Parsing Natural Language

Introduction to Parsing

Parsing Natural Language

- The term „parsing“ is deeply embedded in theoretical computer science

- Definition:

  In laymans terms, we are given a grammar G and an input word x and:

  1. We verify if x is part of the language of G (we will call this algorithm **recognizer**)
  2. We derive the ***sequence of operations***, necessary to generate x from G (**parsing**)

➔ Theoretical computer science provides the algorithms

Parsing Natural Language

- From the perspective of a linguist, the grammar G is primarily what is in focus

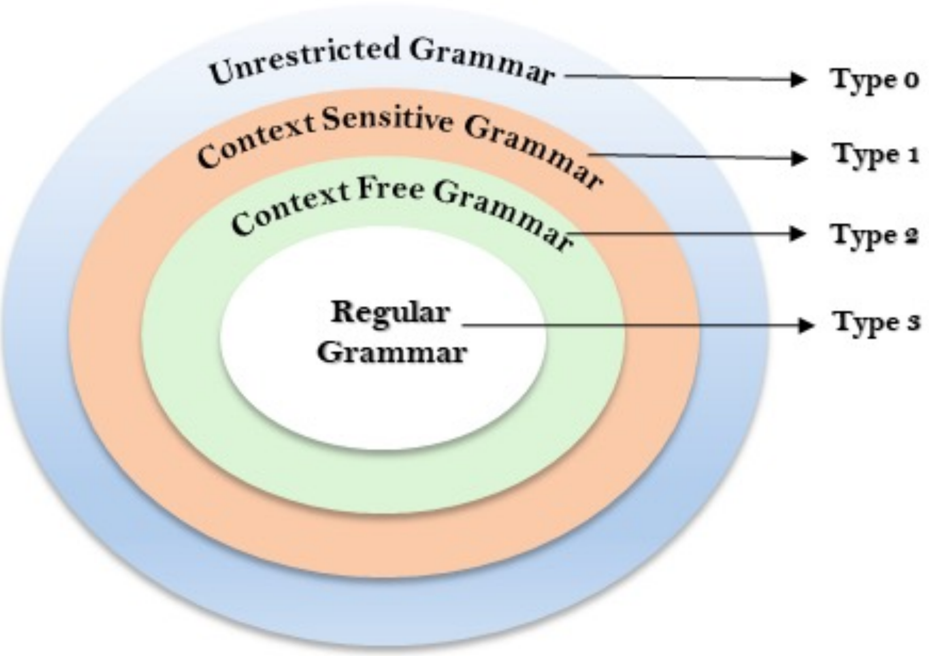
"How do words group together in English"

- The grammar contains all sort of constraints:
  - Morphological constraints between subject, and verb
  - Morphological constraints in the same ***phrase***
  - …

➔ Linguistics provides (sadly many different) theories about the grammar G

More about the grammar

- In theoretical computer science, grammars can be grouped in different types:

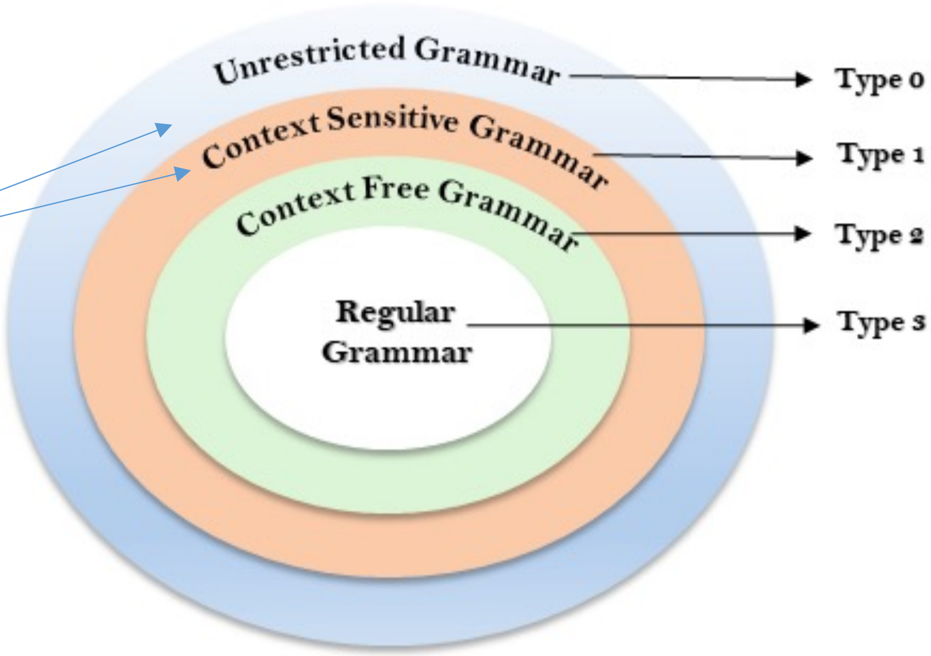


https://www.javatpoint.com/automata-chomsky-hierarchy

More about the grammar

- In theoretical computer science, grammars can be grouped in different types:

No efficient parsing algorithms available!

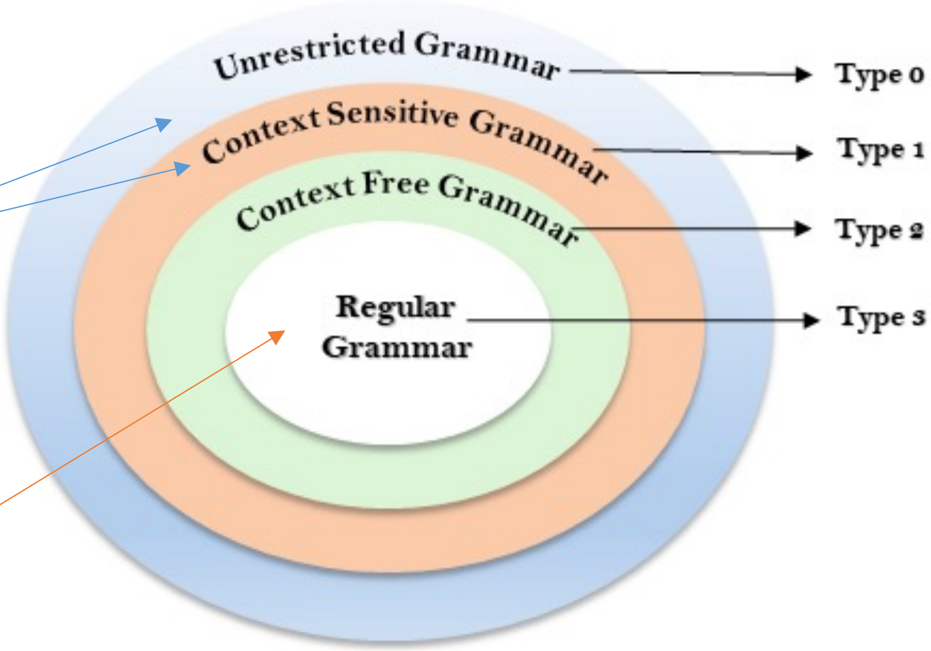


https://www.javatpoint.com/automata-chomsky-hierarchy

More about the grammar

- In theoretical computer science, grammars can be grouped in different types:

No efficient parsing algorithms available!



