EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

**Faculty of Philosophy**
General Linguistics


photo by Abdullah Mohiuddin
CC BY-NC-SA Hajime Yamauchi

# Language Evolution WiSe 2023/2024

## Lecture 8: Formal Language Theory (FLT)

**16/11/2023, Christian Bentz**

# Overview

Recap

Section 1: Basics of FLT
- Grammar in FLT
- Terminal Symbols
- Non-Terminal Symbols
- Rewrite Rules
- Exercise

Section 2: The Chomsky Hierarchy
- Regular Languages (Type 3)
- Context-Free Languages (Type 2)
- Context-Sensitive Languages (Type 1)
- Recursively Enumerable Languages (Type-0)
- The Classical Hierarchy

Summary

References

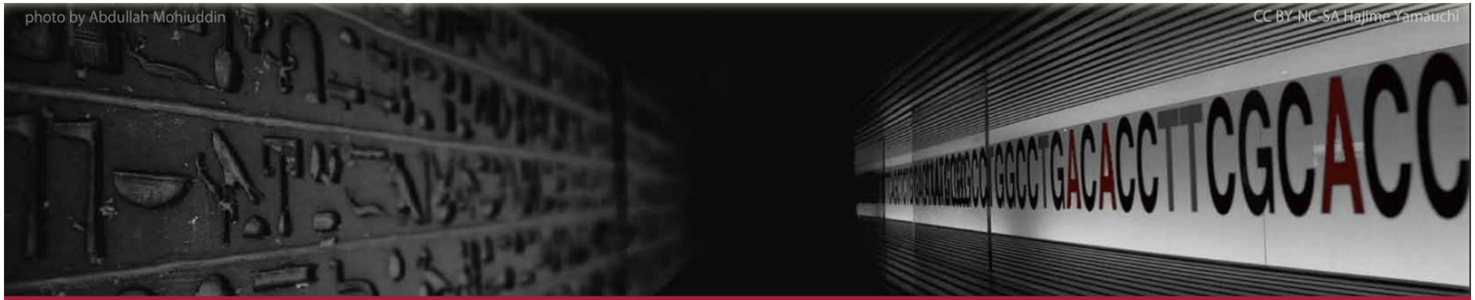


photo by Abdullah Mohiuddin

CC BY-NC-SA Hajime Yamauchi

# Recap

# *What* is unique about human language?

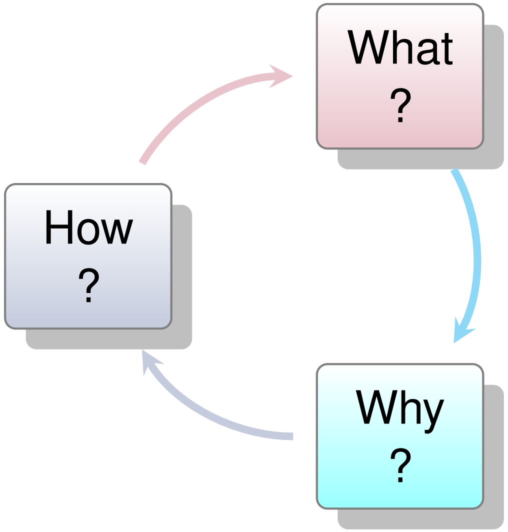
> "If a Martian scientist [...] received from Earth the broadcast of an extensive speech [...] what criteria would [...] determine whether the reception represented the effect of an animate process on Earth, or merely the latest thunderstorm on Earth?"

Zipf (1936). The psycho-biology of language, p. 187.

Recap
Section 1: Basics of FLT
Section 2: The Chomsky Hierarchy
Summary
References

# Three Questions

1. **What** evolved, i.e. what is “language” in the first place?
2. **Why** did it evolve, i.e. did it have particular functions?
3. **How** did it evolve?



# *What* is Language?

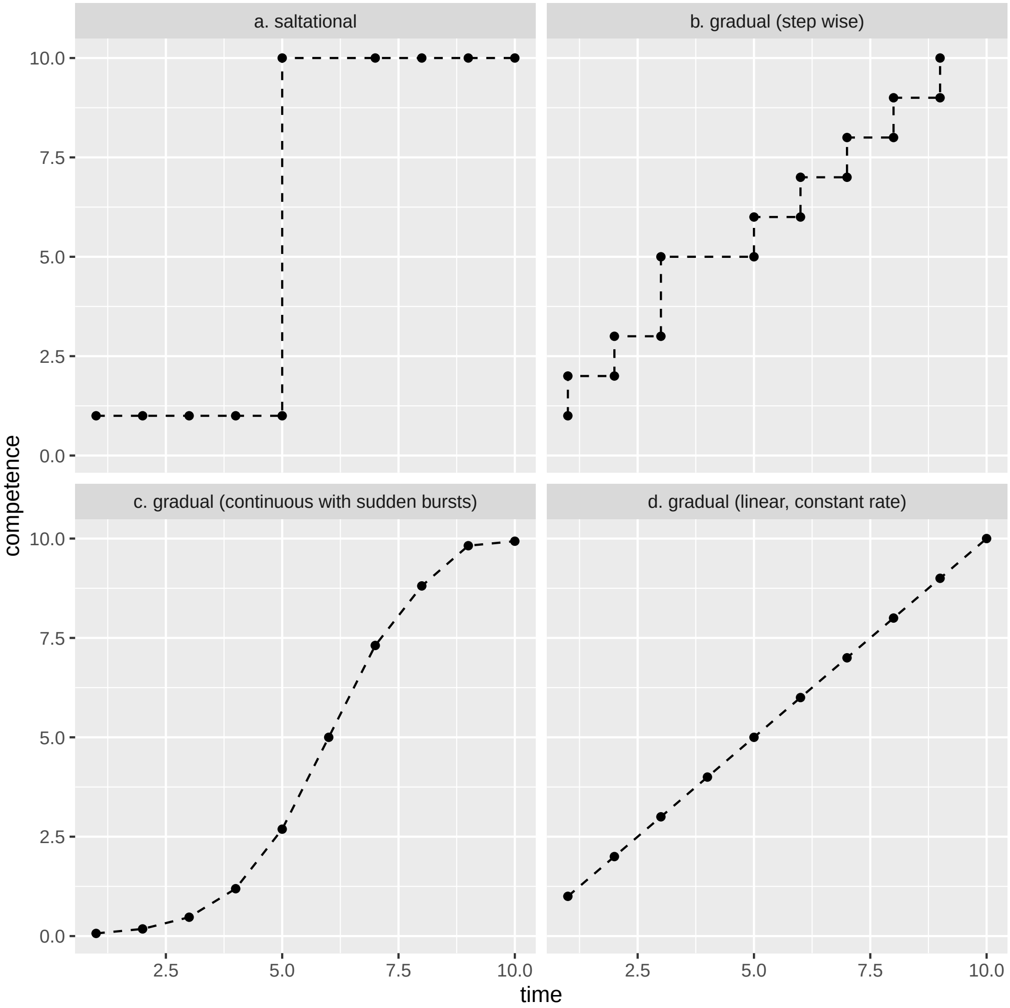
# Competing Definitions of *Language*

- **Formal Language Theory**
- **Faculty of Language**
    - Recursion
    - Rich Language Faculty (Narrow Sense)
- **Minimalism**
    - Strong Minimalist Thesis
    - Minimalist Layers Hypothesis
- **Usage-Based Grammar**

# Evolutionary Functions: *Why* did Language evolve?

- **Vocal Communication**
- **Gestural Communication**
- **Social Bonding** (Grooming/Gossiping)
- **Thinking**
- **No Function**

# Evolutionary Scenarios



# Summary

Is language more like growing a wing, or like learning to play chess?

