Vorlesung "Modellierung"
Wintersemester 2014/15

Einführung
(Folien von Prof. B. König)

Prof. Norbert Fuhr

Was ist Modellierung?

Modell
Ein Modell ist eine Repräsentation eines Systems von Objekten, Beziehungen und/oder Abläufen. Ein Modell vereinfacht und abstrahiert dabei im allgemeinen das repräsentierte System.

System
Der Begriff System wird hier sehr allgemein verwendet. Er kann entweder

- einen Teil der Realität *oder*
- ein noch nicht bestehendes Gebilde, das noch erstellt werden muss,

bezeichnen.

Was ist Modellierung?

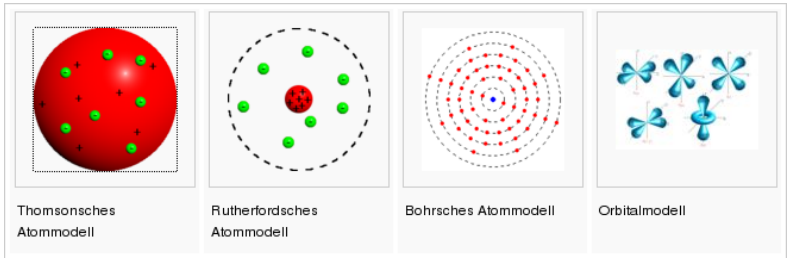
Modellierung
Modellierung ist der Prozess, bei dem ein Modell eines Systems erstellt wird.

Warum sollte man modellieren?
$\rightsquigarrow$ Um ein System zu entwerfen, besser zu verstehen, zu visualisieren, zu simulieren, ...

Um etwas konkreter zu werden betrachten wir den Begriff der Modellierung in verschiedenen Disziplinen (Physik, Biologie, Klimaforschung, ...)

Modellierung in der Physik

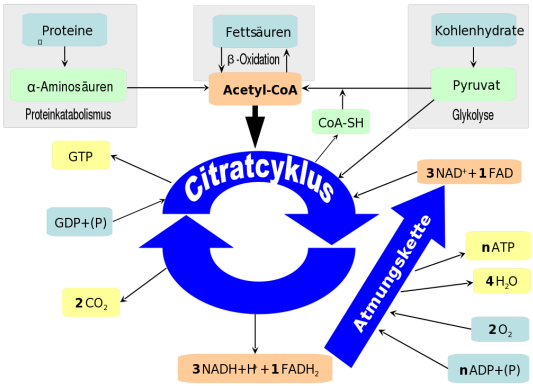
Atommodelle
Atome bestehen aus Protonen, Neutronen und Elektronen. Wie diese Teilchen zusammenwirken, wird in verschiedenen Atommodellen beschrieben, die sich im Laufe der Zeit immer wieder geändert haben.



Modellierung in der Biologie

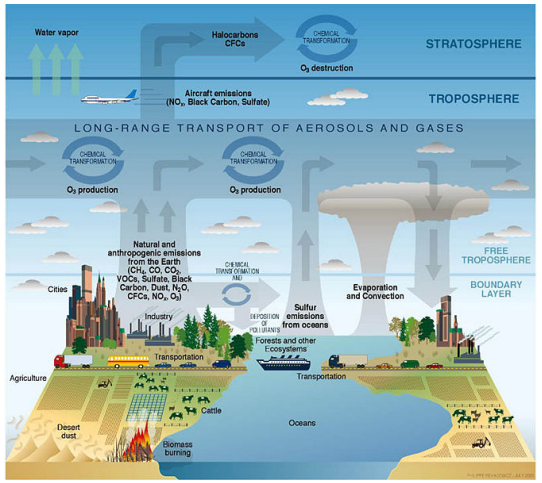
Zitronensäurezyklus

Der Zitronensäurezyklus oder Citratzyklus modelliert den Abbau organischer Stoffe im Körper.



Modellierung in der Klimaforschung

Modell des Transports von Gasen in der Atmosphäre



Arten von Modellen

visuell vs. textuell

Nicht alle Modelle sind visuell bzw. graphisch. Auch mit textuellen Beschreibunge und Formeln kann man modellieren (siehe beispielsweise mathematische Modelle).
Dennoch werden häufig graphische Darstellungen benutzt, auch aus didaktischen Gründen und um sich besser über die Modelle verständigen zu können.

Arten von Modellen

qualitativ vs. quantitativ

- qualitative Modelle: Welche Objekte gibt es? Was passiert? Warum passiert es? In welcher Reihenfolge geschehen die Ereignisse? Was sind die kausalen Zusammenhänge? Welche Phänomene treten auf?
- quantitative Modelle: Wieviele Objekte gibt es? Wie lange dauert ein Vorgang? Wie wahrscheinlich ist ein bestimmtes Ereignis?

