat (i, j) ∈ E

a	(b, c, d)
b	(b, c)
c	()
d	(b, c)

Ungerichtete Graphen als gerichtete mit symmetrischer Kantenmenge darstellen

Vorlesung Modellierung WS 2001/2002 / Folie 405

Ziele:

Datenstrukturen für Graphen kennenlernen

in der Vorlesung:

Erläuterungen zu den beiden Darstellungen

- Adjazenzmatrix: direkter Zugriff aber redundant
- Adjazenzlisten: kompakt aber Suche nach Kanten.

nachlesen:

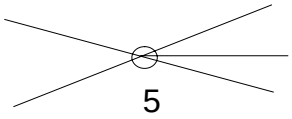
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Teilgraph, Knotengrad

Seien G = (V, E) und G' = (V', E') gerichtete (bzw. ungerichtete) Graphen und seien $V' \subseteq V$ und $E' \subseteq E$, dann heißt G' **Teilgraph von G**.

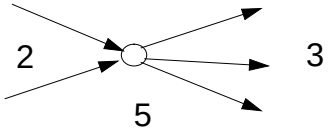
Zu einem Graphen G = (V, E) **induziert** eine Teilmenge der Knoten $V' \subseteq V$ den **Teilgraphen** G' = (V', E'), wobei E' alle Kanten aus E enthält, deren Enden in V' liegen.

Sei G = (V, E) ein **ungerichteter** Graph:



Der **Grad** eines Knotens v ist die Anzahl der Kanten {x, v}, die in v enden.

Sei G = (V, E) ein **gerichteter** Graph:



Der **Eingangsgrad** eines Knotens v ist die Anzahl der Kanten $(x, v) \in E$, die in v münden.

Der **Ausgangsgrad** eines Knotens v ist die Anzahl der Kanten $(v, x) \in E$, die von v ausgehen.

Der Grad eines Knotens v ist die Summe seines Eingangs- und Ausgangsgrades.

Der Grad eines gerichteten oder ungerichteten Graphen ist der maximale Grad seiner Knoten.

Vorlesung Modellierung WS 2001/2002 / Folie 406

Ziele:

Begriffe verstehen

in der Vorlesung:

Erläuterungen und Beispiele dazu

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Markierte Graphen

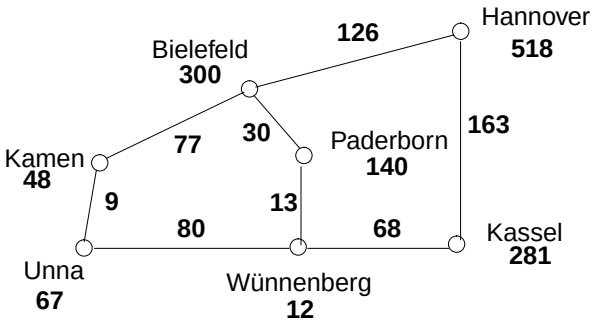
Ein Graph G = (V, E) modelliert eine Menge von Objekten V und die Existenz von Beziehungen zwischen ihnen.

Viele Aufgaben erfordern, dass den **Knoten und/oder den Kanten weitere Informationen** zugeordnet werden.

Dies leisten **Markierungsfunktionen**

Knotenmarkierung MV : V $\rightarrow$ WV,
z.B. Einwohnerzahl: V $\rightarrow \mathbb{N}$

Kantenmarkierung ME : E $\rightarrow$ WE,
z.B. Entfernung: E $\rightarrow \mathbb{N}$



Spezielle Kantenmarkierungen:

Ordnung: E $\rightarrow \mathbb{N}$ legt die **Reihenfolge der Kanten** fest, die von einem Knoten ausgehen (z.B. im Kantorowitsch-Baum siehe Mod-2.19)

Anzahl: E $\rightarrow \mathbb{N}$ modelliert **mehrfache Verbindungen zwischen denselben Knoten**; G ist dann ein **Mehrfachgraph (Multigraph)**.
In der graphischen Darstellung schreibt man die Anzahl an die Kante oder zeichnet mehrere Kanten.

Vorlesung Modellierung WS 2001/2002 / Folie 407

Ziele:

Markierungen verstehen

in der Vorlesung:

Begriffe und Beispiele erläutern

nachlesen:

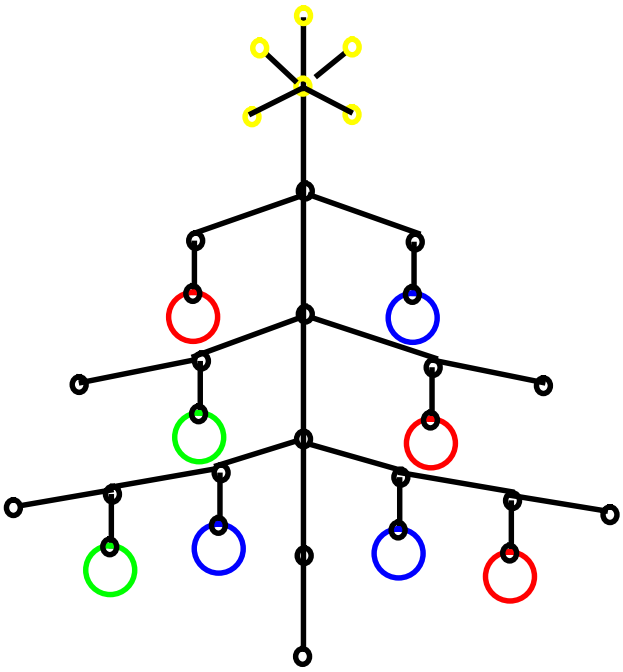
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Verständnisfragen:

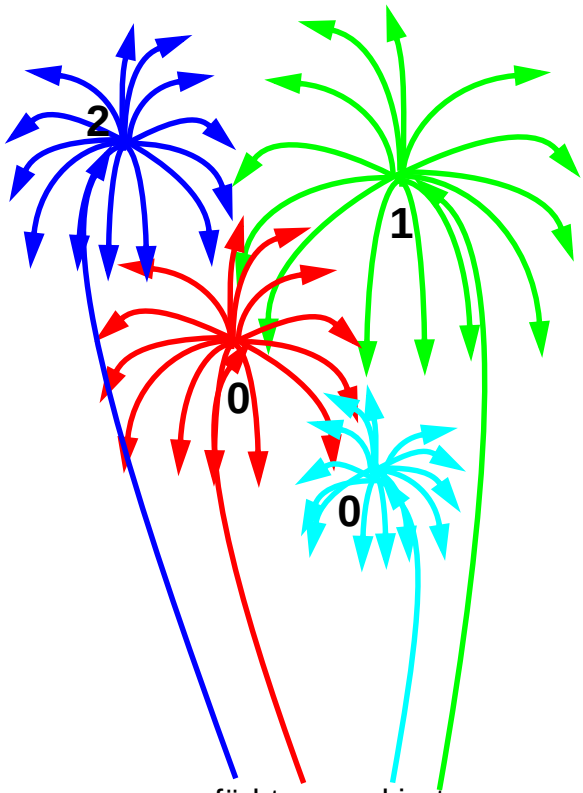
Geben Sie weitere Beispiel für Markierungen

Mod-4.xmas

Jahreszeitgemäße Spezialgraphen



gewurzelter, dekorierter Baum



gefärbter, markierter
Multi-Explosionsgraph

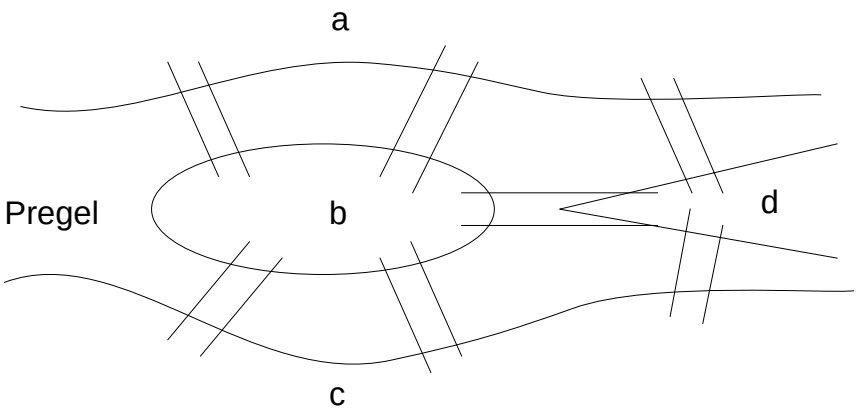
Vorlesung Modellierung WS 2001/2002 / Folie 407x

Ziele:

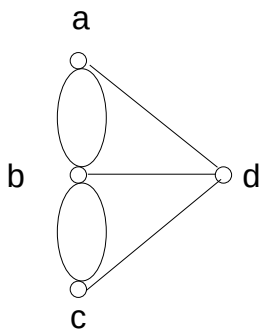
Frohe Weihnachten und ein gutes Neues Jahr!

4.2 Wegeprobleme

Beispiel: Königsberger Brückenproblem (Euler, 1736)



Skizze von Königsberg



Multigraph dazu

a. Gibt es einen Weg, der jede der 7 Brücken genau einmal überquert und zum Ausgangspunkt zurückkehrt?

b. Gibt es einen Weg, der jede der 7 Brücken genau einmal überquert?

Vorlesung Modellierung WS 2001/2002 / Folie 408

Ziele:

Berühmtes Modellierungsbeispiel

in der Vorlesung:

- Modellierung zeigen
- Lösung erarbeiten

nachlesen:

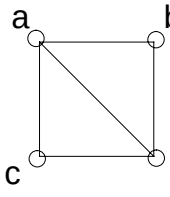
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Verständnisfragen:

Hinweis:

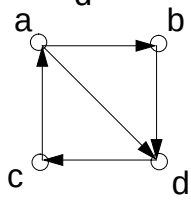
- a: Begründen Sie Ihre Antwort mit dem Grad der Knoten auf solch einem Rundweg.
- b: Für die Knoten, die nicht Endpunkte sind, gilt das Gleiche wie in (a).

Wege in Graphen



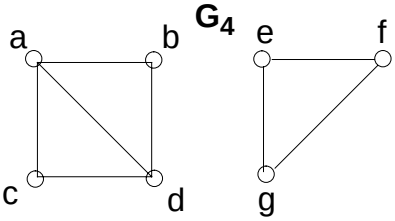
G_1

Sei G = (V, E) ein ungerichteter Graph.
Eine **Folge von Knoten (v_0, v_1, ..., v_n)** mit $\{v_i, v_{i+1}\} \in E$
heißt ein **Weg von v_0 nach v_n**. Er hat die **Länge n** ≥ 0.
Entsprechend gerichtete Graphen: mit $(v_i, v_{i+1}) \in E$ für i = 1,..., n-1



G_2

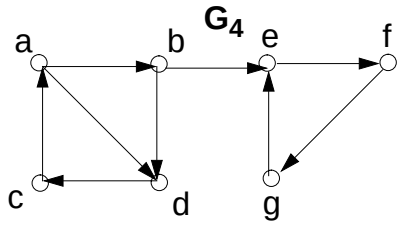
Ein Weg (v_0, v_1, ..., v_n) einer Länge $n \geq 1$ mit $v_0 = v_n$
und paarweise verschiedenen Kanten (v_0, v_1), ..., (v_{n-1}, v_n) heißt
Kreis im ungerichteten Graphen und
Zyklus im gerichteten Graphen.



G_4

Ein gerichteter Graph der keinen Zyklus enthält heißt
azyklischer Graph (engl. **directed acyclic graph, DAG**).

Ein Graph G = (V, E) heißt **zusammenhängend**, wenn es für
beliebige Knoten v, w $\in$ V einen Weg von v nach w gibt.
(Ein gerichteter Graph heißt dann **stark zusammenhängend**.)



G_4

Ein Teilgraph G' = (V', E') von G = (V, E) heißt
Zusammenhangskomponente, wenn
G' **zusammenhängend** ist und wenn
G' durch **Hinzunehmen** beliebiger **Knoten**
$v \in V \setminus V'$ oder **Kanten** $e \in E \setminus E'$ **unzusammenhängend** wird.

Ziele:

Begriffe zu Wegen

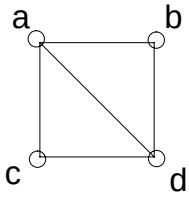
in der Vorlesung:

Begriffe an Beispielen erläutern,

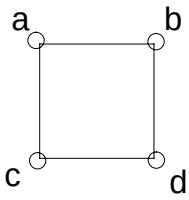
Spezielle Wege und Kreise

Sei G = (V, E) ein ungerichteter, zusammenhängender, schleifenfreier Graph.

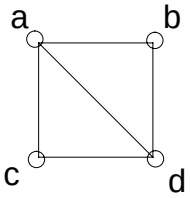
Ein **Euler-Weg** bzw. ein **Euler-Kreis** in G enthält **jede Kante aus E genau einmal**.



(a, b, d, a, c, d)
Euler-Weg



(a, b, d, c, a)
Euler-Kreis



(a, b, d, c, a)
Hamilton-Kreis

G hat einen **Euler-Kreis** genau dann, wenn **alle Knoten geraden Grad** haben.

G hat einen **Euler-Weg**, der kein Kreis ist, genau dann,
wenn G genau **2 Knoten mit ungeradem Grad** hat.

Ein **Hamilton-Kreis** enthält **jeden Knoten aus V genau einmal**.

Vorlesung Modellierung WS 2001/2002 / Folie 410

Ziele:

Wege mit speziellen Eigenschaften

in der Vorlesung:

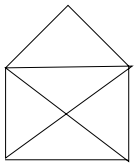
- Begriffe an Beispielen erläutern,
- Königsberger Brückenproblem lösen

nachlesen:

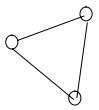
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Wegeprobleme mit Euler-Wegen

1. Königsberger Brückenproblem (Mod-4.1):
   Euler-Weg, Euler-Kreis

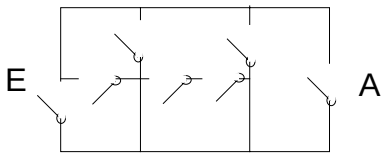


2. Kann man diese Figur in einem Zuge zeichnen?



3. Eine Inselgruppe mit n > 1 Inseln benötigt direkte Schiffsverbindungen zwischen allen Paaren von Inseln. Es gibt nur ein einziges Schiff. Kann es auf einer Tour alle Verbindungen genau einmal abfahren? Für welche n ist das möglich?

4. Planen Sie ein Gruselkabinett:
   Ein Haus mit n > 1 Räumen, 1 Eingangstür, eine Ausgangstür, beliebig vielen Innentüren. Jede Tür schließt nach Durchgehen endgültig. Die Besucher gehen einzeln durch das Haus. Es soll niemand eingesperrt werden.



Vorlesung Modellierung WS 2001/2002 / Folie 411

Ziele:

Aufgaben modellieren lernen

in der Vorlesung:

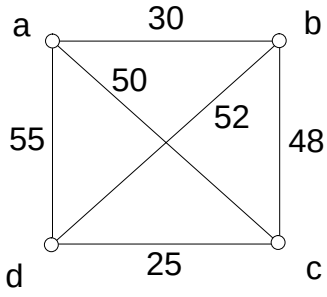
- Erläuterungen zu den Aufgaben.
- Die Graphen zu den Aufgaben zeigen;
- ihre Eigenschaften erkennen.
- In (4) ist wird jeder Raum und die Umgebung als ein Knoten modelliert.

nachlesen:

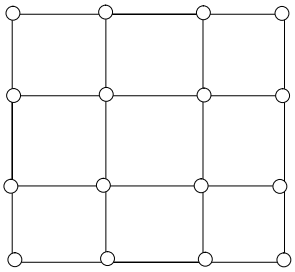
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Wegeprobleme mit Hamilton-Kreisen

1. Traveling Salesman's Problem (Handlungsreisender): n Städte sind mit Straßen bestimmter Länge verbunden. Gesucht ist eine kürzeste Rundreise durch alle Städte.



2. In einem n * n Gitter von Prozessoren soll eine Botschaft sequentiell von Prozessor zu Prozessor weitergegeben werden. Sie soll jeden Prozessor erreichen und zum Initiator zurückkehren. Für welche n ist das möglich?



Vorlesung Modellierung WS 2001/2002 / Folie 412

Ziele:

Aufgaben modellieren lernen

in der Vorlesung:

- Erläuterungen zu den Aufgaben.
- Die Eigenschaften der Graphen erkennen.
- In (1) wird die Entfernung als Kantenmarkierung modelliert.

Allgemeine Hinweise zum Modellieren mit Graphen:

- Rolle der Kanten sorgfältig klären, gerichtet, ungerichtet, markiert.
- Häufig wird der Graph selbst nicht gebraucht, sondern nur bestimmte Eigenschaften, wie Knotengrad.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

4.3 Verbindungsprobleme

Modellierung durch Graphen wie bei Wegeproblemen (4.2), aber hier interessiert die **Existenz von Verbindungen** (Wegen) zwischen Knoten, nicht bestimmte Knotenfolgen.

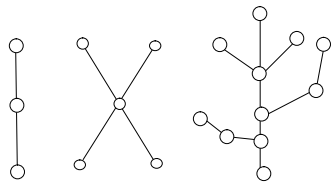
Sei G = (V, E) ein
ungerichteter zusammenhängender Graph
für alle folgenden Begriffe:

Wenn G **keine Kreise** enthält,
heißt er **(ungerichteter) Baum**.

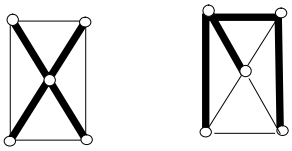
In Bäumen heißen **Knoten mit Grad 1 Blätter**.

Für jeden ungerichteten Baum G = (V, E) gilt | E | = |V| - 1

Bäume



Ein Teilgraph von G, der jeden Knoten aus V enthält und ein Baum ist, heißt **Spannbaum** zu G.



2 Spannbäume zu demselben Graphen

Vorlesung Modellierung WS 2001/2002 / Folie 413

Ziele:

Begriff Spannbaum verstehen

in der Vorlesung:

Erläuterungen und Beispiele zu den Begriffen.

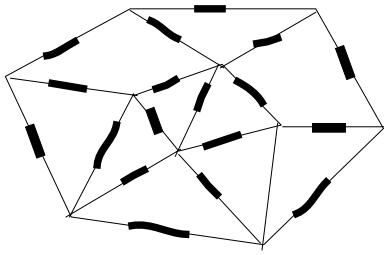
nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Modellierung mit Spannbäumen zu Graphen

Ein Spannbaum ist ein zusammenhängender Teilgraph mit der kleinsten Anzahl Kanten.
Er modelliert kostengünstigen Zusammenhang.

