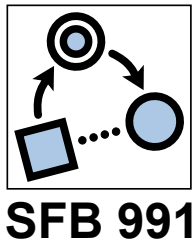
Language modeling with tree-adjoining grammars

Day0

Kata Balogh & Simon Petitjean

University of Düsseldorf

NASSLLI 2018



Formal complexity of natural languages

- computational complexity
- structural complexity

Structural complexity

- Natural languages are modeled as abstract symbol systems with construction rules.
- Questions about the grammaticality of natural sentences correspond to questions about the syntactic correctness of programs or about the well-formedness of logic expressions.

What a grammar theory has to explain:
The number of grammatical sentences is small compared to all possible word sequences.

How complex are English sentences?

(1) Anne sees Peter.

Anne sees Peter in the garden with the binoculars.

Anne who dances sees Peter whom she met yesterday in the garden with the binoculars.

Anne sees Peter and Hans and Sabine and Joachim and Elfriede and Johanna and Maria and Jochen and Thomas and Andrea.

The length of a sentence influences the processing complexity, but it is not a sign of structural complexity!

Natural Language Theories vs. Formal Language Theory

Natural Language Theories

- grammar theories
- explain natural language data
- are language specific (Latvian, German, ...)

Formal Language Theory

- a theory about the structure of symbol strings
- not language specific
- allows statements about the mechanisms for generating and recognizing sets of symbol strings

Natural Languages and Formal Languages

- Generative Grammar (linguistics): from a finite number of words + finite number of rules $\longrightarrow$ infinite number of sentences
- Standard (GG) Assumptions: (about any natural language)
  - The length of any sentence is finite. (whether letters, phonemes, morphemes, or words)
  - There is no longest sentence. (because of recursion)
- from these two assumptions it follows that the cardinality of the set of sentences in any natural language is infinite

Natural Languages and Formal Languages

- modeling any natural language as a set of strings (made of words, morphemes etc.)
- the set of possible strings formed from a vocabulary can be grammatical or ungrammatical
- language: the set of all grammatical strings
- grammar: determines the set of all grammatical strings

Grammars

Grammar

- A formal grammar is a **generating device** which can generate (and analyze) strings/words.
- Grammars are finite rule systems.
- The set of all strings generated by a grammar is a formal language (= generated language).

- Example grammar:

  $S \rightarrow NP\ VP,\ VP \rightarrow V,\ NP \rightarrow DET\ N,\ NP \rightarrow PN,$
  $DET \rightarrow the,\ N \rightarrow cat,\ V \rightarrow sleeps,\ PN \rightarrow Mia$

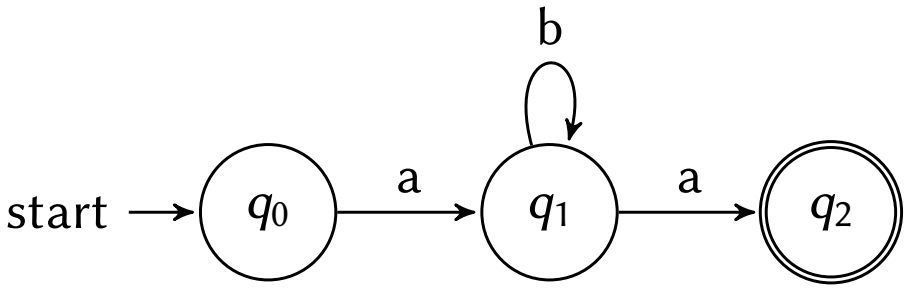
- generates the sentences (strings of words):

  the cat sleeps, Mia sleeps

Automata

Automaton

- An automaton is a **recognizing device** which accepts strings/words.
- The set of all strings accepted by an automaton is a formal language (= accepted language).



$\Rightarrow$ accepts strings: aa, aba, $abba$, $abbba$, ab^4a, ab^5a, ...etc.

$\Rightarrow$ accepts the language $L(ab^\star a)$

Formal grammar

A **formal grammar** is a 4-tuple $G = (N, T, S, R)$ with

- an alphabet of nonterminals N,
- an alphabet of terminals T with $N \cap T = \emptyset$,
- a start symbol $S \in N$,
- a finite set of rules/productions
  $R \subseteq \{\langle \alpha, \beta \rangle \mid \alpha, \beta \in (N)^* \text{ and } \alpha \notin T^*\}$.

Instead of $\langle \alpha, \beta \rangle$ we often write $\alpha \longrightarrow \beta$.

