# Natural Language Processing

- man-machine communication
  - spoken dialogue with information systems or robots
- information extraction from text
  - question answering
  - summarization
  - structuring of textual data

# Levels of Language Description

- syntax: What is a well-formed sentence?
- semantics: Whats the meaning of a sentence?
- pragmatics: Whats the purpose of a sentence?

- practical solutions require task specific constraints
- e.g. fixed pragmatics: question answering
- e.g. restricted syntax: machine translation for controlled language
- e.g. restricted semantics: ignore lexical or scopus ambiguity

# Context-Free Grammars

- Compositionality: complex constituents are built by *concatenation* of elementary constituents
  - grammar rules:

    s $\rightarrow$ np vp
  - axiomatization: $s(X) \leftarrow np(Y), vp(Z), append(Y, Z, X).$
  - pre-terminal rules (dictionary):

    n $\rightarrow$ frau
  - axiomatization: $n([frau]) \leftarrow true.$

# Context-Free Grammars

- sample grammar:

$$s(X) \leftarrow np(Y), vp(Z), append(Y, Z, X).$$
$$np(X) \leftarrow d(Y), n(Z), append(Y, Z, X).$$
$$vp(X) \leftarrow v(X).$$

$$d([die]) \leftarrow true.$$
$$n([frau]) \leftarrow true.$$
$$v([liest]) \leftarrow true.$$

# Context-Free Grammars

- Generation (with SLR resolution):

$$
\begin{array}{l}
?- s(X). \\
\quad ?- np(Y1), vp(Z1), append(Y1, Z1, X1). \\
\quad\quad ?- d(Y2), n(Z2), append(Y2, Z2, X2). \\
\quad\quad ?- n(Z2), append([die], Z2, X2). \\
\quad\quad ?- append([die], [frau], X2). \\
\quad ?- vp(Z1), append(Y1, Z1, X1). \\
\quad ?- append([die, frau], [liest], X1). \\
X = [die, frau, liest]
\end{array}
$$

# Context-Free Grammars

- Recognition (with SLR resolution):

$$?- s([die, frau, liest]).$$
$$?- np(Y1), vp(Z1), append(Y1, Z1, [die, frau, liest]).$$
$$?- d(Y2), n(Z2), append(Y2, Z2, X2).$$
$$?- n(Z2), append([die], Z2, X2).$$
$$?- append([die], [frau], X2).$$
$$?- vp(Z1), append(Y1, Z1, [die, frau, liest]).$$
$$?- v(X3).$$
$$?- append([die, frau], [liest], [die, frau, liest]).$$

# Context-Free Grammars

- recognition is blind search (generate-and-test):
  - generation of all sentences licensed by the grammar
  - subsequent comparison with the the input sentence
  - no consideration of intermediate results
  - processing is not guided by the input
  - termination problems for grammars with recursive rules

# Context-Free Grammar

- alternatives in the dictionary

$$s(X) \leftarrow np(Y), vp(Z), append(Y, Z, X).$$
$$np(X) \leftarrow d(Y), n(Z), append(Y, Z, X).$$
$$vp(X) \leftarrow v(X).$$

$$d([der]) \leftarrow true.$$
$$d([die]) \leftarrow true.$$
$$n([mann]) \leftarrow true.$$
$$n([frau]) \leftarrow true.$$
$$v([lacht]) \leftarrow true.$$
$$v([liest]) \leftarrow true.$$