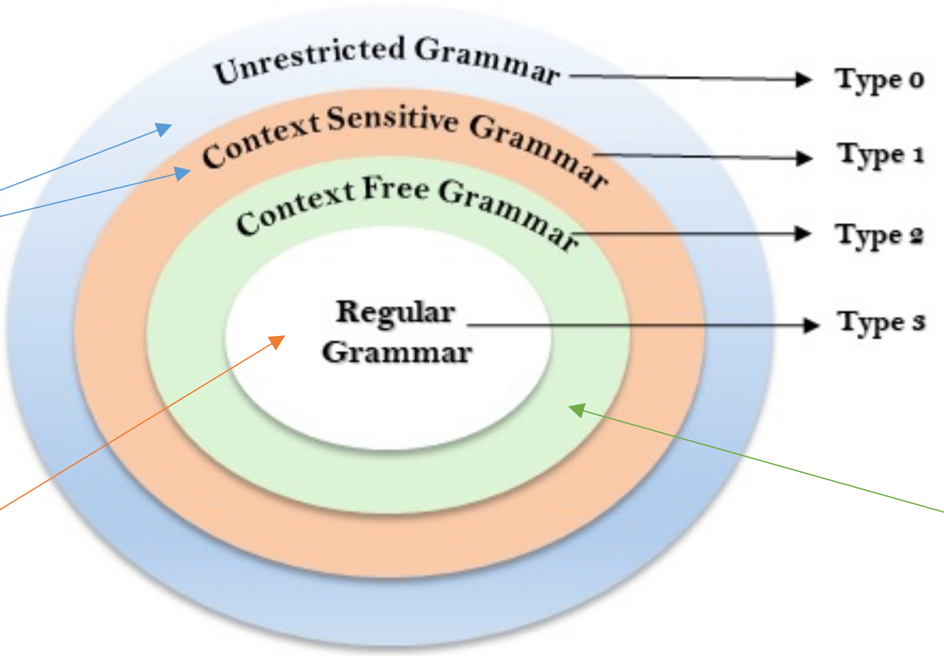
Regular Expressions

https://www.javatpoint.com/automata-chomsky-hierarchy

More about the grammar

- In theoretical computer science, grammars can be grouped in different types:

No efficient parsing algorithms available!



Regular Expressions

Context-Free Grammars, where polynomial algorithms exist!

https://www.javatpoint.com/automata-chomsky-hierarchy

More about the grammar

- So from theoretical computer science, we are bound to work with **Context-Free Grammars**

- This means, all linguistic knowledge we gathered has to be modelled inside this framework, which is not only challenging, but:

  - Also might yield insanely large grammars (since not everything can be expressed in an elegant fashion)

Context Free Grammar

- Definition:

A Context Free Grammar (CFG) is a 4-Tuple $G = (N, \Sigma, R, S)$ having:

N : **finite** set of non-terminal symbols

Σ : **finite** set of terminal symbols

R : a finite set of rules of the form:

$X \rightarrow Y_1 Y_2 \ldots Y_n$ with $X \in N, n \geq 0, and\ Y_i \in (N \cup \Sigma)$

$S \in N$, being a special start symbol

Context Free Grammar - Example

N = {S, NP, VP, PP, DT, Vi, Vt, NN, IN}
S = S
Σ = {sleeps, saw, man, woman, dog, telescope, the, with, in}

R =

S	$\rightarrow$	NP	VP
VP	$\rightarrow$	Vi	
VP	$\rightarrow$	Vt	NP
VP	$\rightarrow$	VP	PP
NP	$\rightarrow$	DT	NN
NP	$\rightarrow$	NP	PP
PP	$\rightarrow$	IN	NP

Vi	$\rightarrow$	sleeps
Vt	$\rightarrow$	saw
NN	$\rightarrow$	man
NN	$\rightarrow$	woman
NN	$\rightarrow$	telescope
NN	$\rightarrow$	dog
DT	$\rightarrow$	the
IN	$\rightarrow$	with
IN	$\rightarrow$	in

Ignore the meaning behind NP, PP for now …

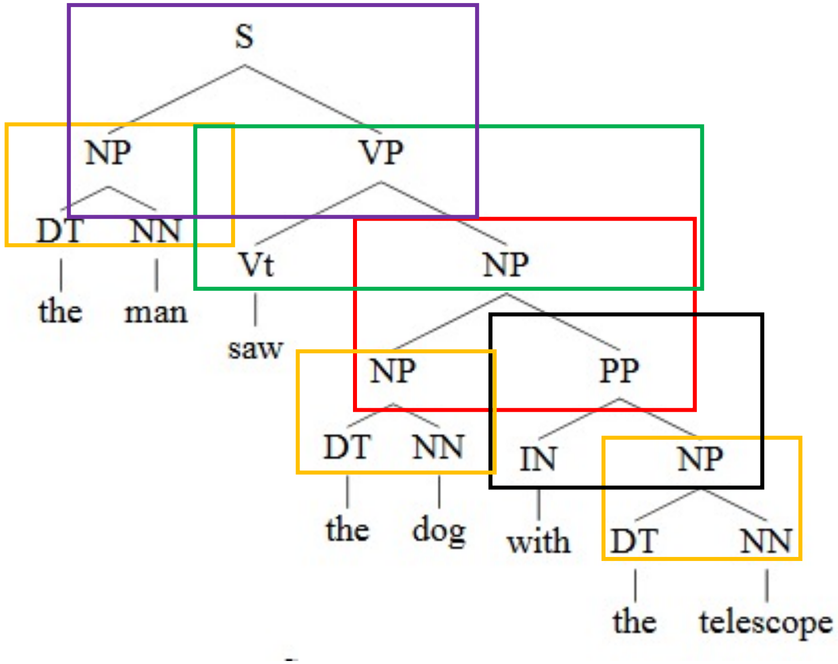
Context Free Grammar - Example

- Parsing using this grammar would produce the following **tree**!

$S \rightarrow NP\ VP$

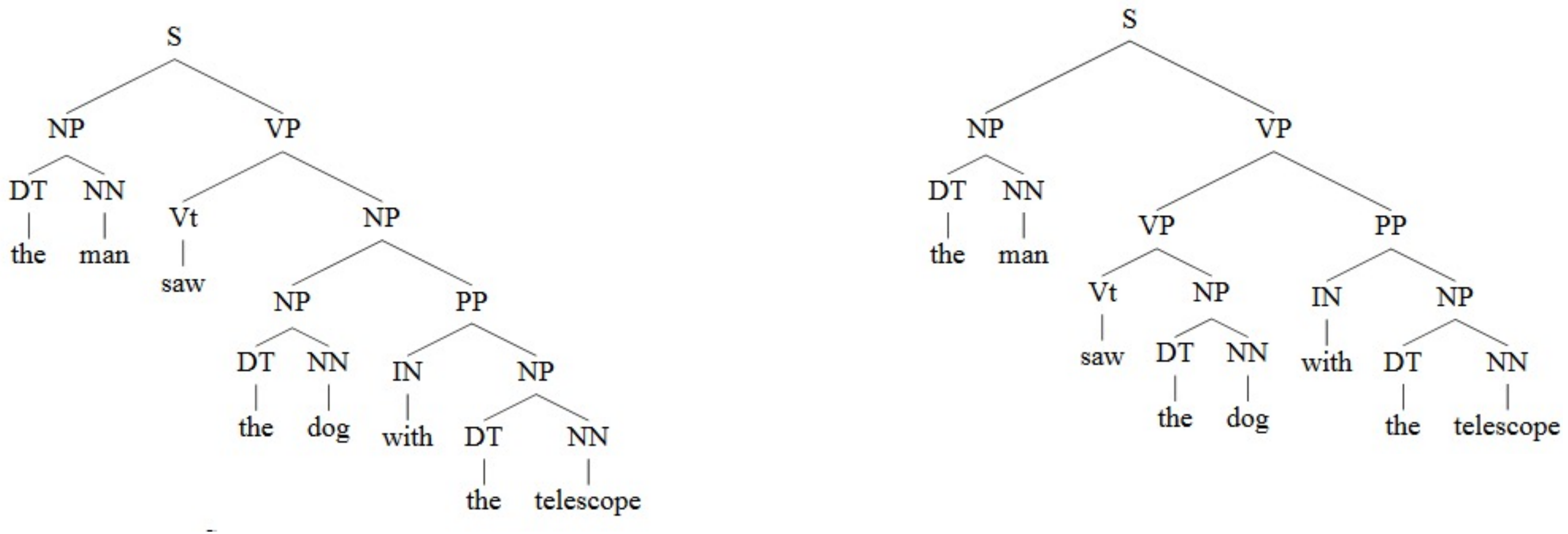
$NP \rightarrow DT\ NN$
$VP \rightarrow Vt\ NP$

$NP \rightarrow NP\ PP$
$PP \rightarrow IN\ NP$



Context Free Grammar - Example

- Problem: Grammars usually tend to allow more than just a single parse!