Saltational Account

Gradual Account

Co-evolution Account

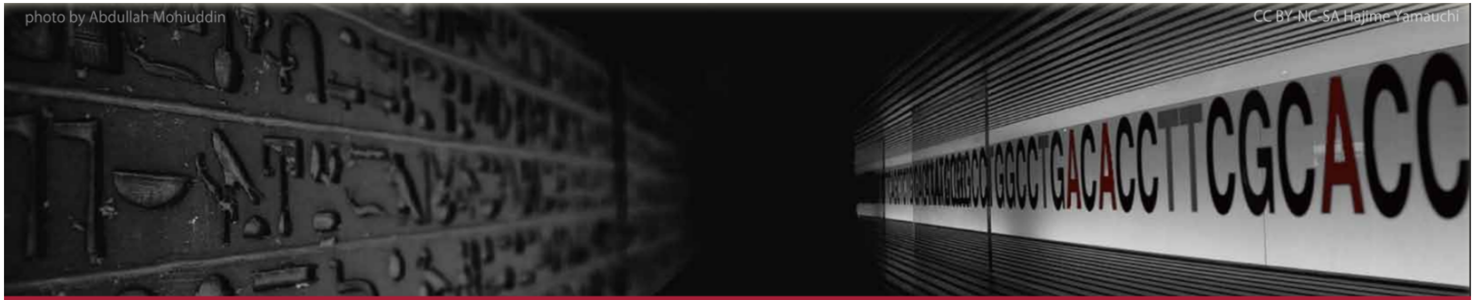
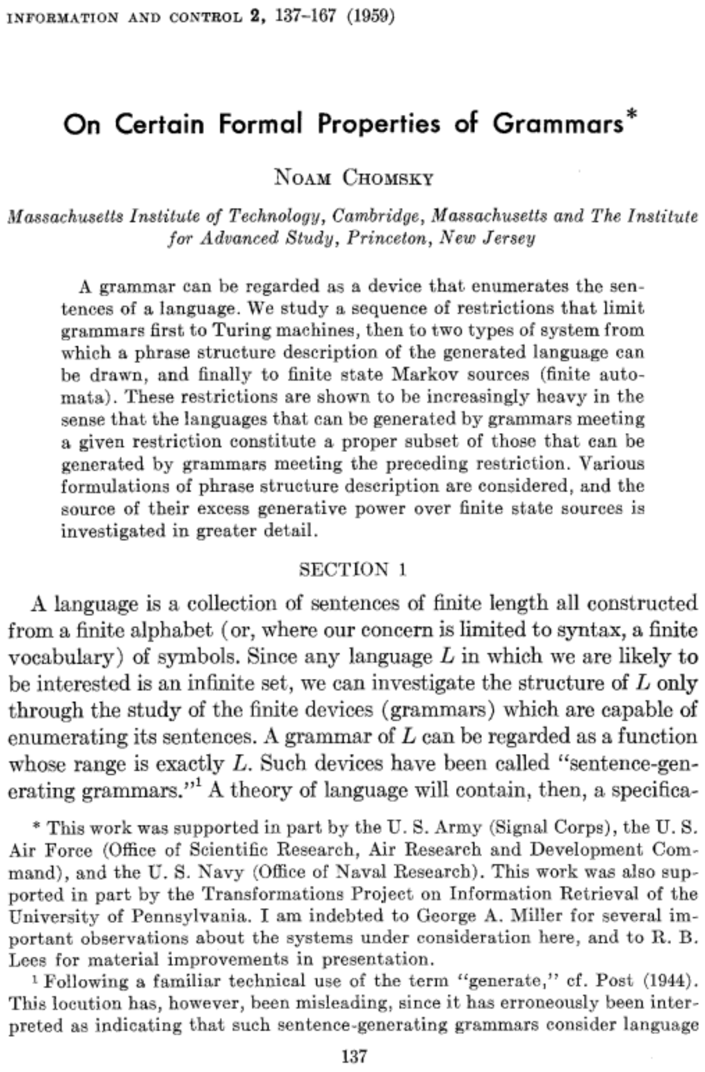


photo by Abdullah Mohiuddin

CC BY-NC-SA Hajime Yamauchi

# Section 1: Basics of FLT



INFORMATION AND CONTROL **2,** 137–167 (1959)

## On Certain Formal Properties of Grammars*

NOAM CHOMSKY

*Massachusetts Institute of Technology, Cambridge, Massachusetts and The Institute for Advanced Study, Princeton, New Jersey*

A grammar can be regarded as a device that enumerates the sentences of a language. We study a sequence of restrictions that limit grammars first to Turing machines, then to two types of system from which a phrase structure description of the generated language can be drawn, and finally to finite state Markov sources (finite automata). These restrictions are shown to be increasingly heavy in the sense that the languages that can be generated by grammars meeting a given restriction constitute a proper subset of those that can be generated by grammars meeting the preceding restriction. Various formulations of phrase structure description are considered, and the source of their excess generative power over finite state sources is investigated in greater detail.

### SECTION 1

A language is a collection of sentences of finite length all constructed from a finite alphabet (or, where our concern is limited to syntax, a finite vocabulary) of symbols. Since any language *L* in which we are likely to be interested is an infinite set, we can investigate the structure of *L* only through the study of the finite devices (grammars) which are capable of enumerating its sentences. A grammar of *L* can be regarded as a function whose range is exactly *L*. Such devices have been called "sentence-generating grammars."[1] A theory of language will contain, then, a specifica-

---

* This work was supported in part by the U. S. Army (Signal Corps), the U. S. Air Force (Office of Scientific Research, Air Research and Development Command), and the U. S. Navy (Office of Naval Research). This work was also supported in part by the Transformations Project on Information Retrieval of the University of Pennsylvania. I am indebted to George A. Miller for several important observations about the systems under consideration here, and to R. B. Lees for material improvements in presentation.

[1] Following a familiar technical use of the term "generate," cf. Post (1944). This locution has, however, been misleading, since it has erroneously been interpreted as indicating that such sentence-generating grammars consider language

137

# **Grammar** in Formal Language Theory

A **grammar** $\mathcal{G}$ in formal language theory is a quadruple consisting of the set of terminal symbols, non-terminal symbols, a starting symbol $S$, and a set of rewrite rules $R$:

$$\langle T, NT, S, R \rangle^{1} \tag{1}$$

Jäger and Rogers (2012). Formal language theory: refining the Chomsky hierarchy.
Partee et al. (1990). Mathematical methods in linguistics.

---

[1] S is a “distinguished member” of NT.

# Symbols: **Terminals**

Firstly, we have a finite set of so-called **terminal symbols** ($T$). In classical phrase structure grammars (PSG) these are typically words,[2] but it could be any set of signs:

$$T_1 = \{a, book, child, reads, the, \dots\}^3 \tag{2}$$

$$T_2 = \{aa, ae, ad, be, bf, cd, \dots\}^4 \tag{3}$$

$$T_3 = \{\star, \circ, \diamond, \odot, \Box, \heartsuit, \oplus, \dots\} \tag{4}$$

---

[2] Words are typically assumed as terminals for the analysis of natural language, but note that we could also choose morphemes, syllables, characters, etc.

[3] I here order them alphabetically, but note that the order in a set does not matter.

[4] Birdsongs and other types of animal communication are sometimes transcribed into strings of two or three letters representing song “syllables”.

# Symbols: **Non-Terminals**

Secondly, we define a finite set of so-called **non-terminal symbols** (*NT*). *Non-terminal* means that these symbols are not to be found in an actual terminal string derivation of a language. All non-terminal symbols have to be eventually replaced by terminals.

In phrase structure grammars, these are symbols for phrases (e.g. NP, VP, AP, etc.), parts of speech (N, V, A, etc.), as well as the starting symbol *S*. For example:

$$NT_1 = \{NP, VP, AP, \ldots N, V, A, \ldots S\} \tag{5}$$

# Rewrite Rules

In the most general definition, **rewrite rules** define how we can rewrite a string of symbols into another string of symbols. We formally have

$$\alpha \rightarrow \beta, \tag{6}$$

where $\alpha$ is a string of $n$ symbols $(x_1, x_2, x_3, \ldots, x_n)$ , with $n \geq 1$, for which $x_i \in (T \cup NT)$, and, likewise, $\beta$ is a string of symbols $(y_1, y_2, y_3, \ldots, y_n)$ for which $y_i \in (T \cup NT)$.