Formal grammar

Let $G = (N, T, S, R)$ be a grammar and $v, w \in (T \cup N)*$:

- v is **directly derived** from w (or w directly generates v), $w \Rightarrow v$ if $w = w_1 \alpha w_2$ and $v = w_1 \beta w_2$ such that $\langle \alpha, \beta \rangle \in R$
- v is **derived** from w (or w **generates** v), $w \Rightarrow^* v$ if there exists $w_0, w_1, \ldots w_k \in (T \cup N)^*$ $(k \geq 0)$ such that $w = w_0$, $w_k = v$ and $w_{i-1} \Rightarrow w_i$ for all $k \geq i \geq 0$
- $\Rightarrow^*$ denotes the reflexive, transitive closure of $\Rightarrow$
- $L(G) = \{w \in T^* | S \Rightarrow^* w\}$ is the **formal language generated by the grammar** G
- Two grammars G_1 and G_2 are **weakly equivalent** if and only if (iff) they generate the same language, i.e. $L(G_1) = L(G_2)$.

Chomsky-hierarchy

- The Chomsky-hierarchy is a hierarchy over structure conditions on the productions.
- Constraining the structure of the productions results in a restricted set of languages.
- The language classes correspond to conditions on the right- and left-hand sides of the productions.
- The Chomsky-hierarchy reflects a special form of complexity, other criteria are possible and result in different hierarchies.
- Linguists benefit from the rule-focussed definition of the Chomsky-hierarchy.

