

Ringvorlesung “Automatische Verarbeitung natürlicher Sprache”

Sprachen in der Informatik und ihre Verarbeitung

Herbert Kuchen

Universität Münster, Institut für Wirtschaftsinformatik

0. Sprachen in der Informatik

Natürliche Sprachen

- Nachteil: historisch gewachsen
- Aufgaben und Probleme:
 - ★ Erkennung gesprochener Sprache
 - ★ Verarbeitung von Sprachdokumenten

0. Sprachen in der Informatik

Natürliche Sprachen

- Nachteil: historisch gewachsen
- Aufgaben und Probleme:
 - ★ Erkennung gesprochener Sprache
 - ★ Verarbeitung von Sprachdokumenten

Künstliche Sprachen

- zur Beschreibung z.B. von Abläufen, Daten, Funktionen, . . .
- Vorteil: Syntax lässt sich geeignet wählen
- Aufgaben und Probleme:
 - ★ Analyse von Syntax und Semantik
 - ★ Übersetzung

1. Künstliche Sprachen in der Informatik

- Programmiersprachen (z.B. Java, C, Cobol, Fortran, Prolog, Lisp)
- Skriptsprachen (z.B. PHP, Perl, JavaScript)
- Datenbankabfragesprachen (SQL)
- Spezifikationssprachen (z.B. B, VDM, Z)
- Webseitenbeschreibungssprachen (HTML, XML)
- Modellierungssprachen (z.B. UML)
- Geschäftsprozessbeschreibungssprachen (z.B. BPEL)
- Konfigurationssprachen (z.B. in J2EE)
-

Beispiel: Java

```
public <T extends Comparable<T>> int binsearch(T v, T[] a){
    int mid;
    int low = 0; int up = a.length-1;
    while (low <= up){
        mid = (low + up) / 2;
        if (v.compareTo(a[mid])<0) up = mid - 1;
        else if (v.compareTo(a[mid])>0) low = mid + 1;
        else return mid;}
    return -1;}

```

- fixiert Reihenfolge von elementaren Verarbeitungsschritten

Beispiel: Structured Query Language (SQL)

Veranstaltungen			Studierende			Hörende	
VNr	Dozent	Veranstaltung	MatNr	Name	Vorname	VNr	MatNr
18	Vossen	DBT	120312	Schmidt	Willy	18	120312
27	Kuchen	Rechnernetze	165214	Sommer	Karin	18	165214
8	Becker	IZI	218960	Winter	Uli	8	165214
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

gesucht: Wie heißen die Hörenden von Prof. Vossen?

SQL:

```
SELECT Studierende.Name
FROM Studierende, Veranstaltungen, Hörende
WHERE Veranstaltungen.Dozent = "Vossen" AND
      Studierende.MatNr = Hörende.MatNr AND
      Hörende.VNr = Veranstaltungen.VNr
```

- beschreibt **deklarativ** das gewünschte Ergebnis

Beispiel: Z

Überweisung

Δ Bank

betrag?: \mathbb{N}

von?, nach?: Konto

von? \neq nach?

$\exists k:0..anzahl-1 \bullet knr(k) = nach?$

$\exists j:0..anzahl-1 \bullet knr(j) = von?$

betrag? \leq kstand(j)

$kstand' = kstand \oplus \{j \mapsto kstand(j) - betrag?, k \mapsto kstand(k) + betrag?\}$