1. Aufständische Gefangene wollen eine minimale Anzahl von Gefängnistüren sprengen, so dass alle Gefangenen freikommen:



2. Alle Agenten A, ..., H sollen direkt oder indirekt miteinander kommunizieren.
   Die Risikofaktoren jeder paarweisen Verbindung sind:

A	A	A	A	A		B	B	C	C	C	C	D	D	E
B	C	E	F	G		C	F	D	F	G	H	E	H	H
9	3	8	3	4	10	6	6	4	5	7	6	3	5	

   Es soll ein Netz mit geringstem Risiko gefunden werden.

Vorlesung Modellierung WS 2001/2002 / Folie 414

Ziele:

Anwendung von Spannbäumen erkennen

in der Vorlesung:

Erläuterungen zu den Modellierungen.

- zu 1: Knoten modellieren Räume und Umgebung, Kanten modellieren die Türen.
- zu 2: Graph mit Kantenmarkierung aufstellen; Spannbaum mit minimaler Kantensumme suchen.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Verbindung und Zusammenhang

Sei G = (V, E) ein ungerichteter, zusammenhängender Graph.

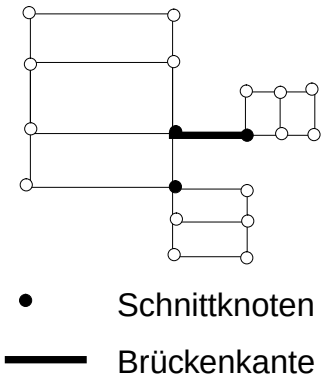
v ist ein **Schnittknoten** in G, wenn G **ohne v nicht mehr zusammenhängend** ist.

e ist eine **Brückenkante** in, wenn G **ohne e nicht mehr zusammenhängend** ist.

G heißt **orientierbar**, wenn man für **jede Kante eine Richtung** so festlegen kann, dass der entstehende **gerichtete Graph stark zusammenhängend** ist.

G ist genau dann **orientierbar**, wenn G **keine Brückenkante** hat.

1. In der Innenstadt sollen zur Hauptverkehrszeit alle Straßen zu Einbahnstraßen werden. Bleiben alle Plätze von überall erreichbar?
2. In einer Stadt sollen einzelne Straßen zur Reparatur gesperrt werden. Bleiben alle Plätze von überall erreichbar?



Vorlesung Modellierung WS 2001/2002 / Folie 415

Ziele:

Zusammenhang zerstören

in der Vorlesung:

Erläuterungen zu den Begriffe und Modellierungen.

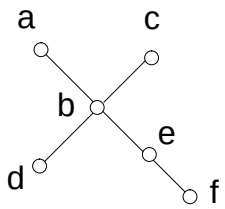
zu 1, 2: Gerichtete Brückenkante zerstört den Zusammenhang.

nachlesen:

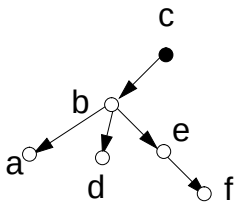
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

4.4 Modellierung mit Bäumen

In einem **ungerichteten Baum** gibt es
zwischen zwei beliebigen Knoten genau einen Weg.



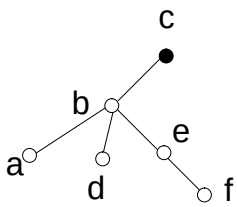
Ein gerichteter, **azyklischer** Graph G ist ein **gerichteter Baum**, wenn alle Knoten einen **Eingangsgrad** $\leq$ **1** haben und es genau einen Knoten mit **Eingangsgrad 0 gibt,**
er ist die Wurzel von G. G ist ein **gewurzelter Baum**.



Wurzel

Man kann aus einem **ungerichteten Baum** in eindeutiger Weise einen gerichteten machen, indem man **einen Knoten zur Wurzel bestimmt**.

Deshalb wird in gewurzelten Bäumen häufig die **Kantenrichtung nicht angegeben**.



In einem gewurzelten Baum ist die **Höhe eines Knotens v** die größte Länge eines Weges von v zu einem Blatt.
Die Höhe der Wurzel heißt **Höhe des Baumes**.

Knoten, die weder Wurzel noch Blatt sind heißen **innere Knoten**.

Vorlesung Modellierung WS 2001/2002 / Folie 416

Ziele:

Zusammenhang: ungerichtet, gerichtet, gewurzelt

in der Vorlesung:

- Erläuterung der Begriffe an dem Beispiel.
- Andere Wurzeln zum selben ungerichteten Graphen
- Höhen bestimmen.

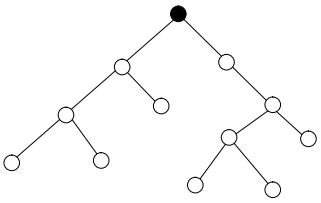
nachlesen:

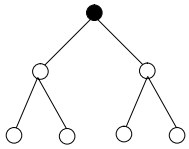
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Binärbäume

Ein gewurzelter Baum heißt **Binärbaum**, wenn
seine Knoten einen **Ausgangsgrad von höchstens 2** haben.

Ein **Binärbaum heißt vollständig**, wenn jeder Knoten außer den Blättern den **Ausgangsgrad 2** hat und die **Wege zu allen Blättern gleich lang** sind.





Höhe 2

Knoten: 7

Blätter: 4

Ein vollständiger **Binärbaum der Höhe h** hat **2^h Blätter** und **2^{h+1}-1 Knoten**

Vorlesung Modellierung WS 2001/2002 / Folie 417

Ziele:

Knotenzahlen in Binärbäumen verstehen

in der Vorlesung:

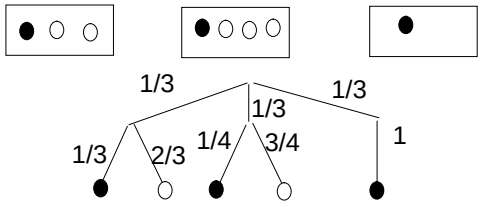
- Rekursive Struktur zeigen.
- Rekursive Berechnung der Knotenzahlen

nachlesen:

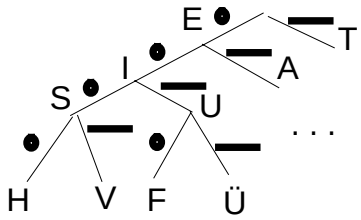
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 6.3

Modellierung von Entscheidungsbäumen

Knoten modelliert **Zwischenstand** einer mehrstufigen Entscheidungsfolge
Kante modelliert eine der wählbaren **Alternativen**

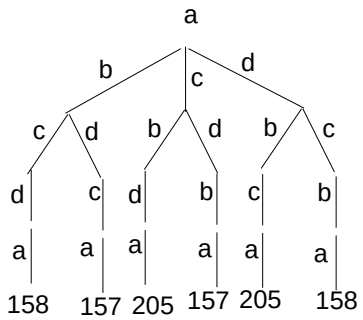


1. **Wahrscheinlichkeiten**,
   z. B. erst Schachtel, dann Kugel ziehen:



2. **Codierungen**,
   Z. B. Morse-Code

3. **Lösungsbaum** für kombinatorische Probleme,
   z. B. Traveling Salesman's Problem (Mod-4.12)
   Blätter repräsentieren einen Rundwege von a aus,
   Kanten sind mit Entscheidungen markiert



4. **Spielzüge**, z. B. Schach (ohne Bild)

Wird **derselbe Zwischenstand** durch verschiedene Entscheidungsfolgen erreicht,
kann man **Knoten identifizieren**.
Es entsteht ein azyklischer oder zyklischer Graph.

Vorlesung Modellierung WS 2001/2002 / Folie 418

Ziele:

Verschiedene Einsatzgebiete von Entscheidungsbämen kennenlernen

in der Vorlesung:

Erläuterungen zu den Beispielen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

Modellierung von Strukturen durch Bäume

Knoten modelliert ein **Objekt**.
Kante modelliert **Beziehung „besteht aus“**, „enthält“, „spezialisiert zu“, ...

Beispiele:

- **Typhierarchie**: Typ - Untertypen
- **Klassenhierarchie**: Oberklasse als Abstraktion ihrer Unterklassen (Mod-4.19a)
  Vererbungshierarchie: Unterklassen erben von ihrer Oberklasse
- **Objektbaum**: Objekt enthält (Referenzen auf) Teilobjekte
- **Kantorowitsch-Baum**: Operator mit seinen Operanden (Mod-4.19b)
- **Strukturbaum**: (Programm-)Struktur definiert durch eine kontextfreie Grammatik (Mod-4.19c)

Identifikation gleicher Teilbäume führt zu azyklischen Graphen (DAGs).

Vorsicht:
Identifikation muss mit der **Bedeutung der Kanten verträglich** sein;
z. B. Ein Gegenstand kann nicht dasselbe Objekt mehrfach als Teil enthalten, wohl aber mehrere Objekte derselben Art.

Vorlesung Modellierung WS 2001/2002 / Folie 419

Ziele:

Varianten von Baumstrukturen

in der Vorlesung:

- Prinzip der Modellierung von Baumstrukturen
- Varianten auf den folgenden 3 Folien

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

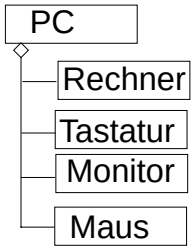
Verständnisfragen:

Kennen Sie weitere Varianten von Baumstrukturen?

Klassen- und Objekthierarchien

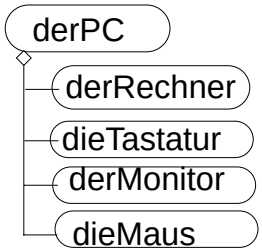
Kompositionsbeziehung im Klassendiagramm (UML Notation):
Knoten: **Klassen**
Kanten: definieren, **aus welcher Art von Objekten** ein Objekt besteht
z. B. ein Objekt der Klasse PC besteht aus
einem Rechner-Objekt, einem Tastatur-Objekt, ...

Diese Beziehung zwischen den Klassen könnte
auch ein allgemeiner Graph sein



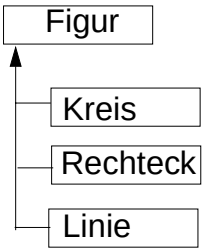
Objektbaum im Objektdiagramm (fast UML):
Knoten: **Objekte**
Kanten: definieren, **aus welchen Objekten** ein Objekt besteht
z. B. dieser PC besteht aus, diesem Rechner, ...

Diese Beziehung muss **konzeptionell ein Baum** sein.



Vererbungsbeziehung im Klassendiagramm (UML Notation):
Knoten: **Klassen**
Kanten: Unterklasse **erbt von** -> Oberklasse
Oberklasse **ist Abstraktion** <- ihrer Unterklassen
Kanten sind zur Wurzel hin gerichtet

Baum bei Einfachvererbung (Java)
azyklischer Graph bei Mehrfachvererbung (C++)



Vorlesung Modellierung WS 2001/2002 / Folie 419a

Ziele:

Klassendiagramme kennenlernen

in der Vorlesung:

Erläuterungen dazu:

- UML Klassendiagramme: Wichtiges Beschreibungsmittel in der Software-Technik.
- Klassendiagramme sind aus ER-Modell abgeleitet (siehe Kapitel 5)
- Klassendiagramme: nicht nur Bäme
- Unterscheidung von Objektiagrammen und Klassendiagrammen

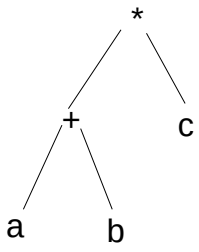
Kantorowitsch-Bäume

Darstellung der Struktur von Termen, Formeln, Ausdrücken (siehe Mod-2.19)

Knoten: Operator

Kanten: Verbindung zu den Operanden eines Operators
Die Kanten sind geordnet (Kantenmarkierung):
erster, zweiter, ... Operand

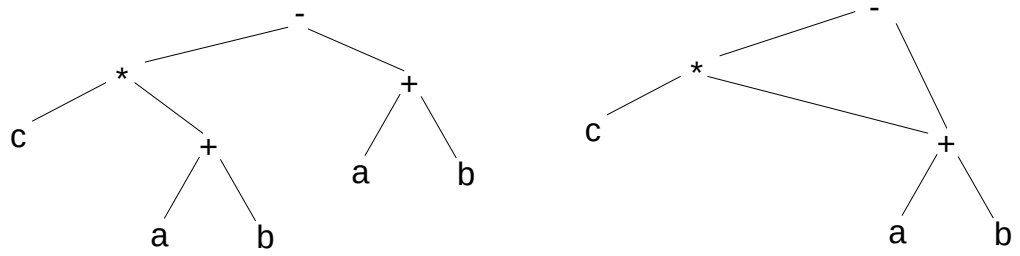
(a + b) * c



Identifikation gleicher Teilbäume führt zu azyklischen Graphen (DAGs):

Z. B. identifizieren Übersetzer gleiche Teilbäume, um Code zu erzeugen, der sie nur einmal auswertet:

c * (a + b) - (a + b)



Vorlesung Modellierung WS 2001/2002 / Folie 419b

Ziele:

Erinnerung an Kantorowitsch-Bäme

in der Vorlesung:

Erläuterungen zu

- Ordnung der Kanten
- Identifikation von Teilbämen

Strukturbäume zu kontextfreien Grammatiken

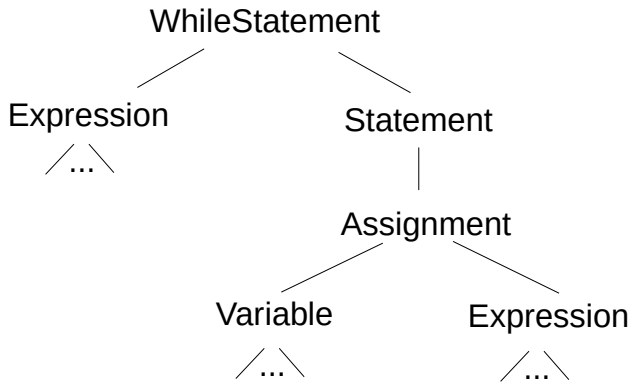
Kontextfreie Grammatiken definieren die Struktur von Programmen, Texten oder Daten.
Ein Programm, Text oder strukturierte Daten werden als Strukturbaum dargestellt.

Knoten: Programmkonstrukt (Nichtterminal der Grammatik)

Kante: Bezug zu Bestandteilen des Programmkonstruktes (Produktion der Grammatik)

Für die Repräsentation von Texten sind die **Kanten geordnet** (Kantenmarkierung)

Strukturbaum:



Produktionen aus der kontextfreien Grammatik:

Statement ::= Assigment

Statement ::= WhileStatement

...

WhileStatement ::= Expression Statement

Assignment ::= Variable Expression

```
while(i<10)  a[i]   =    2*i    ;
```


Vorlesung Modellierung WS 2001/2002 / Folie 419c

Ziele:

Strukturbaum am Beispiel kennenlernen

in der Vorlesung:

Erläuterungen dazu:

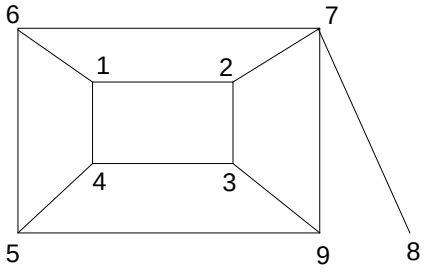
- Kontextfreie Grammatiken werden in Kapitel 5 eingeführt
- Bedeutung von Produktionen informell: "WhileStatement besteht aus Expression und Statement".
- Bezug zum Strukturbaum.

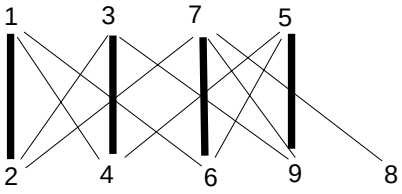
4.5 Zuordnungsprobleme

Aufgabenklasse **paarweise Zuordnung (Matching):**

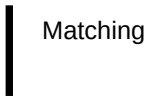
Im ungerichteten Graphen G = (V, E) modelliert eine Kante {a, b} „a passt zu b“, ggf. mit einer Kantenmarkierung als Abstufung

Gesucht ist eine **maximale Menge unabhängiger Kanten**,
das ist einTeilgraph M mit allen Knoten aus V und möglichst vielen Kanten aus E,
so dass der **Grad der Knoten höchstens 1** ist. M heißt ein **Matching** der Knoten von G.





bipartiter Graph



Graph G heißt **bipartit**, wenn V in **2 disjunkte Teilmengen $V = V_1 \cup V_2$** zerlegt werden kann, so dass jede Kante zwei Knoten aus verschiedenen Teilmengen verbindet.

Häufig liefert die Aufgabenstellung schon bipartite Graphen, sogenannte **Heiratsprobleme**:

Mann - Frau Aufgabe - Bearbeiter Verbraucher - Produkte

Vorlesung Modellierung WS 2001/2002 / Folie 420

Ziele:

Paarweise Zuordnung verstehen

in der Vorlesung:

- Matching-Begriff erläutern,
- bipartit erläutern,
- Beispiele angeben

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.2

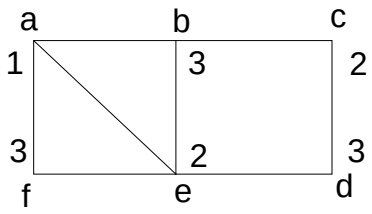
Konfliktfreie Knotenmarkierung (Färbung)

Aufgabenklasse **konfliktfreie Knotenmarkierung (Färbung):**

Im ungerichteten Graphen G = (V, E) modelliert eine Kante {a, b} „a ist unverträglich mit b“,

Gesucht ist eine Knotenmarkierung Färbung: V -> ℕ („Farben“),
so dass durch eine Kante verbundene Knoten verschiedene Marken haben

Die chromatische Zahl eines Graphen G ist die minimale Zahl verschiedener „Farben“, die nötig ist, um G konfliktfrei zu markieren.