Arten von Modellen

black box vs. white box

- black box: nur das von außen beobachtbare Verhalten wird beschrieben
- white box: es wird auch beschrieben, wie das von außen beobachtbare Verhalten im "Inneren" des Systems erzeugt wird

Arten von Modellen

statisch vs. dynamisch

- ein statisches Modell beschreibt einen Zustand des Systems zu einem bestimmten Zeitpunkt
- ein dynamisches Modell beschreibt hingegen auch, wie das System sich entwickelt (ein oder mehrere mögliche Abläufe oder sogar das gesamte Systemverhalten)

Arten von Modellen

nicht-formell vs. semi-formal vs. formal

Je nach Exaktheit der Modelle erhält man:

- formale Modelle, die vollkommen exakt in ihren Aussagen sind (vor allem mathematische Modelle)
- semi-formale Modelle, die teilweise exakt sind, jedoch nicht alles vollständig spezifieren
- nicht-formale Modelle, die als grobe Richtlinie dienen können, jedoch eher vage Aussagen machen

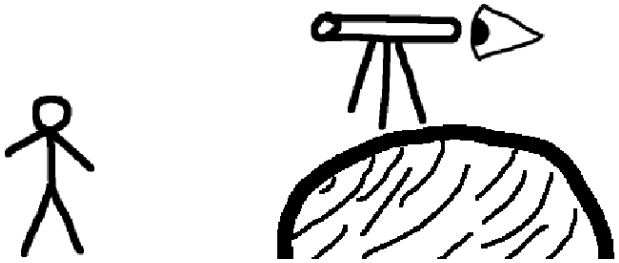
Probleme mit nicht-formalen Modellen

Natürliche Sprache ist nicht immer eindeutig.

Beispiel:

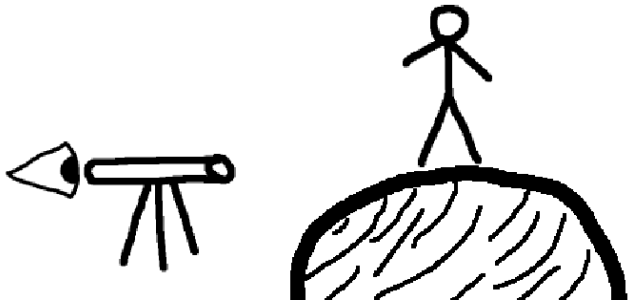
> Ich sah den Mann auf dem Berg mit dem Fernrohr.

Probleme mit nicht-formalen Modellen



(((Ich sah den Mann) auf dem Berg) mit dem Fernrohr)

Probleme mit nicht-formalen Modellen

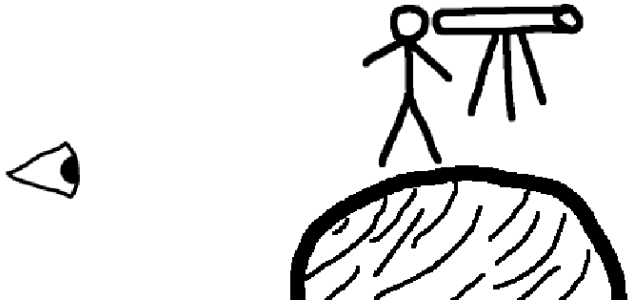


((Ich sah (den Mann auf dem Berg)) mit dem Fernrohr)

Probleme mit nicht-formalen Modellen

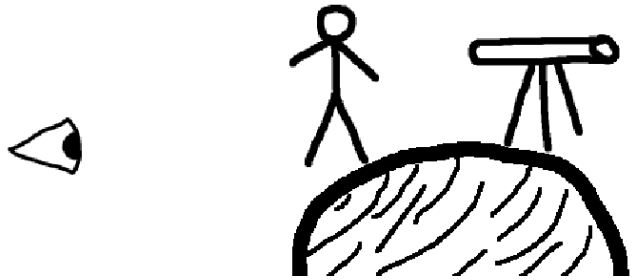
((Ich sah den Mann) (auf dem Berg mit dem Fernrohr))

Probleme mit nicht-formalen Modellen



(Ich sah ((den Mann auf dem Berg) mit dem Fernrohr))

Probleme mit nicht-formalen Modellen



(Ich sah (den Mann (auf dem Berg mit dem Fernrohr)))

Probleme mit nicht-formalen Modellen

(((Ich sah den Mann) auf dem Berg) mit dem Fernrohr)

((Ich sah (den Mann auf dem Berg)) mit dem Fernrohr)

((Ich sah den Mann) (auf dem Berg mit dem Fernrohr))

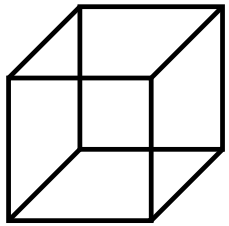
(Ich sah ((den Mann auf dem Berg) mit dem Fernrohr))

(Ich sah (den Mann (auf dem Berg mit dem Fernrohr)))

5 mögliche Interpretationen!