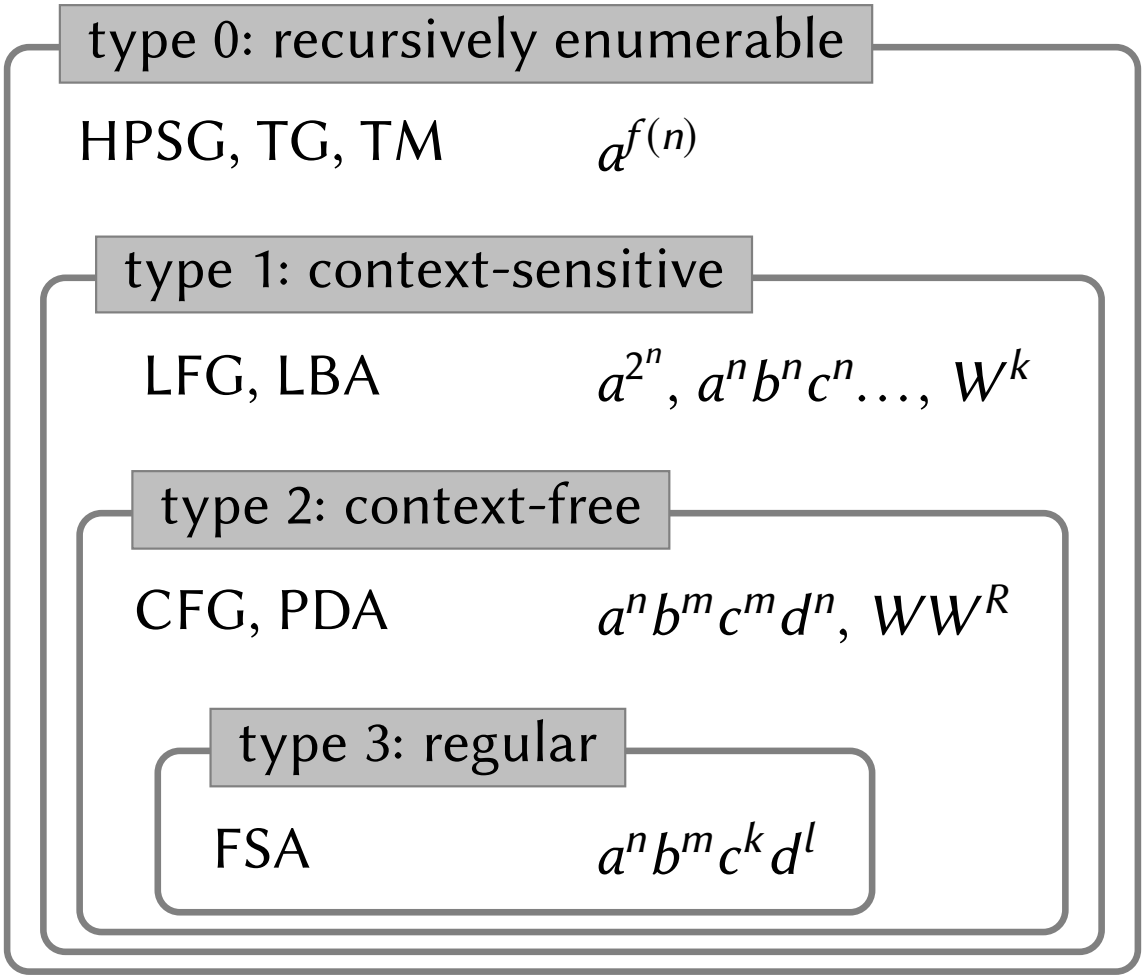
Chomsky-hierarchy

A grammar (N, T, S, R) is a

- Type 0 or unrestricted (phrase structure) grammar iff every production is of the form $\alpha \longrightarrow \beta$ with $\alpha \in (N \cup T)^* \setminus T^*$ and $\beta \in (N \cup T)^*$;
  generates a recursively enumerable language (RE).
- Type 1 or context-sensitive grammar iff every production is of the form $\gamma A \delta \longrightarrow \gamma \beta \delta$ with $\gamma, \delta, \beta \in (N \cup T)^*$, $A \in N$ and $\beta \neq \epsilon$;
  generates a context-sensitive language (CS).
- Type 2 or context-free grammar iff every production is of the form $A \longrightarrow \beta$ with $A \in N$ and $\beta \in (N \cup T)^* \setminus \{\epsilon\}$;
  generates a context-free language (CF).
- Type 3 or right-linear grammar iff every production is of the form $A \longrightarrow \beta B$ or $A \longrightarrow \beta$ with $A, B \in N$ and $\beta \in T^* \setminus \{\epsilon\}$;
  generates a regular language (REG).

For Type 1-3 languages a rule $S \longrightarrow \epsilon$ is allowed if S does not occur in any rule's right-hand side.

Chomsky-hierarchy



Chomsky-hierarchy: overview

type	grammar	rules	machine	word problem
RE	unrestricted	$\alpha \longrightarrow \beta$	TM	undecidable
CS	context-sensitive	$\gamma A \delta \longrightarrow \gamma \beta \delta$	LBA	exponential
CF	context-free	$A \longrightarrow \beta$	PDA	cubic
REG	right-linear	$A \longrightarrow aB \vert b$	FSA	linear

TM: Turing machine
LBA: linerar bounde automaton (a restricted TM)
PDA: push-down automaton
FSA: finite-state automaton

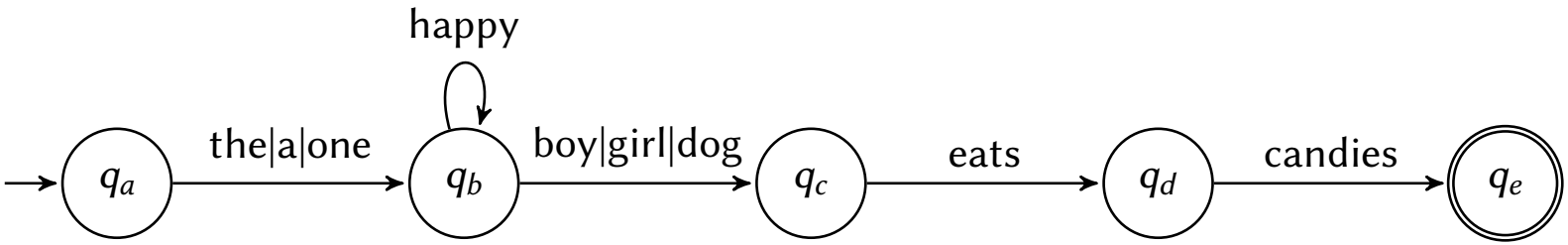
Natural Language is not regular

Hypothesis

All natural languages can be accepted by a finite state automaton (FSA).

FSA:

- finite set of states (including one start state and at least one end state)
- finite set of transitions between states
- On every transition, a word is read from the input.



Potential problems: recursion, constituency, long-distance dependencies
...whenever we might need a storage.

How to **proof** the inadequacy of FSA on the level of string languages?