Es gilt: chromatische Zahl ≤ 1 + maximaler Knotengrad

Anwendungen:

Knoten:	**Kante:**	**Farbe / Marke:**
Staat auf Landkarte	gemeinsame Grenze	Farbe
Partygast	unverträglich	Tisch
Kurs	haben gemeinsame Teilnehmer	Termin
Prozess	benötigen gleiche Ressource	Ausführungszeitpunkt
Variable im Programm	gleichzeitig lebendig	Registerspeicher

Vorlesung Modellierung WS 2001/2002 / Folie 421

Ziele:

Konzept der Färbung verstehen

in der Vorlesung:

- Erläuterung der Unverträglichkeitsrelation.
- Chromatische Zahlen einer Graphen.
- Erläuterung der Anwendungen.

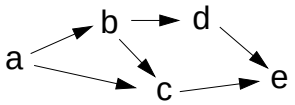
4.6 Abhängigkeitsprobleme

1. Abhängigkeiten zwischen Operationen modellieren (**Abhängigkeitsgraphen**)
2. Ausführungsreihenfolgen von Operationen modellieren (**Ablaufgraphen**)

1. Abhängigkeitsgraph: gerichtet, azyklisch, voneinander abhängige Operationen.

Aufgaben dazu: sequentielle oder parallele **Anordnungen finden** (engl. **scheduling**).

Knoten: Operation, Ereignis; ggf. mit Dauer markiert
Kante: a -> b a ist **Vorbedingung** für b oder
b **benutzt** Ergebnis von a oder
a liest oder schreibt Ressource
bevor b sie überschreibt

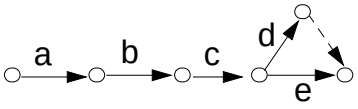


Anwendungen:

- **Projektplanung** mit abhängigen Teilaufgaben (PERT, CPM)
- abhängige **Transaktionen** mit einer Datenbank
- **Anordnung von Code** für die parallele Auswertung von Ausdrücken (Übersetzer)

Kritischer Pfad: längster Weg von einem Anfangsknoten zu einem Endknoten

Duale Modellierung:
Knoten: Ereignis, Anfang und Ende einer Operation
Kante: Operation, ggf. mit Dauer markiert



Vorlesung Modellierung WS 2001/2002 / Folie 422

Ziele:

Prinzip der Abhängigkeitsgraphen verstehen

in der Vorlesung:

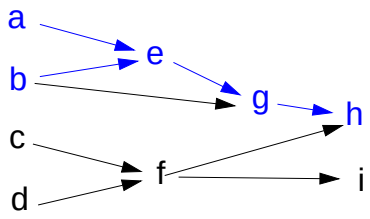
Erläuterungen zu

- Bedeutung von Knoten und Kanten
- Markierungen
- kritischem Pfad
- dualer Modellierung

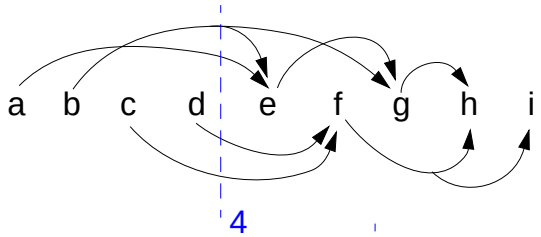
Anordnung von Abhängigkeitsgraphen

Anordnungsaufgaben:
gegebener Abhängigkeitsgraph

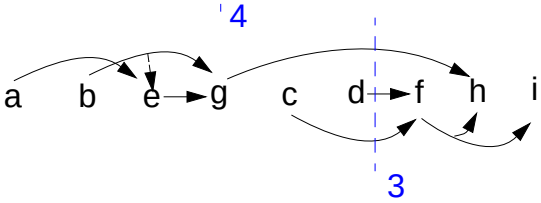
kritischer Pfad



sequentielle Anordnung der Knoten,
so dass **alle Kanten vorwärts** zeigen.

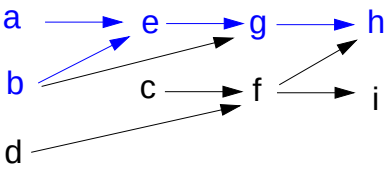


Meist sollen **Randbedingungen** erfüllt werden,
z. B. geringste Anzahl gleichzeitig benötigter Zwischenergebnisse



parallele Anordnung mit
beschränkter Paralellität 3

Länge: 4 Schritte (Operationen)



Vorlesung Modellierung WS 2001/2002 / Folie 423

Ziele:

Anordnungsaufgaben verstehen

in der Vorlesung:

Erläuterungen zu

- Kanten nur vorwärts
- sequentielle Anordnung: Anzahl der Operationen bestimmt die Länge
- Anzahl der Zwischenergebnisse
- parallele Anordnung: kritischer Pfad bestimmt die Länge

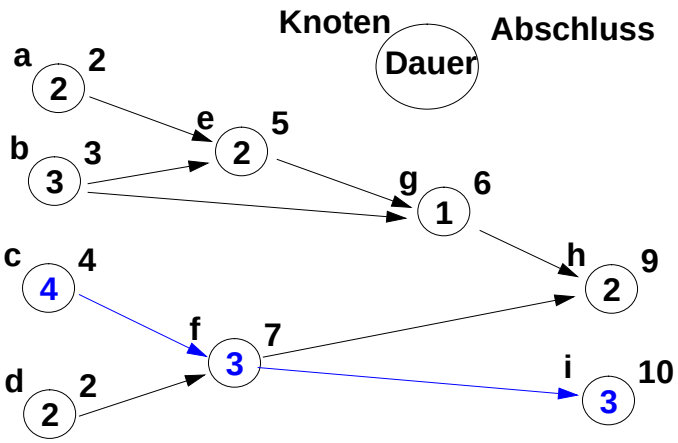
Operationen unterschiedlicher Dauer

Zwei **Knotenmarkierungen**:

Dauer der Operation und

frühester Abschlusstermin
= max. Abschluss der Vorgänger
+ Dauer des Knotens

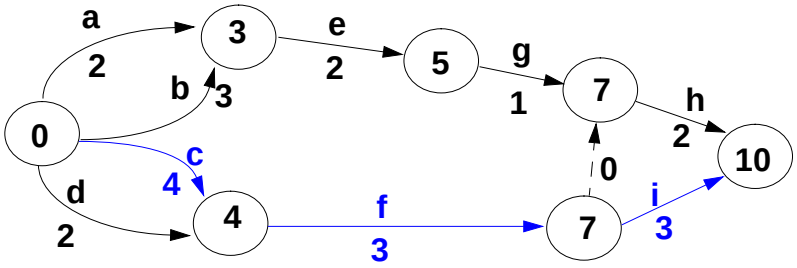
Kritischer Pfad gemäß Summe der Dauer der Operationen



Duale Modellierung:

Kante: Operation
mit **Dauer** als Marke
Mehrfachkanten, Multigraph

Knoten: Ereignis
„vorangehende Operationen sind abgeschlossen“
mit frühestem **Abschlusstermin** als Marke



Vorlesung Modellierung WS 2001/2002 / Folie 424

Ziele:

Ausführungsdauer modellieren

in der Vorlesung:

Erläuterungen zu

- den Knotenmarkierungen,
- der Berechnung des Abschlusstermins,
- dem Kritischen Pfad,
- der dualen Modellierung,
- der Notwendigkeit der zusätzlichen (gestrichelten) Kante

Ablaufgraphen

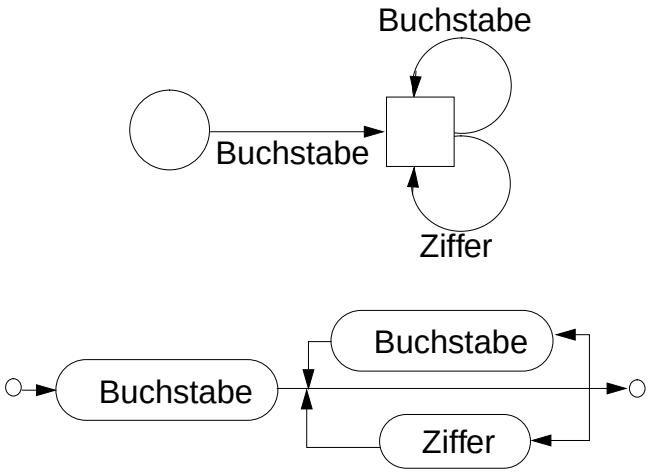
Gerichteter Graph (auch zyklisch) modelliert Abläufe.

Knoten: Verzweigungsstelle, Zustand
Kanten: Fortsetzungsmöglichkeit

Jeder **Weg durch den Graphen** beschreibt einen **potenziellen Ablauf**
Die **Folge der Markierungen eines Weges** kann einen **Satz einer Sprache** modellieren.

Anwendungen:

- **Endlicher Automat** (siehe Kapitel 6)
  modelliert **Folgen von Zeichen**, Symbolen, ...
  Knoten: Zustand
  Kante: Übergang markiert mit Zeichen
- **Syntaxdiagramm**
  modelliert **Folgen von Zeichen**, Symbolen, ...
  Knoten: markiert mit Zeichen
  Kante: „kann folgen auf“
  dual zum endlichen Automaten
- **Aufrufgraphen** (siehe Mod-4.26)
- **Ablaufgraphen** (siehe Mod-4.27)



Vorlesung Modellierung WS 2001/2002 / Folie 425

Ziele:

Prinzip der Ablaufgraphen verstehen

in der Vorlesung:

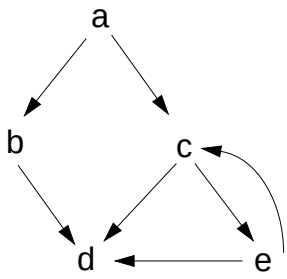
- Erläuterungen am Beispiel des endlichen Automaten.
- Die Zeichnung hat keine Knotennamen, nur eine Kantenmarkierung.
- Syntaxdiagramme als dualen Beschreibung zum endlichen Automaten erklären,
- Die Zeichnung hat keine Knotennamen, nur eine Knotenmarkierung.
- Viele einzelne Kanten sind in der Zeichnung zu einem "Gleissystem" zusammengefasst.

Aufrufgraphen

Gerichteter Aufrufgraph: **Aufrufbeziehung zwischen Funktionen** in einem Programm; wird benutzt in **Übersetzern** und in **Analysewerkzeugen** zur Software-Entwicklung.

Knoten: **Funktion** im Programm
Kante a -> b: Rumpf der Funktion a enthält einen Aufruf der Funktion b; **a könnte b aufrufen**

Zyklus im Aufrufgraph:
Funktionen, die sich **wechselweise rekursiv** aufrufen, z. B. (c, e, c)



Fragestellungen z. B.

- Welche Funktionen sind nicht rekursiv?
- Welche Funktionen sind nicht (mehr) erreichbar?
- Indirekte Wirkung von Aufrufen,
  z. B. nur e verändere eine globale Variable x;
  welche Aufrufe lassen x garantiert unverändert? b, d

Vorlesung Modellierung WS 2001/2002 / Folie 426

Ziele:

Prinzip des Aufrufgraphen verstehen

in der Vorlesung:

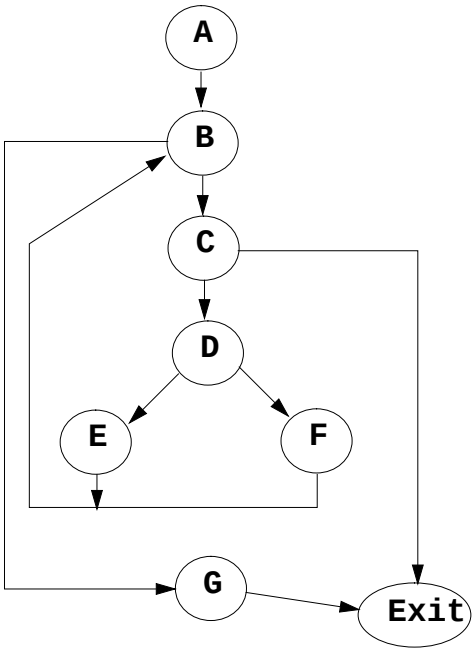
- Erläuterungen dazu
- Weitere Eigenschaften und Anwendungen in der Vorlesung Übersetzer.

Programmablaufgraphen

Gerichteter Graph, modelliert **Abläufe durch ein verzweigtes Programm** (bzw. Funktion); wird benutzt in **Übersetzern** und in **Analysewerkzeugen** zur Software-Entwicklung.

Knoten: unverzweigte Anweisungsfolge (Grundblock), mit Verzweigung (Sprung) am Ende
Kante: potenzieller Nachfolger im Ablauf

```
ug = 0;                              A
og = obereGrenze;
while (ug <= og)                     B
{   mitte = (ug + og) / 2;           C
    if (a[mitte] == x)
          return mitte;
    else if (a[mitte] < x)           D
          ug = mitte + 1;            E
    else   og = mitte - 1;           F
}
return nichtGefunden;                G
```



Fragestellungen, z. B.

- Menge von Wegen, die den **Graph überdecken**, **Software-Testen**
- Wege mit bestimmten Eigenschaften, **Datenflussanalyse**

Vorlesung Modellierung WS 2001/2002 / Folie 427

Ziele:

Prinzip des Programmablaufgraphen verstehen

in der Vorlesung:

- Erläuterungen zum Prinzip,
- Auch andere Abläufe als Programme können so modelliert werden.
- Weitere Eigenschaften und Anwendungen in der Vorlesung Übersetzer.

Zusammenfassung zu Graphen

Problemklassen:

- Wegeprobleme
- Verbindungsprobleme
- Entscheidungsbäume
- hierarchische Strukturen
- Zuordnungsprobleme
- Abhängigkeitsprobleme
- Anordnungen in Folgen
- verzweigte Abläufe

Kanten- und Knotenbedeutung:

- verbunden, benachbart, ...
- Entscheidung, Alternative, Verzweigung
- Vorbedingung, Abhängigkeit
- (Un-)Verträglichkeit
- allgem. symmetrische Relation
- besteht aus, enthält, ist-ein
- (Halb-)Ordnungsrelation

Kanten-, Knotenmarkierungen:

- Entfernung, Kosten, Gewinn, ... bei Optimierungsproblemen
- „Färbung“, disjunkte Knotenmengen bei Zuordnungsproblemen
- Symbole einer Sprache

Vorlesung Modellierung WS 2001/2002 / Folie 428

Ziele:

Übersicht zu Modellierungsaspekten

in der Vorlesung:

- Stichworte zum Einordnen von Modellierungsaufgaben,
- Hilfe zur Wahl einer passenden Variante von Graphen

5. Modellierung von Strukturen
5.1 Kontextfreie Grammatiken

Kontextfreie Grammatik (KFG): formaler Kalkül, Ersetzungssystem; definiert

- **Sprache** als Menge von Sätzen; jeder **Satz** ist eine **Folge von Symbolen**
- **Menge von Bäumen**; jeder Baum repräsentiert die **Struktur eines Satzes** der Sprache

Anwendungen:

- Programme einer **Programmiersprache** und deren Struktur, z. B. Java, Pascal, C
- Sprachen als Schnittstellen zwischen Software-Werkzeugen, **Datenaustauschformate**, z. B. HTML, XML
- Bäume zur Repräsentation **strukturierter Daten**, z. B. in HTML
- Struktur von **Protokollen** beim Austausch von Nachrichten zwischen Geräten oder Prozessen

Beispiel zu HTML:

```
<table>
    <tr>
        <td>Mo</td>
        <td>11-13</td>
        <td>AM</td>
    </tr>
    <tr>
        <td>Fr</td>
        <td>9-11</td>
        <td>AM</td>
    </tr>
</table>
```


Vorlesung Modellierung WS 2001/2002 / Folie 501

Ziele:

Einsatz von KFGn kennenlernen

in der Vorlesung:

Erläuterungen zu den Anwendungen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 1.6.3

Kontextfreie Grammatik

Eine kontextfreie Grammatik G = (T, N, P, S) besteht aus:

T	**Menge der Terminalsymbole** (kurz: Terminale)
N	**Menge der Nichtterminalsymbole** (kurz: Nichtterminale)
	T und N sind disjunkte Mengen
$S \in N$	**Startsymbol** (auch Zielsymbol)
$P \subseteq N \times V^*$	**Menge der Produktionen**
$V = T \cup N$	heißt auch **Vokabular**, seine Elemente heißen **Symbole**

Eine Produktion wird notiert **A ::= x** wobei x eine evtl. leere Folge von Symbolen aus V ist.
Man sagt „A steht auf der **linken Seite** und x auf der **rechten Seite** der Produktion.“
Man gibt Produktionen häufig **Namen**: p1: A ::= x

Beispiel:

Terminale	**T** = { **(** , **)** }
Nichtterminale	**N** = { **Klammern**, **Liste** }
Startsymbol	**S** = **Klammern**

Name	**Produktionsmenge P**
	{
p1:	**Klammern::= ’(’ Liste ’)’**
p2:	**Liste ::= Klammern Liste**
p3:	**Liste ::=**
	}

Vorlesung Modellierung WS 2001/2002 / Folie 502

Ziele:

KFG Definition lernen

in der Vorlesung:

- Erläuterung der Bergriffe an dem Beispiel
- Erläuterung der Notation von Produktionen
- Kennzeichnung von Terminalen, um Verwechselungen zu vermeiden: ’(’

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 1.6.3

Bedeutung der Produktionen

Eine Produktion A ::= x ist eine **Strukturregel**: A **besteht aus** x

Beispiele:

DeutscherSatz	::=	Subjekt Prädikat Objekt
Ein DeutscherSatz	*besteht aus (der Folge)*	Subjekt Prädikat Objekt
Klammern	::=	'(' Liste ')'
Zuweisung	::=	Variable ':=' Ausdruck
Variable	::=	Variable '[' Ausdruck ']'

Produktion als gewurzelter Baum mit geordneten Kanten und mit Symbolen als Knotenmarken:

Eine Produktion A ::= x ist auch eine **Ersetzungsregel**:
In einer Symbolfolge u A v kann A **ersetzt werden durch** x; Ergebnis: u x v

Unter Anwendung der Produktion **`Klammern ::= '(' Liste ')'`** kann man z. B. in
`Klammern Klammern Liste` das 2. Symbol ersetzen: **`Klammern ( Liste ) Liste`**

Vorlesung Modellierung WS 2001/2002 / Folie 503

Ziele:

Produktionen verstehen

in der Vorlesung:

Erläuterungen der beiden Rollen von Produktionen:

- Definition von Struktur: "besteht aus"
- Definition von Ersetzungen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 1.6.3

Ableitungen

Produktionen sind **Ersetzungsregeln**: Ein Nichtterminal A in einer Symbolfolge u A v kann durch die rechte Seite x einer Produktion A ::= x ersetzt werden.
Das ist ein **Ableitungsschritt**; er wird notiert als u A v => u x v

z. B. **`Klammern Klammern Liste`** => **`Klammern ( Liste ) Liste`**
mit Produktion p1