Probleme mit nicht-formalen Modellen

Auch graphische Darstellungen können uneindeutig sein:



Modellierung in der Informatik

In dieser Vorlesung geht es um Modellierungsmethoden in der Informatik. Diese werden zum Entwurf folgender Systeme eingesetzt:

- (Objekt-orientierte) Programme
- (Große) Software-Systeme
- Benutzeroberflächen
- Datenbanken
- Virtual Reality, Computer-Spiele
- ...

Wozu ist Modellierung gut?

Wozu benötigt man Modelle?

$\rightsquigarrow$ je komplexer ein System ist, desto wichtiger ist es, einen Plan zu erstellen, bevor man beginnt das System zu konstruieren

$\rightsquigarrow$ dies führt zu:

- Vermeidung von Fehlern
- besserer Qualität
- niedrigeren Kosten
- besserer Dokumentation und Wiederverwendbarkeit

Wozu ist Modellierung gut?

Analogie: Bau eines Hauses

Beim Bau einer Hundehütte kann man zumeist ohne große Planung vorgehen. Die Hütte kann von einer einzelnen Person erstellt werden und ein Hund hat zumeist keine großen Anforderungen.

Wozu ist Modellierung gut?

Analogie: Bau eines Hauses

Beim Bau eines Einfamilienhauses ist Planung viel wichtiger. Die Familie ist anspruchsvoller als ein Hund, Bauvorschriften müssen eingehalten werden und vermutlich werden nicht alle Arbeiten von derselben Person durchgeführt.

Wozu ist Modellierung gut?

Analogie: Bau eines Hauses

Beim Bau eines Hochhauses ist ohne Erstellung eines detaillierten Plans bzw. Modells nicht möglich. Das Risiko, Fehler zu machen ist sehr groß, und Fehler können extrem kostspielig werden.

Wozu ist Modellierung gut?

Modellierung ist in der Informatik weniger verbreitet als in den Ingenieurwissenschaften, aber ebenso wichtig.

Schwierigkeiten beim Entwurf komplexer Systeme

- Menschen können sich komplexe Systeme normalerweise nicht in vollem Umfang vorstellen
- Bei mehreren Menschen/Entwicklern gibt es unterschiedliche Meinungen darüber, wie das System aussehen muss
  $\rightsquigarrow$ Modelle dienen zu Kommunikation!
- Es ist schwer ein System zu dokumentieren und zu warten, das nicht explizit modelliert ist.

Feststellung (nach Glinz)

Die Entwicklung von Klein-Software unterscheidet sich fundamental von der Entwicklung größerer Software.

Probleme bei der Entwicklung großer Systeme

Klein-Groß-Gegensätze in der Software-Entwicklung:

klein	**groß**
Programme bis ungefähr 300 Zeilen	Längere Programme
Für den Eigengebrauch	Für den Gebrauch durch Dritte
Vage Zielsetzung genügt, das Produkt ist seine eigene Spezifikation	Genaue Zielbestimmung, d.h. die Spezifikation von Anforderungen, erforderlich

Probleme bei der Entwicklung großer Systeme

klein	**groß**
Ein Schritt vom Problem zur Lösung genügt: Lösung wird direkt programmiert	Mehrere Schritte vom Problem zur Lösung erforderlich: Spezifikation, Konzept, Entwurf und Programmieren der Teile, Zusammensetzen, Inbetriebnahme
Validierung (Überprüfung/Testen) und nötige Korrekturen finden am Endprodukt statt	Auf jeden Entwicklungsschritt muss ein Prüfschritt folgen, sonst kann Endergebnis unbrauchbar werden

Probleme bei der Entwicklung großer Systeme

klein	**groß**
Eine Person entwickelt: keine Kooperation und Kommunikation erforderlich	Mehrere Personen entwickeln gemeinsam: Koordination und Kommunikation notwendig
Komplexität des Problems in der Regel klein, Strukturieren und Behalten der Übersicht nicht schwierig	Komplexität des Problems größer bis sehr groß, explizite Maßnahmen zur Strukturierung und Modularisierung erforderlich