- alternatives can also be in the grammar

# Context-Free Grammars

$?- s([die, frau, liest]).$

$?- np(Y1), vp(Z1), append(Y1, Z1, [die, frau, liest]).$

$?- d(Y2), n(Z2), append(Y2, Z2, X2).$

$?- n(Z2), append(Y2, Z2, X2).$

$?- append([der], [mann], X2).$

$?- vp(Z1), append(Y1, Z1, [die, frau, liest]).$

$?- v(X3).$

$?- append([der, mann], [lacht], [die, frau, liest]).$

$?- v(X3).$

$?- append([der, mann], [liest], [die, frau, liest]).$

$?- n(Z2), append(Y2, Z2, X2).$

$?- append([der], [frau], X2).$

$?- vp(Z1), append(Y1, Z1, [die, frau, liest]).$

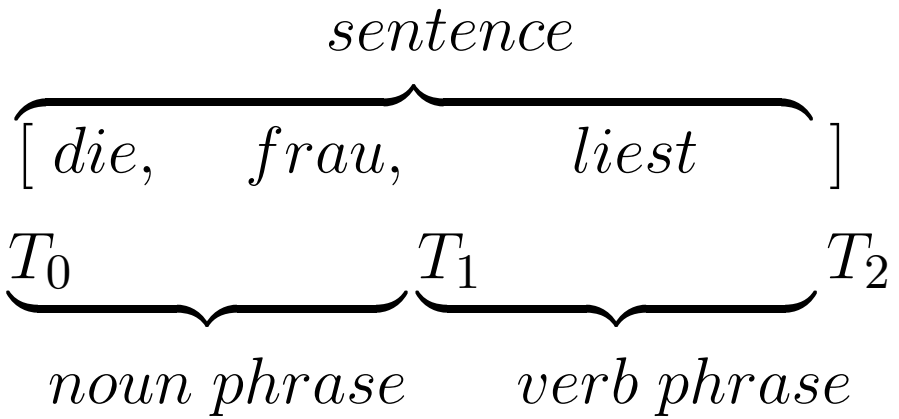
$?- v(X3).$

$?- append([der, frau], [lacht], [die, frau, liest]).$

... ...

# Difference Lists

- a (partial) list is described by two arguments
  - its beginning
  - its end point in that list
- efficient concatenation of partial lists
- partial lists can be used to guide the analysis



# Difference Lists

- Interpretation: $np(list1, list2)$
  - $list1$: list, to be analysed, for which the top-down hypothesis $np$ has to be verified
  - $list2$: remainder of the input which is not covered by the current constituent $np$

# DCG with Difference Lists

- grammar rules:

$$s(T_0, T_2) :- np(T_0, T_1), vp(T_1, T_2).$$
$$np(T_0, T_2) :- d(T_0, T_1), n(T_1, T_2).$$
$$vp(T_0, T_1) :- v(T_0, T_1).$$

- dictionary:

$$d([der|T], T) \leftarrow true.$$
$$d([die|T], T) \leftarrow true.$$
$$n([mann|T], T) \leftarrow true.$$
$$n([frau|T], T) \leftarrow true.$$
$$v([lacht|T], T) \leftarrow true.$$
$$v([liest|T], T) \leftarrow true.$$

# DCGs with Difference Lists

- recognition:

$$?- s([die, frau, liest], [\,]).$$
$$?- np([die, frau, liest], T_1^1), vp(T_1^1, [\,]).$$
$$?- d([die, frau, liest], T_1^2), n(T_1^2, T_1^1).$$
$$?- n([frau, liest], T_1^1).$$
$$?- vp([liest], [\,]).$$
$$?- v([liest], [\,]).$$

- search is guided by the input
- termination problems only for left-recursive rules

# DCGs in Prolog

- special notation which is transformed into difference list predicates when consulted

```
s --> np, vp.
np --> d, n.
vp --> v.


d --> [die].
n --> [frau].
v --> [liest].
```

# Parsing

- augmentation: construction of a structural description using an additional argument position
- most simple case: structural description reflects the structure of the rules

$$s(T_0, T_2, s(S_{np}, S_{vp})) \leftarrow np(T_0, T_1, S_{np}), vp(T_1, T_2, S_{vp}).$$
$$np(T_0, T_2, np(S_d, S_n)) \leftarrow d(T_0, T_1, S_d), n(T_1, T_2, S_n).$$
$$vp(T_0, T_2, vp(S_v, S_{np})) \leftarrow v(T_0, T_1, S_v), np(T_1, T_2, S_{np}).$$
$$vp(T_0, T_1, vp(S_v)) \leftarrow v(T_0, T_1, S_v).$$