Beliebig viele Ableitungsschritte nacheinander angewandt heißen **Ableitung**; notiert als u =>* v

Eine kontextfreie Grammatik **definiert eine Sprache**; das ist eine Menge von Sätzen.
Jeder Satz ist eine Folge von Terminalsymbolen, die aus dem Startsymbol ableitbar ist:
L(G) = { w | w ∈ T* und S =>* w }

Die Grammatik auf Mod-5.2 definiert z. B. geschachtelte Folgen paariger Klammern als Sprachmenge:

{ (), (()), (()()()),((()())(())), ... } ⊆ L(G)

Ziele:

Ableitungsbegriff verstehen

in der Vorlesung:

Erläuterungen dazu

- Beispiele für Ableitungen
- Beispiele für Sprachen

nachlesen:

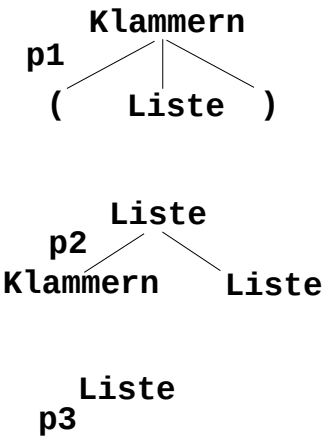
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 1.6.3

Ableitungsbäume

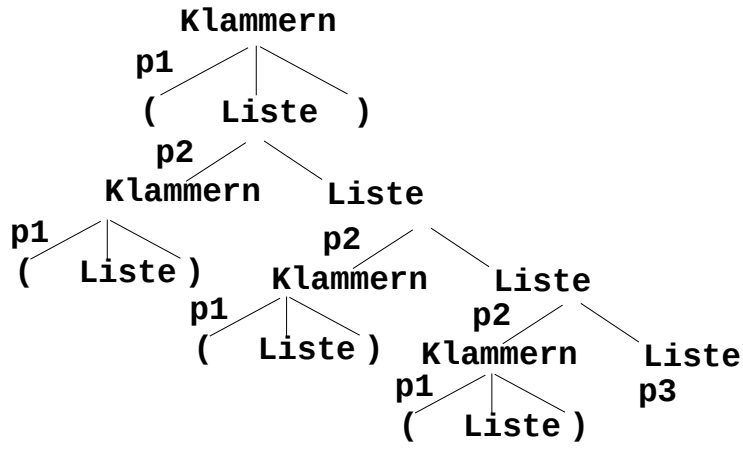
Jede Ableitung kann man als **gewurzelten Baum** darstellen:

- Die **Knoten** mit ihren Marken repräsentieren **Vorkommen von Symbolen**.
- Ein Knoten mit seinen direkten Nachbarn repräsentiert die **Anwendung einer Produktion**.
- Die **Wurzel** ist mit dem **Startsymbol** markiert.
- **Terminale** kommen nur an **Blättern** vor.

Produktionen:



ein Ableitungsbaum:



der Satz dazu: **(()())**

Satz zum Baum: Terminale im links-abwärts Durchgang

Vorlesung Modellierung WS 2001/2002 / Folie 505

Ziele:

Ableitungsbaum verstehen

in der Vorlesung:

- Konstruktion des Baumes durch Zusammensetzen von Produktionsanwendungen,
- Zusammenhang zum Satz der Sprache

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 1.6.3

Beispiel: Ausdrucksgrammatik

p1: Ausdruck ::= Ausdruck BinOpr Ausdruck

p2: Ausdruck ::= Zahl

p3: Ausdruck ::= Bezeichner

p4: Ausdruck ::= '(' Ausdruck ')'

p5: BinOpr ::= '+'

p6: BinOpr ::= '-'

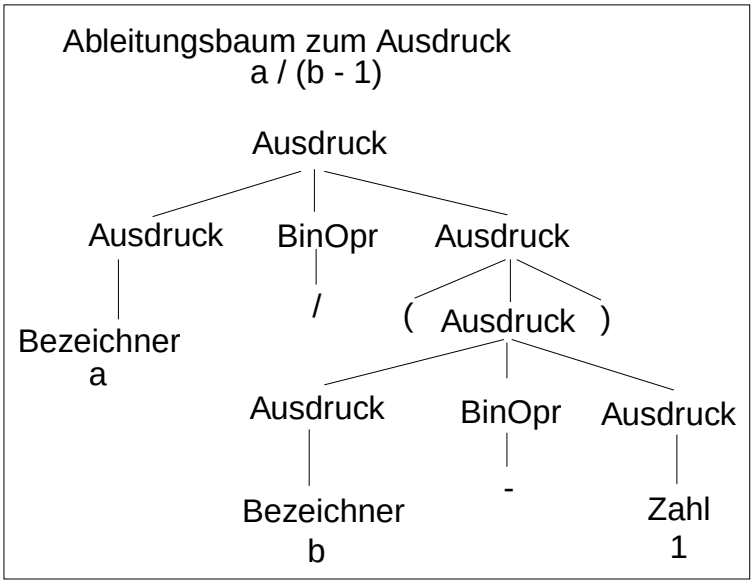
p7: BinOpr ::= '*'

p8: BinOpr ::= '/'

Startsymbol: Ausdruck

Terminale:
{ Zahl, Bezeichner, (,), +, -, *, / }

Schreibweise der Terminale Zahl und Bezeichner wird nicht in der KFG definiert.



Vorlesung Modellierung WS 2001/2002 / Folie 506

Ziele:

Vollständiges Beispiel sehen

in der Vorlesung:

- Erläuterungen dazu.
- Vergleich mit Kantorowitsch-Bäumen.
- Es gibt Sätze zu der Grammatik, die mehrere Ableitungsbäume haben, z. B. a+b+c. Solche Grammatiken nennt man "mehrdeutig".

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 1.6.3

Beispiel: Tabellen in HTML

HTML: Hypertext Markup Language zur Darstellung von verzeigerten Dokumenten, insbesondere im WWW verwendet.
typisch: geklammerte Strukturen mit Klammern der Form **<x>...</x>**.
hier: vereinfachter Ausschnitt aus der Sprache zur Darstellung von Tabellen.

Produktionen der kontextfreien Grammatik:

Table	::= '<table>' Rows '</table>'
Rows	::= Row *****
Row	::= '<tr>' Cells '</tr>'
Cells	::= Cell *****
Cell	::= '<td>' Text '</td>'
Cell	::= '<td>' Table '</td>'

Erweiterung der Notation vonKFGn:

X * auf der rechten Seite einer Produktion steht für eine **beliebig lange Folge von X**

(gleiche Bedeutung wie bei Wertebereichen)

Beispieltext in HTML:

```
<table>
    <tr> <td>Tag</td>
         <td>Zeit</td>
         <td>Raum</td></tr>
    <tr> <td>Mo</td>
         <td>11:00-12.30</td>
         <td>AM</td></tr>
    <tr> <td>Fr</td>
         <td>9:15-10:45</td>
         <td>AM</td></tr>
</table>
```

Darstellung der Tabelle:

Tag	Zeit	Raum
Mo	11:00-12.30	AM
Fr	9:15-10:45	AM

Ziele:

HTML-Ausschnitt verstehen

in der Vorlesung:

Erläuterungen

- zum *-Operator (siehe Mod-2.8a),
- zur Struktur von HTML,
- zum Beispiel,
- zur Baumdarstellung

5.2 Entity-Relationship-Modell

Entity-Relationship-Modell, ER-Modell (P. Chen 1976): Kalkül zur Modellierung von **Aufgabenbereichen mit ihren Objekten, Eigenschaften und Beziehungen**.

Weitergehende Zwecke:

- **Entwurf von Datenbanken**;
  Beschreibung der Daten, die die DB enthalten soll, „konzeptionelles Schema“
- **Entwurf von Software-Strukturen**
  Entwurfssprache UML basiert auf ER

Grundbegriffe

- **Entity** **Objekt** des Aufgabenbereiches
- **Relation** **Beziehung** zwischen Objekten
- **Attribut** Beschreibt ein **Eigenschaft** eines Objektes durch einen **Wert**

Graphische und textuelle **Notationen** für ER-Modellierungen; hier graphische

Ziele:

Zweck des ER-Modells

in der Vorlesung:

Erläuterungen dazu

nachlesen:

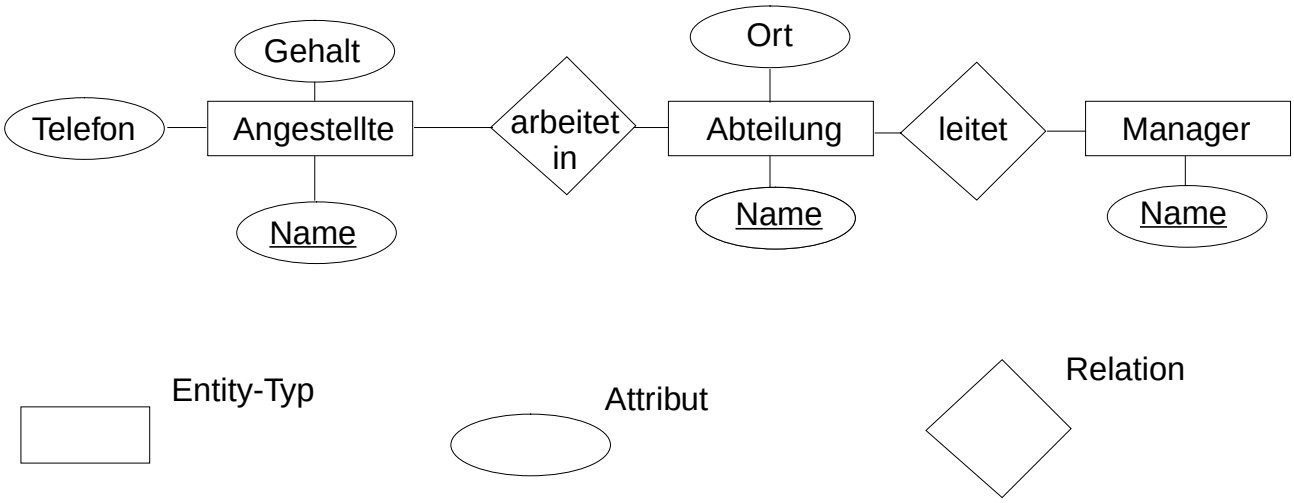
G. Engels: Skript zu TSE II, 2. Datenbankmodelle für den Entwurf

J. D. Ullman: Principles of Database and Knowledge-Base Systems, Vol. I, Computer Science Press, 1988; Ch. 2.2

A.L.Furtado, E. J. Neuhold: Formal Techniques for Data Base Design, Springer, 1986; Ch. 9

Einführendes Beispiel

Ausschnitt aus der Modellierung einer Firmenorganisation:



[Beispiel nach J. D. Ullman: Principles ...]

Vorlesung Modellierung WS 2001/2002 / Folie 509

Ziele:

Erster Eindruck vom ER-Modell

in der Vorlesung:

- Erläuterungen zu dem Beispiel
- Graphiken für die 3 Grundbegriffe

Entities

Entity:
Objekt, Gegenstand aus dem zu modellierenden **Aufgabenbereich**
Jede Entity hat eine **eindeutige Identität**, verschieden von allen anderen

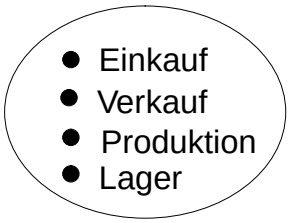
Entity-Menge (auch **Entity-Typ**):
Zusammenfassung von Objekten, die im Modell als **gleichartig** angesehen werden,
z. B. Angestellte, Abteilung, Manager

Im **Modell steht eine Entity-Menge** für die ggf. nicht-endliche Menge aller infrage kommenden Objekte dieser Art.

Eine **konkrete Ausprägung zu der Entity-Menge** ist eine endliche Teilmenge davon.

Abteilung

steht im Modell für die Menge aller in Unternehmen möglichen Abteilungen



konkrete Ausprägung dazu:
die Menge der Abteilungen eines konkreten Unternehmens

Vorlesung Modellierung WS 2001/2002 / Folie 510

Ziele:

Entity-Mengen verstehen

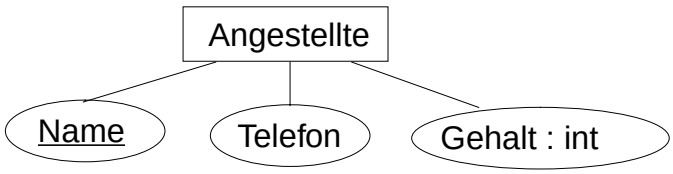
in der Vorlesung:

Erläuterungen dazu

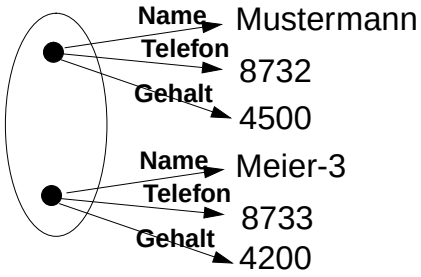
- zur Eindeutigkeit von Entities; Vergleich mit Objekten in Java,
- zu Entity-Mengen;
- Vorsicht beim Vergleich mit Wertebereichen: Dort haben wir Potenzmengen als Wertebereich von konkreten Ausprägungen, die Mengen sind; hier haben wir auch im Modell Entity-Mengen.

Attribute

Attribute beschreiben Eigenschaften von Entities.
Einer Entity-Menge im Modell können Attribute zugeordnet werden, z. B.



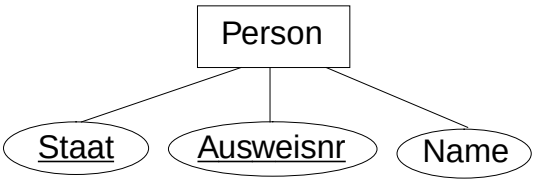
eine konkrete Ausprägung:



Ein Attribut ordnet jeder Entity aus der konkreten Entity-Menge einen Wert zu.
Der **Wertebereich eines Attributes** kann explizit angegeben sein, z. B. int für Gehalt, oder er wird passend angenommen.

Ein Attribut, dessen **Wert jede Entity eindeutig identifiziert**, heißt **Schlüsselattribut**.
Es wird im Modell unterstrichen.

Auch **mehrere Attribute zusammen** können den Schlüssel bilden:



Vorlesung Modellierung WS 2001/2002 / Folie 511

Ziele:

Attribute und ihre Werte verstehen

in der Vorlesung:

- Attribute bilden Entities auf Werte ab.
- Wertebereiche von Attributen wie in Kapitel 2 der Vorlesung.
- Derselbe Attributwert kann vielfach im System vorkommen - im Unterschied zu Objekten, die eindeutig identifizierbar sind.
- Wenn sich ein Schlüsselattribut bei der Modellierung nicht ohnehin natürlich ergibt, sollte man eines einführen (z. B. Nummer der Entities).

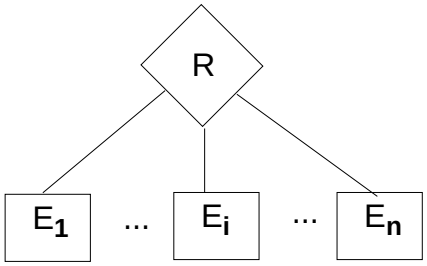
Relationen

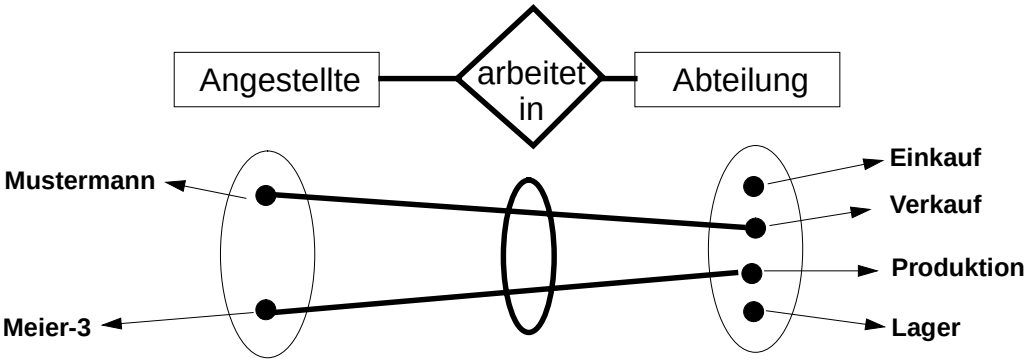
Relationen modellieren Beziehungen zwischen den Entities der Entity-Mengen.

n-stellige Relation R über n Entity-Mengen E_1,..., E_n, mit $n \geq 2$:

Im Modell wird dadurch der **Typ der Relation** angegeben.

Eine **konkrete Ausprägung von R** ist eine **Menge von n-Tupeln (e_1, ... , e_n)**, wobei die e_i Entities aus den konkreten Ausprägungen der Entity-Mengen E_i sind.





Vorlesung Modellierung WS 2001/2002 / Folie 512

Ziele:

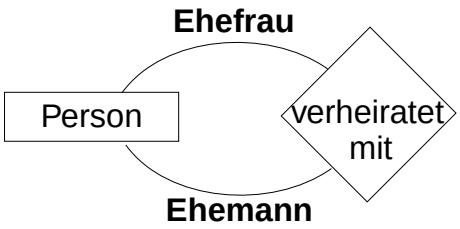
Relationen im ER-Modell verstehen

in der Vorlesung:

Relationsbegriff entspricht dem aus Kapitel 2. Allerdings sind die Wertebereiche auf Entity-Mengen eingeschränkt.

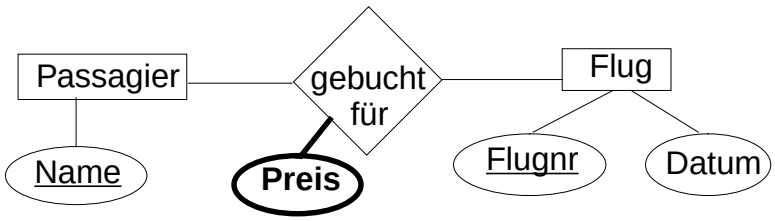
Rollen und Attribute in Relationen

Für manche Relationen wird aus ihrem Namen und der Graphik nicht klar, welche Bedeutung die Entity-Mengen in der Relation haben. Man kann das durch **Rollennamen an den Kanten** verdeutlichen.