Probleme bei der Entwicklung großer Systeme

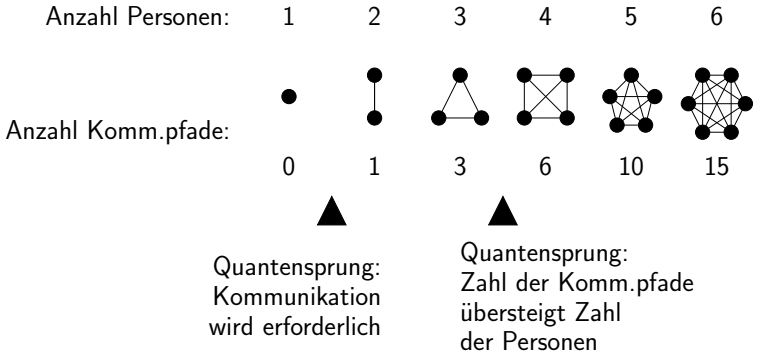
klein	groß
Software besteht aus wenigen Komponenten	Software besteht aus vielen Komponenten, die spezielle Maßnahmen zur Komponentenverwaltung erfordern
In der Regel wird keine Dokumentation erstellt	Dokumentation dringend erforderlich, damit Software wirtschaftlich betrieben und gepflegt werden kann

Probleme bei der Entwicklung großer Systeme

klein	groß
Keine Planung und Projektorganisation erforderlich	Planung und Projektorganisation zwingend erforderlich für eine zielgerichtete, wirtschaftliche Entwicklung

Probleme bei der Entwicklung großer Systeme

Explosion der Anzahl der Kommunikationspfade bei Anstieg der Entwicklerzahl:



Probleme bei der Entwicklung großer Systeme

Die Problematik bei der Erstellung großer Programme sieht man auch an folgenden Zahlen (nach Boehm 1981):

- 40% der Zeit wird mit der Entwicklung verbracht (davon: 15% Spezifikation; 8% Codierung; 16% Test)
- 60% der Zeit wird für die Wartung aufgewendet (12% Anpassung; 36% Erweiterung und Verbesserung; 12% Fehlerbehebung)

Wozu ist Modellierung gut?

Aus diesen Fakten und Zahlen ergibt sich folgende Konsequenz:

Vor allem bei der Erstellung großer Systeme ist es unbedingt erforderlich, zunächst das System zu modellieren, bevor es implementiert bzw. konstruiert wird.

Meistens sind auch mehrere verschiedene Modelle erforderlich.

Aus Gründen der Übersichtlichkeit und Didaktik werden wir uns in der Vorlesung jedoch hauptsächlich mit kleinen Modellen befassen.

Beispiel: Wolf, Ziege, Kohlkopf

Wir modellieren folgendes System, um eine möglich Lösung zu finden.

Wolf-Ziege-Kohlkopf-Problem

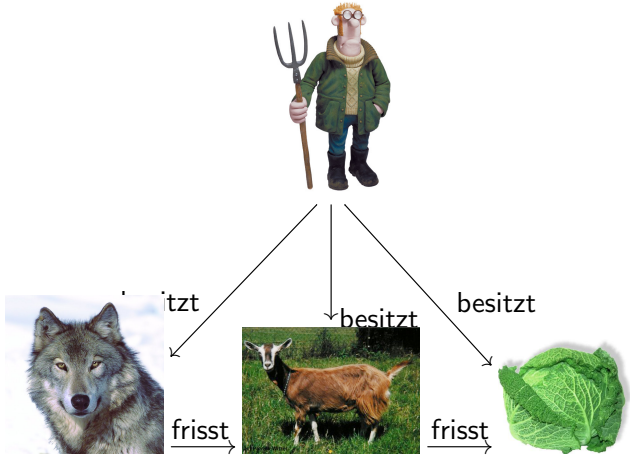
Ein Bauer will einen Fluss überqueren. Er hat einen Wolf, eine Ziege und einen Kohlkopf bei sich. Wenn sie alleingelassen werden, so frisst der Wolf die Ziege, und die Ziege den Kohlkopf. Zur Überquerung des Flusses steht ein Boot mit zwei Plätzen zur Verfügung. Nur der Bauer kann rudern und er kann das Boot entweder allein benutzen oder ein Tier oder den Kohlkopf mitnehmen.

