

Interpretation setzt Daten untereinander und mit Gegenständen anderer Wissensgebiete in Beziehung

- Ziel: Wissensgewinn über Gegenstände in anderen Wissensgebieten

Syntax: Beziehungen, die sich aus Anordnung der Zeichen ergeben

„das Haus“: Folge von 7 Buchstaben und einem Zwischenraum

+₁
Symbol

Semantik: Beziehungen, die sich aus der Interpretationsvorschrift ergeben

„das Haus“: Zwischenraum trennt Wörter; also zwei Wörter

a + b
Gleitpunkt.

Pragmatik: Beziehungen zwischen Daten und Gegenständen bzw. Sachverhalten außerhalb der Datenmenge

„das Haus“: Begriff für einen Gegenstand der realen Welt

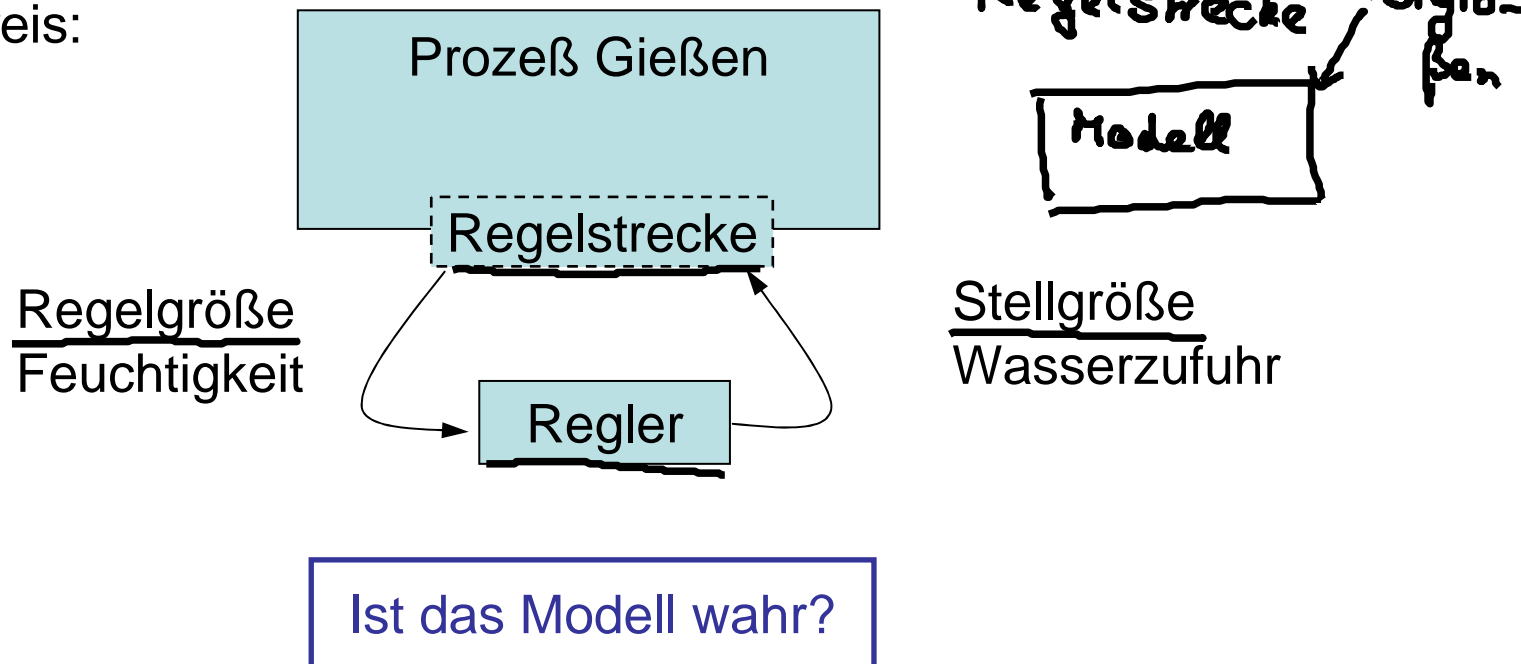
wie führt
man
Glp.add.
aus

Semiotik besteht aus Syntax, Semantik, Pragmatik



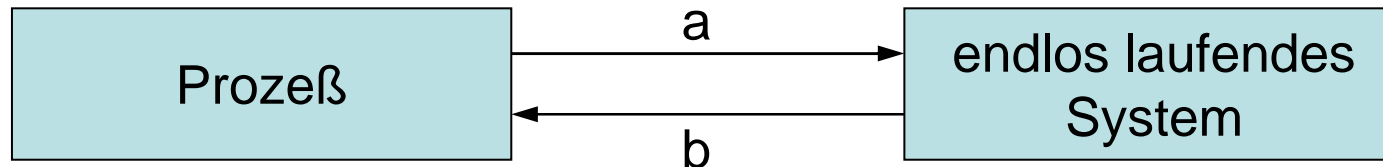
- Garten gießen, wenn nach Modellwelt zu trocken
- Garten gießen überführt trockenen Zustand in weniger trockenen Zustand
- Nach 20 Minuten gießen kommt Gärtner zum Schluß, daß erwünschter Zustand des Gartenbodens erreicht ist

- Regelkreis:



Berechnung von Funktionen, Algorithmus: berechne $f : A \rightarrow B$

Prozeßüberwachung:



b hängt von a und der Systemhistorie ab

eingebettetes System: löst Aufgaben im Verbund mit Systemkomponenten, die nicht der Datenverarbeitung dienen

adaptives System: eingebettetes System, das sich Veränderungen der Wirklichkeit anpaßt

- auch durch das System selbst hervorgerufene

reaktives System: Prozeßüberwachung, eingebettetes oder adaptives System
mehrere Ausgabeströme, mehrere Eingabeströme



1. Entwurf des inneren Aufbau des Systems:
Komponenten, Beziehungen zwischen diesen sowie der Umgebung, Zustände, Zustandsübergänge
 2. rekursiver Entwurf der Teilsysteme (falls notwendig)
 3. Festlegung der Verfahren zur Realisierung der Komponenten und ihrer Beziehungen
 4. Konstruktion der Komponenten und Funktionen, die das Zusammenspiel regeln
