Einführung in die Computerlinguistik und Sprachtechnologie

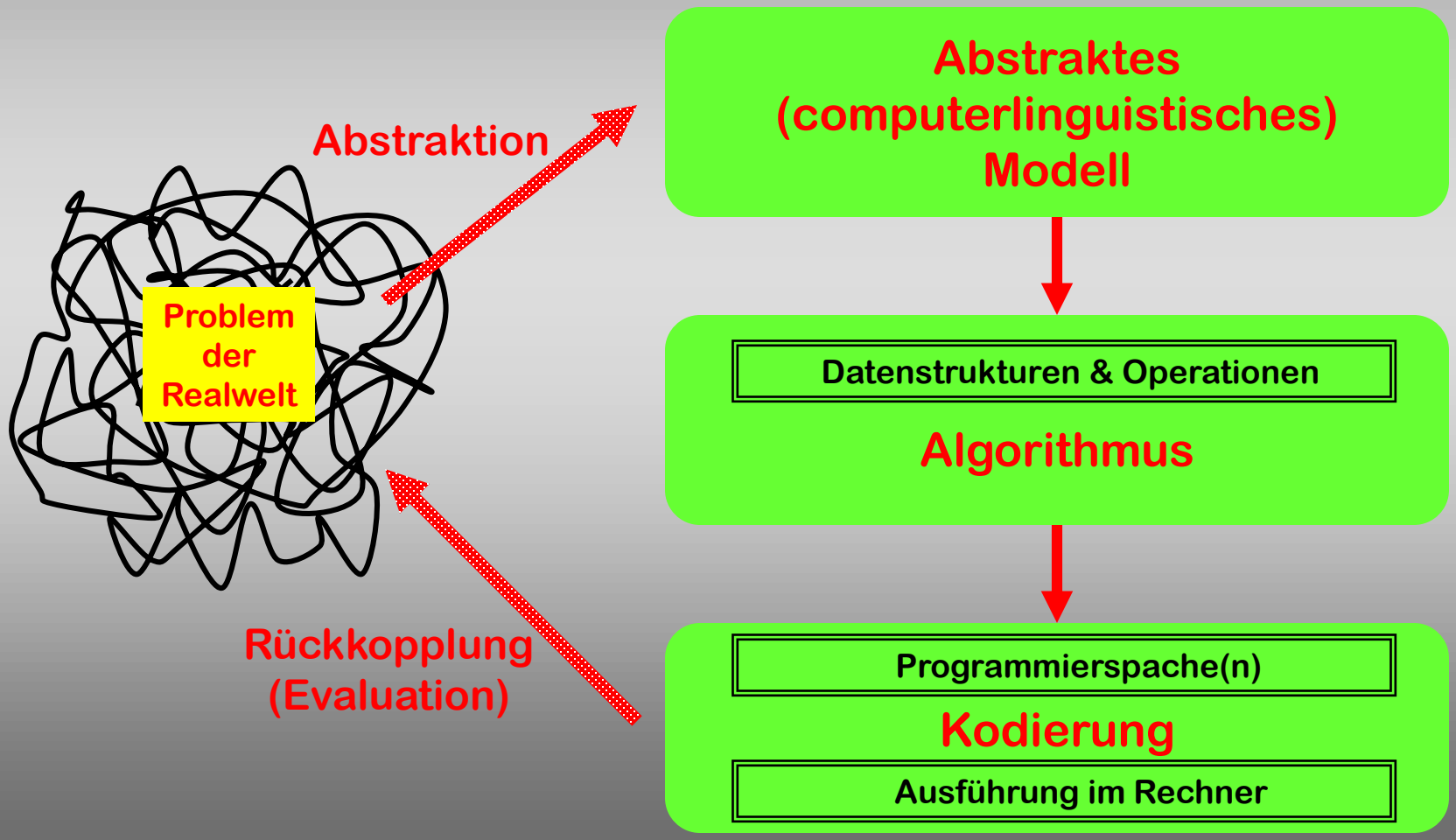
Vorlesung im WiSe 2018/19
(B-GSW-12)