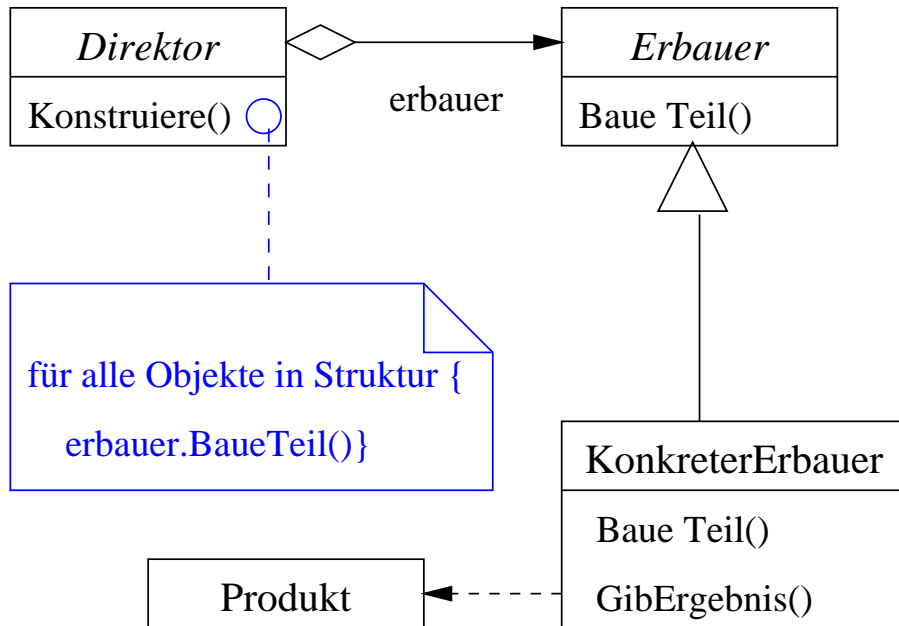
$knr' = knr$

$anzahl' = anzahl$

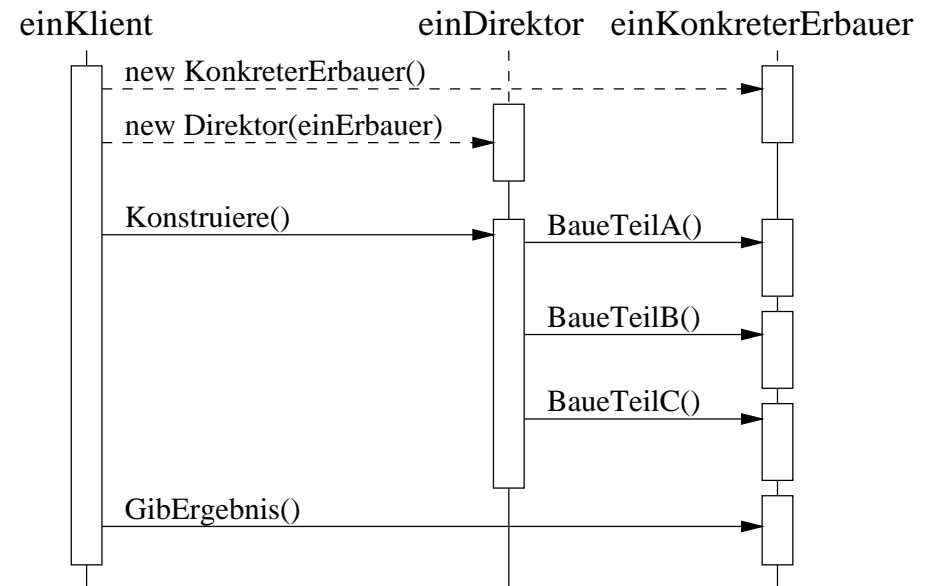
- spezifiziert System durch logische Formel
- umfasst graphische Elemente

