Computerlinguistische Grundlagen

Jürgen Hermes

Sommersemester 19

Sprachliche Informationsverarbeitung

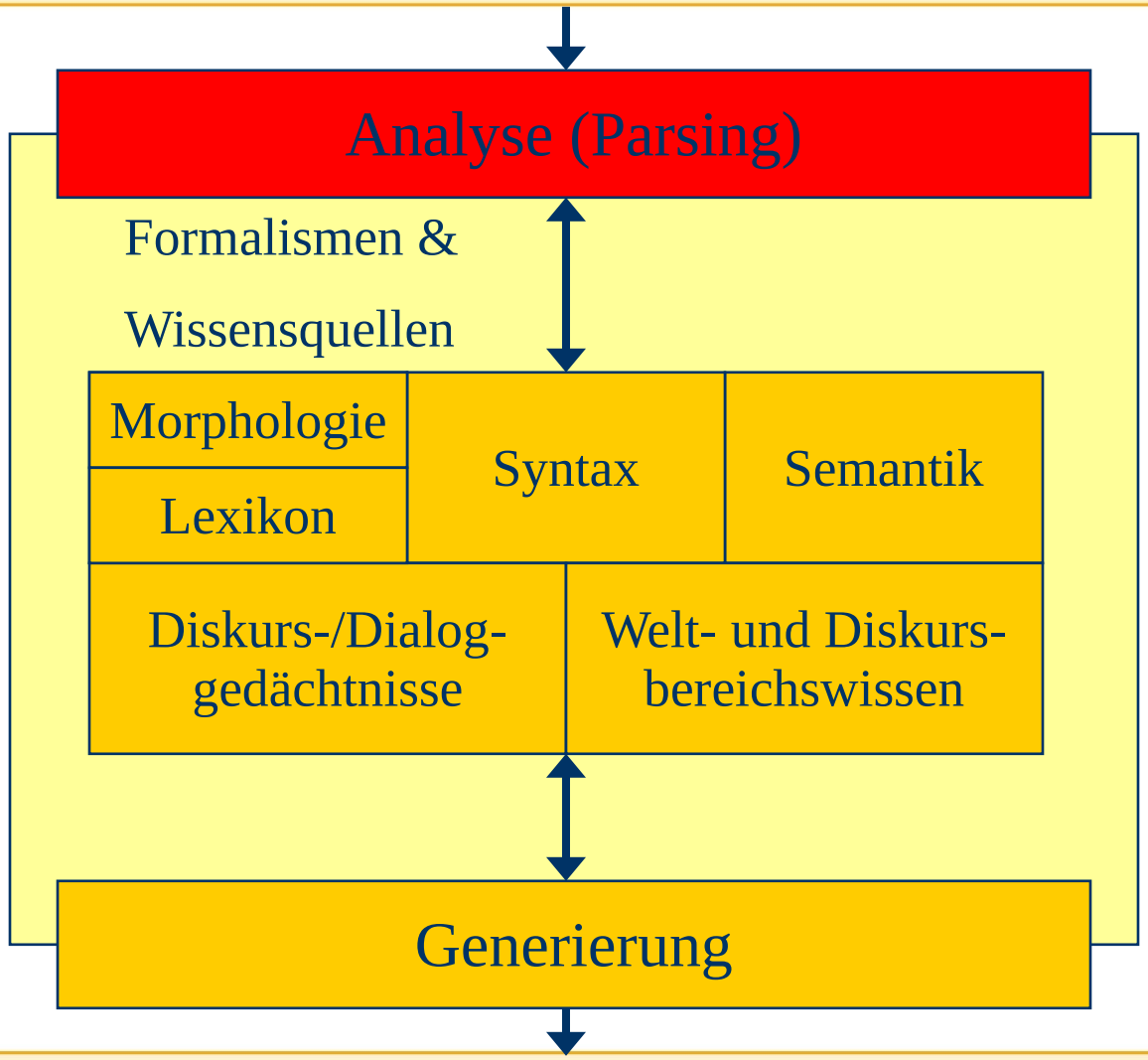
Institut für Digital Humanities

Universität zu Köln

Parser

- **Begriffe, Anwendungen, Strategien, Überblick**
- **Elementare Parsingalgorithmen**
  - **Top-Down-Parser**
  - **Bottom-Up-Parser**
- **Komplexere Parser**
  - **Chart-Parser**
  - **Dependenz-Parser**

Wiederauflage: Architektur eines natürlichsprachlichen Systems



Parsing – Begriffe und Motivation (I)

- Parsing ist ein **algorithmisches Verfahren**, bei dem eine natürlichsprachliche Eingabekette auf eine strukturelle Beschreibung in einer formalen Repräsentationssprache abgebildet wird (vs. *Recognizer*)
- **Motivation** für Parsing in der Computerlinguistik ist eine theoretisch fundierte und algorithmisch präzise Rekonstruktion des Sprachverstehens
- **Wissensquellen** für Parser sind die Wissensquellen eines natürlichsprachlichen Systems

Parsing – Begriffe und Motivation (II)

- **Komplexität** von Parsern: Zeit- und Raumbedarf der zugrundeliegenden Algorithmen. Kontextfreie bzw. schwach kontextsensitive Grammatiken gelten als effizient berechenbar und genügend ausdrucksstark.
- **Hauptproblem** für Parser ist die Disambiguierung auf lexikalischer, syntaktischer, semantischer und pragmatischer Ebene.