Prof. Dr. Udo Hahn

Lehrstuhl für Computerlinguistik
Institut für Germanistische Sprachwissenschaft
Friedrich-Schiller-Universität Jena

http://www.julielab.de

Informatischer Problemlösungszyklus



Informatischer Problemlösungszyklus

- **Modellbildung**
  - **Abstraktion von allen unwesentlichen Details der Problemstellung im Hinblick auf die algorithmische Lösung**
  - **Spezifikation der logischen Abhängigkeiten zwischen problemlösungsrelevanten Objekten**
  - **(computer)linguistisches Wissen**

Informatischer Problemlösungszyklus

- Algorithmisierung
  - Übersetzung der modellbezogenen Spezifikation in
    - eine Menge von Objekten (Datenstrukturen) mit bestimmten Eigenschaften und Beziehungen zueinander
    - die erlaubten Operationen auf diesen Objekten
  - Algorithmus: (möglichst präzise) Beschreibung einer Folge zulässiger Operationen auf den Objekten, um das Problem zu lösen
  - Computerlinguistische Kernexpertise – verlangt informatisches Grundlagenwissen

Informatischer Problemlösungszyklus

- Kodierung (Programmierung)
  - Übersetzung der algorithmischen Spezifikation in Konstrukte einer (geeigneten) Programmiersprache
- Ausführung des Programms
  - Hier erst Bezug auf konkrete Maschinen (Datenstrukturen und Algorithmen sind abstrakte Konstruktionen)
  - Test-Modifikationszyklus ... Dokumentation !
  - Informatisches Know-How

Morphologische Prozesse: Flexion - Deflexion

- Kombination von Grundformen mit Flexionsaffixen (Kasus, Numerus, Tempus)
  - Deklination
    - Land: Land, Landes, Lande, Länder, Ländern
  - Konjugation
    - landen: lande, landest, landet, landeten , gelandet
- primär syntaktische, nur minimale semantische Information, kein grundlegender Wortartwechsel

Morphologische Prozesse: Derivation - Dederivation

- Kombination von Grundformen mit Derivationsaffixen
  - Land: landen, verlanden, anlanden,
  - Land: Landung, Verlandung , Anlandung
  - Land: ländlich, verländlichen, Verländlichung
- modifizierende semantische Information, häufig mit Wortartwechsel verbunden

Morphologische Prozesse: Komposition - Dekomposition

- Kombination von Grundformen mit Grundformen (mittels Fugeninfixen)
  - Land: Landnahme, Landflucht, Landgang
  - Land: Heimatland, Ausland, Bauland
  - Land: Landesrekord, Landesverrat, Landsmann
  - Land: Inlandsflug, Landesratspräsidentengattin
- starke semantische Modifikation, fast keine Wortartwechsel
  - … aber: Rotkehlchen, Weichteile

Lemmatisierung

Eingabe	Lemma	
Töchtern	Tochter	
Hauses	Haus	
sagte	sagen	
Spiegelungen	Spiegelung	
leichter	leicht	
verlängerte	verlängert	
	verlängern	

Lemmatisierung vs. Stemming

Eingabe	Lemma	Stem/Stemming
Töchtern	Tochter	Töchter
Hauses	Haus	Haus
sagte	sagen	sagen, sag
Spiegelungen	Spiegelung	Spiegel
leichter	leicht	leicht
verlängerte	verlängert	läng
	verlängern	läng

Lemmatisierung vs. Stemming

Eingabe	Lemma	Stem/Stemming
Töchtern	Tochter	Töchter
Hauses	Haus	Haus
sagte	sagen	sagen, sag
Spiegelungen	Spiegelung	Spiegel
leichter	leicht	leicht
verlängerte	verlängert	läng
	verlängern	läng

Bestandteile der Problemlösung für morphologisches Stemming

- Linguistisches Wissen
  - Morphologische Struktur von Wörtern:

    $\underline{\text{WORT}} \eqsim \text{AFFIX}_1 \otimes \ldots \otimes \text{AFFIX}_k \otimes \text{STAMM} \otimes$