Beispiel: UML

Klassendiagramm



Sequenzdiagramm



- graphische Modellierung

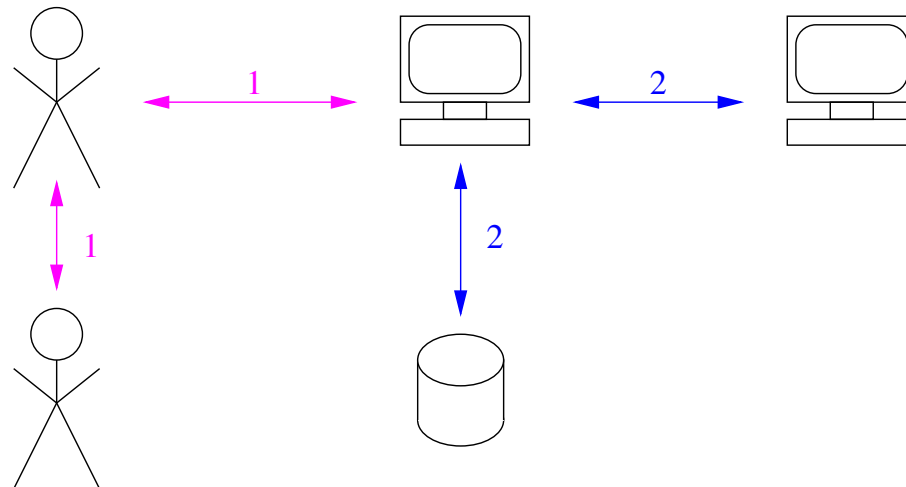
Beispiel: XML

```
<?xml:stylesheet type="text/xsl" href="simple.xsl" ?>
<breakfast-menu>
<food>
<name>Belgian Waffles</name>
<price>$5.95</price>
<description>
two of our famous Belgian Waffles with plenty of real maple syrup
</description>
<calories>650</calories>
</food>
<food>
<name>Strawberry Belgian Waffles</name>
<price>$7.95</price>
<description>
light Belgian waffles covered with strawberries and whipped cream
</description>
<calories>900</calories>
</food>
...
</breakfast-menu>
```

- schwer lesbar; benutzerdefinierte Struktur möglich (→ DTD)
- Verarbeitungswerkzeuge (z.B. Parser) verfügbar

Klassifikation künstlicher Sprachen nach Lesbarkeit

- menschenlesbare künstliche Sprachen
 - ★ z.B. Java, SQL, Z, UML
 - ★ zur Kommunikation unter Menschen bzw. zwischen Mensch und Rechner
- nicht bzw. kaum menschenlesbare Sprachen
 - ★ z.B. HTML, XML, BPEL
 - ★ zum Informationsaustausch zwischen Softwaresystemen
 - ★ oft auf XML basierend (→ Syntaxanalyse geschenkt)



Anforderungen an künstliche Sprachen

- bei Typ 1: intuitive, leicht verständliche Syntax
 - ★ dazu Anlehnung an natürliche Sprache (z.B. `if`, `while`)
- hinreichende Ausdruckskraft für betrachtete Anwendung
- effizient und unzweideutig automatisch verarbeitbar

Exkurs: Formale Sprachen und Grammatiken

- Alphabet Σ (endl. Menge, z.B. $\Sigma = \{a, b, c, \dots, A, \dots, 0, 1, \dots, \dots, !\}$)
- Def.: $L \subset \Sigma^*$ heißt „formale Sprache“ ($\Sigma^* := \bigcup_{i \in \mathbb{N}} \Sigma^i$)

Exkurs: Formale Sprachen und Grammatiken

- Alphabet Σ (endl. Menge, z.B. $\Sigma = \{a, b, c, \dots, A, \dots, 0, 1, \dots, \dots, !\}$)
- Def.: $L \subset \Sigma^*$ heißt „formale Sprache“ ($\Sigma^* := \bigcup_{i \in \mathbb{N}} \Sigma^i$)
- eine Grammatik G besteht aus:
 - N : endl. Menge von Nichtterminal-Symbolen
 - Σ : endl. Menge von Terminalsymbolen, $N \cap \Sigma = \emptyset$
 - $P \subseteq (N \cup \Sigma)^* \cdot N \cdot (N \cup \Sigma)^* \times (N \cup \Sigma)^+$ endliche Menge von Regeln
(Produktionen; Schreibweise: $w \rightarrow w'$)
 - $S \in N$ Startsymbol
- Grammatiken erlauben eine endlich lange Darstellung auch unendlicher Sprachen

Beispiel 1: Grammatik und Ableitung

$$G_1 := (\{E, T, F\}, \{(\,), 1, +, *\}, P, E)$$

$$P := \left\{ \begin{array}{lll} E \rightarrow T, & T \rightarrow F * T, & F \rightarrow 1, \\ E \rightarrow T + E, & T \rightarrow F, & F \rightarrow (E) \end{array} \right\}$$