In words: $\alpha$ and $\beta$ are strings which are made up of terminal symbols, non-terminal symbols, or both. For example, a noun phrase involving a determiner and a noun can be rewritten as follows:

NP → DET N
NP → the N
NP → the tree

| Example: Rewrite | Rule |
|---|---|
| S | – |
| NP V NP | 6 |
| DET N V NP | 7 |
| DET N V DET N | 7 |
| DET N reads DET N | 5 |
| the N reads DET N | 1 |
| the child reads DET N | 3 |
| the child reads a N | 2 |
| the child reads a book | 4 |

## Terminals

$T = \{a, book, child, reads, the\}$

## Non-Terminals

$NT = \{DET, N, NP, V\}$

## *R* (Terminals)

1. DET → the
2. DET → a
3. N → child
4. N → book
5. V → reads

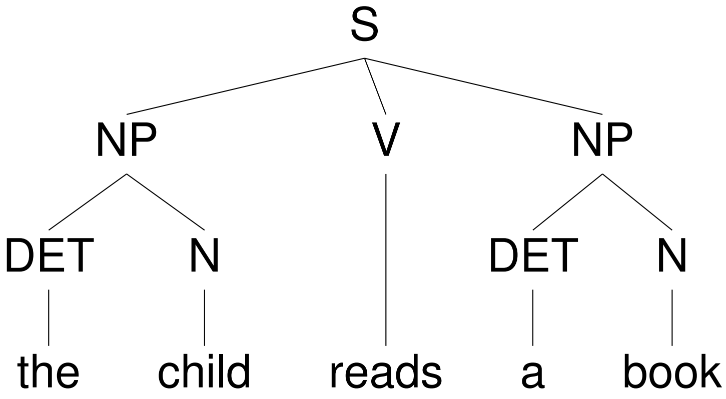
## *R* (Non-Terminals)

6. S → NP V NP
7. NP → DET N

Note: The horizontal line indicates the point where rules exclusively defined with non-terminals ($R(NT)$) end, and rules involving terminals ($R(T)$) start. While the order of application of non-terminal rules is often important, the order of the application of terminal rules is irrelevant.

Recap
Section 1: Basics of FLT
Section 2: The Chomsky Hierarchy
Summary
References

# Trees and Bracket Notation



**Rewrite Notation**

S
NP V NP
DET N V NP
DET N V DET N

---

DET N reads DET N
the N reads DET N
the child reads DET N
the child reads a N
the child reads a book

[S [NP [DET [the]][N [child]]][V [reads]][NP [DET [a]][N [book]]]][5]

---

[5] Note: The *Bracket Notation* is yet another equivalent way to visualize the same structure. In fact, the latex code generating this slide takes the bracket notation as input to generate the above tree. There is also an online tool at ironcreek.net/syntaxtree to generate trees based on bracket notation input.

# Exercise

Assume the following sets of terminal and non-terminal symbols:

$$T = \{\times, +, \circ, \odot, \diamond, \triangleleft\}, \tag{7}$$

$$NT = \{\mathrm{X}, \bigcirc, \Box, S\}. \tag{8}$$

The non-terminal symbols stand in for terminal symbols with a similar shape (i.e. X for the first two, ◯ for the middle two, and □ for the last two). *S* is the starting symbol.

Further consider the following rules:

- Symbols with lines crossing can occur *exactly once* in a sequence, and only *at the end* of a sequence.
- Symbols with a rounded shape can occur *exactly once* in a sequence, and have to occur *before* all signs with an angled shape.
- Symbols with an angled shape can occur an *infinite* number of times in the sequence.
- A symbol of each type has to occur *at least once* in the sequence.

**Give the rewrite rules matching the verbal rules defined above.**
**Give the rewrite derivation of the string $\odot \diamond \diamond \diamond \triangleleft +$.**

# One Possible Solution (Context-Free)

Rewrite rules:

$S \rightarrow \bigcirc\Box X$
$\Box \rightarrow \diamond\Box$ (or $\Box\diamond$)
$\Box \rightarrow \triangleleft\Box$ (or $\Box\triangleleft$)
$\Box \rightarrow \diamond$
$\Box \rightarrow \triangleleft$
$\bigcirc \rightarrow \circ$
$\bigcirc \rightarrow \odot$
$X \rightarrow +$
$X \rightarrow \times$

Rewrite derivation:

$S$
$\bigcirc\Box X$
$\bigcirc \diamond \Box X$ (rule: $\Box \rightarrow \diamond\Box$)
$\bigcirc \diamond \diamond \Box X$
$\bigcirc \diamond \diamond \diamond \Box X$
$\bigcirc \diamond \diamond \diamond \triangleleft X$ (rule: $\Box \rightarrow \triangleleft$)
$\odot \diamond \diamond \diamond \triangleleft X$ (rule: $\bigcirc \rightarrow \odot$)
$\odot \diamond \diamond \diamond \triangleleft +$ (rule: $X \rightarrow +$)

# Another Possible Solution (Regular)

Recap
Section 1: Basics of FLT
Section 2: The Chomsky Hierarchy
Summary
References

Rewrite rules:

$S \rightarrow \circ\bigcirc$
$S \rightarrow \odot\bigcirc$
$\bigcirc \rightarrow \diamond\Box$
$\bigcirc \rightarrow \lhd\Box$
$\Box \rightarrow \diamond\Box$
$\Box \rightarrow \lhd\Box$
$\bigcirc \rightarrow \diamond X$
$\bigcirc \rightarrow \lhd X$
$\Box \rightarrow \diamond X$
$\Box \rightarrow \lhd X$
$X \rightarrow +$
$X \rightarrow \times$

Thanks to John Wang for this solution.

# Important Take-Home-Message

We can introduce **recursion** into a formal grammar by any rule which has the same non-terminal(s) on the left and right hand side:

$$\square \rightarrow \triangleleft \square \qquad (9)$$

So this is already possible in **regular** grammars, i.e. the lowest level of the traditional Chomsky Hierarchy. Arguably, there are some natural language structures where such a recursive pattern is needed. For instance, when a number of adjectives (potentially arbitrarily large) is added before a noun in the English noun phrase (e.g. *a bright, friendly, welcoming, ... friend*).

$$\overline{N} \rightarrow A\,\overline{N} \qquad (10)$$

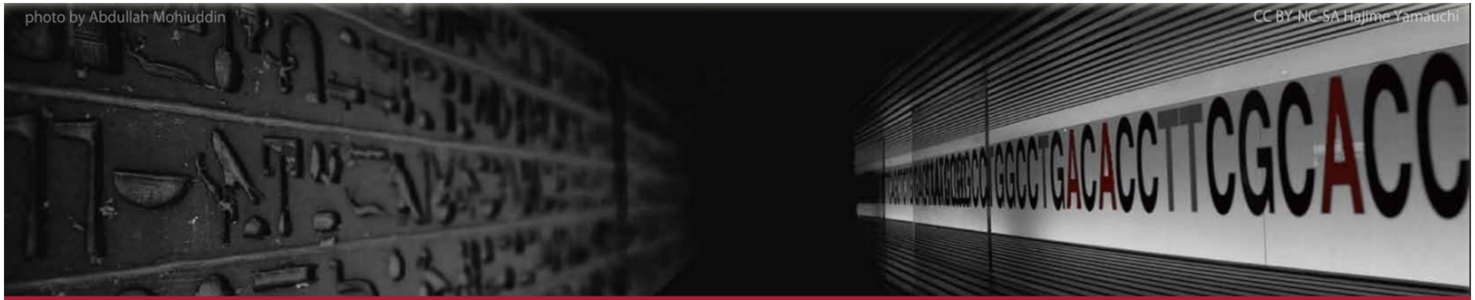


photo by Abdullah Mohiuddin

CC BY-NC-SA Hajime Yamauchi

# Section 2: The Chomsky Hierarchy

# Notational Conventions

- Lower case Latin letters of the beginning of the alphabet are **terminal symbols**, i.e. $a, b, c \in T$. The ones from the end of the alphabet, i.e. $x, y, z$ are used as *variables* representing any possible terminal symbol.
- Upper case Latin letters represent **non-terminal symbols**, i.e. $A, B, C \in NT$, with $X, Y, Z$ being *variables* again. $S$ is the starting symbol.
- Lower case Greek letters, i.e. $\alpha$, $\beta$, $\omega$, represent **strings of terminal and non-terminal symbols**.
- We use the symbol $\epsilon$ for the **empty string**.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