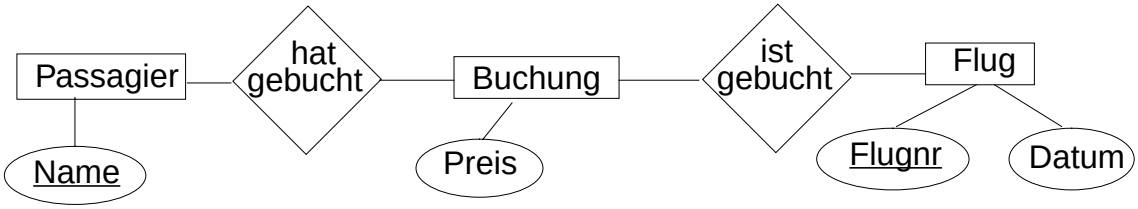


Auch **Relationen können Attribute haben**. Sie beschreiben **Eigenschaften zu jedem Tupel der Relation**.

Der Preis ist eine **Eigenschaft der Buchung** - nicht des Passagieres oder des Fluges.



Man könnte natürlich auch **Buchungen als Entities** modellieren:



Vorlesung Modellierung WS 2001/2002 / Folie 513

Ziele:

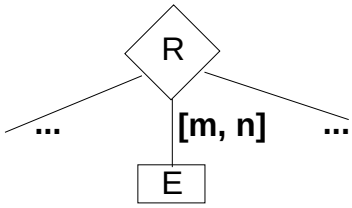
Modellierung von Relationen

in der Vorlesung:

- Erläuterungen zu Rollen,
- zu Attributen von Relationen.
- Mit den beiden Varianten der Modellierung von Flugbuchungen kann man Unterschiedliches ausdrücken: In der unteren Variante kann derselbe Passagier denselben Flug mehrfach buchen. Inder oberen Variante geht das nicht.

Kardinalität von Relationen

In Relationen wird duch Angaben zur **Kardinalität** bestimmt, wie oft eine Entity in den Tupeln der Relation vorkommen kann bzw. vorkommen muss:



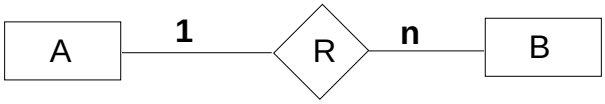
Für jede konkrete Ausprägung der Relation R muss gelten:
Jede Entity e aus der konkreten Entity-Menge zu E kommt
in mindestens m und höchstens n Tupeln vor.

Spezielle Kardinalitäten:

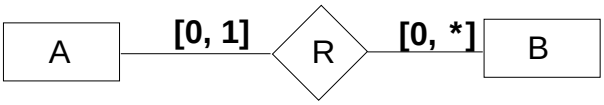
- [1, 1] in **genau einem** Tupel: totale Funktion von E auf die übrigen Rollen der Relation
- [0, 1] in **höchstens einem** Tupel: partielle Funktion von E auf die übrigen Rollen
- [0, *****] in **beliebig vielen** Tupeln

Ohne Angabe wird [0, *] angenommen.

Kurznotation für 2-stellige Relationen:



bedeutet:



Vorlesung Modellierung WS 2001/2002 / Folie 514

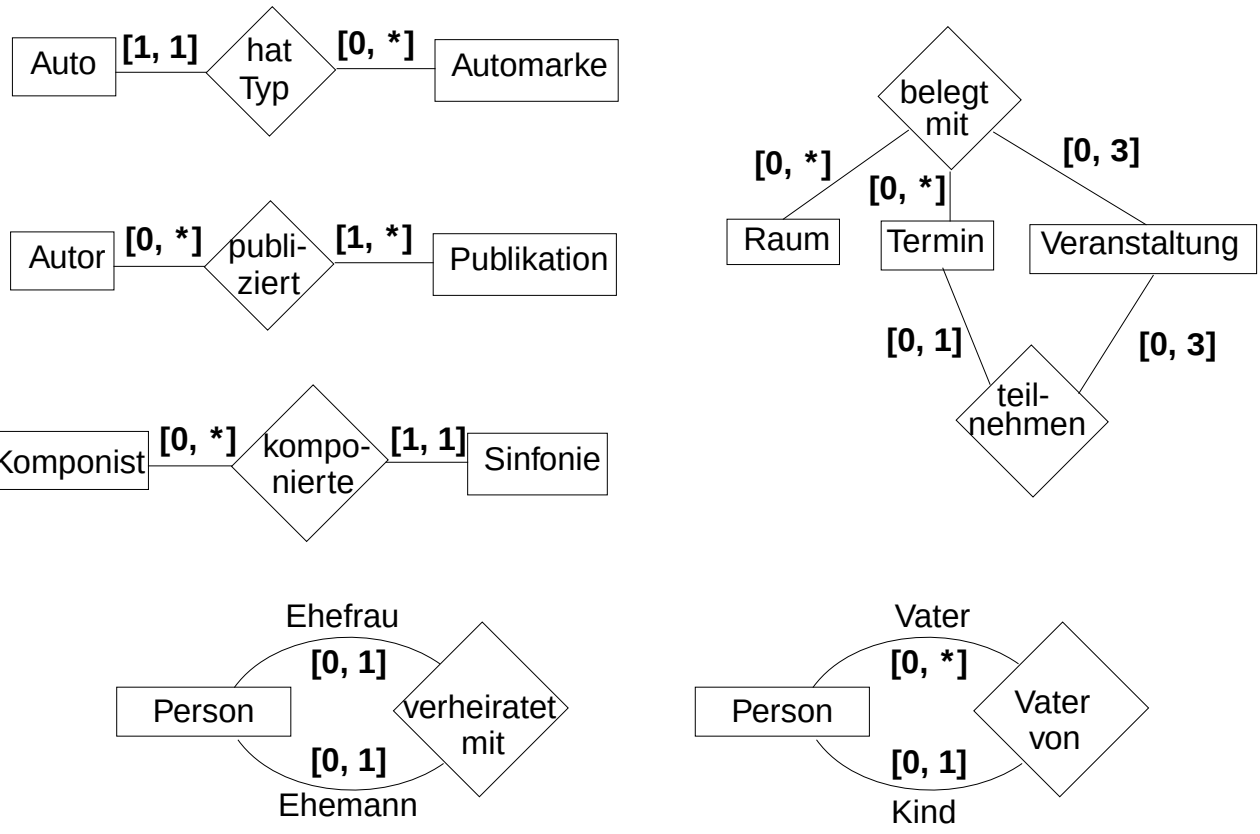
Ziele:

Kardinalitäten verstehen

in der Vorlesung:

- Erläuterung von Kardinalitäten als einschränkende Präzisierung des Modells.
- Erläuterung an Beispielen von Mod-5.15
- Achtung: Es gibt ER-Dialekte, in denen dieselben Notationen eine andere Bedeutung haben: Anzahl der Tupel, die sich nur in Werten aus E unterscheiden. Wir verwenden sie hier nicht.

Beispiele zu Kardinalitäten in Relationen



Vorlesung Modellierung WS 2001/2002 / Folie 515

Ziele:

Kardinalitäten üben

in der Vorlesung:

Erläuterungen zu den Relationen:

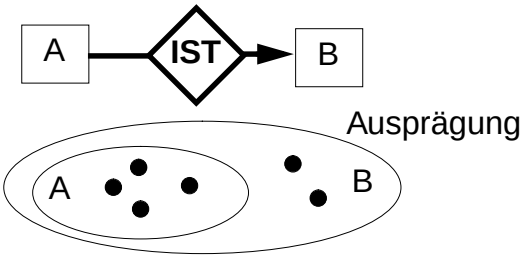
- Jedes Auto-Exemplar hat genau eine Automarke.
- Zu einer Automarke können beliebig viele Autos modelliert sein.
- Eine Publikation hat mindestens einen Autor.
- Eine Sinfonie stammt von genau einem Komponisten.
- Es gibt auch unverheiratete Personen.
- Polygamie ist in diesem Modell nicht vorgesehen.
- Die Väter mancher Personen sind nicht modelliert.
- Veranstaltungen werden höchstens dreimal pro Woche angeboten.
- Im Stundenplan sind Termine nicht mehrfach belegt.

IST-Hierarchie

Die spezielle **Relation IST** (engl. is-a) definiert eine **Spezialisierungs-Hierarchie** für Entity-Mengen:

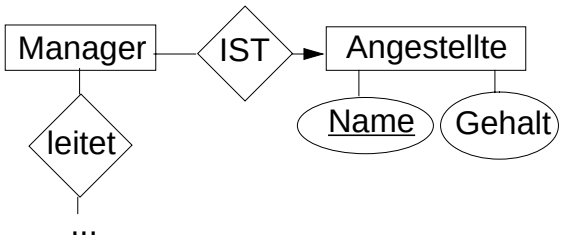
A IST B: Einige Entities der **allgemeineren Menge B** gehören auch der **spezielleren Menge A** an.

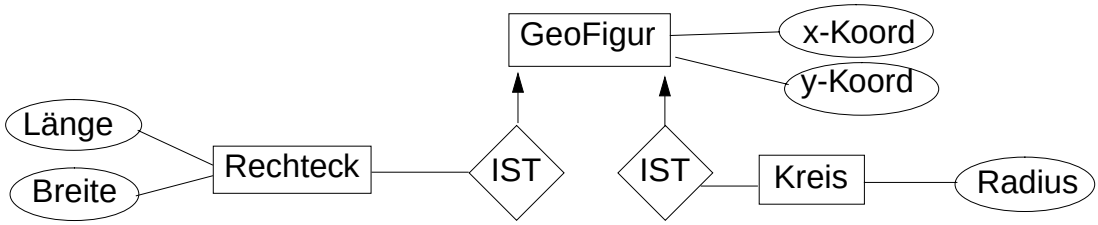
Jede konkrete Ausprägung zu A ist **Teilmenge** der konkreten Ausprägung zu B.
Es kann Entities in B geben, die nicht in A sind.



Die **Entities in A „erben“ alle Attribute von B** und können noch weitere Attribute haben, die **spezielle A-Eigenschaften** beschreiben.

Auch **Schlüsselattribute** werden als solche **geerbt**.





Vorlesung Modellierung WS 2001/2002 / Folie 516

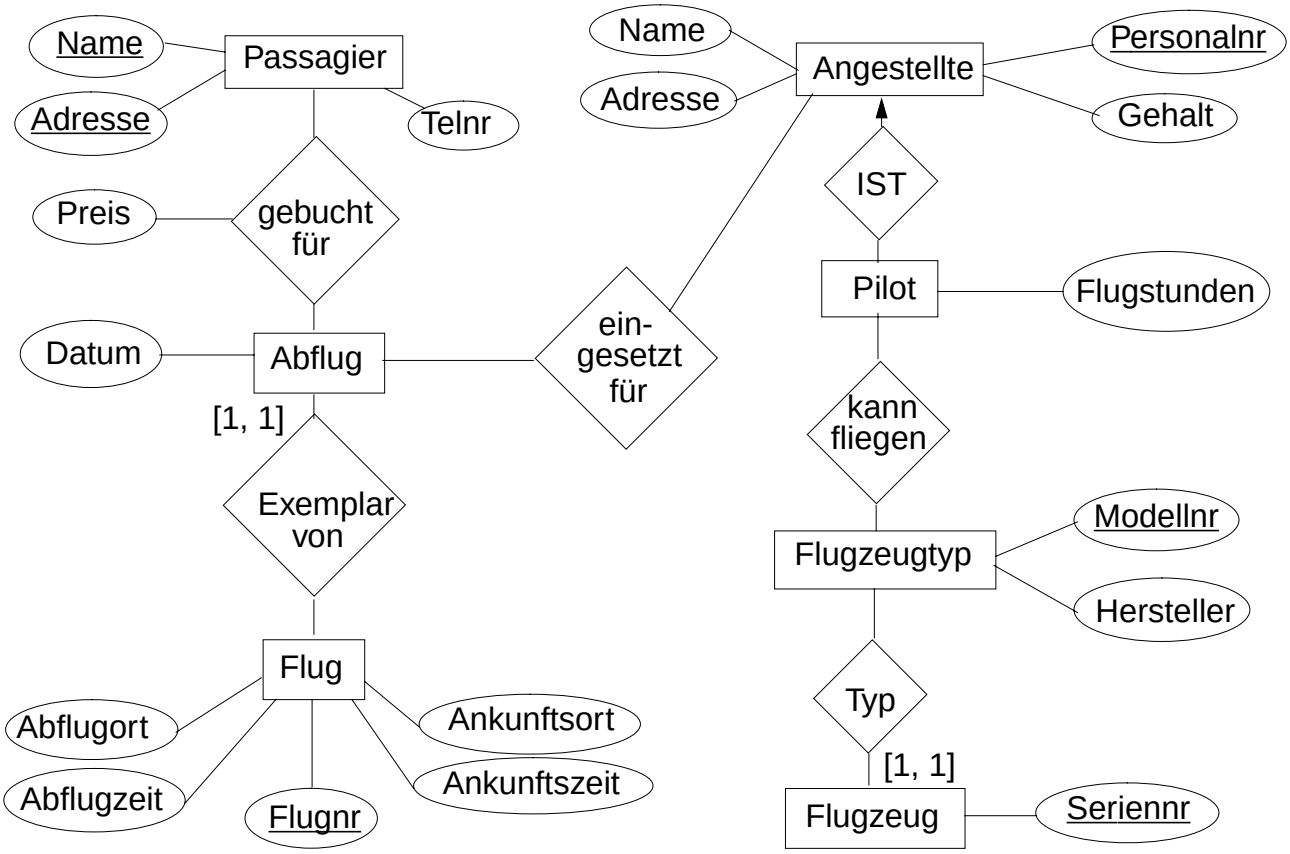
Ziele:

Konzept der Spezialisierung verstehen

in der Vorlesung:

- Erläuterungen dazu.
- Jede Entity existiert weiterhin nur einmal. Sie kann aber zu mehreren Mengen (A und B) gehören.
- Bei der Modellierung von mehreren IST-Relationen zu derselben allgemeinen Entity-Menge sind die speziellen Mengen meist disjunkt (z. B. Rechteck und Kreis). Das ist aber formal nicht vorgeschrieben.
- Entspricht der Vererbung zwischen Ober- und Unterklassen in objektorientierten Programmiersprachen.

Beispiel: Fluggesellschaft



Vorlesung Modellierung WS 2001/2002 / Folie 517

Ziele:

ER-Modellierung im Zusammenhang sehen

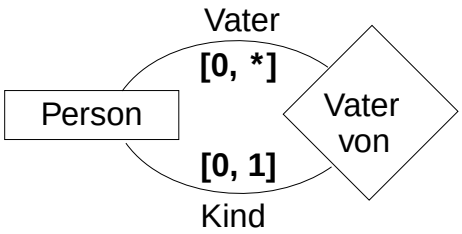
in der Vorlesung:

Erläuterungen zu

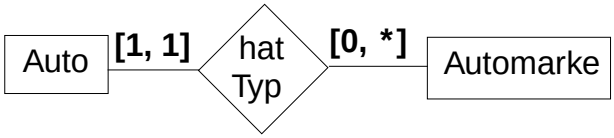
- Schema: "Exemplar von", "Typ"
- Schlüsselattributen

Hinweise zur Modellierung mit ER

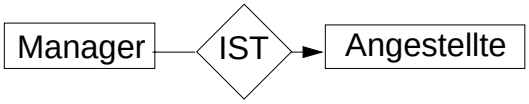
- In einem ER-Modell kommt **jede Entity-Menge nur einmal** vor.
- **Rollen** zu Relationen **angeben**, wo es nötig ist.
- Bedeutung der Kardinalitäten klarstellen.



- **Typ - Exemplar - Relationen** bewusst einsetzen.



- **Spezialisierung** sinnvoll einsetzen.



- Typ - Exemplar - Relation **nicht** mit Spezialisierung **verwechseln**

Vorlesung Modellierung WS 2001/2002 / Folie 518

Ziele:

Einige Modellierungsregeln

in der Vorlesung:

Erläuterungen dazu mit Hinweis auf Beispiele

6. Modellierung von Abläufen
6.1 Endliche Automaten

Endlicher Automat:
Formaler Kalkül zur **Spezifikation von realen oder abstrakten Maschinen**. Sie

- reagieren auf **äußere Ereignisse**,
- ändern ihren **inneren Zustand**,
- produzieren ggf. **Ausgabe**.

Endliche Automaten werden **eingesetzt**, um

- das **Verhalten realer Maschinen** zu spezifizieren, z. B. Getränkeautomat,
- das **Verhalten von Software-Komponenten** zu spezifizieren,
  z. B. Reaktionen von Benutzungsoberflächen auf Bedienereignisse,
- **Sprachen zu spezifizieren**: Menge der Ereignis- oder Symbolfolgen, die der Automat akzeptiert, z. B. Schreibweise von Bezeichnern und Zahlwerten in Programmen

Zunächst definieren wir nur die **Eingabeverarbeitung** der Automaten;
das Erzeugen von **Ausgabe** fügen wir **später** hinzu.

Vorlesung Modellierung WS 2001/2002 / Folie 601

Ziele:

Charakterisierung endlicher Automaten

in der Vorlesung:

Erläuterungen dazu

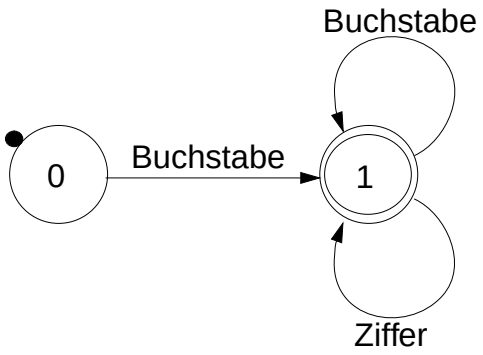
nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.4

Zwei einführende Beispiele

Endlicher Automat definiert eine **Sprache**, d. h. eine Menge von Wörtern.
Ein Wort ist eine Folge von Zeichen.

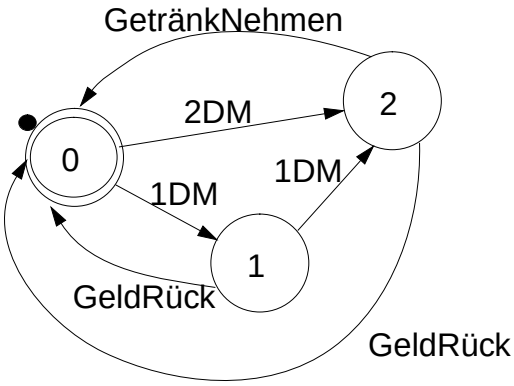
Hier: **Bezeichner** in Pascal-Programmen:



Akzeptiert Folgen von Buchstaben und Ziffern beginnend mit einem Buchstaben.