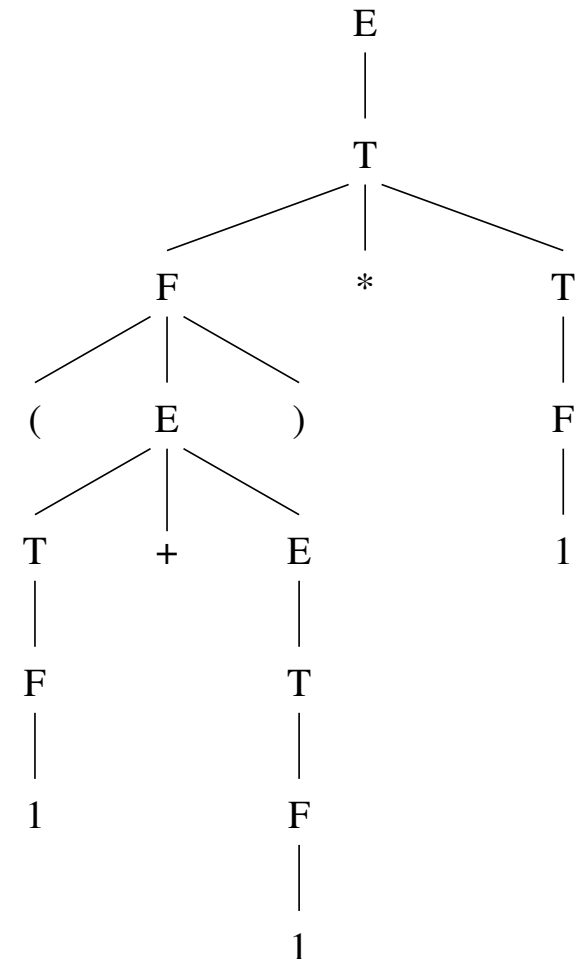
Beispiel 1: Grammatik und Ableitung

$$G_1 := (\{E, T, F\}, \{(\,), 1, +, *\}, P, E)$$

$$P := \left\{ \begin{array}{lll} E \rightarrow T, & T \rightarrow F * T, & F \rightarrow 1, \\ E \rightarrow T + E, & T \rightarrow F, & F \rightarrow (E) \end{array} \right\}$$

es gilt: $(1 + 1) * 1 \in L(G_1)$ denn:

$$\begin{aligned} E &\Rightarrow T \Rightarrow \\ &F * T \Rightarrow \\ &(E) * T \Rightarrow \\ &(T + E) * T \Rightarrow \\ &(F + E) * T \Rightarrow \\ &(1 + E) * T \Rightarrow \\ &(1 + T) * T \Rightarrow \\ &(1 + F) * T \Rightarrow \\ &(1 + 1) * T \Rightarrow \\ &(1 + 1) * F \Rightarrow \\ &(1 + 1) * 1 \end{aligned}$$



- beachte: Nichtdeterminismus; hier: Linksableitung

Beispiel 2: Grammatik und Ableitung

$$G_2 := (\{S, B, C\}, \{a, b, c\}, P, S)$$

$$P : \begin{array}{lll} S \rightarrow aSBC & CB \rightarrow BC & bB \rightarrow bb \\ S \rightarrow aBC & aB \rightarrow ab & bC \rightarrow bc \\ cC \rightarrow cc \end{array}$$

$$\text{z.B.: } S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow aaBBCC \Rightarrow \\ aabBCC \Rightarrow aabbCC \Rightarrow aabbcC \Rightarrow aabbcc$$

$$\text{es lässt sich zeigen: } L(G_2) = \{a^n b^n c^n \mid n \in \mathbb{N}_+\}$$

Chomsky-Hierarchie von Grammatiken

Typ	Bezeichnung	typische Regeln
0	uneingeschränkt	$AbC \rightarrow dA$
1	kontextsensitiv	$Aa \rightarrow ABc$
2	kontextfrei	$A \rightarrow aBCd$
3	regulär	$A \rightarrow a, A \rightarrow aB$



- den Grammatik-Klassen entsprechen Sprachklassen
- nicht alle formalen Sprachen lassen sich durch Grammatiken darstellen

Äquivalente Beschreibungsmechanismen für Sprachklassen

- Sprachen der gleichen Klasse durch **unterschiedliche Mechanismen** beschreibbar

Äquivalente Beschreibungsmechanismen für Sprachklassen

- Sprachen der gleichen Klasse durch **unterschiedliche Mechanismen** beschreibbar
- einige sind gut lesbar für Menschen → z.B. Verwendung in Handbüchern

Äquivalente Beschreibungsmechanismen für Sprachklassen

- Sprachen der gleichen Klasse durch **unterschiedliche Mechanismen** beschreibbar
- einige sind gut lesbar für Menschen → z.B. Verwendung in Handbüchern
- andere eignen sich für die effiziente, automatische Sprachverarbeitung
 - ★ z.B. beim Wortproblem (ist $w \in L$?)