**Faculty of Philosophy**
General Linguistics

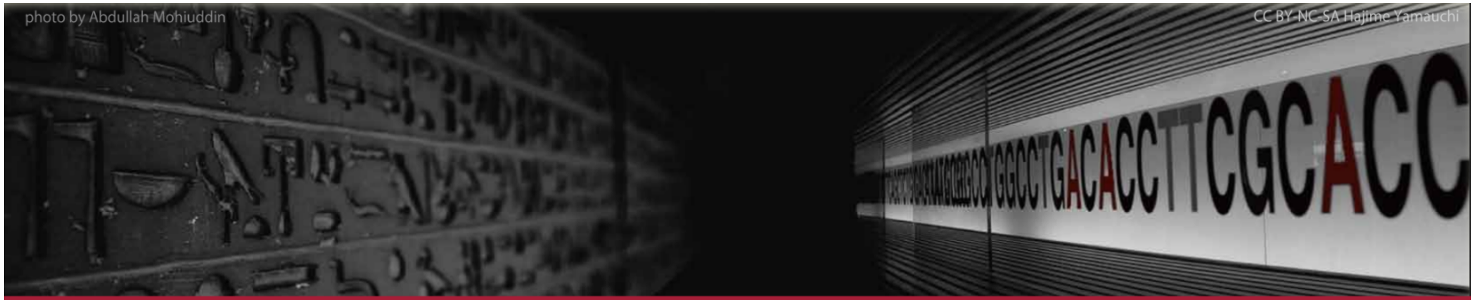


photo by Abdullah Mohiuddin

# Regular Languages (Type 3)

# Definition: Finite State Grammar

In so-called **regular**, or **finite state grammars**, the rewrite rules are restricted to two forms:

$$\mathrm{X} \rightarrow \mathrm{x}$$
$$\mathrm{X} \rightarrow \mathrm{xY}$$

In words, any non-terminal on the left-hand side of a rule can only be rewritten either into a terminal, or into a terminal followed by a non-terminal.

Jäger & Rogers (2012), p. 1958.

Notes: Jäger & Rogers (2012) just use Latin letters from the beginning of the alphabet here, i.e. $\mathrm{A} \rightarrow \mathrm{a}$, $\mathrm{A} \rightarrow \mathrm{aB}$. Moreover, remember from typical mathematical functions like $f(x, y) = x + y^2$, that $x$ and $y$ might represent different numbers, *but they do not have to*, i.e. it is possible that $x = y$. Also, we could have the rule $\mathrm{X} \rightarrow \mathrm{Yx}$ instead of the one above (but we could not mix them according to Jäger & Rogers 2012, footnote 6).

# Abstract Example

| Rewrite | Rule |
|---|---|
| S | – |
| aS | 1 |
| aaS | 1 |
| aaaS | 1 |
| aaaa | 2 |

**Terminals**

$T = \{a\}$

**Non-Terminals**

$NT = \{S\}$

***R***

1. S → aS
2. S → a

Note: The language generated with this grammar is $L(\mathcal{G}_R) = \{a, aa, aaa, \ldots, a^n\}$, with $n \in \mathbb{N}$, and hence in theory we can have $n = \infty$ due to the recursive rule. Information encoding based on this grammar already achieves **discrete infinity**, as it represents the natural numbers.

# Natural Language Example

| Rewrite | Rule |
|---|---|
| S | – |
| she VP | 1 |
| she saw NP | 2 |
| she saw the N | 3 |
| she saw the axolotl | 4 |

**Terminals**

$T = \{axolotl, bunny, saw, she, the\}$

**Non-Terminals**

$NT = \{N, NP, VP, S\}$

***R***

1. S $\rightarrow$ she VP
2. VP $\rightarrow$ saw NP
3. NP $\rightarrow$ the N
4. N $\rightarrow$ axolotl
5. N $\rightarrow$ bunny

The language generated with this grammar is
$L(\mathcal{G}_R) = \{\text{she saw the axolotl}, \text{she saw the bunny}\}$.

Recap
Section 1: Basics of FLT
Section 2: The Chomsky Hierarchy
Summary
References

# Further Examples of Regular Languages $L(\mathcal{G}_R)$

- The set of strings which follows the pattern $x^n y^m$, e.g. $L(\mathcal{G}_R) = \{ab, aab, abb, aabb, \dots\}$[6]
- The set of strings such that the number of ‘a’s in them is a multiple of 4, i.e. $L(\mathcal{G}_R) = \{aaaa, aaaaaaaa, aaaaaaaaaaaa, \dots\}$
- The set of natural numbers that leave a remainder of 3 when divided by 5, i.e. $L(\mathcal{G}_R) = \{8, 13, 18, \dots\}$
- etc.

Jäger & Rogers (2012), p. 1958.

---

[6] If we include a rule which yields an empty element, e.g. $x \rightarrow \epsilon$, then *a*, *b*, and $\epsilon$ would also be part of this set.

# Are Natural Languages Fully **Regular**?[7]

While there are certainly many sentences in natural languages which can be generated by regular grammars (maybe even the vast majority?), Chomsky argued in some of his earliest work that structures such as

1. **If** $S_1$, **then** $S_2$,
2. **Either** $S_3$, **or** $S_4$,
3. The **man** who said that $S_5$, **is** arriving today,

bear dependencies which hinder a strictly regular generation.

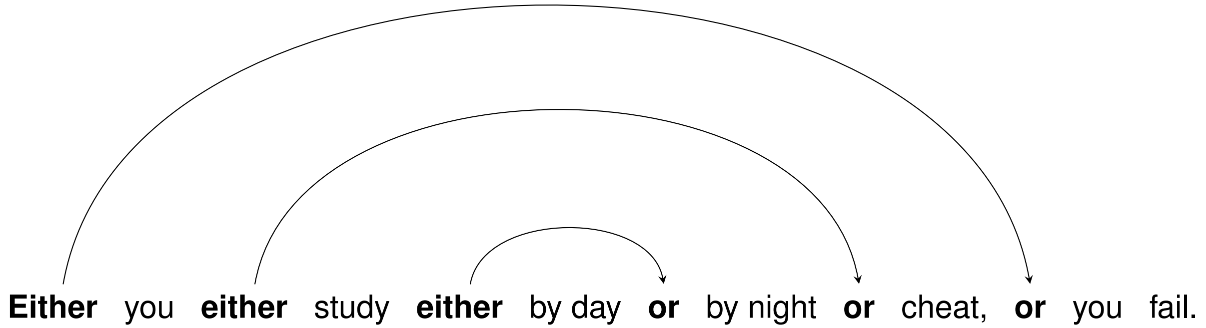
Chomsky (1956), p. 115.

Note: The $S$s here stand in for declarative sentences. The dependencies connect the words in bold face.

---

[7] In the sense "equivalent to finite state languages".

# Non-Regular Example



(1) **Either** you **either** study **either** by day **or** by night **or** cheat, **or** you fail.

either ... either ... either ... or ... or ... or
a ... a ... a ... b ... b ... b

$a^n b^n$

Under the assumption that such constructions can be applied productively *ad infinitum*, a regular grammar could not generate *these and only these* sentences. While the *non-regular* language $L(\mathcal{G}_{NR}) = \{ab, aabb, aaabbb, \ldots, a^nb^n\}$ is a proper subset of the regular language $L(\mathcal{G}_R) = \{ab, aab, abb, aabb, \ldots, a^nb^m\}$, i.e. $L(\mathcal{G}_{NR}) \subset L(\mathcal{G}_R)$, there is no way to identify this subset with a finite state automaton.