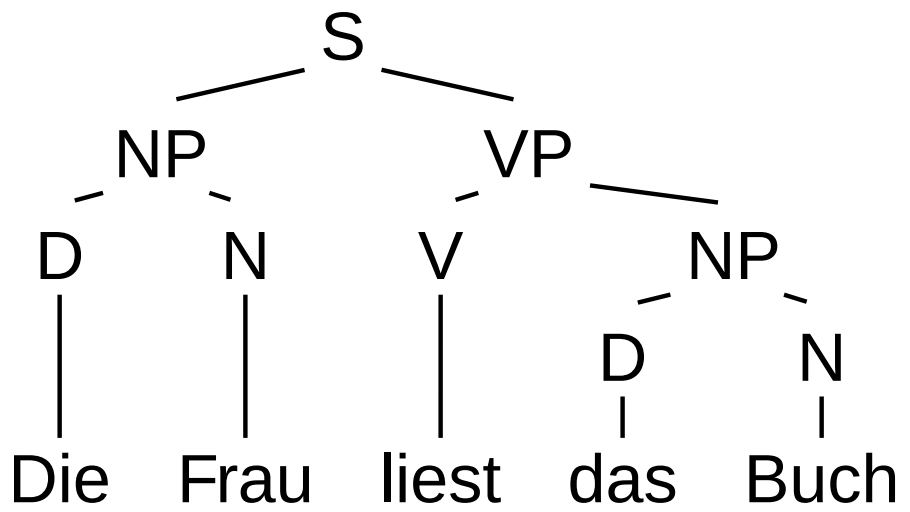
$$d([die|T], T, d(die)) \leftarrow true.$$
$$d([das|T], T, d(das)) \leftarrow true.$$
$$n([frau|T], T, n(frau)) \leftarrow true.$$
$$n([buch|T], T, d(buch)) \leftarrow true.$$
$$v([liest|T], T, v(liest)) \leftarrow true.$$

# Parsing

- invoking with
  - the input sentence
  - an empty (remaining) list, and
  - a non-instantiated variable to be bound with the structural description

$$?- s([die, frau, liest, das, buch], [\,], S_s).$$

$$S_s = s(np(d(die), n(frau)),\\ vp(v(liest),\\ np(d(das), d(buch))))$$



# Parsing

- generated structural description can also be independent from the structure of the rules
  - avoiding termination problems with left-recursive rules while keeping the left-recursive structure
  - building a semantic description

# Parsing

- generating left-recursive structures using right-recursive rules
- complex noun phrases with left recursion

$$np(np(S_d, S_n)) \leftarrow d(S_d), n(S_n).$$
$$np(np(S_{np}, S_{pp})) \leftarrow np(S_{np}), pp(S_{pp}).$$
$$pp(pp(S_p, S_{np})) \leftarrow p(S_p), np(S_{np}).$$