Äquivalente Beschreibungsmechanismen für Sprachklassen

- Sprachen der gleichen Klasse durch **unterschiedliche Mechanismen** beschreibbar
- einige sind gut lesbar für Menschen → z.B. Verwendung in Handbüchern
- andere eignen sich für die effiziente, automatische Sprachverarbeitung
 - ★ z.B. beim Wortproblem (ist $w \in L$?)
- diese Darstellungen lassen sich **automatisch ineinander transformieren**

Äquivalente Beschreibungsmechanismen für Sprachklassen

- Sprachen der gleichen Klasse durch **unterschiedliche Mechanismen** beschreibbar
- einige sind gut lesbar für Menschen → z.B. Verwendung in Handbüchern
- andere eignen sich für die effiziente, automatische Sprachverarbeitung
 - ★ z.B. beim Wortproblem (ist $w \in L$?)
- diese Darstellungen lassen sich **automatisch ineinander transformieren**

Sprachklasse	leicht lesbar	für automatische Verarbeitung
0	Grammatik	Turing-Maschine
1	kontextsensitive Gr.	linear beschränkte TM
2	EBNF, Syntaxdiagramm	Kellerautomat (LL,LR,LALR)
3	regulärer Ausdruck	endlicher Automat (DEA,NEA)

Alternative Beschreibungsmechanismen für reguläre Sprachen

Reguläre Grammatik:

$$G_3 := (\{S, A, B\}, \{0, \dots, 9, \cdot, e\}, P, S)$$

$$P := \left\{ \begin{array}{lll} S \rightarrow 0S, & \dots & S \rightarrow 9S, \quad S \rightarrow \cdot A, \\ A \rightarrow 0A, & \dots & A \rightarrow 9A, \quad A \rightarrow eB, \\ B \rightarrow 0B, & \dots & B \rightarrow 9B, \quad B \rightarrow \varepsilon \end{array} \right\}$$

Alternative Beschreibungsmechanismen für reguläre Sprachen

Reguläre Grammatik:

$$G_3 := (\{S, A, B\}, \{0, \dots, 9, \cdot, e\}, P, S)$$

$$P := \left\{ \begin{array}{lll} S \rightarrow 0S, & \dots & S \rightarrow 9S, \quad S \rightarrow \cdot A, \\ A \rightarrow 0A, & \dots & A \rightarrow 9A, \quad A \rightarrow eB, \\ B \rightarrow 0B, & \dots & B \rightarrow 9B, \quad B \rightarrow \varepsilon \end{array} \right\}$$

Regulärer Ausdruck:

$$\rho = (0 \mid \dots \mid 9)^* \cdot (0 \mid \dots \mid 9)^* e (0 \mid \dots \mid 9)^*$$

Alternative Beschreibungsmechanismen für reguläre Sprachen

Reguläre Grammatik:

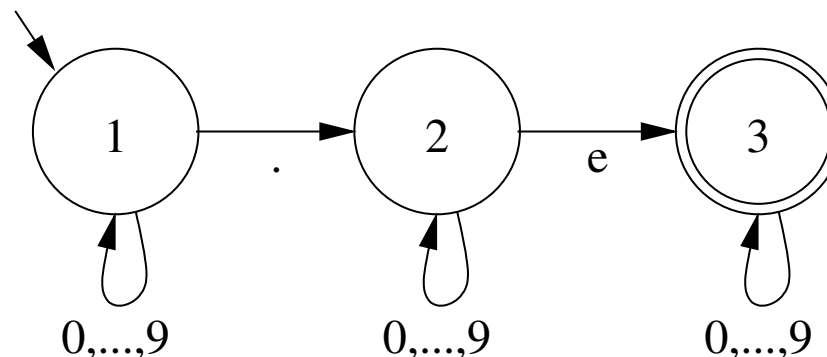
$$G_3 := (\{S, A, B\}, \{0, \dots, 9, \cdot, e\}, P, S)$$