# Summary: Regular Grammars (Type 3)

- **Regular grammars** are already very powerful. They can generate, for instance, a set of strings which reflects the natural numbers $\mathbb{N}$, i.e. $L(\mathcal{G}_R) = \{a, aa, aaa, \ldots, a^n\}$.
- They can (in principle) generate all sentences in natural languages too, however...
- Firstly, for certain constructions, e.g. of the $a^nb^n$ type, they will also generate ungrammatical sentences...
- Secondly, since at least one terminal symbol has to be produced in every rewrite, the generation of sentences is not very “elegant”, in the sense that no higher level patterns (phrase structures) can be captured.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

**Faculty of Philosophy**
General Linguistics

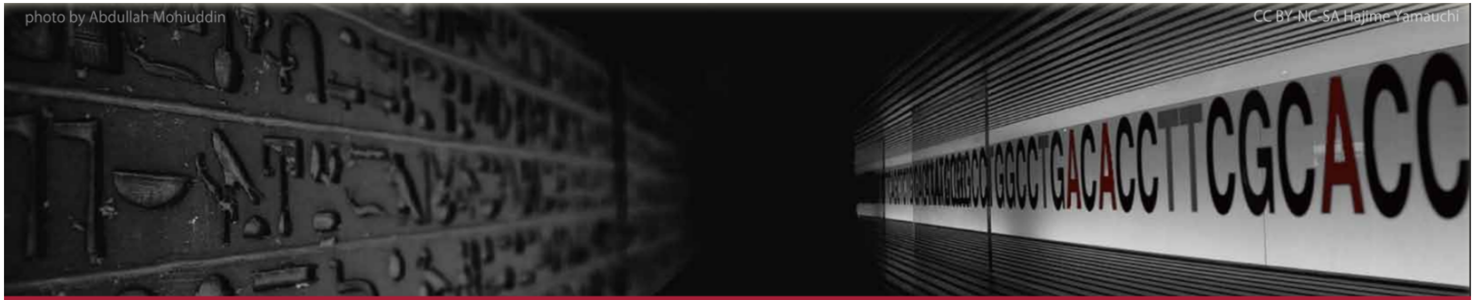


photo by Abdullah Mohiuddin

CC BY-NC-SA Hajime Yamauchi

# Context-Free Languages (Type 2)

# Definition: Context-Free Grammar

In the original work of Chomsky, the most general rewrite rule from above, i.e.

$$\alpha \rightarrow \beta, \tag{11}$$

is further restricted by, firstly, adding a **context** to each side, i.e. $\varphi_1\_\varphi_2$, and secondly, by requiring that $\alpha$ is a single non-terminal $X$ such that we have

$$\varphi_1 X \varphi_2 \rightarrow \varphi_1 \beta \varphi_2 \tag{12}$$

Now, if this context is **defined to be *null***, we call this a **context-free** grammar (hence the name). We thus have rewrite rules of the general form

$$X \rightarrow \beta \tag{13}$$

In words, we only allow one *single non-terminal symbol* on the left-hand side of the arrow, but an arbitrary string of terminals and non-terminals on the right-hand side.

Chomsky (1959), p. 142.
Jäger and Rogers (2012), p. 1957.

# Abstract Example

| Rewrite | Rule |
|---|---|
| S | – |
| aSb | 1 |
| aaSbb | 1 |
| aaaSbbb | 1 |
| aaabbb | 2 |

### Terminals

$$T = \{a, b, \epsilon\}$$

### Non-Terminals

$$NT = \{S\}$$

### $R$

1. S $\rightarrow$ aSb
2. S $\rightarrow$ $\epsilon$

The language generated with this grammar is
$L(\mathcal{G}_{CF}) = \{ab, aabb, aaabbb, \ldots, a^n b^n\}$, with $n \in \mathbb{N}$.

# Natural Language Example

| Rewrite | Rule |
|---|---|
| S | – |
| NP V NP | 1 |
| PRON V NP | 3 |
| PRON V DET N | 2 |
| she saw the axolotl | 4, 5, 6, 8 |

**Terminals**

$T = \{axolotl, bunny, saw, she, the\}$

**Non-Terminals**

$NT = \{N, NP, V, PRON, S\}$

***R***

1. S $\rightarrow$ NP V NP
2. NP $\rightarrow$ DET N
3. NP $\rightarrow$ PRON
4. DET $\rightarrow$ the
5. V $\rightarrow$ saw
6. N $\rightarrow$ axolotl
7. N $\rightarrow$ bunny
8. PRON $\rightarrow$ she

# Natural Language Example

The language generated with this grammar is $L(\mathcal{G}_{CF}) =$ $\{$she saw the axolotl,
she saw the bunny,
the axolotl saw she,
the bunny saw she,
the axolotl saw the axolotl,
the axolotl saw the bunny,
the bunny saw the bunny, the bunny saw the axolotl$\}$.

**Terminals**

$T = \{axolotl, bunny, saw, she, the\}$

**Non-Terminals**

$NT = \{N, NP, V, PRON, S\}$

**$R$**

1. S → NP V NP
2. NP → DET N
3. NP → PRON
4. DET → the
5. V → saw
6. N → axolotl
7. N → bunny
8. PRON → she

# Further Examples of Context-Free Languages

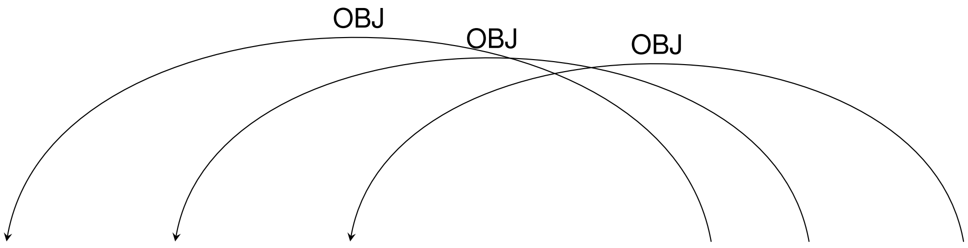
- *Mirror language*: the set of strings $\gamma\omega$ over a set of terminals $T$ such that $\omega$ is the mirror image of $\gamma$, e.g. $L(\mathcal{G}_{CF}) = \{abba, abccba, abcddcba, \dots\}$
- *Palindrome language*: the set of strings $\gamma$ that are identical to their mirror image, e.g. $L(\mathcal{G}_{CF}) = \{aba, bab, abba, baab, aabaa, \dots\}$
- Languages with strings of the form $x^n y^m z^m w^n$.
- etc.

Jäger & Rogers (2012), p. 1958.

# Are Natural Languages **Context-Free**?

This was a debated topic in *Formal Language Theory* (FLT) since the first formulations of the types of languages in the 1950s and 1960s. It took until the mid 1980s (!) for an alleged **non-context-free sentence structure** to be proposed and (apparently) accepted by most syntacticians.

Swiss German (Indo-European)



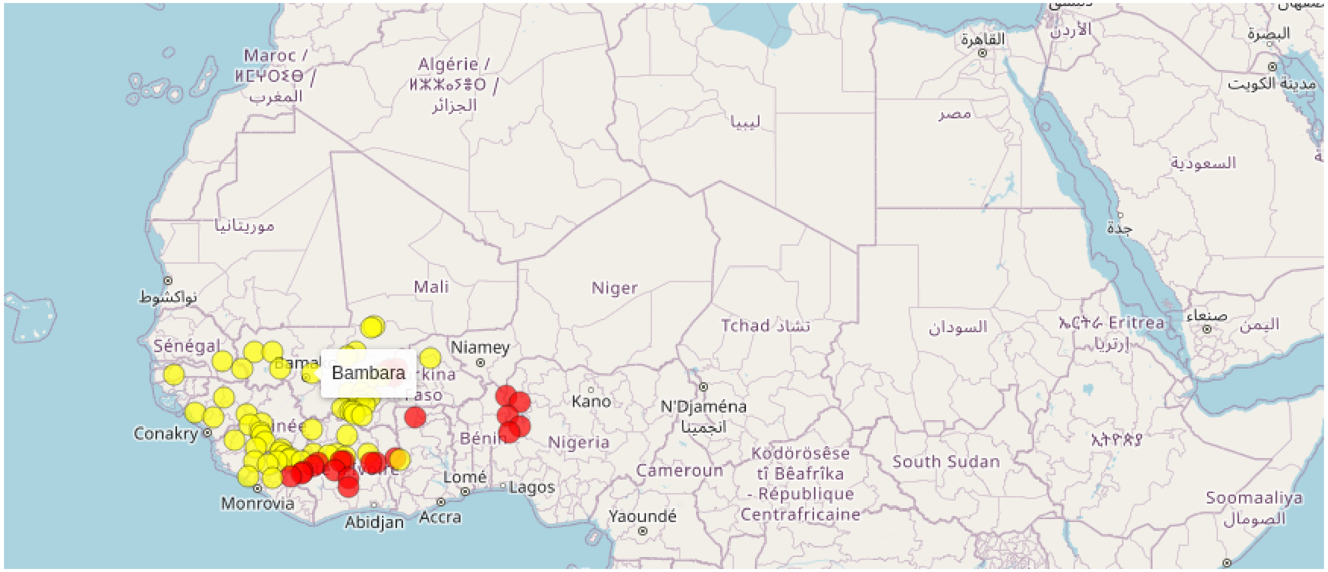
[...] das mer d'chind em Hans es huus haend wele laa hälfe aastriiche

"[...] that we have wanted to let the children help Hans paint the house."