    $\otimes \text{AFFIX}_{k+1} \otimes \ldots \otimes \text{AFFIX}_n$

  - $\text{AFFIX}_{1..k}$ heißen Präfixe, $\text{AFFIX}_{k+1..n}$ Suffixe
  - Deklarativ (Strukturbeschreibung)
- Computerlinguistisches Wissen
  - Suffixabtrennungsalgorithmus (suffix stripping):

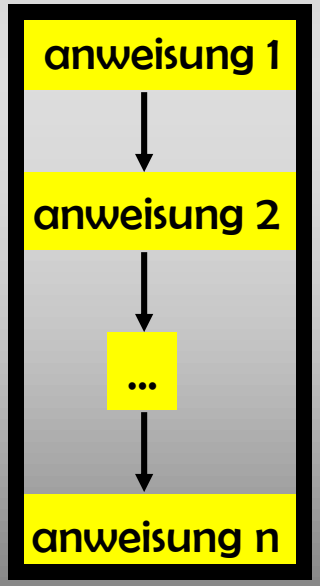
    $\text{STAMM} \otimes \text{AFFIX}_{k+1} \otimes \ldots \otimes \text{AFFIX}_n \rightarrow \text{STAMM}$
  - Prozedural (Aktionsbeschreibung)

Algorithmische Sprachkonstrukte
Anweisungsfolge

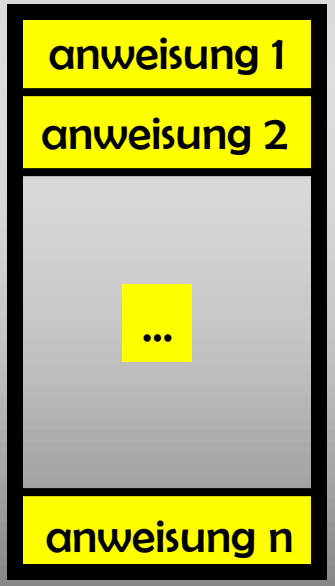
PSEUDOCODE

```
anweisung 1;
anweisung 2;
...
...
anweisung n;
```