Anwendungsgebiete für Parser

- **Compilerbau**: Quellcode → Compilierung → Objektcode (syntaktische & semantische Analyse)
- **Kognitive Psychologie**: Interaktion syntaktischer und semantischer Prozesse
- **Computerlinguistik**: Wohlgeformtheitsüberprüfung; Disambiguierung; Strukturbeschreibungsaufbau; Identifikation und Korrektur der Eingabe

Überblick über die gängisten Parser
(*schwarz* diejenigen, die wir behandeln)

- **Elementare Parser**: Top-Down, Bottom-Up, Left-Corner
- **Chart-Parser:** Earley Algorithmus, Active Chart Parsing
- **Statistische Parser**
- **Deterministische Parser**
- **Konnektionistische Parser**
- **Dependenzparser**

Parsing-Strategien

- **Analyserichtung**: top-down vs. bottom-up
- **Suchstrategien**: breadth-first (parallel) vs. depth-first (sequenziell)
- **Verfahrensweise**: deterministisch vs. nicht-deterministisch
- **Verarbeitungsrichtung**: uni- vs. bidirektional
- **Kommunikation**: online vs. offline

Analyserichtungen

- **Top-Down-Parsing**
  - zielgesteuert, von oben nach unten
  - Regelexpansion: linke Regelseite → rechte Regelseite
  - terminiert, wenn abgeleiteter Ausdruck dem Eingabesatz entspricht
  - Probleme: Erzeugung evtl. nicht-terminierbarer Knoten; keine links- bzw. rechtsrekursiven Regeln zugelassen
- **Bottom-Up-Parsing**
  - datengesteuert, von unten nach oben
  - Regelreduktion: rechte Regelseite → linke Regelseite
  - terminiert, wenn abgeleiteter Ausdruck dem Startsymbol entspricht
  - Probleme: Bildung evtl. nicht reduzierbarer Konstituenten; keine Tilgungsregeln zugelassen

Suchstrategien

- **Breadth-First:** parallel: mehrere Alternativen der Regelanwendung werden simultan verfolgt
- **Depth-First:** sequenzielles Abarbeiten der Alternativen mit Protokoll der Auswahlentscheidungen und eventuellem Zurücksetzen
- „**Best-First“:**
  - Heuristisches Parsen
  - Aufwandsreduktion, Analysenrobustheit
  - Typen: partielles Parsing – Insel-Parsing – Skimming

Verfahrensweise

- **Nichtdeterministisches Parsen**:
  - Grammatiken erlauben mehrere Alternativen
  - Suchstrategien erforderlich
- **Deterministisches Parsen**:
  - In jedem Zustand ist nur ein einziger Ableitungsschritt möglich
  - Verwendet v.a. im Compilerbau
  - Bei Einsatz für natürliche Sprache: Aufschieben der Entscheidung (look-ahead; wait & see)
  - Vertreter: PARSIFAL, Tomita-Algorithmus

Top-Down Recognizer

Daten:	Lexikon und kontextfreie Syntax
Eingabe:	Satz (w) mit der Länge n>=0
Ausgabe:	TRUE/FALSE
Arbeitsstrukturen:	nächstes Wort (Anfangswert: 1)
	Struktur (Anfangswert: Startsymbol [S])
Methode MAIN:	WENN Ableitung leer und Wort == n+1
	→Return TRUE
	SONST Reduziere(Expandiere(Ableitung))
PROZEDUR Expandiere:	Wende Regel an, übergebe Ableitung
PROZEDUR Reduziere:	Ersetze lexikalische Kategorien, gehe zum nächstem Wort

Top-Down Parser

Daten:	Lexikon und kontextfreie Syntax
Eingabe:	Satz (w) mit der Länge n>=0
Ausgabe:	Strukturbeschreibung für w
Arbeitsstrukturen:	Wort (Anfangswert 1)
	Struktur (Anfangswert Startsymbol S)
	Position (Anfangswert 1)
Methode MAIN:	WENN Struktur == w liefere sie zurück
	SONST Reduziere(Expandiere(Struktur))
PROZEDUR Reduziere:	Ändere Variable Position
PROZEDUR Expandiere:	Ändere Variablen Struktur & Position

Bottom-Up Recognizer

Daten:	Lexikon und kontextfreie Syntax
Eingabe:	STACK1([Eingabesatz w]) mit n>=1
	STACK2([ ])
Ausgabe:	TRUE/FALSE
Methode MAIN:	WENN STACK2 nicht leer
	DANN Reduce(STACK2)
	SONST WENN STACK1 nicht leer
	DANN Shift(STACK1,STACK2)
	SONST Return TRUE/FALSE
PROZEDUR Shift:	Schreibe Elemente von STACK1 in STACK2
PROZEDUR Reduce:	Ersetze Elemente in STACK2 nach Regeln der Syntax oder des Lexikons

Literatur / Hausaufgabe

Zur Nachbereitung:

Naumann, Langer (1994): Kapitel 1 und 2 (S. 3-36)

Geben Sie Syntax und Lexikon an, mit dem man den Satz
„Das trächige Wildschwein schläft in dem Unterholz“
parsen kann. Führen Sie jeweils einen *Top-Down*- und einen *Bottom-Up-Recognizing*-Durchgang durch. Welche Methode kommt ohne Suchstratiegie aus?

Zur Vorbereitung:

Naumann, Langer (1994): Chart-Parsing (S. 102-106)