$$P := \left\{ \begin{array}{lll} S \rightarrow 0S, & \dots & S \rightarrow 9S, & S \rightarrow \cdot A, \\ A \rightarrow 0A, & \dots & A \rightarrow 9A, & A \rightarrow eB, \\ B \rightarrow 0B, & \dots & B \rightarrow 9B, & B \rightarrow \varepsilon \end{array} \right\}$$

Regulärer Ausdruck:

$$\rho = (0 \mid \dots \mid 9)^* \cdot (0 \mid \dots \mid 9)^* e (0 \mid \dots \mid 9)^*$$

Endlicher Automat:



Verwendung der Sprachklassen in der Informatik

Klasse	eingesetzt in	Aufwand des Wortproblems
0	theoretischer Informatik	unentscheidbar
1	theoretischer Informatik	exponentiell
2	Syntax von Kunstsprachen, Data Dictionaries in Software Engineering	i.a. kubisch; oft linear
3	Mikrosyntax von Symbolen Software Engineering, Schaltkreisentwurf	lineare Zeit

Statische Semantik

- Teil der Semantik, der bereits bei der Übersetzung geprüft werden kann
- z.B.: stimmen Typen?
 - ★ $2 + \text{false}$ verboten, $2+5$ erlaubt

Statische Semantik

- Teil der Semantik, der bereits bei der Übersetzung geprüft werden kann
- z.B.: stimmen Typen?
 - ★ `2 + false` verboten, `2+5` erlaubt
- sind verwendete Bezeichner deklariert?
 - ★ `int x; . . . x = 0;`

Statische Semantik

- Teil der Semantik, der bereits bei der Übersetzung geprüft werden kann
- z.B.: stimmen Typen?
 - ★ $2 + \text{false}$ verboten, $2+5$ erlaubt
- sind verwendete Bezeichner deklariert?
 - ★ $\text{int } x; \dots x = 0;$
- statische Semantik wird beschrieben durch **attributierte Grammatik**, d.h.
 - ★ kontextfreie Grammatik mit semantischen Zusatzregeln
 - ★ z.B. $E \rightarrow E + E$, $\$0 = \begin{cases} \text{int,} & \text{falls } \$1 == \text{int und } \$2 == \text{int} \\ \text{undefiniert,} & \text{sonst} \end{cases}$
- geeignete attributierte Grammatiken erlauben eine Auswertung in Linearzeit

Dynamische Semantik

- Teil der Semantik, der zur Laufzeit bearbeitet wird
- exakt beschreibbar durch Mathematik (z.B. Funktion, Relation)
 - ★ $f(\text{Eingabe}) = \text{Ausgabe}$
 - ★ $f(\text{Zustand}) = \text{Nachfolgezustand}$

Dynamische Semantik

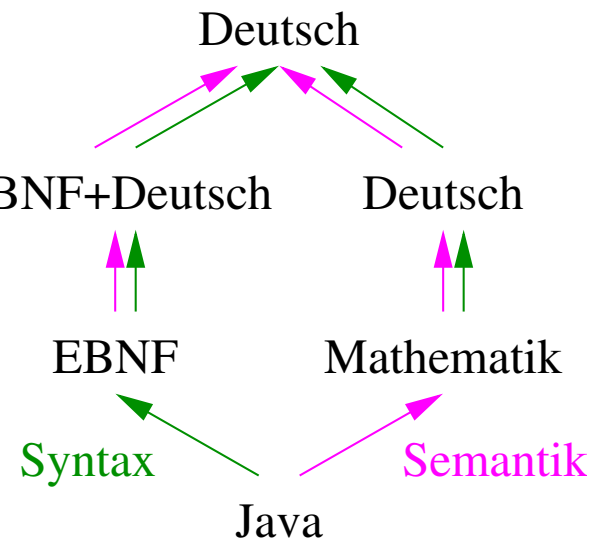
- Teil der Semantik, der zur Laufzeit bearbeitet wird
- exakt beschreibbar durch Mathematik (z.B. Funktion, Relation)
 - ★ $f(\text{Eingabe}) = \text{Ausgabe}$
 - ★ $f(\text{Zustand}) = \text{Nachfolgezustand}$