Beispiel: Wolf, Ziege, Kohlkopf

Statisches Modell I: Beteiligte Akteure/Objekte

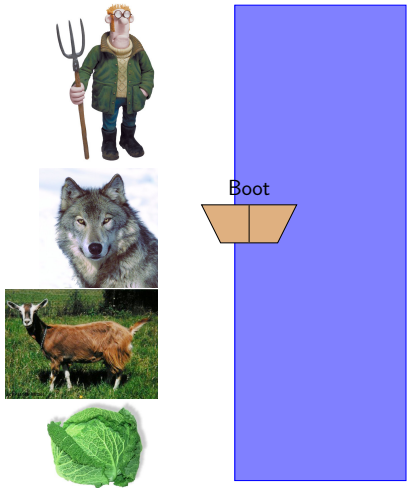
Beispiel: Wolf, Ziege, Kohlkopf

Statisches Modell II: Fress- und Eigentumsbeziehungen zwischen den Akteuren



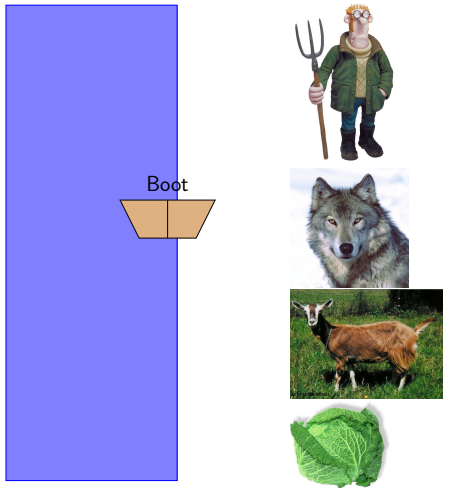
Beispiel: Wolf, Ziege, Kohlkopf

Ausgangssituation: vor Überquerung des Flusses



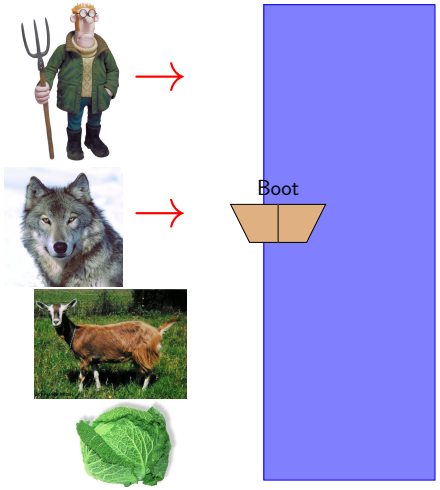
Beispiel: Wolf, Ziege, Kohlkopf

Zielsituation: nach Überquerung des Flusses



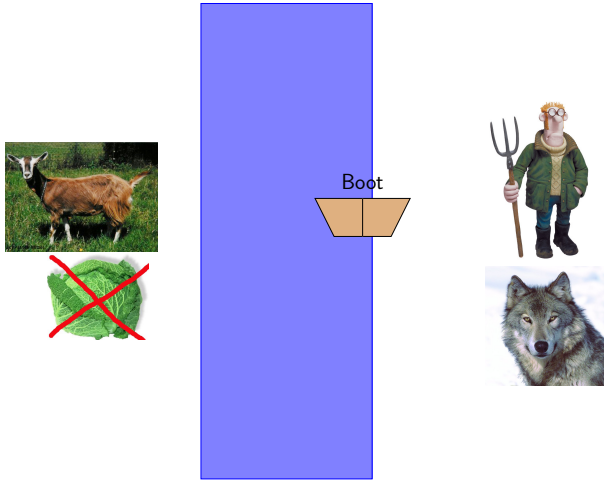
Beispiel: Wolf, Ziege, Kohlkopf

Dynamisches Modell: Beispielablauf, erster Schritt
Bauer und Wolf setzen gemeinsam über



Beispiel: Wolf, Ziege, Kohlkopf

Dynamisches Modell: Beispielablauf, zweiter Schritt
Ziege frisst Kohlkopf



Syntax und Semantik

Man unterscheidet bei der Modellierung zwischen:

- Syntax: Symbole und Diagramme, die für die Darstellung des Modells genutzt werden dürfen
  Im Beispiel: Bild der Ziege, blaue Fläche, etc.
- Semantik: Bedeutung, die sich hinter den Symbolen verbirgt
  Im Beispiel: Die blaue Fläche symbolisiert den Fluss.
  Die Pfeile bedeuten: "Fluss wird überquert"

Zu einer Syntax gibt es nicht immer eine dazugehörige Semantik (im Beispiel ist die Semantik sehr vage).
Wünschenswert ist jedoch im allgemeinen, dass die Bedeutung aller Symbole möglichst präzise festgelegt wird.
Einigung auf eine gemeinsame Sprache/Notation, auf gemeinsame visuelle Beschreibungen zur Vermeidung von Missverständnissen.

Weitere Aspekte

Weitere wichtige Gesichtspunkte sind:

- Analyse:
  - Ist das Modell korrekt? Ist es in sich konsistent?
  - Stimmt das Modell mit der späteren Implementierung überein? (Hier werden Verfahren zum Testen und zur Verifikation benötigt)
- Werkzeuge, Software-Tools: werden benötigt zum Zeichnen, zum Darstellen (Wechsel zwischen verschiedenen Darstellungen), zum Archivieren, zur Code-Generierung, zur Analyse, ...