Endlicher Automat spezifiziert das **Verhalten einer Maschine**.

Hier: einfacher **Getränkeautomat**:



Akzeptiert Folgen von Ereignissen zur Bedienung eines Getränkeautomaten

Endliche Automaten können durch **gerichtete, markierte Graphen** dargestellt werden, **Ablaufgraphen**.

Vorlesung Modellierung WS 2001/2002 / Folie 602

Ziele:

Eindruck von Automaten und ihrer Darstellung

in der Vorlesung:

Informelle Erläuterungen zu

- Zuständen,
- Übergängen,
- äßeren Ereignissen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.4

Alphabete

Alphabet: Eine **Menge von Zeichen** zur Bildung von Zeichenfolgen, häufig mit Σ bezeichnet.

Wir betrachten hier nur endliche Alphabete, z. B.

{0, 1}, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, {a, b, ..., z}

Ein **Wort über einem Alphabet** Σ ist eine **Zeichenfolge** aus Σ^*

statt $(a_1, a_2, ..., a_n) \in \Sigma^*$ schreiben wir auch $a_1\ a_2\ ...\ a_n$, z. B. $10010 \in \{0, 1\}^*$

für die leere Folge ( ) schreiben wir ε (epsilon)

Vorlesung Modellierung WS 2001/2002 / Folie 603

Ziele:
Wörter über Alphabeten

in der Vorlesung:
Erläuterungen und Beispiele dazu

nachlesen:
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.4

Deterministischer endlicher Automat

Deterministischer endlicher Automat (engl.: deterministic finite automaton, DFA):

Quintupel $A = (\Sigma, Q, \delta, q_0, F)$ mit

Σ	endliches **Eingabealphabet**
Q	endliche **Menge von Zuständen**
δ	**Übergangsfunktion** aus $Q \times \Sigma \rightarrow Q$
$q_0 \in Q$	**Anfangszustand**
$F \subseteq Q$	**Menge der Endzustände** (akzeptierend)

Wir nennen $r = \delta(q, a)$ **Nachfolgezustand von q unter a**.

A heißt **deterministisch, weil** es **zu jedem Paar (q, a)**, mit $q \in Q$, $a \in \Sigma$, **höchstens einen Nachfolgezustand** $\delta(q, a)$ gibt.

A heißt **vollständig, wenn** die **Übergangsfunktion** δ eine **totale** Funktion ist.

Vorlesung Modellierung WS 2001/2002 / Folie 604

Ziele:

Formale Definition verstehen

in der Vorlesung:

Erläuterungen zu

- den Komponenten des 5-Tupels,
- dem Begriff "deterministisch",
- der Eigenschaft "vollständig"

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.4

Gerichteter Graph zu endlichem Automaten

Knoten: **Zustände** des Automaten; Anfangszustand und Endzustände werden speziell markiert
Kanten: **Übergangsfunktion**, q -> r markiert mit a, genau dann wenn δ(q, a) = r
Es gibt Kanten, die sich nur durch ihre Markierung unterscheiden, deshalb: **Multigraph**

Beispiele von Mod-6.2:

Σ := Menge der ASCII-Zeichen

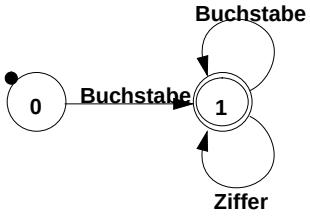
Q := {0, 1}

δ :=

	a...zA...Z	0...9	sonstige
0	1		
1	1	1	

q_0 = 0

F = {1}



Buchstabe, Ziffer sind **Zeichenklassen**

Σ := {**1DM, 2DM, GeldRück, GetränkNehmen**}

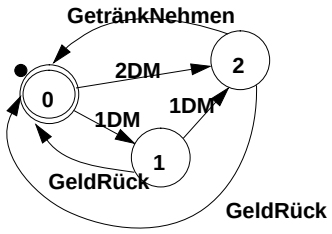
Q := {0, 1, 2}

δ :=

	1DM	**2DM**	**GeldRück**	**GetränkNehmen**
0	**1**	**2**		
1	**2**		**0**	
2			**0**	**0**

q_0 = 0

F = {0}



Vorlesung Modellierung WS 2001/2002 / Folie 605

Ziele:

Graphdarstellung verstehen

in der Vorlesung:

- Übergangsfunktion ist als Tabelle angegeben
- Markierung von Anfangs- und Endzuständen
- Zusammenfassung von Zeichen mit gleichen Übergängen zu Zeichenklassen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.4

Akzeptierte Sprache

Die Zeichen einer Zeichenfolge bewirken nacheinander Zustandsübergänge in Automaten.

Zustandsübergangsfunktion erweitert für Zeichenfolgen:

Sei $\delta : Q \times \Sigma \to Q$ eine **Übergangsfunktion für Zeichen**,
dann ist $\hat{\delta} : Q \times \Sigma^* \to Q$ eine **Übergangsfunktion für Wörter**, definiert durch

- Übergang mit dem **leeren Wort**: $\hat{\delta}(q, \varepsilon) = q$ für alle $q \in Q$
- Übergang mit dem **Wort wa**: $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$ für alle $q \in Q$, $w \in \Sigma^*$, $a \in \Sigma$

Statt $\hat{\delta}$ schreiben wir meist auch δ.

Sei $A = (\Sigma, Q, \delta, q_0, F)$ ein deterministischer endlicher Automat und $w \in \Sigma^*$.
A akzeptiert das Wort w genau dann, wenn $\delta(q_0, w) \in F$.
Die Menge $L(A) := \{ w \in \Sigma^* \mid \delta(q_0, w) \in F \}$ heißt die **von A akzeptierte Sprache**.

Beispiele für Sprachen, die von endlichen Automaten akzeptiert werden können:
$L = \{a^n b^m \mid n, m \in \mathbb{N}\}$, $L = \Sigma^*$

Es gibt keinen endlichen Automaten, der $\{a^n b^n | n \in \mathbb{N}\}$ akzeptiert.

Vorlesung Modellierung WS 2001/2002 / Folie 606

Ziele:

Sprache eines endlichen Automaten verstehen

in der Vorlesung:

Erläuterungen

- zur Übergangsfunktion für Wörter,
- zur Sprache des Automaten,
- zu Beispielen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.4

Nicht-deterministischer Automat

Nicht-deterministisch (allgemein) :
Es gibt mehrere Möglichkeiten der Entscheidung bzw. der Fortsetzung, es ist aber nicht festgelegt, welche gewählt wird.

Nicht-deterministischer Automat:
Die **Übergangsfunktion** δ kann einen Zustand q und ein Eingabezeichen a auf **mehrere Nachfolgezustände** abbilden $\delta : Q \times \Sigma \rightarrow Pow(Q)$.
Welcher gewählt wird, ist nicht festgelegt.
Σ , Q, q_0, F sind wie für deterministische endliche Automaten definiert.

Erweiterung von δ auf Zeichenfolgen:
Sei $A = (\Sigma , Q, \delta , q_0, F)$ ein nicht-deterministischer endlicher Automat;

dann ist $\hat{\delta}$ definiert durch

- Übergang mit dem **leeren Wort**: $\hat{\delta}(q, \varepsilon) = q$ für alle $q \in Q$
- Übergang mit dem **Wort wa**: $\hat{\delta}(q, wa) = \{q' \in Q \mid \exists p \in \hat{\delta}(q, w): q' \in \delta(p, a)\}$
  für alle $q \in Q, w \in \Sigma^*, a \in \Sigma$

Wir schreiben meist δ für $\hat{\delta}$

Ein nicht-deterministischer endlicher Automat A **akzeptiert** ein Wort w gdw. $\delta(q_0, w) \cap F \neq \emptyset$
$L(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset \}$ ist **die von A akzeptierte Sprache**.

Vorlesung Modellierung WS 2001/2002 / Folie 607

Ziele:

Nicht-Determiniertheit verstehen

in der Vorlesung:

Erläuterungen

- zur Übergangsfunktion an Beispielen,
- zur Erweiterung der Übergangsfunktion,
- zur Nicht-Determiniertheit im Automaten und im allgemeinen.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.4

Nicht-deterministische und deterministische Automaten

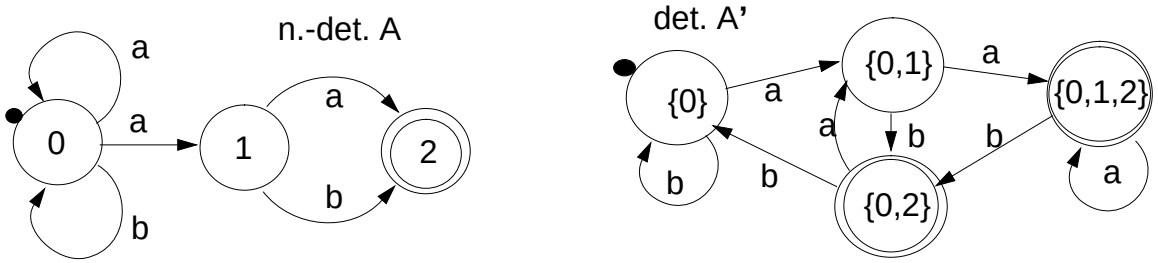
Satz: Sei L(A) die Sprache eines nicht-deterministischen Automaten.
Dann gibt es einen deterministischen Automaten, der L(A) akzeptiert.

Man kann **aus einem nicht-deterministischen Automaten A** = $(\Sigma , Q , \delta, q_0 , F)$
einen **deterministischen A'** = $(\Sigma, Q', \delta', q_0', F')$ systematisch **konstruieren**:

Jeder **Zustand aus Q' repräsentiert eine Menge von Zuständen aus Q, d. h. Q' $\subseteq$ Pow(Q)**

- **Anfangszustand**: $q_0' = \{q_0\}$
- **Übergangsfunktion, Nachfolgezust.**: $\forall q' \in Q': \forall a \in \Sigma: \delta' (q', a) = \bigcup_{q \in q'} \delta (q, a)$
  d. h. q' repräsentiert die Vereinigung aller Zustände, die in A von q unter a erreicht werden.
- **Endzustände**: $F' = \{q' \in Q' \mid q' \cap F \neq \emptyset\}$
  d. h. q' ist Endzustand, wenn seine Zustandsmenge einen Endzustand von A enthält.

Beispiel:



Die Zahl der Zustände kann sich dabei **exponentiell** vergrößern.

Vorlesung Modellierung WS 2001/2002 / Folie 608

Ziele:

Konstruktionsprinzip verstehen

in der Vorlesung:

- Erläuterungen zur Konstruktion,
- Zusammenhang: Zustand - Menge von Zuständen,
- Konstruktion am Beispiel,
- L(A): Wörter über {a, b}*, deren zweitletztes Zeichen ein a ist.
- Bei n-letztem Zeichen benötigt der deterministische Automat 2 hoch n Zustände.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.4

Endliche Automaten mit Ausgabe

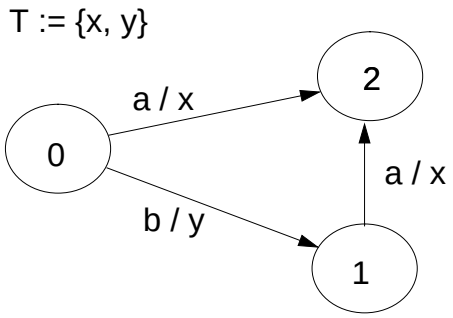
Man kann mit endlichen Automaten auch **Reaktionen der modellierten Maschine** spezifizieren: **Automaten mit Ausgabe**.

Wir erweitern den Automaten um ein **endliches Ausgabealphabet T** und um eine Ausgabefunktion. Es gibt 2 Varianten für die Ausgabefunktion:

Mealy-Automat:

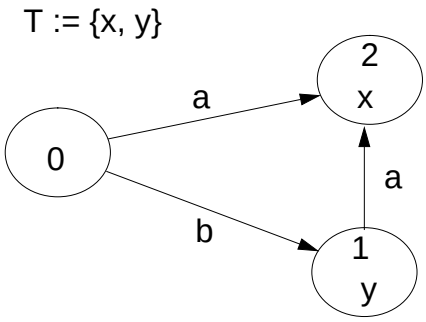
Eine Ausgabefunktion $\lambda : Q \times \Sigma \rightarrow T^*$ ordnet den **Zustandsübergängen** jeweils ein **Wort über dem Ausgabealphabet** zu.

Graphische Notation:



Moore-Automat:

Eine Ausgabefunktion $\mu : Q \rightarrow T^*$ ordnet den **Zuständen** jeweils ein **Wort über dem Ausgabealphabet** zu. Es wird bei Erreichen des Zustands ausgegeben.



Vorlesung Modellierung WS 2001/2002 / Folie 609

Ziele:

Zwei Ausgabevarianten

in der Vorlesung:

- Erläuterungen dazu;
- Wenn keine Ausgabe angegeben ist, wird das leere Wort als Ausgabe angenommen.
- Mealy- und Moore-Automaten werden auch so definiert, dass jeweils ein Zeichen statt ein Wort ausgegeben werden.

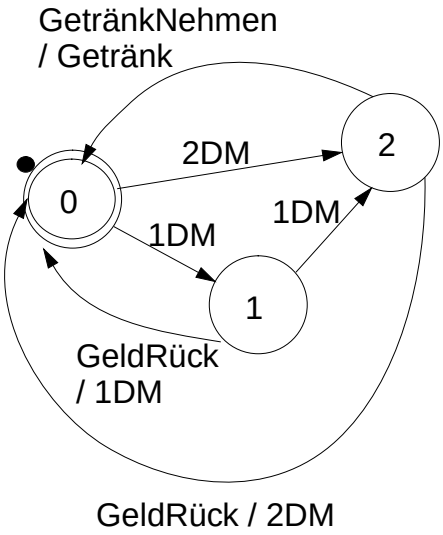
nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.4

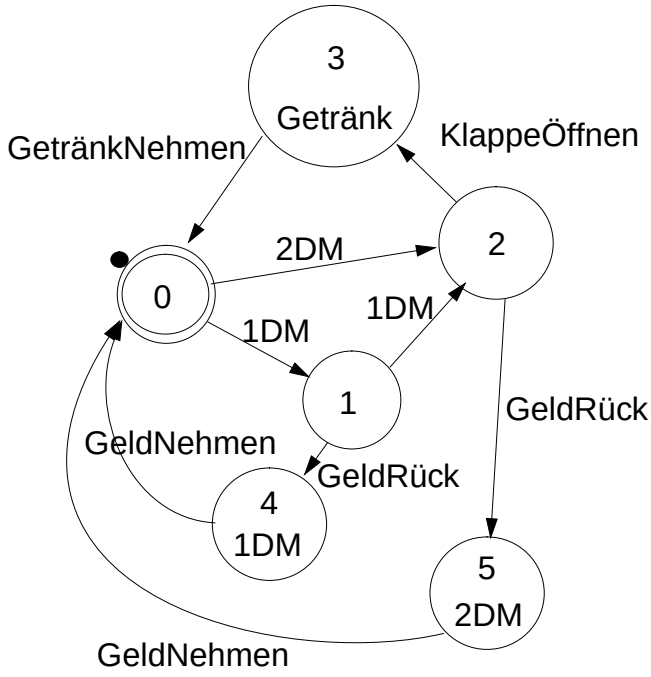
Beispiele für endliche Automaten mit Ausgabe

Die Spezifikation des Getränkeautomaten aus Mod-6.2 wird mit Ausgabe versehen:

Mealy-Automat



Moore-Automat



Vorlesung Modellierung WS 2001/2002 / Folie 610

Ziele:

Ausgabe zuordnen

in der Vorlesung:

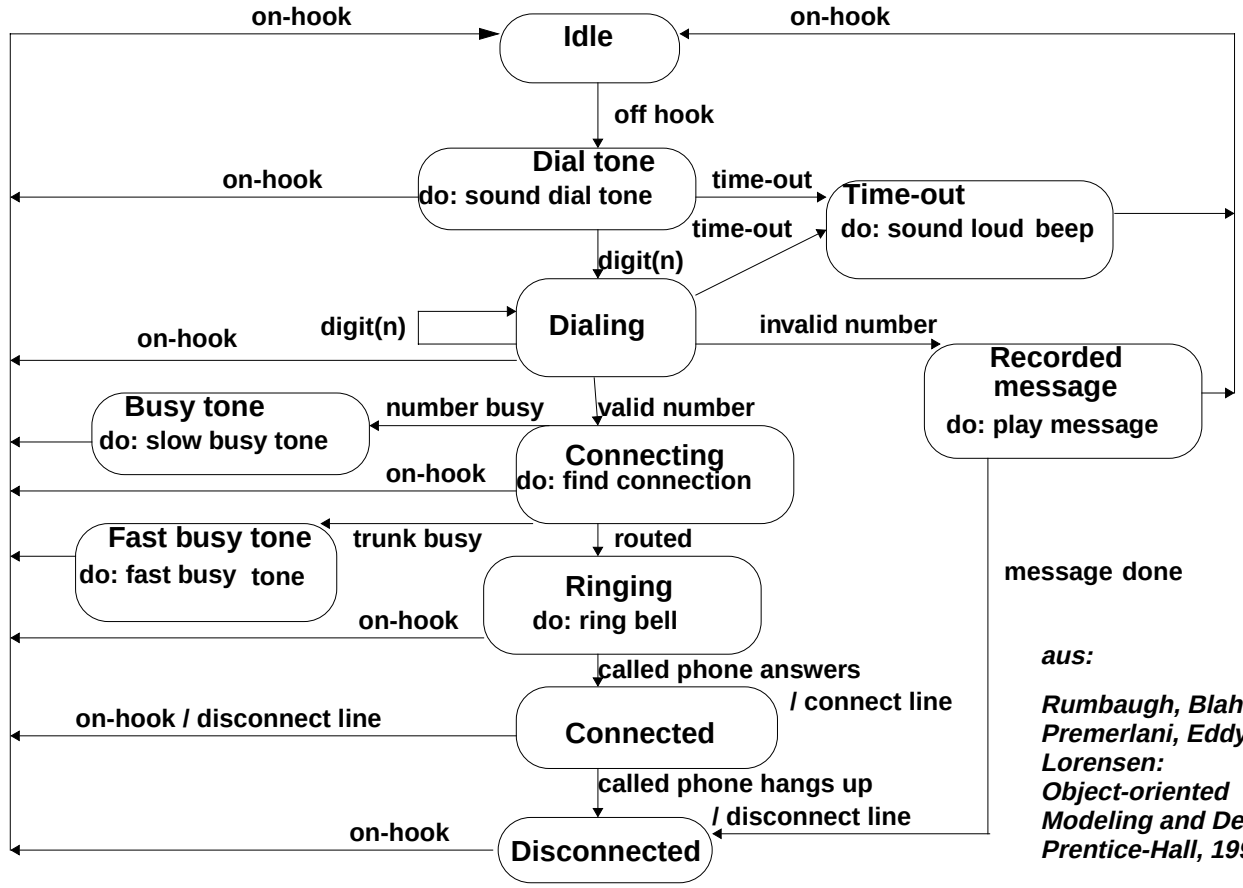
Erläuterungen dazu

- Mealy-Automat erläutern
- An einigen Positionen bleibt die Ausgabe leer.
- Moore-Automat erläutern
- Zusätzliche Zustände begründen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.4

Endlicher Automat zur Telefonbedienung



aus:

Rumbaugh, Blaha, Premerlani, Eddy, Lorensen: Object-oriented Modeling and Design, Prentice-Hall, 1991

Vorlesung Modellierung WS 2001/2002 / Folie 611

Ziele:

Praktisches Modellierungsbeispiel sehen

in der Vorlesung:

- Erläuterungen dazu
- Eingabe sind Ereignisse beim Telefonieren
- Ausgabe sind ausgelöste Aktionen
- Ausgabe ist sowohl einigen Zuständen (do:...) als auch einigen Übergängen (/ ...) zugeordnet.