Shieber (1984). Evidence against the context-freeness of natural language.

# Another proof based on Bambara (Mande, bam)

> “In this paper I look at the possibility of considering the vocabulary of a natural language as a sort of language itself. In particular, I study the weak generative capacity of the vocabulary of Bambara, and show that vocabulary is not context free. This result has important ramification the theory of syntax of natural language.”
>
> Culy (1985). The complexity of the vocabulary of Bambara, p. 345.



# Bambara Vocabulary Structure

Noun *o* Noun construction:

(2)

| | | |
|---|---|---|
| wulu | o | wulu |
| dog | PRT | dog |

“whichever dog”

Agentive construction: Noun + Transitive Verb + *la*

(3) wulunyinina

| | | |
|---|---|---|
| wulu | nyini | la |
| dog | search.for | PRT |

“one who searches for dogs”, i.e. “dog searcher”

Agentive construction with recursive application:

(4) wulunyininanyinina

| | | | | |
|---|---|---|---|---|
| wulu | nyini | la | nyini | la |
| dog | search.for | PRT | search.for | PRT |

“one who searches for dog searchers”

# Bambara Vocabulary Structure

Recap
Section 1: Basics of FLT
Section 2: The Chomsky Hierarchy
Summary
References

Noun o Noun construction and Agentive construction together:

(5) wulunyinina o wulunyinina

| | | | | | | |
|---|---|---|---|---|---|---|
| wulu | nyini | la | o | wulu | nyini | la |
| dog | search.for | PRT | PRT | dog | search.for | PRT |

“whichever dog searcher”

(6) wulunyininanyinina o wulunyininanyinina

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| wulu | nyini | la | nyini | la | o | wulu | nyini | la | nyini | la |
| dog | search.for | PRT | search.for | PRT | PRT | dog | search.for | PRT | search.for | PRT |

“whichever searcher of dog searchers”

# Culy's proof in a nutshell

A subset of the Vocabulary ($\mathcal{B}$) in Bambara is of the form:

$$\mathcal{B}' = \{\text{wulu}^m\text{nyinina}^n\,\text{o}\,\text{wulu}^m\text{nyinina}^n | m, n \geq 1\}. \tag{14}$$

if we assume that this is the case for all nouns *a* and the suffix *-nyinina* (*b*), this yields the more general form:[8]

$$\mathcal{B}' = \{\text{a}^m\text{b}^n\text{a}^m\text{b}^n | m, n \geq 1\}. \tag{15}$$

Since $\mathcal{B}'$ is not context free, and a subset of the overall vocabulary ($\mathcal{B}' \subset \mathcal{B}$), it follows that $\mathcal{B}$ is also not context free.

Culy (1985), p. 349.

**Problem**: The formulations in the above equations are not fully correct, since, according to Culy's analysis of Bambara vocabulary, $m = 1$ and $n \geq 1$. In other words, the noun cannot be multiplied, only the suffix. So, strictly speaking, we look at a pattern $\text{ab}^n\text{ab}^n$.

---

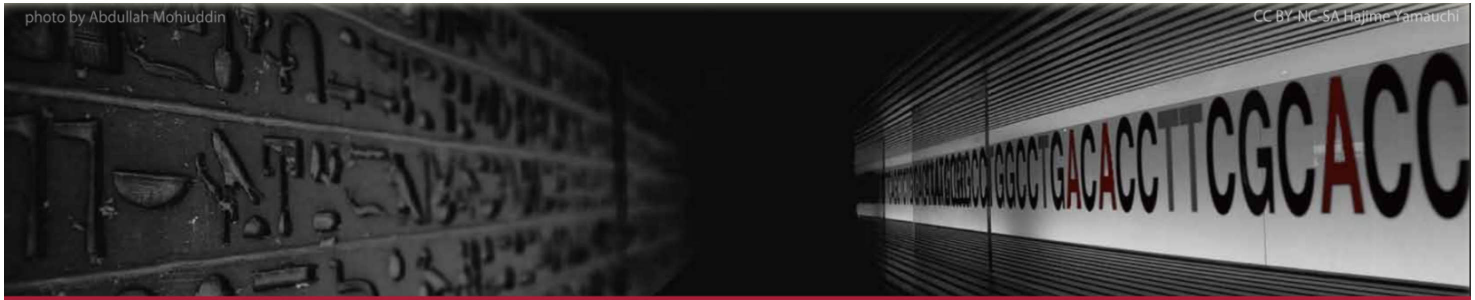
[8] The "o" is here dropped in the general form without loss of generality.

# Summary: Context-Free Grammars (Type 2)

- **Context-free grammars** are more powerful than regular grammars by allowing more diverse sentences to be generated.
- If we take the **binarized version** of CFG, this essentially boils down to having one additional rule pattern compared to regular grammars: $X \rightarrow YZ$[9]
- It has been argued (and largely accepted) that some (very rare) patterns in natural languages might require **non-context-free** rules to be generated.

---

[9] In this case, it is allowed that $Y = Z$.



# Context-Sensitive Languages (Type 1)

# Definition: Context-Sensitive Grammar

In the original work of Chomsky, the most general rewrite rule from above, i.e.

$$\alpha \to \beta, \tag{16}$$

is further restricted by, firstly, adding a **context** to each side, i.e. $\varphi_1 \_ \varphi_2$, and secondly, by requiring that $\alpha$ is a single non-terminal $X$ such that we have

$$\varphi_1 X \varphi_2 \to \varphi_1 \beta \varphi_2 \tag{17}$$

Now, if this context **may be *null*** (but does not have to be), we call this a **context sensitive** grammar (hence the name).

Chomsky (1959), p. 142.

# Definition: Alternative Version

Chomsky subsequently proves that it is possible to give an alternative definition of Type 1 languages by stating the original most general rule:

$$\alpha \rightarrow \beta, \tag{18}$$

with the additional condition that $\beta$ **is at least as long as** $\alpha$, i.e.

$$l(\alpha) \leq l(\beta), \tag{19}$$

where $l()$ is a function which gives the length in number of symbols. More precisely, Chomsky proves that this weakening of the original restriction ($\varphi_1 X \varphi_2 \rightarrow \varphi_1 \beta \varphi_2$) "will not increase the class of generated languages."[10]

We will work with this **alternative definition** in the lecture series.

Chomsky (1959), p. 145.
Jäger & Rogers (2012), p. 1957.

---

[10] Sometimes this latter definition of context-sensitive grammar in equation (19) and with the length restriction is referred to as *non-contracting grammar*.

# Abstract Example

| Rewrite | Rule |
|---|---|
| S | – |
| XY | 1 |
| acY | 3 |
| acbYd | 4 |
| acbbdd | 5 |
| abcbdd | 6 |
| abbcdd | 6 |

**Terminals**

$T = \{a, b, c\}$

**Non-Terminals**

$NT = \{S, X, Y\}$

***R***

1. S → XY
2. X → aXc
3. X → ac
4. Y → bYd
5. Y → bd
6. **cb** → **bc** (context-sensitive !)

The language generated with this grammar is $L(\mathcal{G}_{CS}) = \{abcd, abbcdd, aabbccdd, \ldots, a^m b^n c^m d^n\}$, with $n, m \in \mathbb{N}$. Note that $n = m$ is possible.

# Natural Language Examples

$\alpha \rightarrow \beta$ with $l(\alpha) \leq l(\beta)$

the → a ✓

the tree → this x

the huge tree → the tree x

the huge tree bends in the wind → the x

the → the huge tree bends in the wind ✓

VP → NP bends NP NP ✓

NP VP NP NP → NP VP x

DET → the ✓

the → DET ✓

NP → the N ✓

NP → DET N ✓

VP → NP VP ✓

# Further Examples of Context-Sensitive Languages

- *Copy language*: the set of strings $\omega = \gamma\gamma$ over a set of terminals $T$ such that $\omega$ consists of two identical halves, e.g. $L(\mathcal{G}_{CS}) = \{aa, abab, abcabc, abcdabcd, \dots\}$
- Languages with strings of the form $x^n y^n z^n$, e.g. $L(\mathcal{G}_{CS}) = \{abc, aabbcc, aaabbbccc, \dots\ a^n b^n c^n\}$
- Languages with strings of the form $x^n y^n z^n w^n v^n$, e.g. $L(\mathcal{G}_{CS}) = \{abcde, aabbccddee, \dots\ a^n b^n c^n d^n e^n\}$
- The set of all *prime numbers* (where each number $x$ is represented by a string of length $l(x)$).
- etc.