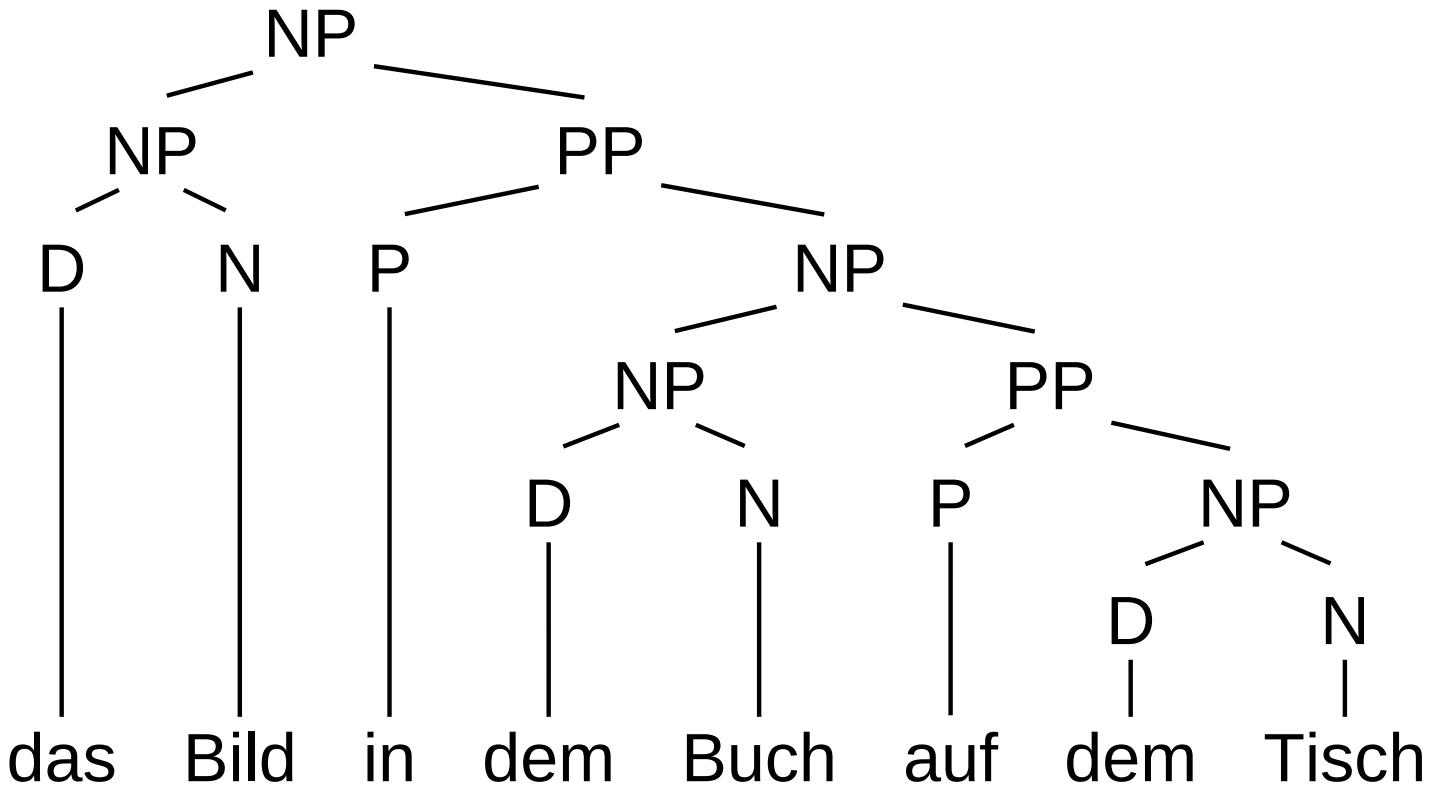
- translated into a right-recursive only one

$$np(np(S_d, S_n)) \leftarrow d(S_d), n(S_n).$$
$$np(S_{pps}) \leftarrow d(S_d), n(S_n), pps(np(S_d, S_n), S_{pps}).$$

$$pps(S_{np}, np(S_{np}, S_{pp})) \leftarrow pp(S_{pp}).$$
$$pps(S_{np}, S_{pps}) \leftarrow pp(S_{pp}), pps(np(S_{np}, S_{pp}), S_{pps}).$$