 5. Integration aller Komponenten und Funktionen
 6. Integration des Systems in seine Umgebung
- jeder Schritt wird validiert
 - Kostenschätzung und Zeitplanung für Gesamtprojekt und Einzelschritte

Systemzerlegung

Programmieren



1.4 Algorithmen

Realisierungsarten für Funktionen $f: A \rightarrow B$

partielle Fkt⁶⁶
totale Fkt.

statisch : (auch **deklarativ**)

- $f(a) =$ **Formel**
- $g(a, b) = 0$, falls $f(a) = b$

$$ax + b = c$$

A

\rightarrow

$$g(x) = ax + b - c = 0$$

tabellarisch :

a_1	a_2	a_3	...	a_{n-1}	a_n
$f(a_1)$	$f(a_2)$	$f(a_3)$...	$f(a_{n-1})$	$f(a_n)$

algorithmisch : (auch **operativ, prozedural, synthetisch**)

- Beschreibung zur Berechnung des Ergebnisses für beliebige Argumente a
- in beliebiger Sprache
- Codierung in endlicher Länge
- Folge von endlich vielen elementaren Operationen

Algorithmus : eine solche Beschreibung
benannt nach Mohamed Ibn Mus Alchwarizimi, ? - 846,
schrieb erstes Lehrbuch der Algebra



effektiv \neq
effizient

- Definition: Algorithmus
 - ein Algorithmus ist ein Verfahren
 - mit einer präzisen endlichen Beschreibung
 - unter Verwendung effektiver Verarbeitungsschritte
ist in endlicher Zeit durchführbar
- Algorithmen lösen i.d.R. eine Klasse von Aufgaben
 - die spezielle aus der Klasse zu bearbeitende Aufgabe wird durch Parameter bestimmt
- Parameter sind die Eingabe des Algorithmus
- das zu der Eingabe gelieferte Resultat ist die Ausgabe des Algorithmus



▪ funktionale Sicht

die Aufgabenstellung, die durch einen Algorithmus bewältigt werden soll

Problem
was?

▪ operationelle Sicht

die spezielle Art und Weise, wie die Aufgabe bewältigt wird

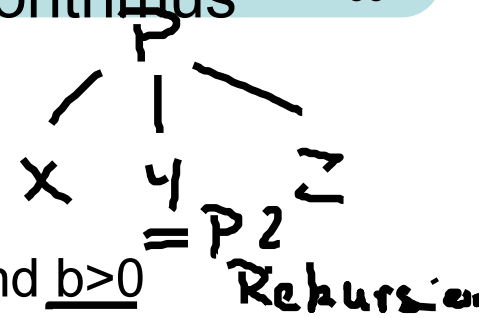
- elementare Verarbeitungsschritte, die im Algorithmus auftreten
- Beschreibung der Auswahl (Ablaufsteuerung) der einzelnen, auszuführenden Schritte
- Daten und Parameter, die durch den Algorithmus bearbeitet werden

wie?