Natural Language is not regular

The case of nested dependency: [Chomsky 1957]
a woman hired another woman
a woman whom another woman hired hired another woman
a woman whom another woman whom another woman hired hired hired another woman

Formal proof by contradiction (using closure properties and the Pumping Lemma):

- Every regular language satisfies the **Pumping Lemma**, hence the pattern $wa^n z$.
- homomorphism f: $f(\text{a woman}) = w$, $f(\text{whom another woman}) = a$, $f(\text{hired}) = b$, $f(\text{hired another woman}) = z$
  - $wa^* b^* z$ is a regular language; and
  - f(English) $\cap$ $wa^* b^* z = wa^n b^n z$ should be regular as well.
- $wa^n b^n z$ contradicts the Pumping Lemma for regular languages.

$\Rightarrow$ **English is not regular!**

Natural Language is not context-free

- a long time debate about the context-freeness of natural languages

Chomsky 1957:34

"Of course there are languages (in our general sense) that cannot be described in terms of phrase structure, but I do not know whether or not English is itself literally outside the range of such analysis."

- several wrong arguments (see Pullum & Gazdar 1982), e.g.:

Bresnan 1978:37–38

"in many cases the number of a verb agrees with that of a noun phrase at some distance from it ... this type of syntactic dependency can extend as memory or patience permits ... the distant type of agreement ... cannot be adequately described even by context-sensitive phrase-structure rules, for the possible context is not correctly describable as a finite string of phrases."

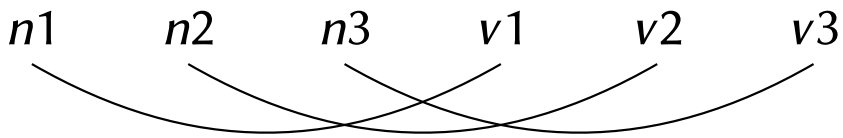
- right proof techniques: pumping lemma and closure properties
- What is a non context-free phenomenon in natural languages?

Natural Language is not context-free

A serious try: Syntax of Dutch [Bresnan et al. 1982]

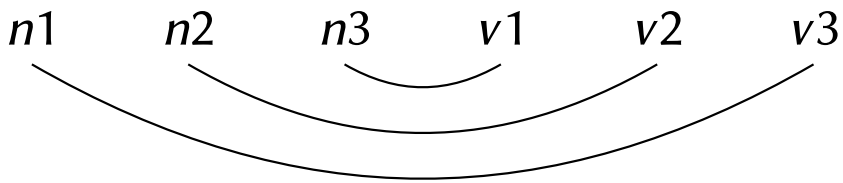
(2) dat Jan Piet de kinderen zag helpen zwemmen.
that Jan Piet the children saw help swim
'that Jan saw Piet helping the children to swim.'

Linguistic dependencies are cross-serial:



However: No reflection on the surface, i. e. in the string language!

⇒ In principle the string can be generated by a CFG, even though the dependencies will get lost.



Natural Language is not context-free

Another try by Culy (1985): Duplication in the morphology of Bambara

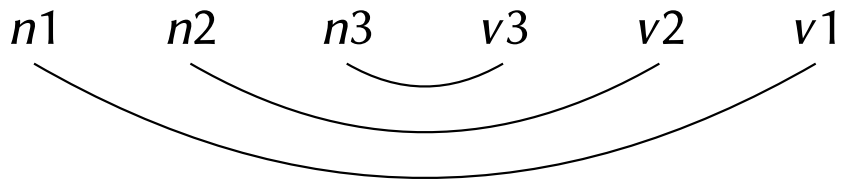
wulu *'dog'*
wulu-lela *'dog watcher'*
wulu-lela-nyinila *'dog watcher hunter'*
wulu-o-wulu *'whatever dog'*
wulu-lela-o-wulu-lela *'whatever dog watcher'*
wulu-lela-nyinila-o-wulu-lela-nyinila *'whatever dog watcher hunter'*

Pattern: $a^n b^m a^n b^m$ or ww (copy language) $\Rightarrow$ not context-free!

Natural Language is not context-free

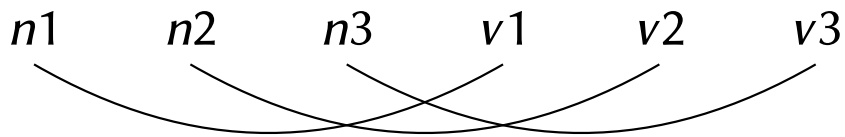
- German: **nested dependency** (subordinate clauses)

(3) (dass) er die Kinder dem Hans das Haus streichen helfen ließ
(that) he the children the Hans the house paint help let
'(that) he let the children help Hans paint the house'



- Schwyzerdütsch: **cross-serial dependency**

(4) …mer d'chind em Hans es huus lönd hälfe aastriiche
…we children.ACC the Hans.DAT the house.ACC let help paint
'…we let the children help Hans paint the house'



Natural Language is not context-free

Proof by Shieber (1985):

(5) Jan säit das mer d'chind em Hans es huus lönd hälfe aastriiche.