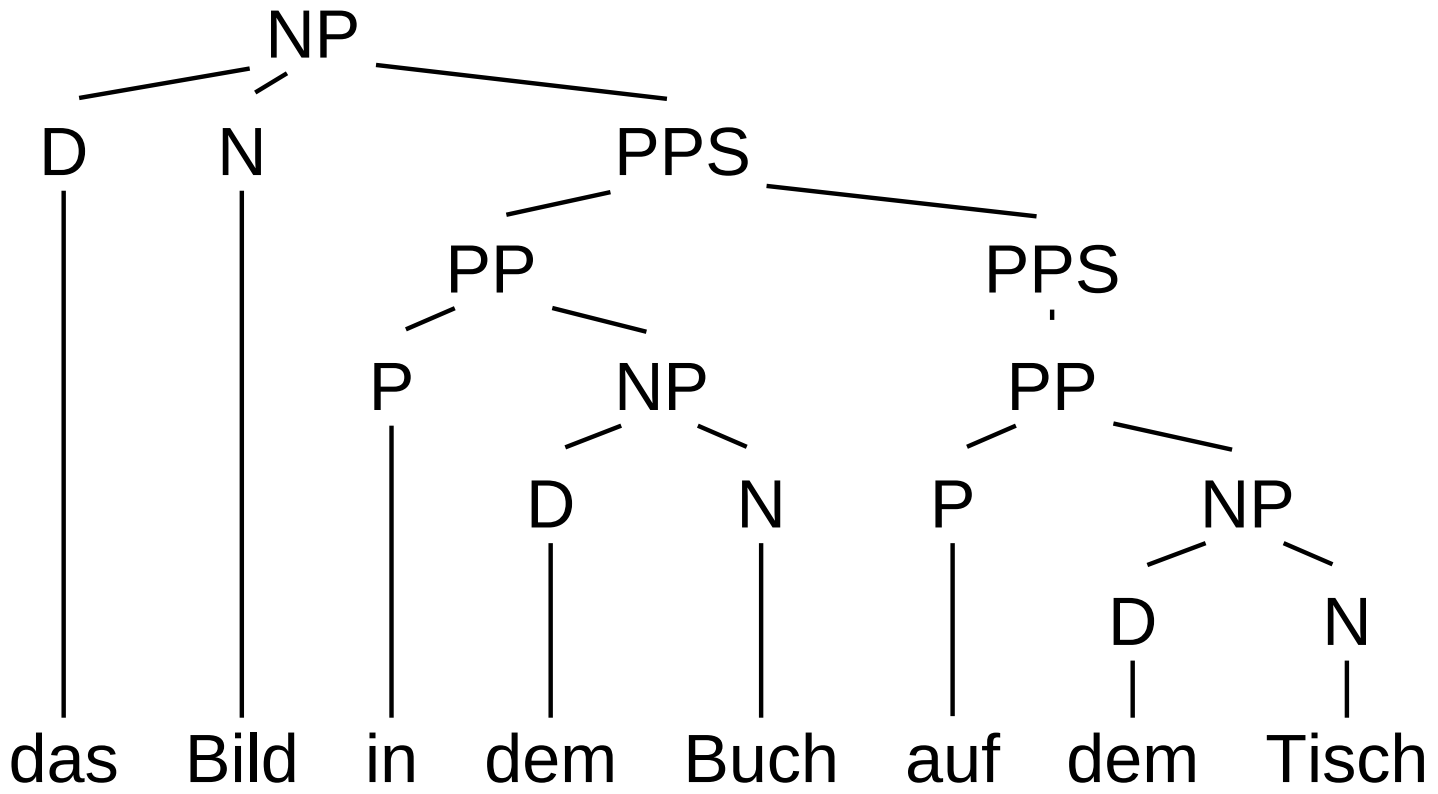
# Parsing

- left-recursive structure



# Parsing

- right-recursive derivation



# Logical Forms

- logical form: description of the propositional content (i.e. the semantic contribution) of a sentence

  e.g.: *Die Frau liest ein Buch.*

  $\exists x \exists y . frau(x) \wedge buch(y) \wedge lesen(x, y)$

- ignoring quantification for simplicity

  $frau(x) \wedge buch(y) \wedge lesen(x, y)$

- also not considered: anaphoric reference, definitness, ...
  e.g.: *Die Frau sitzt auf dem Balkon. Sie liest ein Buch.*

# Logical Forms

- constructing a semantic analysis
  - two additional argument positions for a difference list representation of the logical form
  - more additional argument positions to model referential objects described by the particular constituent