▪ für eine algorithmisch lösbare Aufgabenstellung (→ funktionale Sicht) existieren viele unterschiedliche Lösungswege (→ operationale Sicht)

- daher: unterscheide Spezifikation einer Aufgabe von der (algorithmischen) Lösung der Aufgabe





▪ Aufgabenstellung:

gegeben seien zwei ganze Zahlen a , b mit $a > 0$ und $b > 0$
 gesucht ist der größte gemeinsame Teiler $ggT(a,b)$ der beiden Zahlen a und b

▪ der Algorithmus zur Berechnung von $ggT(a,b)$ nach Euklid lautet:

- (1) falls $a = b$, dann gilt: $ggT(a,b) = a$;
- (2) falls $a < b$, dann wende den Algorithmus ggT auf $(a, b-a)$ an;
- (3) falls $b < a$, dann wende den Algorithmus ggT auf $(a-b, b)$ an.

▪ Analyse von Eigenschaften des Algorithmus

- elementare Verarbeitungsschritte: '-', '<', '='
- für ungleiche negative Eingabezahlen terminiert der Algorithmus nicht
- der Algorithmus ist deterministisch



bedingte Ausführung: Tätigkeit wird nur ausgeführt, wenn bestimmte Bedingung erfüllt ist

Wiederholung bzw. Schleife: Tätigkeit wird wiederholt solange bis eine Bedingung erfüllt oder nicht erfüllt ist

Ausführung von Unterprogrammen : führe Tätigkeit aus, die woanders beschrieben ist

- benutzt Ergebnis
- Parameter sind möglich

+ Elementarop.

sequentielle Ausführung

parallele Ausführung: führe mehrere Tätigkeit gleichzeitig oder zeitlich verzahnt aus

- mit oder ohne Endesynchronisierung

Das sind alle denkbaren Alternativen zur Definition zeitlicher Abläufe



- Ein Algorithmus heißt für eine Eingabe
 - **terminierend**, wenn er stets nach endlich vielen Schritten **halten** endet
 - **deterministisch**, wenn der nächste auszuführende Verarbeitungsschritte immer eindeutig festgelegt ist
 - **determiniert**, wenn das Resultat des Algorithmus eindeutig bestimmt ist
 - **sequentiell**, wenn die Verarbeitungsschritte stets hintereinander ausgeführt werden
 - **parallel**, wenn gewisse Verarbeitungsschritte nebeneinander ausgeführt werden

- Elemente zur Beschreibung von Algorithmen
 - (1) Ausführung elementarer Schritte
 - (2) Fallunterscheidung über Bedingungen
 - (3) sequentielle/parallele Ausführung, Wiederholung, Rekursion



- Algorithmen berechnen Funktionen
- Frage: Lassen sich alle mathematisch spezifizierten Funktionen durch Algorithmen berechnen?
 - Antwort: Nein!
fundamentales Ergebnis der mathematischen Logik (Karl Gödel)
 - Funktionen, für die sich ein Algorithmus zu ihrer Berechnung angeben lässt, heißen berechenbar $f: A \rightarrow B$ $\{y \mid y \in B, y = f(a)\}$
 - Eine Menge M heißt aufzählbar, wenn sie genau aus den Werten y einer berechenbaren Funktion $y = f(x)$ besteht
 - Eine aufzählbare Menge M heißt entscheidbar, wenn man algorithmisch feststellen kann, ob $y \in M$ oder $y \notin M$
- verschiedene Formalisierungen des Begriffs „Algorithmus“ und damit des Begriffs *berechenbar*.
 - Text- und Termersetzung, rekursive Funktionen, Turing-Maschinen, while-Sprachen
- Churchsche These: alle Formalisierungen von „Algorithmus“ sind gleichwertig



- **Endlicher Zeichenvorrat Σ (Alphabet)**

- Elemente $\sigma \in \Sigma$ heißen **Zeichen**

- Ein **Wort** x über Σ ist eine endliche Folge über $\Sigma = \{a, b\}$

- $x = x_1 \dots x_n$ mit $x_i \in \Sigma$

$a b a$

- Die **Länge** $|x|$ eines Wortes ist die Länge der Folge

- Genau ein **leeres Wort** ε mit $|\varepsilon| = 0$ $\varepsilon \notin \Sigma$!