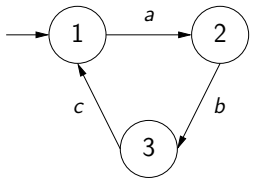
Inhalt der Vorlesung

Inhalt

- Mathematische Grundlagen
- Graphen für statische und dynamische Systembeschreibungen
- Petrinetze
- UML (Unified Modeling Language)
- Markov-Prozesse

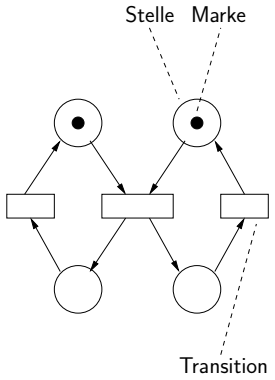
Graphen

- Graphen bestehen aus Knoten und Kanten.
- Sie können eingesetzt werden für
  - Statische Modellierung: Komponenten und Beziehungen zwischen den Komponenten
  - Dynamische Modellierung: Zustände und Zustandsübergänge in Form eines Zustandsübergangsdiagramms



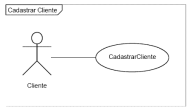
Petrinetze

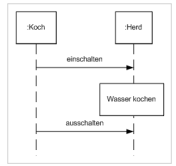
- Modell für nebenläufige und verteilte Systeme, das die gemeinsame Nutzung von Ressourcen beschreibt.
- Schwerpunkt liegt auf der Modellierung des dynamischen Verhaltens.
- Etabliertes Modell, das vielfältig eingesetzt wird.
- Formale Semantik.
- Erfunden von Carl Adam Petri (1962).



UML: Unified Modeling Language

- Standard-Modellierungssprache für Software Engineering.
- Basiert auf objekt-orientierten Konzepten.
- Sehr umfangreich, enthält viele verschiedene Typen von Modellen.
- Entwickelt von Grady Booch, James Rumbaugh, Ivar Jacobson (1997).





Inhalt der Vorlesung

Die vorgestellten Modellierungsmethoden sind nicht die einzigen Modellierungsmethoden in der Informatik.

Hier: Fokus auf visuelle Modellierung mit Hilfe von Diagrammen

Mögliche Alternative: algebraische Modellierungsmethoden, die sich stärker an der Mathematik orientieren

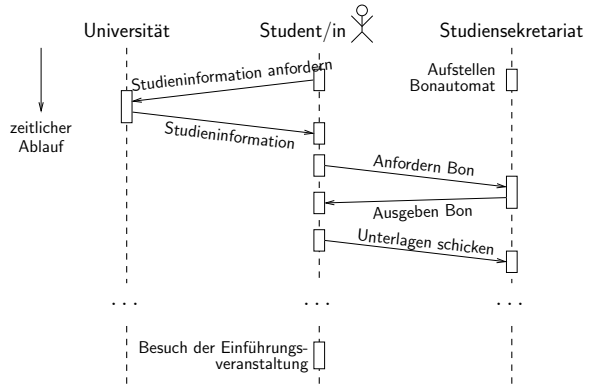
Ein weiteres Beispiel: Einschreiben an der Universität

Szenario: Eine Reorganisation der Universität, und insbesondere des Studiensekretariats, das für Einschreibungen zuständig ist, steht an.

Hierzu soll der Ablauf des Einschreibens neuer Studierender modelliert werden ...

Ein weiteres Beispiel: Einschreiben an der Universität

Darstellung des zeitlichen Ablaufs durch Symbolisieren der beteiligten Partner als Linien und der Kommunikation durch Pfeile (Ausschnitt).



Ein weiteres Beispiel: Einschreiben an der Universität

Solche Diagramme sind übrigens Bestandteil von UML. Sie werden Sequenzdiagramme (engl. sequence diagrams, auch message sequence charts) genannt.

Notation: Mengen und Funktionen

Menge

Menge M von Elementen, wird beschrieben als Aufzählung

$$M = \{0, 2, 4, 6, 8, \dots\}$$

oder als Menge von Elementen mit einer bestimmten Eigenschaft

$$M = \{n \mid n \in \mathbb{N}_0 \text{ und } n \text{ gerade}\}.$$

Allgemeines Format:

$$M = \{x \mid P(x)\}$$

(M ist Menge aller Elemente x, die die Eigenschaft P erfüllen.)

Notation: Mengen und Funktionen

Bemerkungen:

- Die Elemente einer Menge sind ungeordnet, d.h., ihre Ordnung spielt keine Rolle. Beispielsweise gilt:

  $$\{1, 2, 3\} = \{1, 3, 2\} = \{2, 1, 3\} = \{2, 3, 1\} = \{3, 1, 2\} = \{3, 2, 1\}$$

- Ein Element kann nicht "mehrfach" in einer Menge auftreten. Es ist entweder in der Menge, oder es ist nicht in der Menge. Beispielsweise gilt:

  $$\{1, 2, 3\} \neq \{1, 2, 3, 4\} = \{1, 2, 3, 4, 4\}$$

Notation: Mengen und Funktionen

Element einer Menge

Wir schreiben $a \in M$, falls ein Element a in der Menge M enthalten ist.

Anzahl der Elemente einer Menge

Für eine Menge M gibt $|M|$ die Anzahl ihrer Elemente an.