Jäger & Rogers (2012), p. 1958.

# Are Natural Languages **Context-Sensitive**?

Arguments such as the one by Shieber (1984) and Culy (1985) have led most FLT syntactians to assume that natural languages are at least **mildly context-sensitive**. There seem to be no generally accepted proposals of natural language structures which would require to go beyond the context-sensitive level.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

**Faculty of Philosophy**
General Linguistics

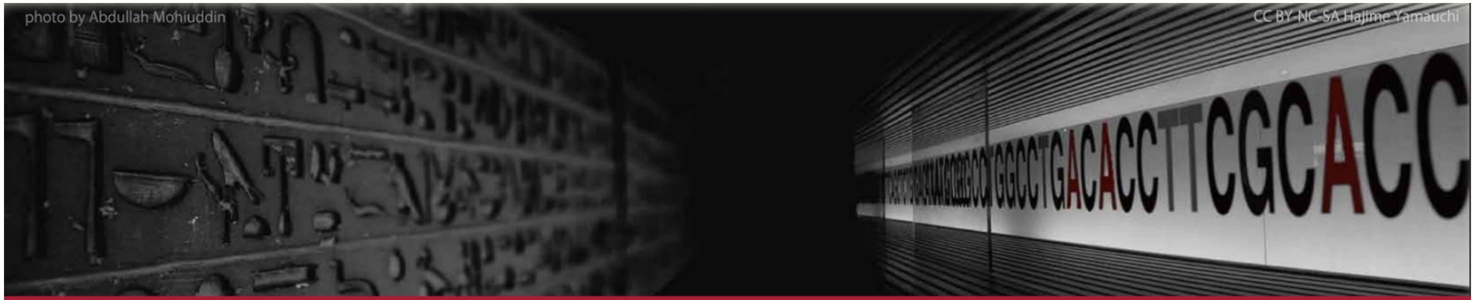


photo by Abdullah Mohiuddin

CC BY-NC-SA Hajime Yamauchi

# Recursively Enumerable Languages (Type 0)

# Definition: Type-0 Grammar

If the most general rewrite rule

$$\alpha \rightarrow \beta, \tag{20}$$

is not further restricted, we would call this a **Type-0 grammar**.

Chomsky (1959), p. 143.

# Natural Language Examples

$\alpha \rightarrow \beta$

the tree → this

the huge tree → the tree

the huge tree bends in the wind → the

NP VP NP NP → NP VP

If we really wanted to perform arbitrary transformations from any string $\alpha$ to any string $\beta$, then this would be a **Type-0 grammar**.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

**Faculty of Philosophy**
General Linguistics

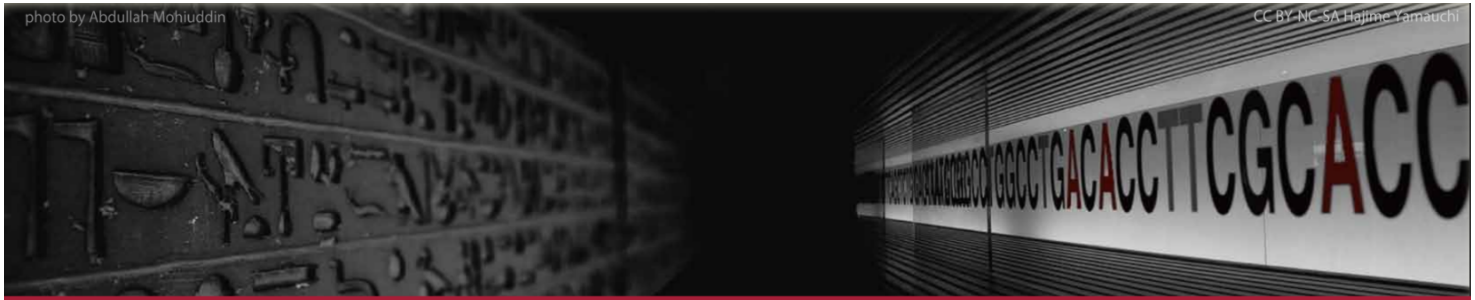
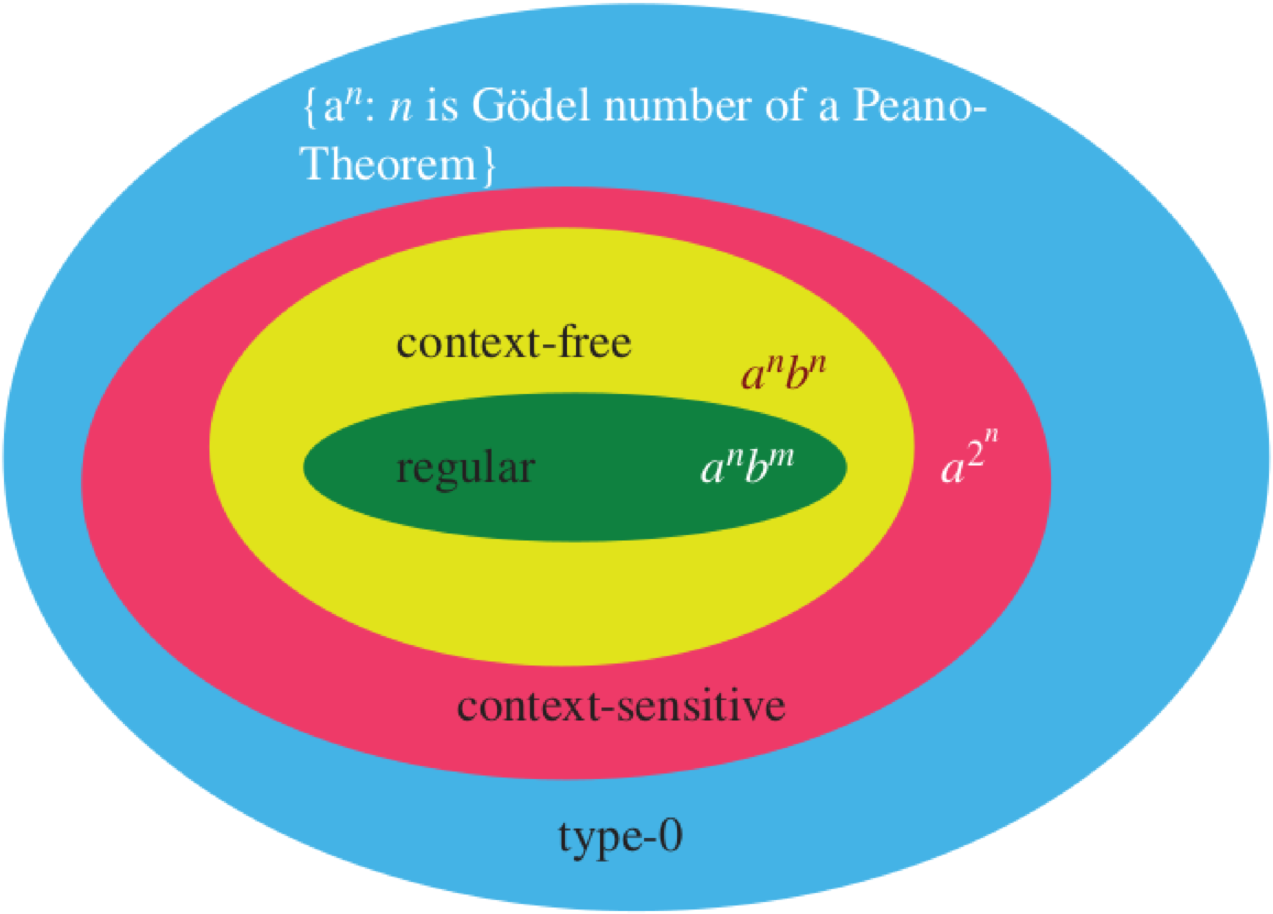