- homomorphism f:

  $f(\text{d'chind}) = a$ $f(\text{em Hans}) = b$ $f(\text{lönd}) = c$ $f(\text{hälfe}) = d$
  $f(\text{Jan säit das mer}) = w$ $f(\text{es huus}) = x$ $f(\text{aastriiche}) = y$
  $f(s) = z$

- $f(\text{Schwyzerdütsch}) \cap wa^*b^*xc^*d^*y = wa^mb^nxc^md^ny$
  - CF languages are closed under intersection with regular languages
  - $wa^*b^*xc^*d^*y$ is regular
  - by Pumping Lemma: $wa^mb^nxc^md^ny$ is not regular

$\Rightarrow$ **Schwyzerdütsch is not context-free**

Mildly context sensitive languages

NL is not mildly context-sensitive?

1. A set L of languages is mildly context-sensitive iff
   a. L contains all context-free languages
   b. L can describe cross-serial dependencies: there is an $n \geq 2$ such that $\{w^k \mid w \in (V_T)^*\} \in L$ for all $k \geq n$
   c. the languages in L are polynomially parseable, i.e., $L \subset$ PTIME
   d. the languages in L have the constant growth property
2. A formalism F is mildly context-sensitive iff the set $\{L \mid L = L(G)$ for some $G \in F\}$ is mildly context-sensitive.

- constant growth property: if we order the words of a language according to their length, then the length grows in a linear way
- there is a finite figure n, that limits the maximum number of instantiations of cross serial dependencies in a sentence of L

NL is not mildly context-sensitive?

Evidence brought forward against semi-linearity:

- case stacking ("Suffixaufnahme") in Old Georgian [Michaelis & Kracht 1997]
- Chinese number-names [Radzinski 1991]
- coordination in Dutch [Groenink 1997]
- relativized predicates in Yoruba [Kobele 2006]

Indexed Grammar (IG)

Indexed Grammar (IG): [Aho 1968]

- $IG = \langle T, N, I, S, R \rangle$, where I = a set of indices
  - context-free rules extended with indices $\Rightarrow$ a kind of stack
  - indices can appear only on non-terminals, e.g. $A[ijkiij]$
  - adding or removing indices on the right-hand side of the rule
  - the index-string can be infinite long
- two kinds of rules in R:

(i) "push and copy": $A[..] \rightarrow B[i..]$ e.g. $\ldots A[jk] \ldots \Rightarrow \ldots B[ijk] \ldots$

(ii) "pop and copy": $A[i..] \rightarrow B[..]$ e.g. $\ldots A[ijk] \ldots \Rightarrow \ldots B[jk] \ldots$

Linear Indexed Grammar (LIG): [Gazdar 1988]

The stack may be copied to at most one non-terminal per rule.

~~$A[..] \rightarrow B[..]C[..]$~~

Indexed Grammar (IG)

LIG for the language $a^n b^n c^n$:

- $G = \langle T, N, I, S, R \rangle$, where

  $T = \{a, b, c\}$, $N = \{S, Q\}$, $I = \{i\}$
  $R = \{\ S[..] \rightarrow aS[i..]c, \quad S[..] \rightarrow Q[..],$
  $R = \{\ Q[i..] \rightarrow Q[..]b, \quad Q[\,] \rightarrow \epsilon \qquad \}$

- example derivation:
  - $S[\,] \Rightarrow aS[i]c \Rightarrow aaS[ii]cc \Rightarrow aaaS[iii]ccc \Rightarrow aaaQ[iii]ccc \Rightarrow aaaQ[ii]bccc \Rightarrow aaaQ[i]bbccc \Rightarrow aaaQ[\,]bbbccc \Rightarrow aaabbbccc$
  - $S[\,] \overset{*}{\Rightarrow} a^n Q[i^n] c^n \overset{*}{\Rightarrow} a^n b^n c^n$

LIG for the language $\{ww | w \in \{a, b\}^*\}$:

- Try yourself!