Teilmengenbeziehung

Wir schreiben $A \subseteq B$, falls jedes Element von A auch in B enthalten ist. Die Relation $\subseteq$ heißt auch Inklusion.

Notation: Mengen und Funktionen

Vereinigung

Die Vereinigung zweier Mengen M_1, M_2 ist die Menge M, die die Elemente enthält, die in M_1 oder M_2 vorkommen. Man schreibt dafür $M_1 \cup M_2$.

$$M_1 \cup M_2 = \{a \mid a \in M_1 \text{ oder } a \in M_2\}$$

Schnitt

Der Schnitt zweier Mengen M_1, M_2 ist die Menge M, die die Elemente enthält, die sowohl in M_1 als auch in M_2 vorkommen. Man schreibt dafür $M_1 \cap M_2$.

$$M_1 \cap M_2 = \{a \mid a \in M_1 \text{ und } a \in M_2\}$$

Notation: Mengen und Funktionen

Kreuzprodukt

Seien A, B zwei Menge. Die Menge $A \times B$ is die Menge aller Paare (a, b), wobei das erste Element des Paars aus A, das zweite aus B kommt.

$$A \times B = \{(a, b) \mid a \in A, b \in B\}$$

Beispiel:

$\{1, 2\} \times \{3, 4, 5\} = \{(1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5)\}$

Es gilt: $|A \times B| = |A| \cdot |B|$ (für endliche Menge A, B).

Notation: Mengen und Funktionen

Bemerkungen:

- Wir betrachten nicht nur Paare, sondern auch sogenannte Tupel, bestehend aus mehreren Elementen. Ein Tupel $(a_1, \ldots, a_n)$ bestehend aus n Elementen heißt auch n-Tupel.
- In einem Tupel sind die Elemente geordnet! Beispielsweise gilt:

$$(1, 2, 3) \neq (1, 3, 2) \in \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0$$

- Ein Element kann "mehrfach" in einem Tupel auftreten. Tupel unterschiedlicher Länge sind immer verschieden. Beispielsweise:

$$(1, 2, 3, 4) \neq (1, 2, 3, 4, 4)$$

Merke: Runde Klammern $(,)$ und geschweifte Klammern $\{,\}$ stehen für ganz verschiedene mathematische Objekte!

Notation: Mengen und Funktionen

Potenzmenge

Sei M eine Menge. Die Menge $\mathcal{P}(M)$ ist die Menge aller Teilmengen von M.

$$\mathcal{P}(M) = \{A \mid A \subseteq M\}$$

Beispiel:
$\mathcal{P}(\{1,2,3\}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$.

Es gilt: $|\mathcal{P}(M)| = 2^{|M|}$ (für eine endliche Menge M).

Notation: Mengen und Funktionen

Funktion

$$\begin{aligned} f : A &\to B \\ a &\mapsto f(a) \end{aligned}$$

Die Funktion f bildet ein Element $a \in A$ auf ein Element $f(a) \in B$ ab. Dabei ist A der *Definitionsbereich* und B der *Wertebereich*.

Beispiel (Quadratfunktion):

$$f : \mathbb{Z} \to \mathbb{N}_0, \quad f(n) = n^2$$
$$\ldots, -3 \mapsto 9, -2 \mapsto 4, -1 \mapsto 1, 0 \mapsto 0, 1 \mapsto 1, 2 \mapsto 4, 3 \mapsto 9, \ldots$$

Dabei ist $\mathbb{N}_0$ die Menge der natürlichen Zahlen (mit der Null) und $\mathbb{Z}$ die Menge der ganzen Zahlen.

Graphen

Graphen sind netzartige Strukturen, bestehend aus Knoten und Kanten. Sie bilden die Grundlagen vieler diagrammatischer Modellierungstechniken.

Gerichteter Graph

Sei L eine Menge von Beschriftungen (oder Labels). Ein gerichteter (beschrifteter) Graph $G = (V, E)$ besteht aus

- einer Knotenmenge V und
- einer Kantenmenge $E \subseteq V \times L \times V$.

Bemerkung: V steht für *vertices* und E für *edges*.

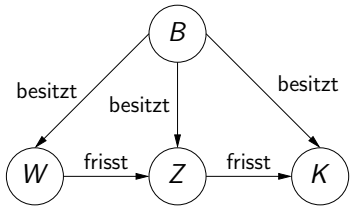
Graphen

Beispiel (Bauer, Wolf, Ziege, Kohlkopf):

$$V = \{B, W, Z, K\} \qquad L = \{\text{besitzt}, \text{frisst}\}$$

$$\begin{aligned} E = \{&(B, \text{besitzt}, W), (B, \text{besitzt}, Z), (B, \text{besitzt}, K), \\ &(W, \text{frisst}, Z), (Z, \text{frisst}, K)\} \end{aligned}$$

Graphische Darstellung:

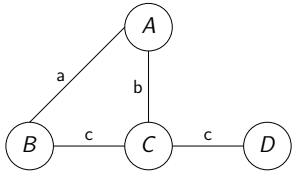


Graphen

Weitere Arten von Graphen:

Ungerichtete Graphen

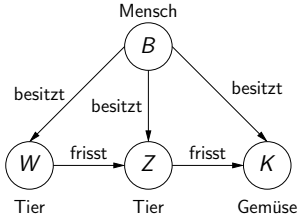
Bei ungerichteten Graphen spielt die Richtung der Kanten keine Rolle. Formal sind Kanten zweielementige Teilmengen der Knotenmenge (statt Tupel).