Context Free Grammar

- Where can we get a grammar for our language?
- Option 1: Intuition:
  - We obtain it from our experts (linguists)
  - We model what we can find in books dedicated to teach grammar

Context Free Grammar

- Option 2: We are going to extract them from a data set, which is labelled by experts (usually called a „Treebank“):

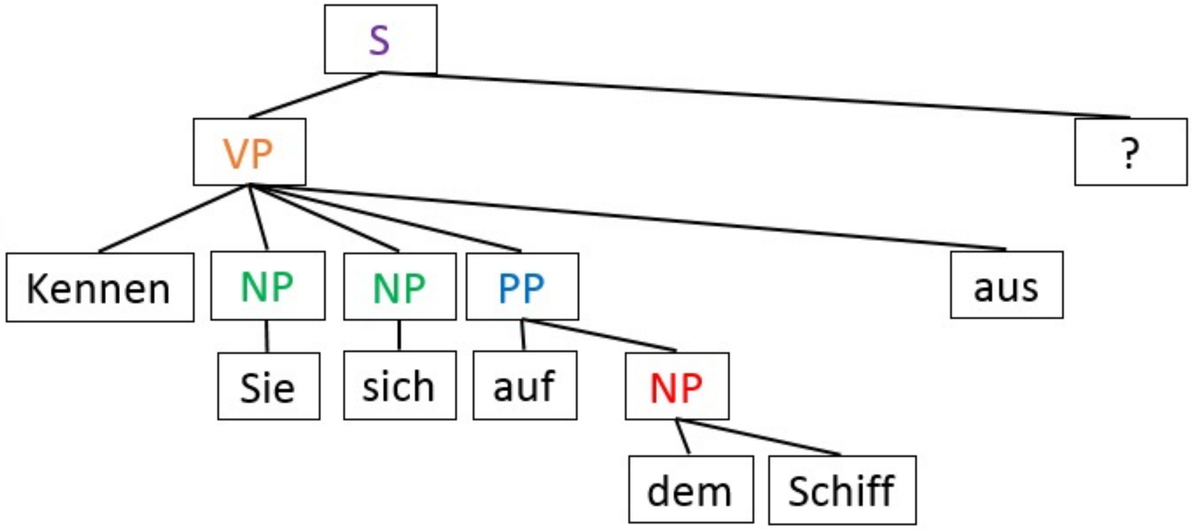
Is it harder to parse Chinese, or the Chinese Treebank?

Roger Levy
Department of Linguistics
Stanford University
rog@stanford.edu

Christopher Manning
Department of Computer Science
Stanford University
manning@cs.stanford.edu

Context Free Grammar

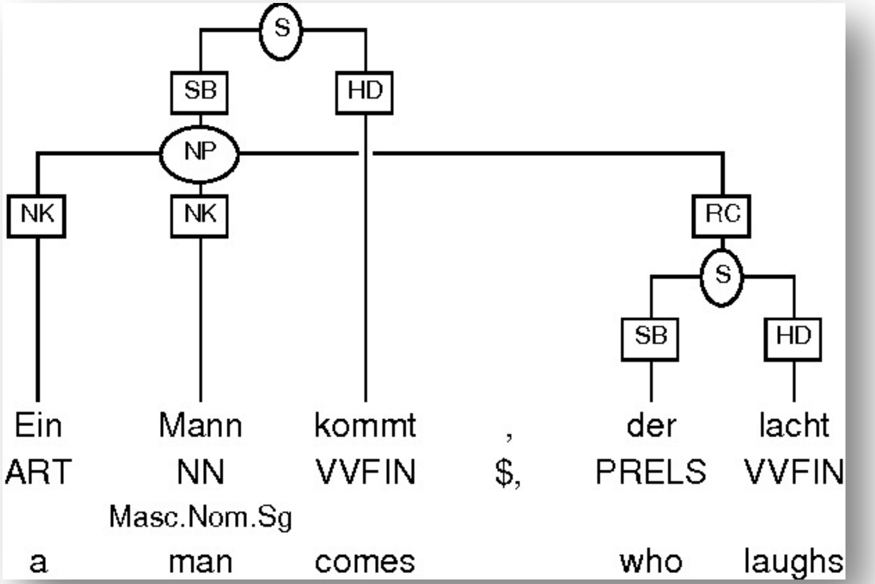
- Example: Extracting the rules from a labelled sentence:
- $\mathbf{S} \rightarrow \mathbf{VP}\ ?$
- $\mathbf{VP} \rightarrow Kennen\ \mathbf{NP}\ \mathbf{NP}\ \mathbf{PP}\ aus$
- $\mathbf{NP} \rightarrow Sie$
- $\mathbf{NP} \rightarrow sich$
- $\mathbf{PP} \rightarrow auf\ \mathbf{NP}$
- $\mathbf{NP} \rightarrow dem\ Schiff$



Context Free Grammar

- We can now apply this procedure to an entire treebank

- E.g. for the German TIGER Treebank (about. 50.000 sentences)
  ➔ results in 30.000 different rules! (excluding terminals!)



Parsing- Valid trees

- In general, in order to get all valid trees (according to our grammar) we can apply one of the following strategies:
  1. Top-Down Search:
     - Start with the symbol S,
     - Apply all rules in R , which start with $S \rightarrow$ ...
     - Continuously expand the resulting symbols in a similar fashion

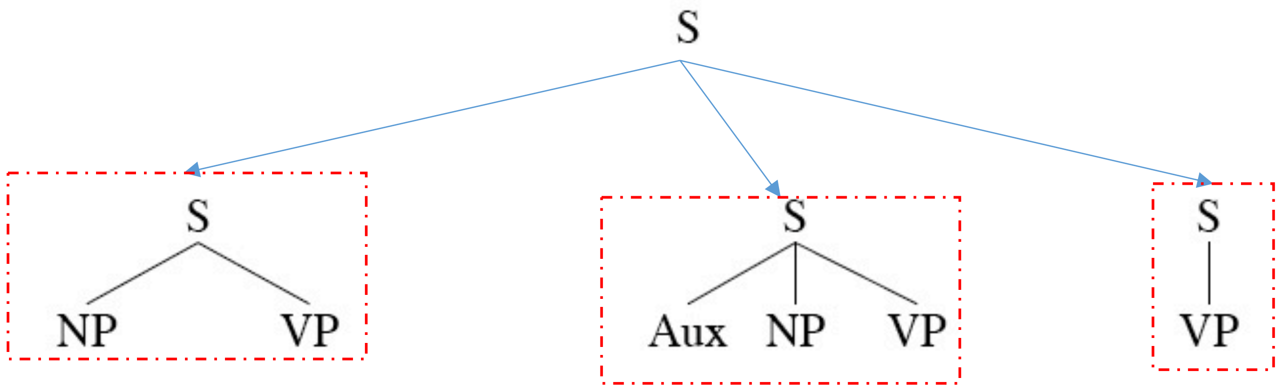
     ➔ Once we produced the sentences we can stop and return the operations
  2. Bottom-Up Search:
     - Start with the tokens,
     - Apply all rules, which produce the tokens
     - Repeat this step, until an S non-terminal was produced