photo by Abdullah Mohiuddin
CC BY-NC-SA Hajime Yamauchi

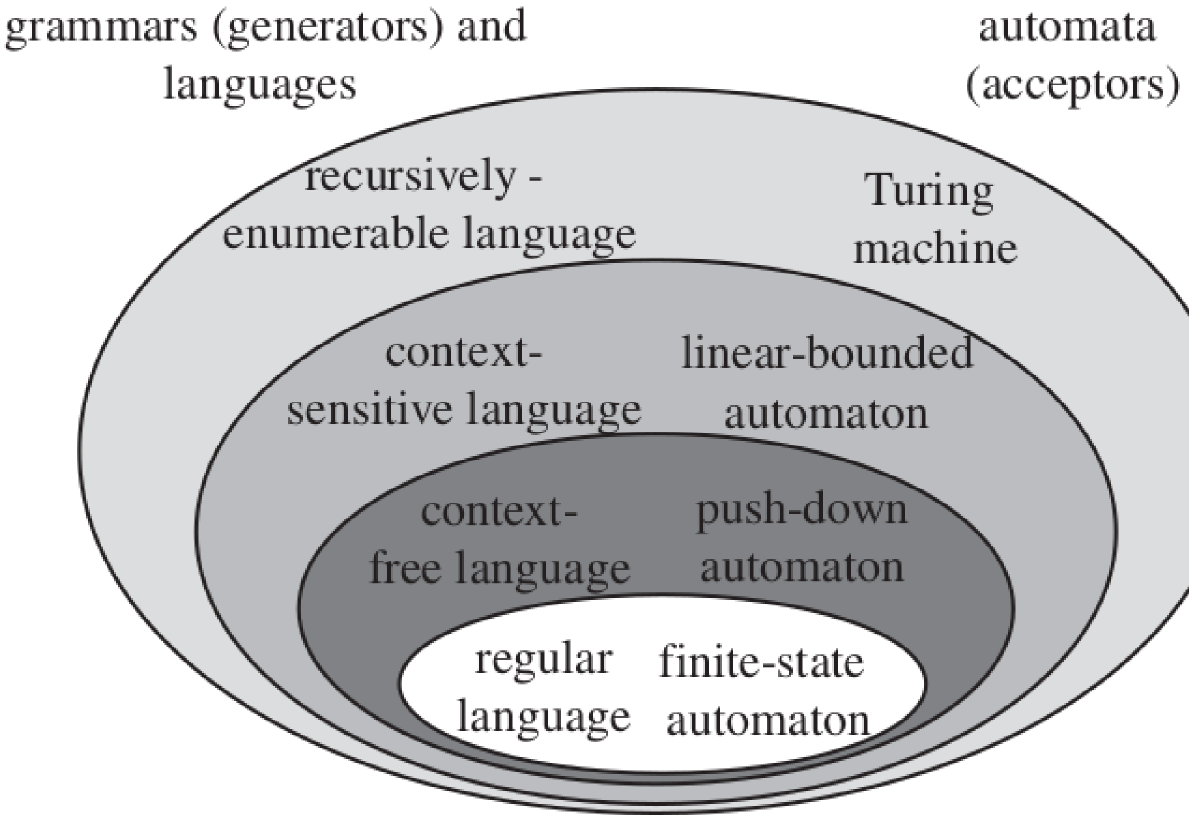
# The Classical Hierarchy

# The Classical Hierarchy



Jäger & Rogers (2012), p. 1959.

# The Classical Hierarchy



Fitch & Friederici (2012).

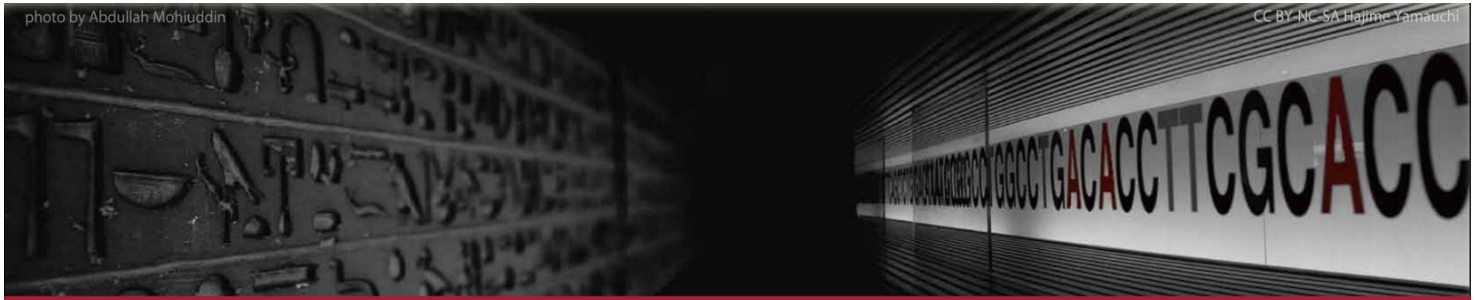


photo by Abdullah Mohiuddin

CC BY-NC-SA Hajime Yamauchi

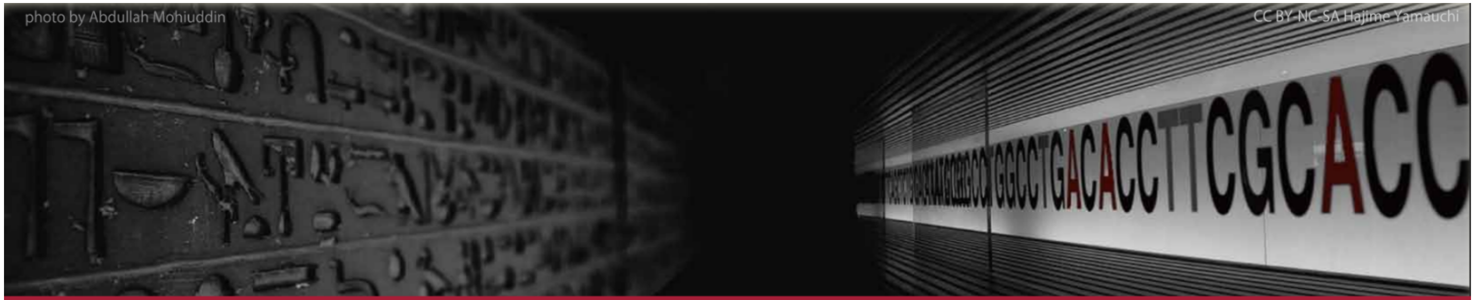
# Summary

# Summary

- FLT models language and other symbol systems based on a set of **terminals**, **non-terminals**, and **rewrite rules**.
- The complexity of the rewrite rules is captured on the **Chomsky hierarchy**, including **regular (type-3)**, **context-free (type-2)**, **context-sensitive (type-1)**, and **type-0**.
- Regular languages already have the capacity for **recursion** and **discrete infinity**.
- **Natural languages** seem to lie somewhere between context-free and context-sensitive, sometimes referred to as **mildly context-sensitive**.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

**Faculty of Philosophy**
General Linguistics



# References

# References

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions in Information Theory* 2, 113–124.

Chomsky, N. (1957). *Syntactic structures*. The Hague, The Netherlands: Mouton.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control* 2, 137–167.

Chomsky, N. & Schützenberger, M. P. (1965). The algebraic theory of context-free languages. *Studies in logic and the foundations of mathematics*. North Holland Publishing Company, Amsterdam.

Culy, C. (1985). The complexity of the vocabulary of Bambara. *Linguistics and Philosophy*, Vol. 8, No. 3, pp. 345-351.

Fitch, T. & Friederici, A. D. (2012). Artificial grammar learning meets formal language theory: an overview. *Philosophical Transactions of the Royal Society*, 367, pp. 1933-1955.

Jäger, Gerhard, and Rogers, James (2012). Formal language theory: refining the Chomsky hierarchy. *Philosophical Transactions of the Royal Society*, 367, pp. 1956-1970.

Shieber, Stuart (1984). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*.

# Thank You.

**Contact:**

Faculty of Philosophy
General Linguistics
Dr. Christian Bentz
SFS Keplerstraße 2, Room 168
chris@christianbentz.de

Office hours:
During term: Wednesdays 10-11am
Out of term: arrange via e-mail