- Schreibweisen:

- x ist Wort über Σ, ε zulässig: $x \in \Sigma^*$ *Kleene'scher Stern*
- x ist Wort über Σ, ε nicht zulässig: $x \in \Sigma^+$

- $v \in \Sigma^*$ ist **Teilwort** eines Wortes x

Teilwort

- v ist Teilfolge $x_i \dots x_{i+k}$ von x
- Schreibweise: $x = u v w, u, w \in \Sigma^*$



1.6 Definition der Semi-Thue Systeme

87

- Ein Semi-Thue System ist ein Paar $T=(\Sigma, T)$ mit

- endlichem Zeichenvorrat Σ
- endlicher Regelmenge T

Produktionen

- Regeln $l \rightarrow r$ mit $l, r \in \Sigma^*$

- **Anwendbar** auf ein Wort $x \in \Sigma^* \Leftrightarrow x = u \circ v$, $u, v \in \Sigma^*$

- Ergebnis der Anwendung: $x' = u \circ r \circ v$

- **Transformation** oder **direkte Ableitung**, $x \Rightarrow x'$

- **Ableitung** $x \Rightarrow^* x'$: Endliche Folge direkter Ableitungen

- Ausführungsmodell für gegebenes Ausgangswort $x \in \Sigma^*$

- Solange anwendbare Regeln vorhanden

- Wähle eine beliebige anwendbare Regel

- Wähle eine beliebige passende Anwendungsstelle

- Wende die Regel an.

1 → 0

1 1
✓ ✓



1.6 Beispiel: Addition

88

- **Zeichenvorrat:** $\Sigma = \{ |, + \}$
- **Aufgabe:** Addition unärer Zahlen
- **Regeln:**

$$\begin{array}{l} \cancel{+| \rightarrow |+} \quad \text{(Regel 1)} \\ + \rightarrow \varepsilon \quad \text{(Regel 2)} \end{array}$$

$$\begin{array}{l} ||| + || \Rightarrow |||| + | \quad \text{(Regel 1)} \\ \Rightarrow |||| + \quad \text{(Regel 1)} \\ \Rightarrow |||| \quad \text{(Regel 2)} \end{array}$$

- Sind die Regeln minimal?

$$||| + || \Rightarrow |||| \quad (R.2)$$



1.6 Beispiel: Grade oder ungrade

89

- **Zeichenvorrat:** $\Sigma = \{ | \}$
- **Aufgabe:** Ist eine unäre Zahl grade oder ungrade?
- **Regeln:**

$$|| \rightarrow \varepsilon$$

$$| | | | \rightarrow | | | \rightarrow |$$

Interpretation des Ergebnisses:

$1 \hat{=} \text{ ungrade}$

$\varepsilon \hat{=} \text{ grade}$



1.6 Bezug zu Algorithmen

90

▪ Semi-Thue-Systeme sind Algorithmen

- Eingabe: Wort x
- Operationen: Ersetzungsregeln
 - i.a. beliebig oft angewendet
- Terminierung: keine Regel mehr anwendbar
- Ausgabe: Ableitung von x

$$R = \left\{ \begin{array}{l} 11 \rightarrow \varepsilon, \\ 11 \rightarrow \varepsilon, \\ 1 \rightarrow \varepsilon \end{array} \right\}$$

$x \in \Sigma^*, \Sigma = \{1\}$

▪ Eigenschaften

- Ausführung i.a. **indeterministisch** (siehe Ausführungsmodell)
- i.a. nichtterminierend
- Ausgabe i.a. nicht determiniert

nicht vorgebestimmt welche Op. als nächstes ausgef. wird

$$R = \left\{ \begin{array}{l} 0 \rightarrow 1, \\ 1 \rightarrow 0 \end{array} \right\}$$

$x = 1$

↘ Ausgabe nicht vorgebestimmt



▪ **Formale Sprache** $L_x = L(\mathbf{T}, x) = \{ y \mid x \Rightarrow^* y \text{ in } \mathbf{T} \}$

▪ Beispiel: Kindersprache

▪ **Zeichenvorrat:**

• $\Sigma = \{ \text{Mama, Papa, essen, trinken, mu\ss mal} \}$

$\Sigma' = \Sigma \cup \{ \text{Appell, Ausruf} \}$

$R = \text{Ausruf} \Rightarrow \text{Appell}$

$\text{Ausruf} \Rightarrow \text{mu\ss mal}$

$\text{Appell} \Rightarrow \text{M. e.} \mid \text{M. t} \mid \text{P. e} \mid \text{P. t}$

$\text{Appell} \Rightarrow \text{M. t}$

" \Rightarrow . .