# Logical Forms

- constructing a semantic analysis

$$s(T_0, T_2, S_0, S_2) \leftarrow np(T_0, T_1, Agens, S_0, S_1), vp(T_1, T_2, Agens, S_1, S_2).$$
$$np(T_0, T_2, RefO, S_0, S_1) \leftarrow d(T_0, T_1), n(T_1, T_2, RefO, S_0, S_1).$$
$$vp(T_0, T_2, Agens, S_0, S_2) \leftarrow$$
$$v(T_0, T_1, Agens, Patiens, S_0, S_1), np(T_1, T_2, Patiens, S_1, S_2).$$

$$d([die|T], T) \leftarrow true.$$
$$d([das|T], T) \leftarrow true.$$
$$n([frau|T], T, X, S, [frau(X)|S]) \leftarrow true.$$
$$n([buch|T], T, X, S, [buch(X)|S]) \leftarrow true.$$
$$v([liest|T], T, Agens, Patiens, S, [lesen(Agens, Patiens)|S]) \leftarrow true.$$

# Semantic Forms

- generating a semantic form from a sentence

  $?- s([die, frau, liest, das, buch], [\,], [\,], S).$

  $S = [buch(Y), lesen(X, Y), frau(X)]$

- semantic form can be
  - instantiated and inserted into e data base
  - used as a query to retrieve instances from a data base

# Semantic Forms

- generating a sentence from a logical form

  $?- s(S, [\,], [\,], [buch(Y), lesen(X, Y), frau(X)]).$

  $S = [die, frau, liest, die, buch]$

  - can be used to generate sentences from inferred information
  - shortcomings:
    - generation is sensitive to the order of conjuncts!
    - grammar needs to be refined

# Augmenting the Grammar

- goal: more precise syntactic modelling
- coarse grained categories ($v, n, d$) are not sufficient
- augmentation: DCG with features (complex categories)
- example: agreement and case government:
  *ich rede/du redest/er redet/...*
  *der alte Mann/des alten Mannes/dem alten Mann/...*
  *mit ihm/durch ihn*
- projection: transferring features from the lexical to the phrase level
  - additional syntaktic features: Case, Geder, Number, Person

# Augmenting the Grammar

- additional argument positions in the dictionary

$d([der|T], T, nom, sg, mas) \leftarrow true.$
$d([der|T], T, gen, sg, fem) \leftarrow true.$
$d([der|T], T, dat, sg, fem) \leftarrow true.$
$d([der|T], T, gen, pl, X) \leftarrow true.$

$n([mann|T], T, gen, sg, mas) \leftarrow true.$
$n([mann|T], T, dat, sg, mas) \leftarrow true.$
$n([mann|T], T, acc, pl, mas) \leftarrow true.$

$v([sieht|T], T, tran, 3, sg) \leftarrow true.$

# Augmenting the Grammar

- modelling the additional well-formedness constraints
  - for agreement: as a coreference between argument positions on the righthand side of a rule
  - for case government: as an instantiated argument position
  - for feature projection: as coreference between argument positions on the right- and lefthand side of a rule

$$s(...) \leftarrow np(..., nom, P, N), vp(..., P, N).$$
$$np(..., K, 3, N) \leftarrow d(..., K, N, G), n(..., K, G, N).$$
$$vp(..., P, N) \leftarrow v(..., tran, P, N), np(..., acc, X, Y).$$

- other syntactic phenomena (subcategorization, movement, selectional restrictions, ...) can be modelled in a similar way

# Natural Language Processing

- the power of CDGs is based on the power of unification
  - simplest form of a family of unification grammars
- it is difficult if not impossible to write a robust and broad coverage unification grammar for unrestricted text
  - application of approximate solution methods, e.g. probabilistic approaches
  - attempting approximate results, e.g. shallow instead of deep understanding
- natural language grammar is inherently inconsistent
  - provide for preferential reasoning and conflict arbitration