     ➔ Return the tree

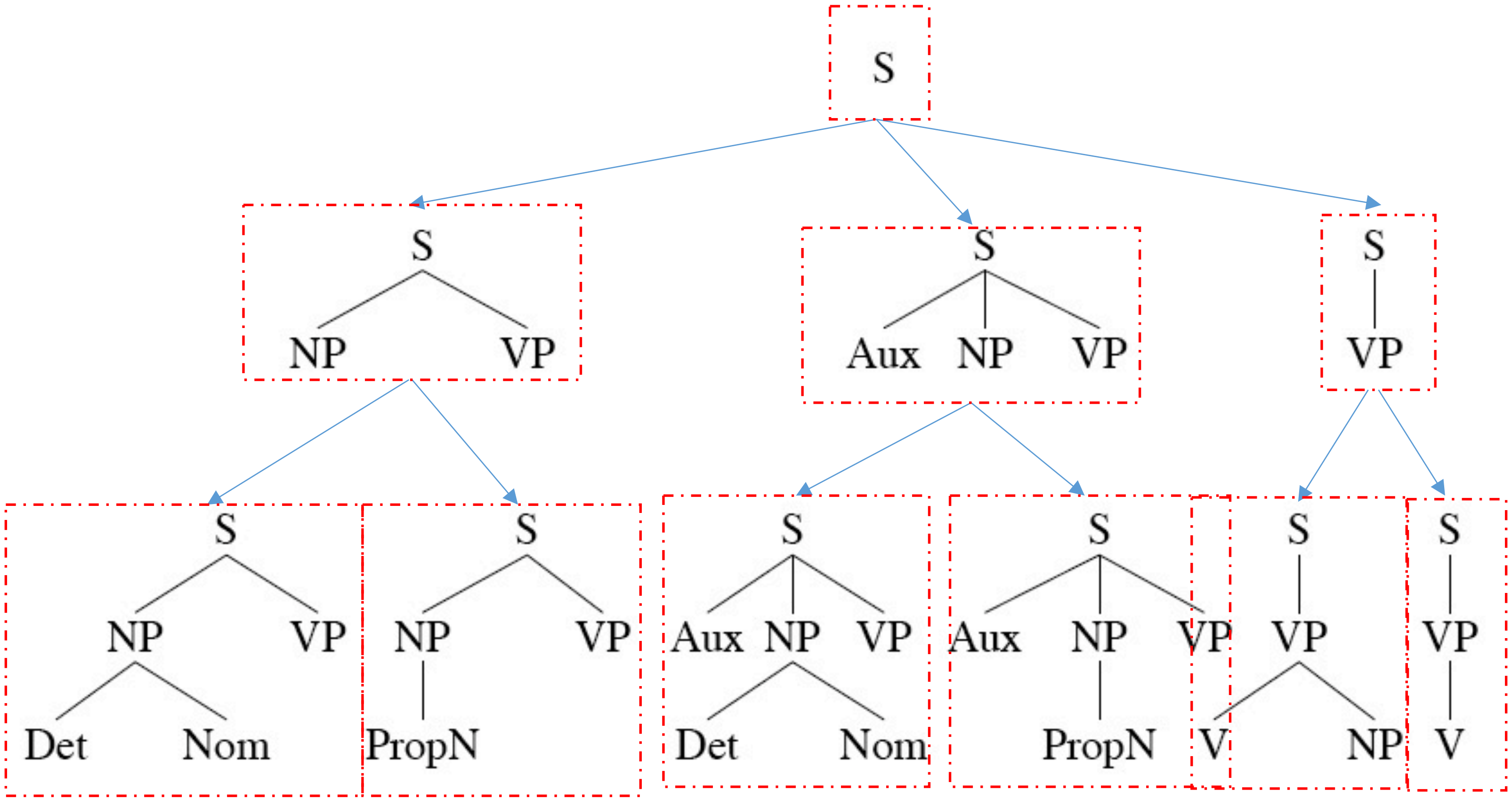
Parsing- Top Down Search

S

Parsing- Top Down Search



Parsing- Top Down Search



Top-Down and Bottom-Up

- Top-down
  - Only searches for trees that can be answers (i.e. S’s)
  - But also suggests trees that are not consistent with any of the words

  ➔ Solved efficiently: **Earley-Algorithm**
- Bottom-up
  - Only forms trees consistent with the words
  - But suggests trees that make no sense globally

  ➔ Solved efficiently: **CYK-Algorithm (also called CKY)**

Parsing Computer Languages

- Wait I know computers and I have never heard of “Earley” or “CKY”

- The current Python parser (< version 3.9) is a LL(1)-parser
- And it is going to be replaced by a parser based on PEG
- Maybe you have heard of LR(k)-parsers
- Or the ALL(*)-parser of ANTLR

➔ Why do computer languages not use Earley or CKY?

Diving into the language

- We made use of cryptic symbols, such as NP, PP, etc…
- We are now going to explain what they are actually representing!

- For this, we need to leave the view on parsing from a computer scientist again and have to dive how linguists model our language
  - And obviously we will find more than just a single theory! We focus on:

1. Constituency Grammar
2. Dependency Grammar

Constituency

- The idea is that „*groups of words behave as a single unit*“

- The most general idea is to divide a sentence into **phrases**

NP

1. Noun phrases: „everything that revolves around a noun“

   [The beautiful pieces of art] shown in …

Constituency

- The idea is that „*groups of words behave as a single unit*“

- The most general idea is to divide a sentence into **phrases**

NP

1. Noun phrases: „everything that revolves around a noun“

[The beautiful pieces of art] shown in …

We call this the „head“ of the phrase

Constituency

NP

- Noun phrases: „everything that revolves around a noun“

[The beautiful pieces of art] shown in …

- A NP can contain different NP‘s (among other stuff)

[The beautiful pieces of [art]] shown in …

Constituency

PP

- Prepositional phrases: „everything that revolves around a preposition“

The beautiful pieces **[of art]** shown in …

We call this the „head“ of the phrase

- A PP usually contains an NP

Constituency

VP

- Verb phrases: „everything that revolves around a verb“

  (usually the subject is excluded)

The beautiful pieces of art **[shown in the gallery of the Louvre]**.

We call this the „head“ of the phrase

- A VP usually contains a verb and other NPs or PPs (or VPs)

Constituency

- This covers the three most basic structures, obviously there are many more

- But it suffices to understand our toy grammar, since the remaining symbols are just POS-Tags of the terminal words!

S	→	NP	VP
VP	→	Vi	
VP	→	Vt	NP
VP	→	VP	PP
NP	→	DT	NN
NP	→	NP	PP
PP	→	IN	NP

Dependency

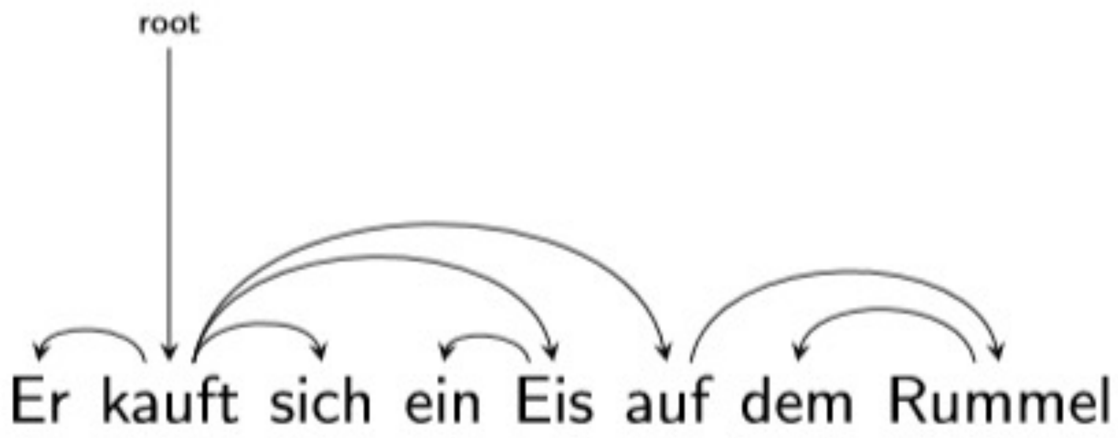
- A related but different approach is the approach of „Dependencies“

Lucien Tesnière:

The sentence is an organized whole, the constituent elements of which are words.
Every word that belongs to a sentence ceases by itself to be isolated as in the dictionary. Between the word and its neighbors, the mind perceives connections, the totality of which forms the structure of the sentence. The structural connections establish dependency relations between the words. Each connection in principle unites a superior term and an inferior term. The superior term receives the name governor. The inferior term receives the name subordinate. Thus, in the sentence "Alfred parle", "parle" is the governor and "Alfred" the subordinate.