FLUSSDIAGRAMM



STRUKTOGRAMM

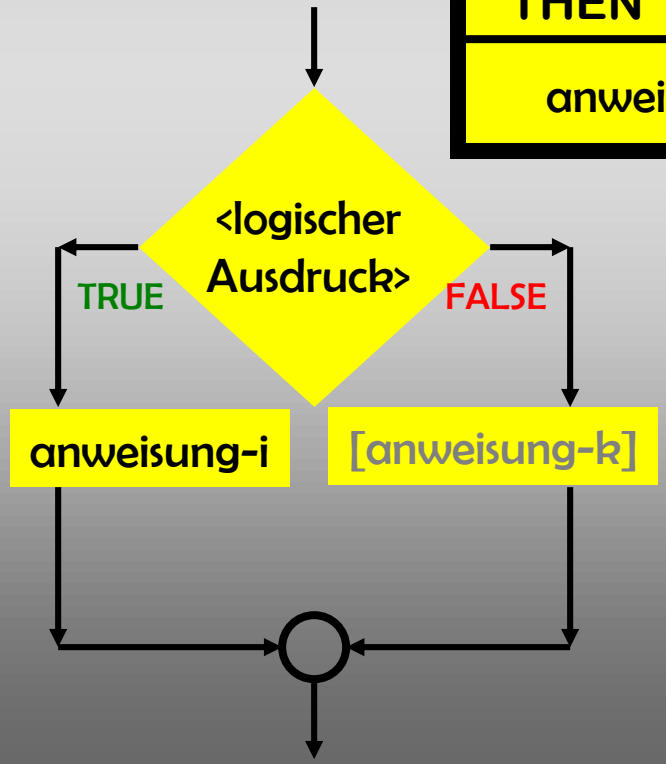


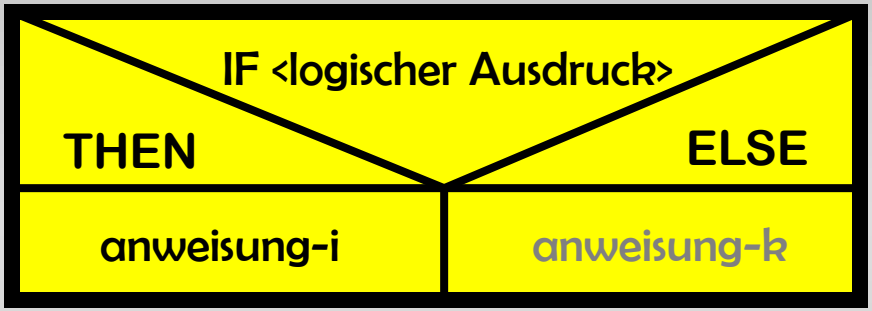
Algorithmische Sprachkonstrukte
Bedingte Anweisungen (IF)

PSEUDOCODE **FLUSSDIAGRAMM** **STRUKTOGRAMM**

IF <logischer Ausdruck>
THEN anweisung-i;
(ELSE anweisung-k;)

„*falls* <logischer Ausdruck>
TRUE *führe aus:*
anweisung-i;
(sonst *führe aus:*
anweisung-k;)“



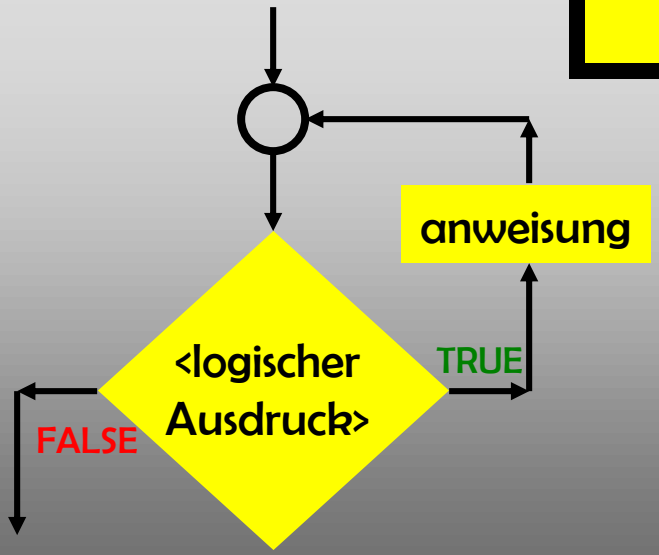


Algorithmische Sprachkonstrukte
Repetierte Anweisungen (WHILE)

PSEUDOCODE **FLUSSDIAGRAMM** **STRUKTOGRAMM**

WHILE <logischer Ausdruck> DO
anweisung;

„*solange* <logischer Ausdruck>
TRUE
führe aus:
anweisung“



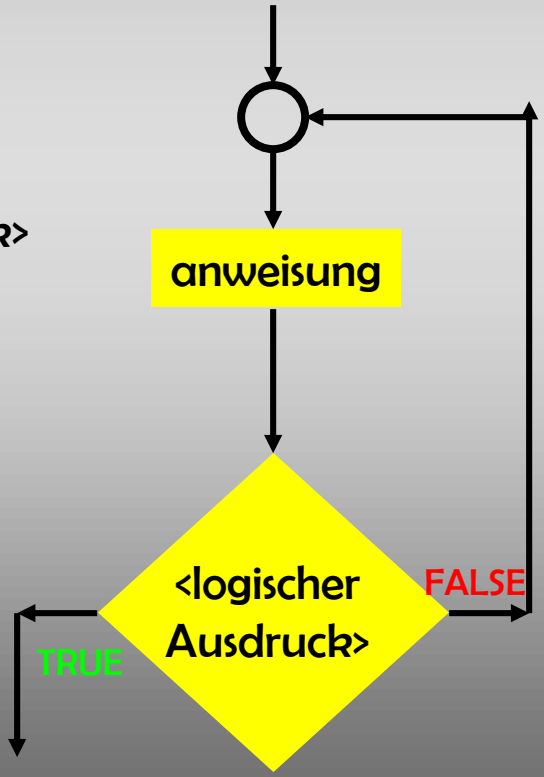
Algorithmische Sprachkonstrukte
Repetierte Anweisungen (REPEAT)