Meta–Meta–Meta–Sprache

Meta–Meta–Sprache

Metasprache

Objektsprache



2. Handhabung natürlicher Sprache in der Informatik

→ z.B. Prof. Ney, RWTH Aachen

- Erkennung gesprochener Sprache
- Verarbeitung von (textuellen) Sprachdokumenten
- heute: Einsatz in Nischen (Auto, Handy, Babelfish . . .)

Erkennung gesprochener Sprache

- basierend auf akustisch-linguistischer Modellierung und Statistik
- Parameter werden ggf. aus Trainingdaten gelernt
- Berücksichtigung des Kontexts

Erkennung gesprochener Sprache

- basierend auf akustisch-linguistischer Modellierung und Statistik
- Parameter werden ggf. aus Trainingdaten gelernt
- Berücksichtigung des Kontexts
- z.B. $Sch\gamma e$ und $\gamma \in \{l, r\} \rightarrow Schule$

Erkennung gesprochener Sprache

- basierend auf akustisch-linguistischer Modellierung und Statistik
- Parameter werden ggf. aus Trainingdaten gelernt
- Berücksichtigung des Kontexts
- z.B. $Schu\gamma e$ und $\gamma \in \{l, r\} \rightarrow$ Schule
- z.B. $\dots tt\gamma$ und $\gamma \in \{er, a\} \rightarrow \dots tter$

Erkennung gesprochener Sprache

- basierend auf akustisch-linguistischer Modellierung und Statistik
- Parameter werden ggf. aus Trainingdaten gelernt
- Berücksichtigung des Kontexts
- z.B. $Schu\gamma e$ und $\gamma \in \{l, r\} \rightarrow$ Schule
- z.B. $\dots tt\gamma$ und $\gamma \in \{er, a\} \rightarrow \dots tter$
- z.B. $helft$ den armen vögeln + Thema Tierschutz \rightarrow

Erkennung gesprochener Sprache

- basierend auf akustisch-linguistischer Modellierung und Statistik
- Parameter werden ggf. aus Trainingdaten gelernt
- Berücksichtigung des Kontexts
- z.B. $Schu\gamma e$ und $\gamma \in \{l, r\} \rightarrow$ Schule
- z.B. $\dots tt\gamma$ und $\gamma \in \{er, a\} \rightarrow \dots tter$
- z.B. $helft$ den armen vögelN + Thema Tierschutz \rightarrow Helft den armen VögelN!

Erkennung gesprochener Sprache

- basierend auf akustisch-linguistischer Modellierung und Statistik
- Parameter werden ggf. aus Trainingdaten gelernt
- Berücksichtigung des Kontexts
- z.B. Schu γ e und $\gamma \in \{l, r\} \rightarrow$ Schule
- z.B. . . . tt γ und $\gamma \in \{er, a\} \rightarrow$. . . tter
- z.B. helf γ t den armen vögel γ n + Thema Tierschutz \rightarrow Helf γ t den armen Vögel γ n!
- Zusammenfügen von Lauten zu Worten und dann zu Sätzen
- Ziel: Überführung in leicht weiterverarbeitbare, rechnerinterne Darstellung
z.B. Syntaxbaum
- möglichst: robuste, sprecherunabhängige, adaptive Erkennung

Verarbeitung von (textuellen) Sprachdokumenten

- Sprachverstehen
 - ★ ggf. OCR (Optical character recognition)
 - ★ Analyse von Syntax und Semantik
- Übersetzung, z.B.:
 - ★ in andere Sprache
 - ★ Dokumentenklassifikation
 - ★ Ausgabe gesprochener Sprache

3. Zusammenfassung

- Künstliche Sprachen
 - ★ i.d.R. gut lesbar
 - ★ ausreichend ausdruckskräftig
 - ★ effizient verarbeitbar
- Natürliche Sprachen
 - ★ Verarbeitung basierend auf akustisch-linguistischer Modellierung + Statistik