6.2 Petri-Netze

Petri-Netz (auch Stellen-/Transitions-Netz):
Formaler Kalkül zur **Modellierung von Abläufen mit nebenläufigen Prozessen** und kausalen Beziehungen

Basiert auf **bipartiten gerichteten Graphen**:

- **Knoten** repräsentieren **Bedingungen**, Zustände bzw. **Aktivitäten**.
- **Kanten** verbinden **Aktivitäten** mit ihren **Vor- und Nachbedingungen**.
- **Knotenmarkierung** repräsentiert den veränderlichen **Zustand des Systems**.
- **graphische Notation**.

C. A. Petri hat sie 1962 eingeführt.
Es gibt zahlreiche Varianten und Verfeinerungen von Petri-Netzen. Hier nur die Grundform.

Anwendungen von Petri-Netzen zur Modellierung von

- realen oder abstrakten Automaten und Maschinen
- kommunizierenden Prozessen in der Realität oder in Rechnern
- Verhalten von Hardware-Komponenten
- Geschäftsabläufe
- Spielpläne

Vorlesung Modellierung WS 2001/2002 / Folie 612

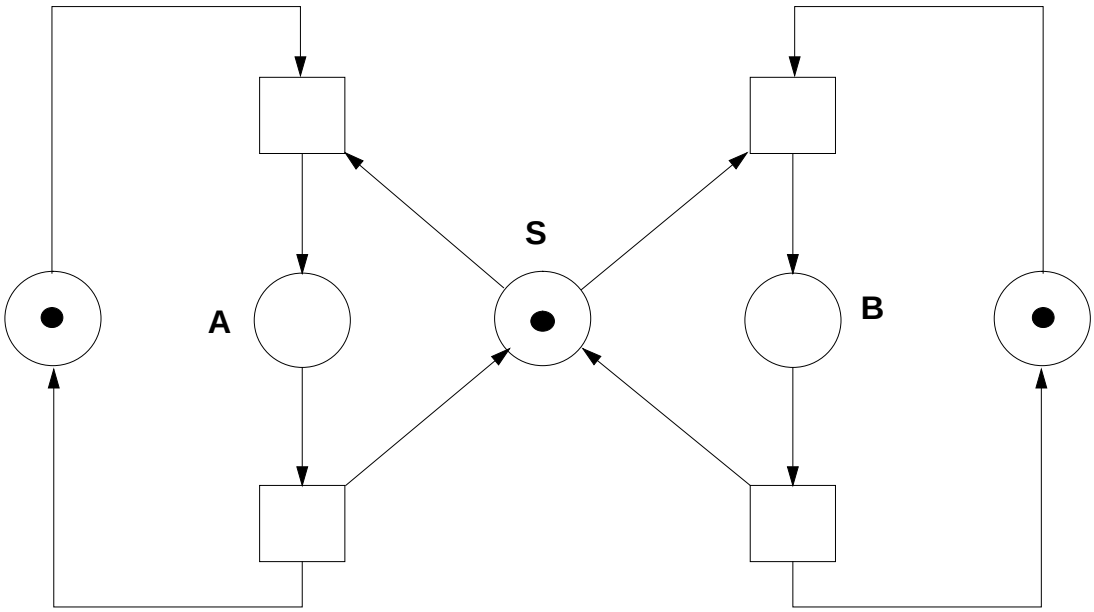
Ziele:
Einführung zu Petri-Netzen

in der Vorlesung:
Erläuterungen dazu

nachlesen:
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.5

Einführendes Beispiel

Das Petri-Netz modelliert zwei **zyklisch ablaufende Prozesse**.
Die mittlere Stelle synchronisiert die beiden Prozesse,
so dass sie sich **nicht zugleich in den Zuständen A und B** befinden können.
Prinzip: **gegenseitiger Ausschluss** durch **Semaphor**



Vorlesung Modellierung WS 2001/2002 / Folie 613

Ziele:

Eindruck von Petri-Netzen

in der Vorlesung:

informelle Erläuterungen zu

- parallelen Prozessen
- gegenseitigem Ausschluss
- Markierung und Schalten in Petri-Netzen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.5

Definition von Petri-Netzen

Ein **Petri-Netz** ist ein Tripel P = (S, T, F) mit

S **Menge von Stellen**,
repräsentieren Bedingungen, Zustände; graphisch Kreise

T **Menge von Transitionen** oder Übergänge,
repräsentieren Aktivitäten; graphisch Rechtecke

F **Relation** mit $F \subseteq S \times T \cup T \times S$
repräsentieren kausale oder zeitliche Vor-, Nachbedingungen von Aktivitäten aus T

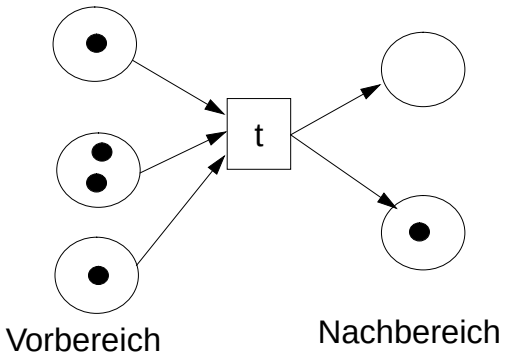
P bildet einen **bipartiten, gerichteten Graphen** mit den Knoten $S \cup T$ und den Kanten F.

Zu einer **Transition t** in einem Petri-Netz P sind die Stellenmengen definiert

Vorbereich (t) := { s | (s, t) ∈ F}

Nachbereich (t) := { s | (t, s) ∈ F}

Der **Zustand des Petri-Netzes** wird durch eine **Markierungsfunktion** angegeben, die jeder Stelle eine **Anzahl von Marken** zuordnet:
$M_P: S \rightarrow \mathbb{N}_0$



Vorlesung Modellierung WS 2001/2002 / Folie 614

Ziele:

Petri-Netz formal verstehen

in der Vorlesung:

Erläuterungen zu den Begriffen

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.5

Verständnisfragen:

Welche Arten von Kanten kann es in einem Petri-Netz nicht geben?

Schaltregel für Petri-Netze

Das **Schalten einer Transition** t überführt eine Markierung M in eine Markierung M‘.

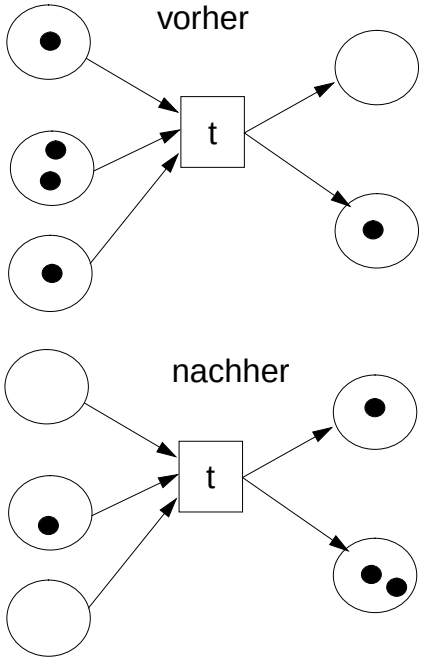
Eine **Transition t kann schalten**, wenn für alle Stellen s ∈ Vorbereich (t) gilt M(s) ≥ 1.

Wenn eine Transition t **schaltet,** gilt für die **Nachfolgemarkierung M‘**:

M‘(v) = M(v) - 1 für alle
v ∈ Vorbereich(t) \ Nachbereich(t)

M‘(n) = M(n) + 1 für alle
n ∈ Nachbereich(t) \ Vorbereich(t)

M‘(s) = M(s) sonst



Wenn in einem Schritt **mehrere Transitionen schalten können**, wird eine davon **nicht-deterministisch ausgewählt**.

Wir sagen, eine **Markierung M_2 ist von einer Markierung M_1 aus erreichbar**, wenn es ausgehend von M_1 eine Folge von Transitionen gibt, die nacheinander schalten und M_1 in M_2 überführen.

Zu jedem Petri-Netz wird eine **Anfangsmarkierung M_0** angeben.

Vorlesung Modellierung WS 2001/2002 / Folie 615

Ziele:

Schaltregel verstehen

in der Vorlesung:

- Schaltregel erläutern
- nicht-deterministische Auswahl zeigen,
- Konflikt zwischen mehreren Transitionen, die Schalten können: Vorbereiche sind nicht disjunkt

nachlesen:

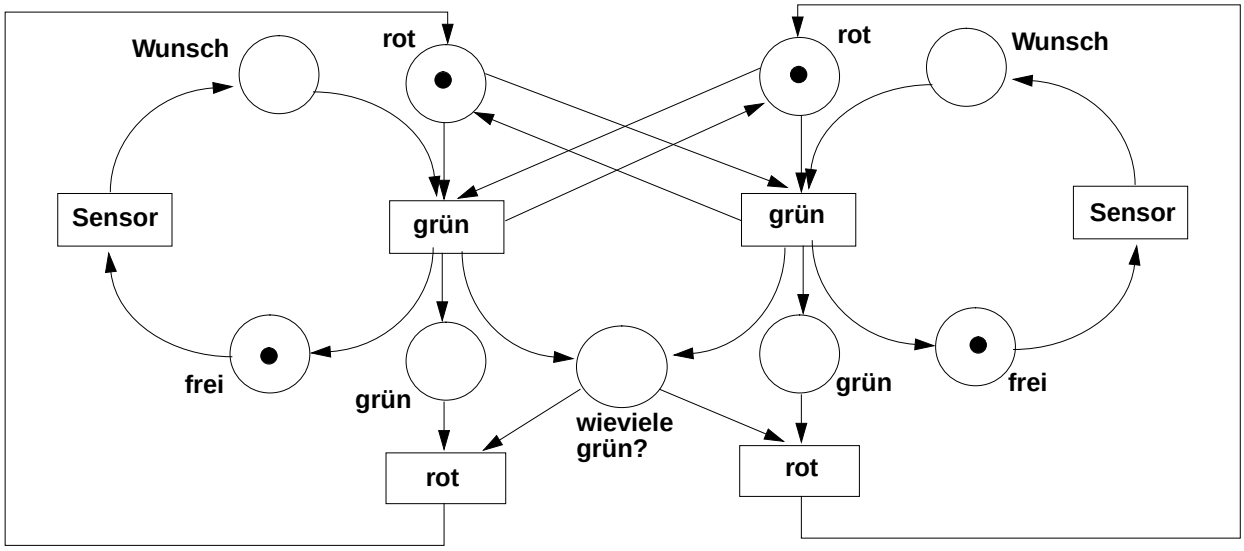
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.5

Beispiel für ein binäres Netz

Ein Petri-Netz heißt **binär (sicher)**, wenn für alle aus M_0 erreichbaren Markierungen M und für alle Stellen s gilt $M(s) \leq 1$.

Petri-Netze, deren **Stellen Bedingungen repräsentieren** müssen binär sein.

Beispiel: Modellierung einer Sensor-gesteuerten Ampelkreuzung:



aus: B. Baumgarten: Petri-Netze, Bibliographisches Institut & F. A. Brockhaus AG, 1990

Vorlesung Modellierung WS 2001/2002 / Folie 616

Ziele:

Stellen als Bedingungen verstehen

in der Vorlesung:

- Erläuterungen zu dem Beispiel,
- Vor- und Nachbedingungen diskutieren,
- Eigenschaften diese Modells diskutieren

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.5

Lebendige Petri-Netze

Petri-Netze modellieren häufig **Systeme, die nicht anhalten** sollen.

Ein Petri-Netz heißt **schwach lebendig**, wenn es zu jeder von M_0 erreichbaren Markierung eine Nachfolgemarkierung gibt.

Eine **Transition t heißt lebendig**, wenn es zu jeder von M_0 erreichbaren Markierung M' eine Markierung M" gibt, die von M' erreichbar ist, und in der t schalten kann.

Ein **Petri-Netz heißt lebendig**, wenn alle seine Transitionen lebendig sind.

Verklemmung: Ein System kann unerwünscht anhalten,
weil das **Schalten einiger Transitionen zyklisch voneinander abhängt**.

Sei: $\sigma \subseteq S$ eine Teilmenge der Stellen eines Petri-Netzes und

Vorbereich (σ) $:= \{t \mid \exists\, s \in \sigma : (t, s) \in F\}$,
d. h. die Transitionen, die auf Stellen in σ wirken

Nachbereich (σ) $:= \{t \mid \exists\, s \in \sigma : (s, t) \in F\}$,
d. h. die Transitionen, die Stellen in σ als Vorbedingung haben

Dann ist σ eine **Verklemmung, wenn Vorbereich** $(\sigma) \subseteq$ **Nachbereich** (σ).

Wenn **für alle** $s \in \sigma$ **gilt M (s) = 0**, dann kann es **keine Marken auf Stellen in** σ in einer Nachfolgemarkierung von M geben.

Vorlesung Modellierung WS 2001/2002 / Folie 617

Ziele:

Begriffe zur Lebendigkeit von Netzen verstehen

in der Vorlesung:

Erläuterungen zu

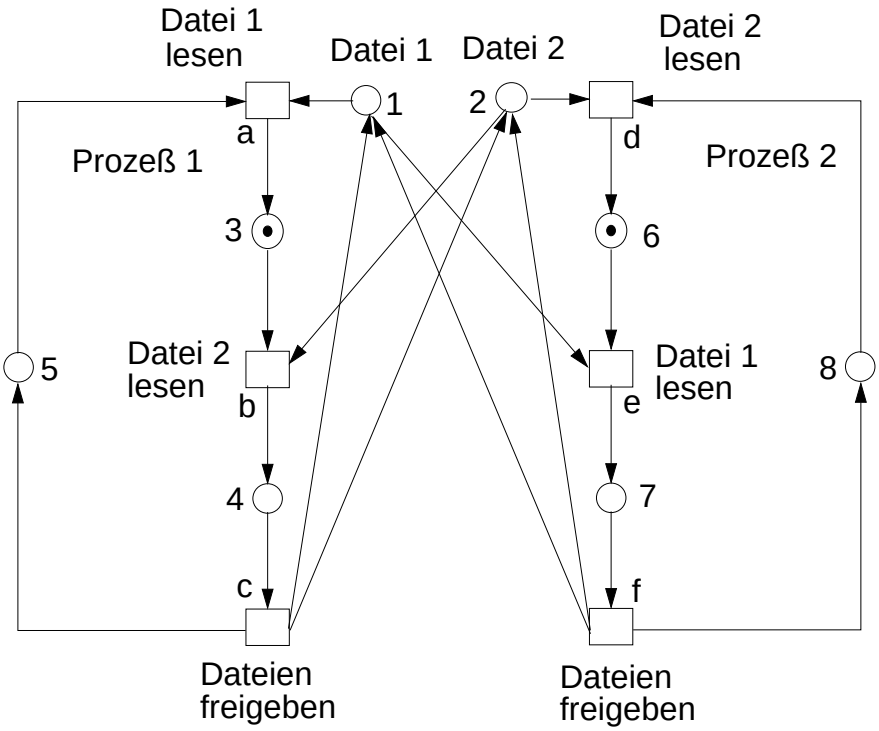
- nicht-terminierenden Systemen,
- Lebendigkeitsbegriffen,
- Verklemmungen

am Beispiel von Mod-6.18

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.5

Verklemmung beim Lesen von Dateien



$\sigma = \{1, 2, 4, 5, 7, 8\}$

Vorbereich (σ) $= \{b, c, e, f\}$

Nachbereich (σ) $= \{a, b, c, d, e, f\}$

$M(\sigma) = 0$

Vorlesung Modellierung WS 2001/2002 / Folie 618

Ziele:

Beispiel für eine Verklemmung

in der Vorlesung:

Erläuterung:

- Jeder der Prozesse fordert nacheinander zwei Dateien an und gibt sie dann beide wieder frei.
- Die Verklemmung tritt ein, wenn jeder Prozess eine Datei belegt und auf die andere wartet.
- Sigma charakterisiert diese Situation.
- Es gibt verschiedene Techniken, die Verklemmung zu vermeiden, z. B.
- Bei einem Prozess die Reihenfolge der Dateien vertauschen.
- Beide Dateien zugleich anfordern.

nachlesen:

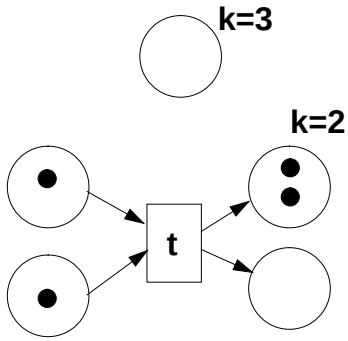
G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.5

Kapazitäten und Gewichte

Man kann **Stellen eine begrenzte Kapazität** von $k \in \mathbb{N}$ Marken zuordnen.