PSEUDOCODE

REPEAT anweisung;
UNTIL <logischer Ausdruck>

„ führe aus:
anweisung
solange <logischer Ausdruck>
FALSE“

FLUSSDIAGRAMM



STRUKTOGRAMM

REPEAT	anweisung
UNTIL	<logischer Ausdruck>

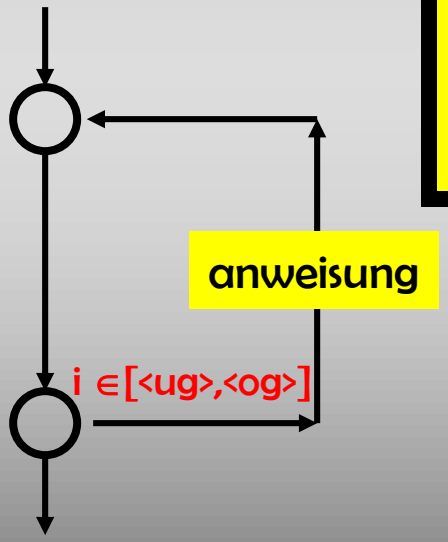
Algorithmische Sprachkonstrukte
Repetierte Anweisungen (FOR)

PSEUDOCODE

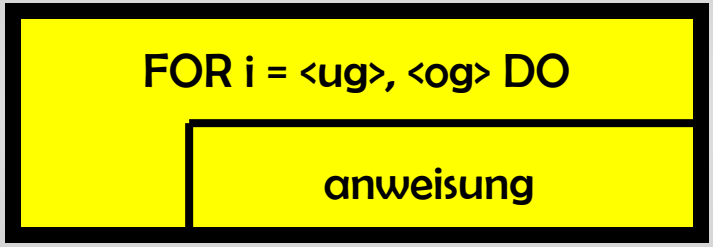
FOR i=<ug>,<og> DO
anweisung;

„ führe aus:
anweisung
solange i ∈[<ug>,<og>]“

FLUSSDIAGRAMM



STRUKTOGRAMM



Porter-Stemmer

[Porter 1980]

- (Vereinfachte) morphologische Struktur von englischen Wörtern
  - Ein Stamm, gefolgt von einem oder mehreren morphologischen Affixen ($\text{STAMM} \otimes \text{AFFIX}_{k+1} \otimes \ldots \otimes \text{AFFIX}_n$)
- (Porter-)Stemming
  - Verfahren zur Reduktion eines beliebigen englischen Eingabeworts auf seinen (morphologisch oft nicht-kanonischen) Stamm durch Eliminierung/Transformation aller Affixe
  - Regelbasierter, heuristischer Ansatz
    - *abate, abatements, abated* → *abat*
    - 6 Regelsätze in sequentieller Ordnung

      (<condition>) IF →THEN-Regeln

Porter-Stemmer – Definitionen I

[Porter 1980]

- VOKAL
  - Die Buchstaben ‚A‘, ‚E‘, ‚I‘, ‚O‘, ‚U‘ und ‚Y‘
- KONSONANT
  - Jeder andere Buchstabe
- Damit haben alle Wörter die Form
  - $(C)\ (VC)^m\ (V);\ m \geq 0$
  - C : Folge von einem oder mehreren Konsonanten
  - V : Folge von einem oder mehreren Vokalen
  - Beispiel: *troubles* = CVCVC

Porter-Stemmer – Definitionen I

[Porter 1980]

- VOKAL
  - Die Buchstaben ‚A‘, ‚E‘, ‚I‘, ‚O‘, ‚U‘ und ‚Y‘
- KONSONANT
  - Jeder andere Buchstabe
- Damit haben alle Wörter die Form
  - $(C)\ (VC)^m\ (V);\ m \geq 0$
  - C : Folge von einem oder mehreren Konsonanten
  - V : Folge von einem oder mehreren Vokalen
  - Beispiel: *troubles* = CVCVC