- **Inverses** T^{-1} eines Semi-Thue Systems T
 - Pfeilrichtung der Regeln umdrehen

- Beispiel: Kindersprache (Netz)

- **Zeichenvorrat:**

- $\Sigma = \{a, l, e, s, 1, 3, 5\}$

$$R = \left\{ \begin{array}{l} l \rightarrow 1, \\ e \rightarrow 3, \\ s \rightarrow 5 \end{array} \right\}$$

$$\text{alles} \Rightarrow^* a1135$$

$$R^{-1} = \left\{ \begin{array}{l} 1 \rightarrow l, \\ 3 \rightarrow e, \\ 5 \rightarrow s \end{array} \right\}$$

$$a1135 \Rightarrow^* \text{alles}$$



- Gegeben eine Dose mit (endlich vielen) weißen und schwarzen Bohnen
- (Spiel-)Regeln
Nehme fortgesetzt blind zwei Bohnen aus der Dose
 - (1) Haben sie die gleiche Farbe, so lege eine schwarze Bohne zurück
 - (2) Haben sie verschiedene Farben, so lege eine weiße Bohne zurück
- Eigenschaften
 - Terminierung mit genau einer Bohne
 - Das Ergebnis ist ^{un}abhängig von der anfangs gewählten Anordnung, *Reihenf. der Züge.*

Semi-Thue-System: $\Sigma = \{\text{schwarz ; weiß}\}$

$\lambda = $	schwarz schwarz	\rightarrow	$ $ schwarz	$= 1$
	weiß weiß	\rightarrow	schwarz	
	schwarz weiß	\rightarrow	weiß	
	weiß schwarz	\rightarrow	weiß	

▪ **Induktionsbehauptung:** Semi-Thue-System terminiert für alle Wörter x mit $|x| = n + 1$

- eine Regel ist immer anwendbar, da $n \geq 2$
- sei y das Wort nach Regelanwendung, also $|y| = n$
- nach Induktionshypothese terminiert das Semi-Thue System für y , $= n$

$x \Rightarrow y \quad |y| = |x| - 1$
 $= n + 1 - 1$
 $= n$

also terminiert das Semi-Thue System für x

Satz: Das Ergebnis (Farbe der letzten Bohne) ist ^{un}abhängig von der Reihenfolge der Anordnung und der Regelauswahl

Beweis: Angabe eines Beispiels. Welches?



$$\Sigma = \{0, L, B\}$$

$$T = \begin{array}{l} \boxed{0} \Rightarrow \boxed{L}, \\ \boxed{B} \Rightarrow \boxed{0}, \end{array}$$

$$BB \Rightarrow \epsilon$$

$$BOLLORB \Rightarrow LBLLORB \Rightarrow LOBLOB \Rightarrow LOORB$$

$$\Rightarrow LOOLBB \Rightarrow LOOL$$



1.6 Addition von 1 in Binärdarstellung ganzer Zahlen

Markov-Algorithmus:

$\alpha L \rightarrow L \alpha$	$\alpha O \rightarrow O \alpha$	$\alpha \rightarrow \beta$
$L \beta \rightarrow \beta O$	$O \beta \rightarrow \cdot L$	$\beta \rightarrow \cdot L$
$\epsilon \rightarrow \alpha$	ϵ -regel	

Handwritten notes:
 - Verschiebe (above first two rules)
 - Zustandswechsel (above third rule)
 - Übertrag (next to $O \beta \rightarrow \cdot L$)
 - Rechenregeln (next to $\beta \rightarrow \cdot L$)

Beispiellauf:

$\sim \cdot 8 \leftarrow 1$
 $LOLL \Rightarrow \alpha LOLL$ (ϵ -regel)
 $\Rightarrow L\alpha OLL$
 $\Rightarrow LO\alpha LL$
 $\Rightarrow LOL\alpha L$
 $\Rightarrow LOLL\alpha$
 $\Rightarrow LOLL\beta$
 $\Rightarrow LOL\beta O$
 $\Rightarrow LO\beta OO$
 $\Rightarrow LLOO$

Handwritten notes:
 - Verschiebe / esse (in a large bracket next to the sequence)
 - Zustandswechsel (in a large bracket next to the sequence)