Dependency Parsing?- Easy explanation

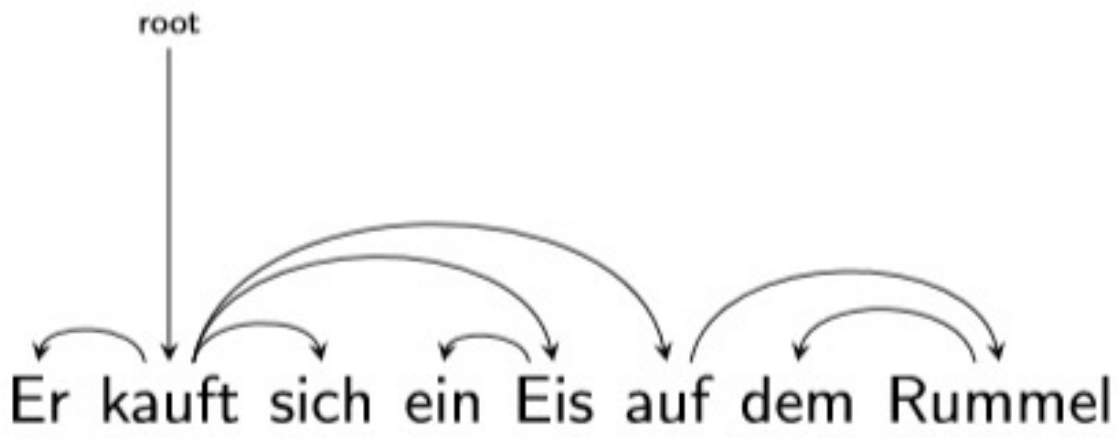
- A sentence consists of tokens, and
  - Every token in the sentence depends on exactly one other token.
  - For this to be possible, we introduce a dummy token „root“



He buys an ice cream at the carnival

Dependency Parsing- Terminology

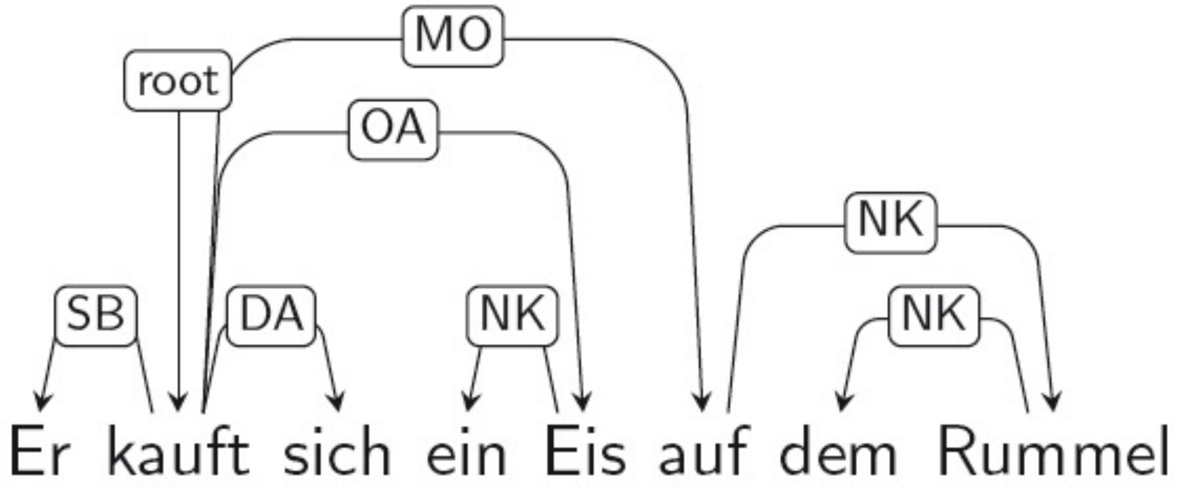
- The token from which the edge starts is denoted „Head|Governor|Regent“ and the token where the edge ends as „Dependent|Modifier|Sub“



He buys an ice cream at the carnival

Dependency Parsing- Terminology

- The relation might be be labelled, the labels are called „DependencyRelations“



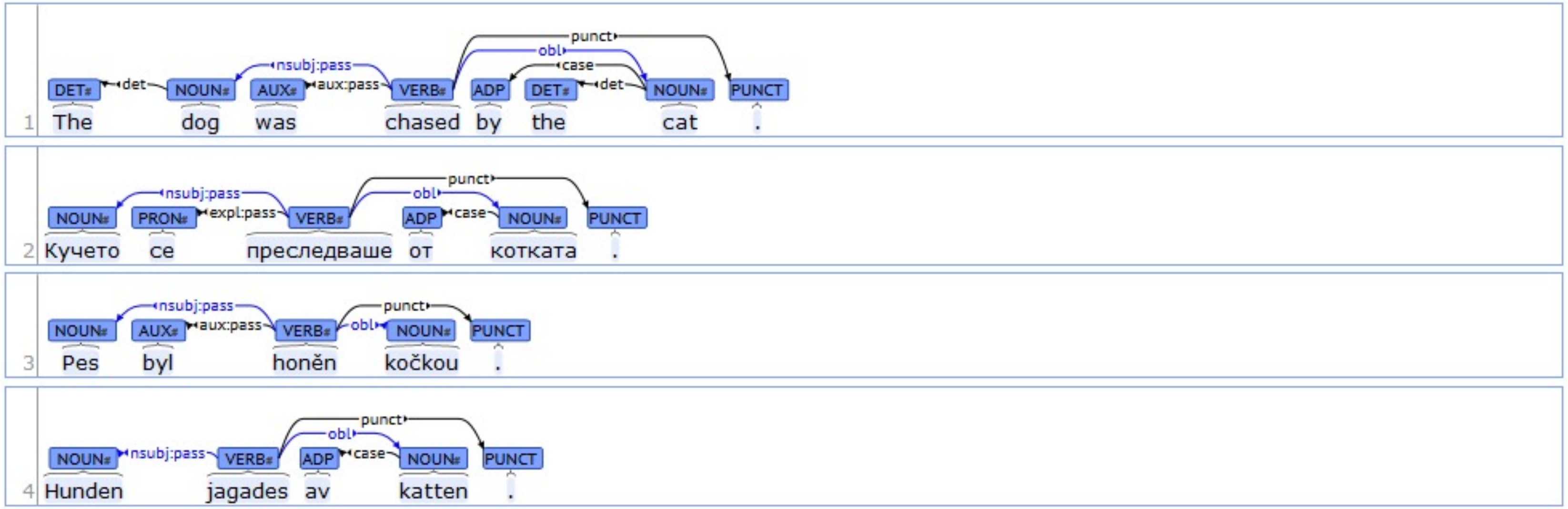
Dependency Parsing- Terminology

- The set of relations is fixed (per schema) for German you usually use the set of the NEGRA project:

AC adpositional case marker
ADC adjective component
AMS measure argument of adj
APP apposition
AVC adverbial phrase component
CC comparative complement
CD coordinating conjunction
CJ conjunct
CM comparative concjunction
CP complementizer
DA dative
DH discourse-level head
DM discourse marker
GL prenominal genitive
GR postnominal genitive
HD head
JU junctor
MC comitative
MI instrumental
ML locative
MNR postnominal modifier
MO modifier
MR rhetorical modifier
MW way (directional modifier)
NG negation
NK noun kernel modifier
NMC numerical component
OA accusative object
OA2 second accusative object
OC clausal object
OG genitive object
PD predicate
PG pseudo-genitive
PH placeholder
PM morphological particle
PNC proper noun component
RC relative clause
RE repeated element
RS reported speech
SB subject
SBP passivised subject (PP)
SP subject or predicate
SVP separable verb prefix
UC (idiosyncratic) unit component
VO vocative

Universal Dependencies

- The projects aims to have a tagset, which is applicable to many languages
- Already available for over 60 languages!
- http://universaldependencies.org/



Dependency Grammars

- Ok nice, every token is related to another one, but how can we formulate this into a grammar?

Dependency Grammars

- Ok nice, every token is related to another one, but how can we formulate this into a grammar?