Graphen

Graphen mit Knotenbeschriftung

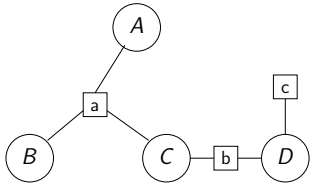
Auch Knoten können Beschriftungen tragen, wobei zwei verschiedene Knoten auch gleich beschriftet sein dürfen.



Graphen

Hypergraphen

Bei Hypergraphen kann eine Kante (symbolisiert durch ein Quadrat oder Rechteck) mit einer beliebigen Anzahl von Knoten verbunden sein. Evtl. sind dabei die Knoten im Bezug auf die Kante geordnet (wie bei gerichteten Graphen).



Graphen

Graphen können in vielfältiger Weise zur Modellierung eingesetzt werden. Wir betrachten zwei typische Fälle.

Graphen zur statischen Modellierung

Knoten sind Komponenten oder Objekte, die untereinander über Kanten verbunden sind bzw. in Beziehung stehen.

Beispiel: Beziehungen zwischen Bauer, Wolf, Ziege, Kohlkopf

Zustandsübergangsdiagramme

Graphen zur dynamischen Modellierung

Knoten sind Zustände und Kanten sind Zustandsübergänge.

Klassischer Vertreter: Zustandsübergangsdiagramme (auch Transitionssysteme genannt)

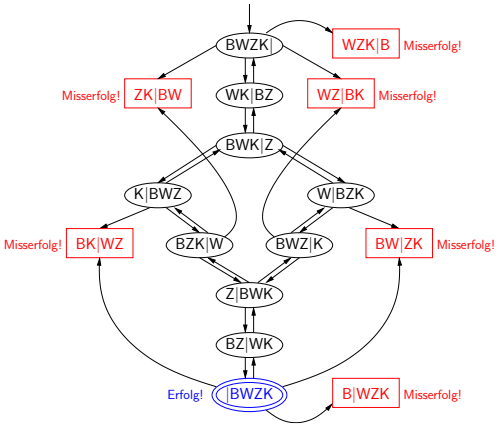
Zustandsübergangsdiagramm

Ein Zustandsübergangsdiagramm besteht aus einem gerichteten Graphen (Z, U), wobei Z (= Zustände) die Knotenmenge des Graphen und U (= Übergänge) die Kantenmenge der Graphen ist, und außerdem aus einem Startzustand $z_0 \in Z$.

Zustandsübergangsdiagramme werden graphisch wie gerichtete Graphen dargestellt. Der Startzustand (auch Anfangszustand genannt) wird dabei meist durch eine eingehende Pfeilspitze gekennzeichnet.

Zustandsübergangsdiagramme

Beispiel: Zustandsübergangsdiagramm für die Übergänge beim Wolf-Ziege-Kohlkopf-Problem.



Zustandsübergangsdiagramm

Bemerkungen:

- Der senkrechte Strich | steht für den Fluss. Links und rechts davon befinden sich die Akteure/Objekte (B = Bauer, W = Wolf, Z = Ziege, K = Kohlkopf)
- Übergänge sind aus Gründen der Übersichtlichkeit nicht beschriftet. Sinnvolle Beschriftungen wären die ausgeführten Aktionen ("Bauer bringt Ziege über den Fluss", etc.)
- Eckige (rote) Zustände symbolisieren hier Misserfolg (z.B. "Ziege frisst Kohlkopf").
  Kanten, die aus solchen Zuständen herausführen, wurden weggelassen.
- Die doppelte (blaue) Ellipse symbolisiert hier Erfolg (erwünschter Zielzustand ist erreicht)

Zustandsübergangsdiagramme

Weitere Bemerkungen:

- Es gibt mehrere (sogar unendlich viele) Wege zum Zielzustand. Die zwei kürzesten enthalten jeweils sieben Übergänge.
- Zustandsübergangsdiagramme selbst relativ einfacher Systeme werden oft erstaunlich groß (sogenannte Zustandsexplosion).

Wichtig:
Zustandsübergangsdiagramme stellen im allgemeinen *alle* Zustände und *alle* Übergänge eines Systems dar.