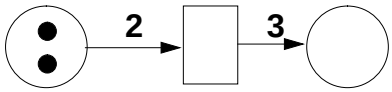
Die Bedingung, dass eine **Transition t schalten kann**, wird erweitert um:

Die **Kapazität keiner der Stellen im Nachbereich von t darf überschritten** werden.

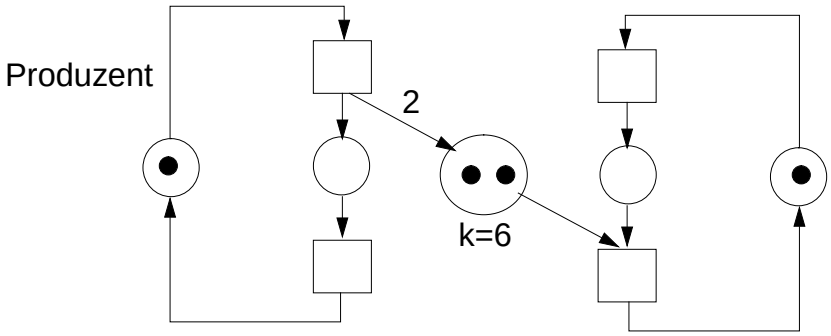


t kann nicht schalten

Kanten kann ein **Gewicht** $n \in \mathbb{N}$ zugeordnet werden: sie bewegen **beim Schalten n Marken**.



Beispiel: **Beschränkter Puffer**



Vorlesung Modellierung WS 2001/2002 / Folie 619

Ziele:

Konzepte verstehen

in der Vorlesung:

Erläuterung der beiden Konzepte am Beispiel.

- Schaltregel wird ergänzt.
- Produzent liefert immer 2 Einheiten zugleich (Kantengewicht 2).
- Produzent kann nur liefern (schalten), wenn die Pufferstelle noch freie Kapazitähat.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.5

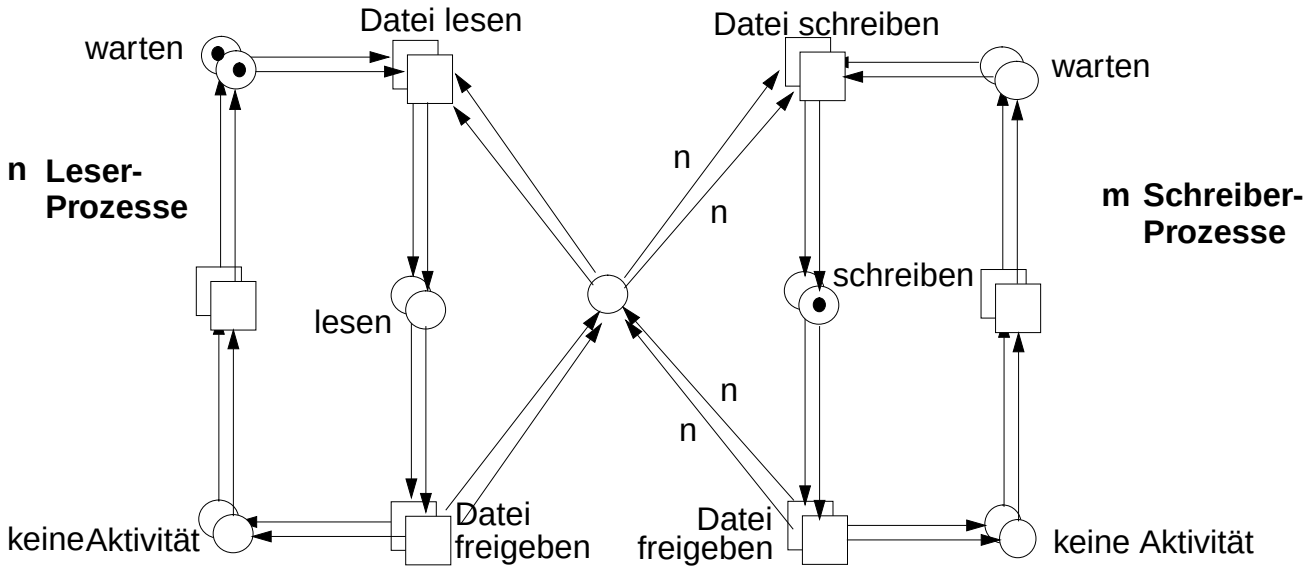
Beispiel: Leser-Schreiber-System

n Leser-Prozesse und **m Schreiber-Prozesse** operieren **auf derselben Datei**.

Mehrere **Leser können zugleich lesen**.

Ein **Schreiber** darf nur dann schreiben, wenn **kein anderer Leser oder Schreiber aktiv** ist.

Modellierung: ein **Schreiber entzieht der Synchronisationsstelle alle n Marken**.



Vorlesung Modellierung WS 2001/2002 / Folie 620

Ziele:

Beispiel für Kapazitäten und Gewichte

in der Vorlesung:

- Erläuterung des Leser-Schreiber-Systems.
- Allerdings können wechselnde Leser die Schreiber auf Dauer blockieren. Das Petri-Netz ist nicht fair.

nachlesen:

G. Goos: Vorl. über Informatik Bd.1, Abschnitt 2.5

7. Zusammenfassung Wunschkonzert

Thema	Häufigkeit	Platz
nichts	22	
Wertebereiche	45	4
Funktionen, Relationen	41	5
Terme, Signaturen	32	
Substitution, Unifikation	22	
Algebren	30	
Aussagenlogische Formeln	11	
Aussagenlogische Schlüsse	**51**	**3**
Verifikation	**157**	**1**
Prädikatenlogik: Interpretation	**59**	**2**
Prädikatenlogik: Umformungen	35	6
Graphen: Wegeprobleme	2	
Graphen: Verbindungsprobleme	0	
Graphen: Bäume	1	
Graphen: Zuordnungsprobleme	2	
Graphen: Abhängigkeiten	5	
KFG: Bäume	5	
KFG: Ableitungen	9	
Entity−Relationship−Modell	15	
ER: Kardinalitäten	7	
Endliche Automaten	12	
Petri−Netze	7	
Sonstige	0	
Summe der Stimmen:	**570**	

Vorlesung Modellierung WS 2001/2002 / Folie 701

Ziele:

Wiederholungswünsche zeigen.

Schlussregeln für Algorithmenelemente

Sequenzregel:

$$\frac{\{P\}\ S_1\ \{Q\} \qquad \{Q\}\ S_2;\ \{R\}}{\{P\}\ S_1; S_2\ \{R\}}$$

Zuweisungsregel:

$\{\ P\ {}^{x}_{e}\ \}\ x := e;\ \{\ P\ \}$

Wo in P x vorkommt, muss in $P\ {}^{x}_{e}$ e stehen.

{P}
S_1;
{Q}
S_2
{R}

{ x = y*(1+q) + (r-y) }
r := r - y;
{ x = y*(1+q) + r }
q := 1 + q;
{ x = y*q + r }

↓ Schlussregel für die Anweisung anwenden

↓ Schlussregel für die Anweisung anwenden

Konsequenzregel:

$\{R\} \rightarrow \{Q\}$

in Ausführungsrichtung einfügen, wo nötig.

Vorlesung Modellierung WS 2001/2002 / Folie 702

Ziele:

Wiederholung mit veränderter Perspektive

2-seitige Alternative

$$\frac{\{P \wedge B\}\ S_1\ \{Q\} \qquad \{P \wedge \neg B\}\ S_2\ \{Q\}}{\{P\}\ \text{falls } B\text{: } S_1 \text{ sonst } S_2\ \{Q\}}$$

{ P }

falls B:

{P ∧ B}

S_1

{ Q }

sonst

{P ∧ ¬ B}

S_2

{ Q }

{ Q }

{ true }

falls a > 0:

{ true ∧ a > 0 }

{a > 0} b := a; {b > 0} → { b ≥ 0 }

{ b ≥ 0 }

sonst

{true ∧ ¬ (a > 0)}

{¬ (a > 0) } ≡ { -a ≥ 0 } b := - a; { b ≥ 0 }

{ b ≥ 0 }

{ b ≥ 0 }

Ziele:

Wiederholung mit veränderter Perspektive

Bedingte Anweisung

(Bedingter Schritt)

$\{P \wedge B\}\ S\ \{Q\}$
$P \wedge \neg B \rightarrow Q$

$\{P\}$ falls B: S $\{Q\}$

{ P }

falls B:
{P ∧ B}

S_1

{ Q }

// leere Alternative:
{P ∧ ¬ B} → { Q }

{ Q }

{ true }

falls a < 0:
{true ∧ a < 0}

{a < 0} → { -a > 0 }
a := - a;
{a > 0} → { a ≥ 0 }

{ a ≥ 0 }

// leere Alternative:
{true ∧ ¬ (a < 0)} → { a ≥ 0 }

{ a ≥ 0 }

Vorlesung Modellierung WS 2001/2002 / Folie 704

Ziele:

Wiederholung mit veränderter Perspektive

Schleifenregel

$$\frac{\{P \wedge B\}\ \ S\ \ \{P\}}{\{P\}\ \text{solange B wiederhole S}\ \{P \wedge \neg B\}}$$

Schleifeninvariante P

im Beispiel ist P:

$z * a^b = x^y \wedge b \geq 0$

{P}
solange B wiederhole
{P ∧ B}
S
{P}
{P ∧ ¬ B}

a := x; b := y; z := 1;
{a = x ∧ b = y ∧ z = 1} →
{$z * a^b = x^y \wedge b \geq 0$}
solange b > 0 wiederhole
{$z * a^b = x^y \wedge b \geq 0 \wedge b > 0$}
≡ {$z * a * a^{b-1} = x^y \wedge (b-1) \geq 0$}
b := b - 1;
{$z * a * a^b = x^y$ b ≥ 0}
z := z · a;
{$z * a^b = x^y \wedge b \geq 0$}

{$z * a^b = x^y \wedge b \geq 0 \wedge b \leq 0$}
≡ {$z * a^b = x^y \wedge b = 0$} → {$z = x^y$ }

Terminierung separat zeigen.

Vorlesung Modellierung WS 2001/2002 / Folie 705

Ziele:

Wiederholung mit veränderter Perspektive

Interpretation prädikatenlogischer Formeln

Prädikatenlogische Formel besteht aus:	**Beispiel:**	**Interpretation $\Im$ legt fest:**
	$\forall x\ P(x, f(x)) \wedge Q(g(a, z))$	Individuenbereich U := { ... }
atomaren Formeln	P(x, f(x)) Q(g(a, z))	
mit Prädikatssymbolen	P Q	Prädikate $\Im(P)$:= { (a, b) | ...}
Termen als Parametern	x f(x) g(a, z)	
Funktionssymbolen	f g a	Funktionen $\Im(f)(n)$ -> n+1
freien Variablen	z	Variablenwerte $\Im(z)$:= 3
gebundenen Variablen	x	
Formeln mit logischen Junktoren	$P(x, f(x)) \wedge Q(g(a, z))$	Verknüpfung gemäß AL
Quantor-Formeln	$\forall x\ P(x, f(x)) \wedge Q(g(a, z))$	Bindung jedes Wertes aus U an Variable; Auswertung gemäß Bedeutung von $\forall$ und $\exists$

Vorlesung Modellierung WS 2001/2002 / Folie 706

Ziele:

Wiederholung mit veränderter Perspektive

Auswertung von Quantor-Formeln

$\forall x\ F(x)$ jeden Wert aus U einsetzen und Ergebnisse mit $\wedge$ verknüpfen: $F(u_1) \wedge F(u_2) \wedge ...$

$\exists x\ F(x)$ jeden Wert aus U einsetzen und Ergebnisse mit $\vee$ verknüpfen: $F(u_1) \vee F(u_2) \vee ...$

Wenn man einen Wert findet, der das Ergebnis bestimmt, $F(u_i) = f$ bei $\forall$ bzw. $F(u_j) = w$ bei $\exists$ braucht man nicht alle aufzuzählen.

Beispiel:

$U := \{ 1, 2, 3 \}$ $\qquad \Im(P) := \{ (a, b) \mid a + b < 5 \}$

$\exists x$	$\forall y$	P(x, y)				
1	1	w	$\wedge$			
1	2	w	$\wedge$			
1	3	w	=	**w**	$\vee$	← hier steht das Ergebnis fest
2	1	w	$\wedge$			
2	2	w	$\wedge$			
2	3	f	=	f	$\vee$	
3	1	w	$\wedge$			
3	2	f	$\wedge$			
3	3	f	=	f	=	**w**

Ziele:

Wiederholung mit veränderter Perspektive

Aussagenlogischer Schluss

Annahme		**Folgerung**
Formelmenge $\mathfrak{A}$		Formel F
$\{A_1, ..., A_n\}$	**|=**	F

Der Schluss ist korrekt, wenn für **alle Interpretationen** $\Im$, die alle Formeln in $\mathfrak{A}$ erfüllen, auch $\Im$ (F) gilt. Die **Korrektheit** kann man so prüfen:

$A_1 \wedge ... \wedge A_n \wedge \neg F$ muss **unerfüllbar** sein.

Beispiele:

$\{ \neg p, \neg q\}$ **|=** $\neg (p \vee q)$

	$\neg p \wedge \neg q \wedge \neg(\neg (p \vee q))$	DeMorgan
$\equiv$	$(\neg p \wedge \neg q) \wedge \neg(\neg p \wedge \neg q)$	
$\equiv$	O unerfüllbar	

$\{ p \rightarrow q, q\}$ **|=** $p \wedge q$

	$(p \rightarrow q) \wedge q \wedge \neg(p \wedge q)$	Implikation
$\equiv$	$(\neg p \vee q) \wedge q \wedge \neg(p \wedge q)$	DeMorgan
$\equiv$	$(\neg p \vee q) \wedge q \wedge (\neg p \vee \neg q)$	Distributiv.
$\equiv$	$(\neg p \vee (q \wedge \neg q)) \wedge q$	Komplem.
$\equiv$	$\neg p \wedge q$ erfüllbar; falscher Schluss	

Gegenbeispiel: p = f, q = w

Vorlesung Modellierung WS 2001/2002 / Folie 708

Ziele:

Wiederholung mit veränderter Perspektive

Zusammenfassung der Themen und Begriffe (1)

1. Modellbegriff

2.1 Wertebereiche beschrieben d. Mengen

Mengen, extensional, intensional, Operationen
Potenzmengen
Kartesisches Produkt
Indexmengen
Folgen
Relationen, Eigenschaften von Relationen
Funktionen, Eigenschaften,
spezielle Funktionen
disjunkte Vereinigung

2.2 Terme

Sorten, Signatur
korrekte Terme, Grundterme
Präfix-, Postfix-, Infix-Form, Funktionsform
Kantorowitsch-Bäume
Substitution
Umfassende Terme
Unifikation, allgemeinster Unifikator
Unifikationsverfahren
Abstrakte Algebra, Axiome
Konkrete Algebra
Datenstrukturen: Keller, Binärbaum
Konstruktor, Hilfskonstruktor, Projektion
Normalform

Ziele:

Verdeutlichen, was wir gelernt haben.

Zusammenfassung der Themen und Begriffe (2)

3.1 Aussagenlogik

AL Formeln, logische Junktoren
Belegung, Interpretation
Wahrheitstafeln
erfüllbar, unerfüllbar, allgemeingültig (Tautologie)
Gesetze der booleschen Algebra
aussagenlogischer Schluss, korrekter Schluss

3.3 Prädikatenlogik

PL Formeln,
gebundene und freie Variable
Wirkungsbereich von Quantoren
Umbenennung von Variablen
Interpretation von PL Formeln
Individuenbereich
Beschränkung von Wertebereichen
Umformungen
erfüllbar, unerfüllbar, allgemeingültig
PL Schluss

3.2 Verifikation (Hoaresche Logik)

Schlussregeln für Sequenz,
Zuweisung,
2-seitige Alternative,
bedingte Anweisung,
Konsequenz, Aufruf,
Schleife, Schleifeninvariante,
Schleife aus Invariante konstruieren
Terminierung von Schleifen

Ziele:

Verdeutlichen, was wir gelernt haben.

Zusammenfassung der Themen und Begriffe (3)

4. Graphen

4.1 Grundlegende Definitionen

Gerichtetet, ungerichteter Graph,
Multigraph, Teilgraph,
Grad, Eingangs-, Ausgangsgrad
Adjazenzmatrix, Adjazenzlisten

4.2 Wegeproblem

Weg, Kreis, Zyklus,
gerichteter azyklischer Graph,
zusammenhängend,
Zusammenhangskomponente,
Euler-Weg, Euler-Kreis, Hamilton-Kreis

4.3 Verbindungsprobleme

Baum, Spannbaum,
Schnittknoten, Brückenkante
orientierbarer Graph

4.4 Modellierung mit Bäumen

Gerichteter Baum, Wurzel, Höhe, Blätter
Binärbäume,
Entscheidungsbäume
Strukturbäume

4.5 Zuordnungsprobleme

Paarweise Zuordnung (Matching),
bipartit,
Färbung

4.6 Abhängigkeitsprobleme

Abhängigkeitsparagraph,
Anordnung (Scheduling),
Ablaufparagraph,
Aufrufgraph,
Programmablaufgraph

Ziele:

Verdeutlichen, was wir gelernt haben.

Zusammenfassung der Themen und Begriffe (4)

5. Modellierung von Strukturen

5.1 Kontextfreie Grammatiken

Terminale, Nichtterminale, Startsymbol
Produktionen,
Ableitung,
Sprache einer KFG,
Ableitungsbaum

5.2 Entity Relationship Modell

Entity-Menge, konkrete Ausprägung,
Attribut, Schlüsselattribut
Relation, Rollen, Kardinalität
IST-Spezialisierung

6. Modellierung von Abläufen

6.1 Endliche Automaten

Alphabet,
deterministisch, nicht-deterministisch
Zustände, Übergangsfunktion
Anfangszustand, Endzustände
akzeptierte Sprache
NEA-DEA-Konstruktion,
Ausgabe, Mealy-Automat, Moore-Automat

6.2 Petri-Netze

Stellen, Transitionen, Vorbereich, Nachbereich,
Markierungsfunktion, Schaltregel,
Anfangsmarkierung, erreichbare Markierungen
zyklische Prozesse,
binäres Netz,
Verklemmung (deadlock),
Kapazitäten, Gewichte,
beschränkter Puffer, Leser-Schreiber-System

Vorlesung Modellierung WS 2001/2002 / Folie 712

Ziele:

Verdeutlichen, was wir gelernt haben.