Porter-Stemmer – Definitionen I

[Porter 1980]

- VOKAL
  - Die Buchstaben ‚A‘, ‚E‘, ‚I‘, ‚O‘, ‚U‘ und ‚Y‘
- KONSONANT
  - Jeder andere Buchstabe
- Damit haben alle Wörter die Form
  - $(C)\ (VC)^m\ (V);\ m \geq 0$
  - C : Folge von einem oder mehreren Konsonanten
  - V : Folge von einem oder mehreren Vokalen
  - Beispiel: *troubles* = CVCVC

Porter-Stemmer – Definitionen I

[Porter 1980]

- VOKAL
  - Die Buchstaben ‚A‘, ‚E‘, ‚I‘, ‚O‘, ‚U‘ und ‚Y‘
- KONSONANT
  - Jeder andere Buchstabe
- Damit haben alle Wörter die Form
  - $(C)\ (VC)^m\ (V);\ m \geq 0$
  - C : Folge von einem oder mehreren Konsonanten
  - V : Folge von einem oder mehreren Vokalen
  - Beispiel: *troubles* = CVCVC

Porter-Stemmer – Definitionen I

[Porter 1980]

- VOKAL
  - Die Buchstaben ‚A‘, ‚E‘, ‚I‘, ‚O‘, ‚U‘ und ‚Y‘
- KONSONANT
  - Jeder andere Buchstabe
- Damit haben alle Wörter die Form
  - $(C)\ (VC)^m\ (V);\ m \geq 0$
  - C : Folge von einem oder mehreren Konsonanten
  - V : Folge von einem oder mehreren Vokalen
  - Beispiel: *troubles* = CVCVC

Porter-Stemmer – Definitionen I

[Porter 1980]

- VOKAL
  - Die Buchstaben ‚A‘, ‚E‘, ‚I‘, ‚O‘, ‚U‘ und ‚Y‘
- KONSONANT
  - Jeder andere Buchstabe
- Damit haben alle Wörter die Form
  - (C) $(VC)^m$ (V); $m \geq 0$
  - C : Folge von einem oder mehreren Konsonanten
  - V : Folge von einem oder mehreren Vokalen
  - Beispiel: *troubles* = CVCVC

Porter-Stemmer – Definitionen I

[Porter 1980]

- VOKAL
  - Die Buchstaben ‚A', ‚E', ‚I', ‚O', ‚U' und ‚Y'
- KONSONANT
  - Jeder andere Buchstabe
- Damit haben alle Wörter die Form
  - $(C)\ (VC)^m\ (V);\ m \geq 0$
  - C : Folge von einem oder mehreren Konsonanten
  - V : Folge von einem oder mehreren Vokalen
  - Beispiel: *troubles* = CVCVC
- *Longest matching* hat Priorität bei Regelauswahl innerhalb einer Klasse

Porter-Stemmer - Definitionen II

[Porter 1980]

- m()
  - Gibt die Anzahl von Vokal-Konsonantensequenzen im aktuellen Stamm zurück (<..> optional)
    - <c><v> ergibt ‚0' (cry ← cry-ing)
    - <c>vc<v> ergibt ‚1' (care ← car-ing, scare ← scar-ing)
    - <c>vcvc<v> ergibt ‚2' (probab ← probab-ility)
- *χ: Stamm endet mit Buchstaben χ (χ aus A..Z)
- *v*: Stamm enthält Vokal
- *d : Stamm enthält Doppelkonsonant (z.B. ‚LL')
- *o : Stamm endet in der Form Konsonant-Vokal-Konsonant; 2. Konsonant nicht ‚W', ‚X' oder ‚Y'

Porter-Stemmer – Schritte für das Englische

[Porter 1980]

1. Eliminierung von Pluralendungen und 3PS
2. Eliminierung von Past Tense und Verlaufsform bei Verben
3. Y→I Transformation
4. Derivationsmorphologie I: Doppelsuffixe
5. Derivationsmorphologie II: Einzelsuffixe
6. Clean-up (Aufräumen)