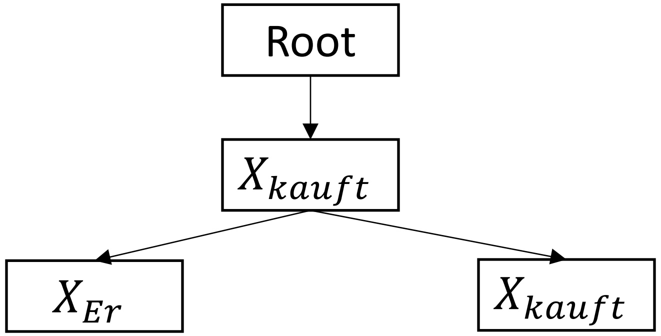
$$\begin{tikzcd} \boxed{\text{Root}} \arrow[d] \\ \boxed{X_{kauft}} \end{tikzcd}$$

Grammar:

Root $\rightarrow X_{kauft}$

Dependency Grammars

- Ok nice, every token is related to another one, but how can we formulate this into a grammar?



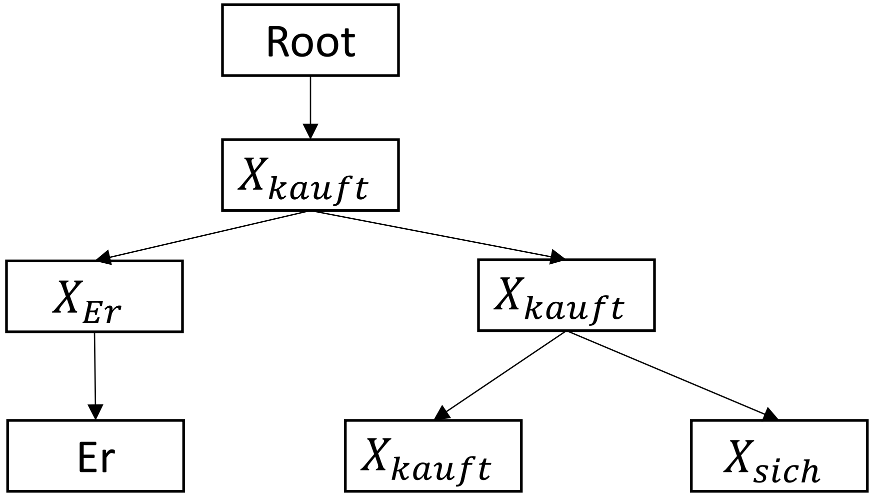
Grammar:

$\text{Root} \rightarrow X_{kauft}$

$X_{kauft} \rightarrow X_{Er} \quad X_{kauft}$

Dependency Grammars

- Ok nice, every token is related to another one, but how can we formulate this into a grammar?



Grammar:

$\text{Root} \rightarrow X_{kauft}$

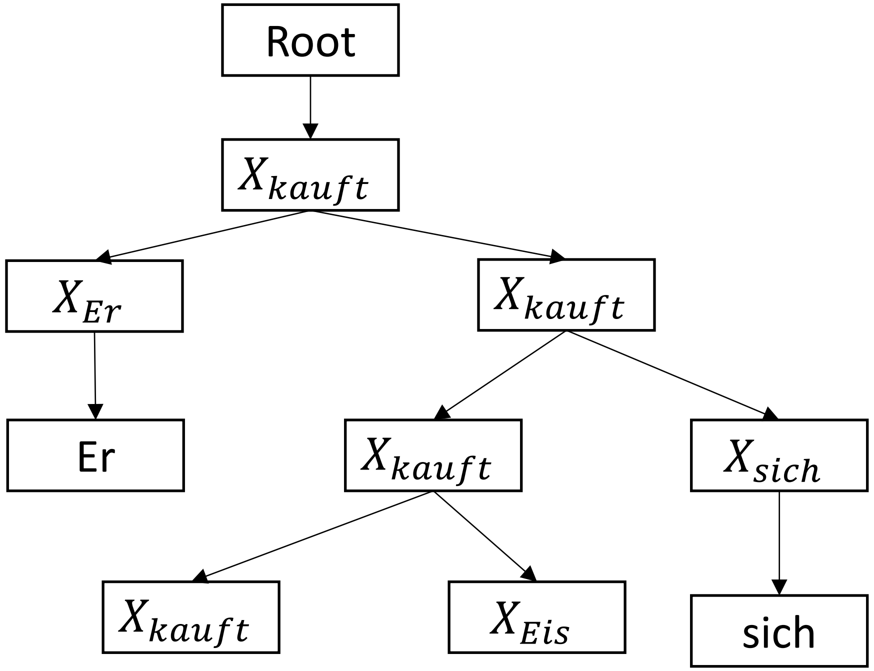
$X_{kauft} \rightarrow X_{Er} \quad X_{kauft}$

$X_{Er} \rightarrow Er$

$X_{kauft} \rightarrow X_{kauft} \quad X_{sich}$

Dependency Grammars

- Ok nice, every token is related to another one, but how can we formulate this into a grammar?



Grammar:

$\text{Root} \rightarrow X_{kauft}$

$X_{kauft} \rightarrow X_{Er} \quad X_{kauft}$

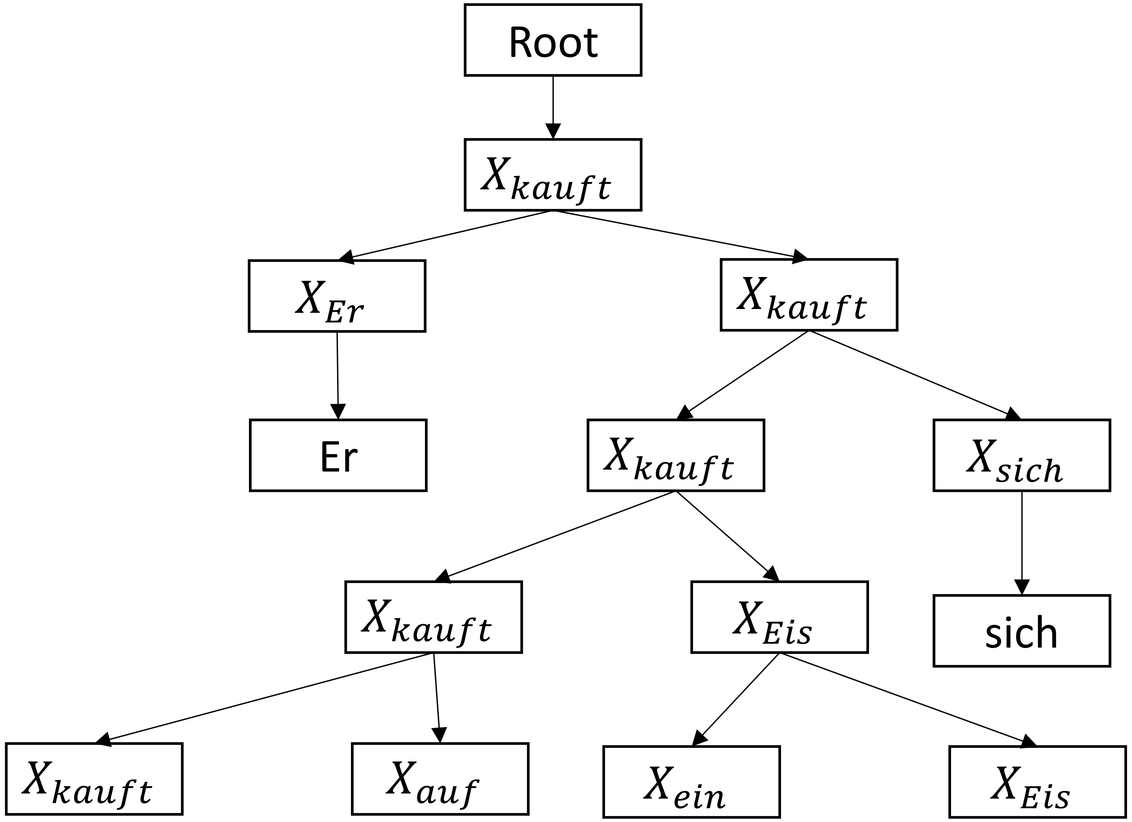
$X_{Er} \rightarrow Er$

$X_{kauft} \rightarrow X_{kauft} \quad X_{sich}$

$X_{sich} \rightarrow \text{sich}$

$X_{kauft} \rightarrow X_{kauft} \quad X_{Eis}$

Dependency Grammars



Grammar:

Root $\rightarrow X_{kauft}$

$X_{kauft} \rightarrow X_{Er} \quad X_{kauft}$

$X_{Er} \rightarrow Er$

$X_{kauft} \rightarrow X_{kauft} \quad X_{sich}$

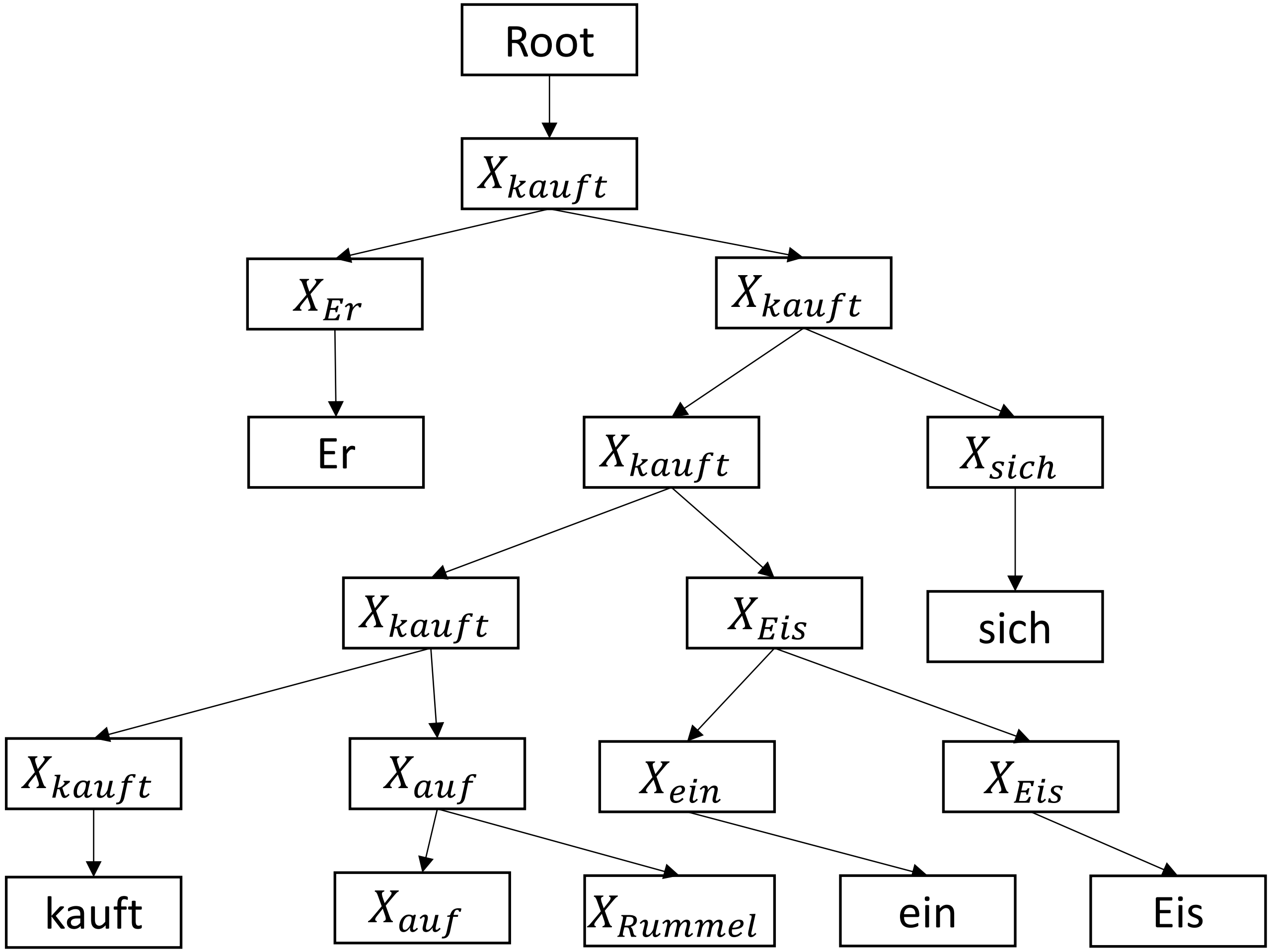
$X_{sich} \rightarrow$ sich

$X_{kauft} \rightarrow X_{kauft} \quad X_{Eis}$

$X_{Eis} \rightarrow X_{ein} \quad X_{Eis}$

$X_{kauft} \rightarrow X_{kauft} \quad X_{auf}$

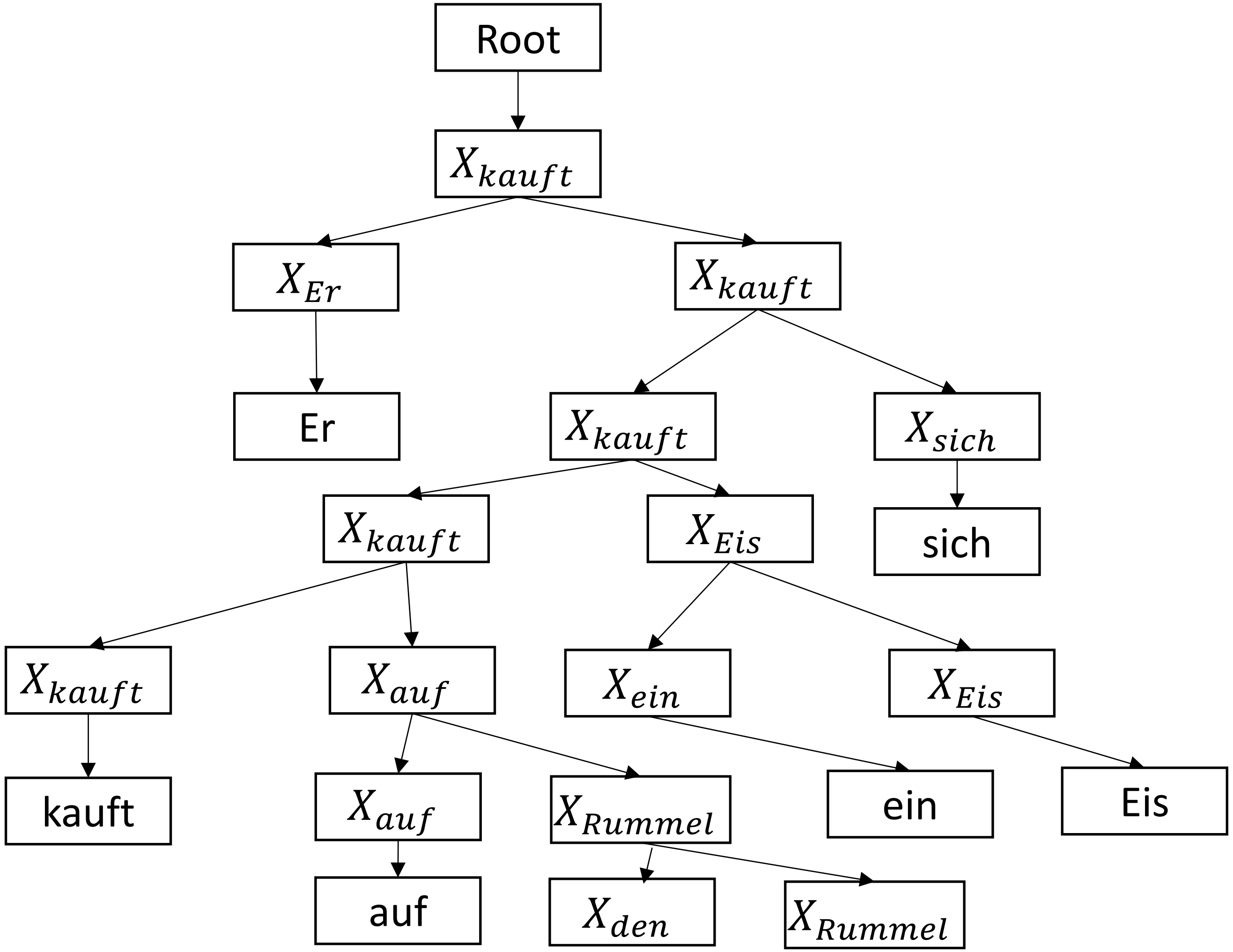
Dependency Grammars



Grammar:

Root $\rightarrow X_{kauft}$
$X_{kauft} \rightarrow X_{Er} \quad X_{kauft}$
$X_{Er} \rightarrow Er$
$X_{kauft} \rightarrow X_{kauft} \quad X_{sich}$
$X_{sich} \rightarrow$ sich
$X_{kauft} \rightarrow X_{kauft} \quad X_{Eis}$
$X_{Eis} \rightarrow X_{ein} \quad X_{Eis}$
$X_{kauft} \rightarrow X_{kauft} \quad X_{auf}$
$X_{kauft} \rightarrow$ kauft
$X_{auf} \rightarrow X_{auf} \quad X_{Rummel}$

Dependency Grammars



Grammar:

Root $\rightarrow X_{kauft}$

$X_{kauft} \rightarrow X_{Er} \quad X_{kauft}$

$X_{Er} \rightarrow Er$

$X_{kauft} \rightarrow X_{kauft} \quad X_{sich}$

$X_{sich} \rightarrow$ sich

$X_{kauft} \rightarrow X_{kauft} \quad X_{Eis}$

$X_{Eis} \rightarrow X_{ein} \quad X_{Eis}$

$X_{kauft} \rightarrow X_{kauft} \quad X_{auf}$

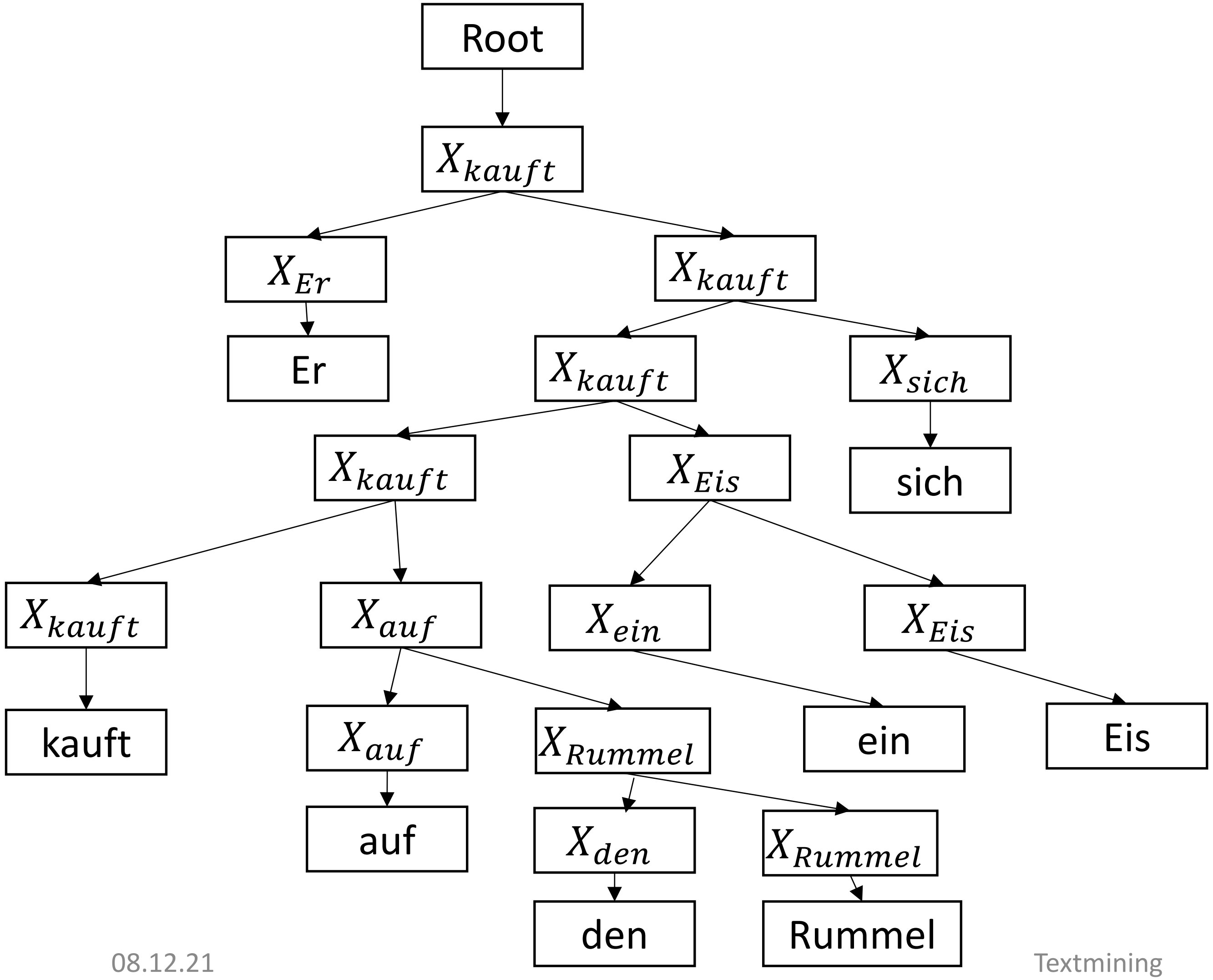
$X_{kauft} \rightarrow$ kauft

$X_{auf} \rightarrow X_{auf} \quad X_{Rummel}$

$X_{auf} \rightarrow$ auf

$X_{Rummel} \rightarrow X_{den} \quad X_{Rummel}$

Dependency Grammars



Grammar:

$\text{Root} \rightarrow X_{kauft}$
$X_{kauft} \rightarrow X_{Er} \quad X_{kauft}$
$X_{Er} \rightarrow Er$
$X_{kauft} \rightarrow X_{kauft} \quad X_{sich}$
$X_{sich} \rightarrow \text{sich}$
$X_{kauft} \rightarrow X_{kauft} \quad X_{Eis}$
$X_{Eis} \rightarrow X_{ein} \quad X_{Eis}$
$X_{kauft} \rightarrow X_{kauft} \quad X_{auf}$
$X_{kauft} \rightarrow \text{kauft}$
$X_{auf} \rightarrow X_{auf} \quad X_{Rummel}$
$X_{auf} \rightarrow \text{auf}$
$X_{Rummel} \rightarrow X_{den} \quad X_{Rummel}$
$X_{den} \rightarrow \text{den}$
$X_{Rummel} \rightarrow \text{Rummel}$

Dependency Grammars

- This particular form of grammar is called „Bilexical Grammar"
- All rules in the form (H: Head, NH: Non-Head):
  - $H \rightarrow NH\ H$ (if left arc)
  - $H \rightarrow H\ NH$ (if right arc)
- The presented tree is not unique under this grammar!
- There are dedicated parsing algorithms for this kind of grammar (Collins and Eisner algorithm)

Grammar:

Root $\rightarrow X_{kauft}$
$X_{kauft} \rightarrow X_{Er}\ \ X_{kauft}$
$X_{Er} \rightarrow Er$
$X_{kauft} \rightarrow X_{kauft}\ \ X_{sich}$
$X_{sich} \rightarrow$ sich
$X_{kauft} \rightarrow X_{kauft}\ \ X_{Eis}$
$X_{Eis} \rightarrow X_{ein}\ \ X_{Eis}$
$X_{kauft} \rightarrow X_{kauft}\ \ X_{auf}$
$X_{kauft} \rightarrow$ kauft
$X_{auf} \rightarrow X_{auf}\ \ X_{Rummel}$
$X_{auf} \rightarrow$ auf
$X_{Rummel} \rightarrow X_{den}\ \ X_{Rummel}$
$X_{den} \rightarrow$ den
$X_{Rummel} \rightarrow$ Rummel

Recap

- We introduced parsing of natural language, where the computer scientist has the algorithms and the linguist a grammar
- A parser is an algorithm, which produces a tree of lexical symbols, which depicts the inner structure of the sentence
- We introduced the two most common formalisms for a natural language grammar:
  - Constituency
  - Dependency
- Which can all be modelled in the framework of Context-Free-Grammars!
- We can now proceed and show the parsing algorithms for general Context-Free-Grammars!