TECHNISCHE UNIVERSITÄT DRESDEN

FORMAL LANGUAGES

WITH RESPECT TO THE CHOMSKY-SCHÜTZENBERGER-HIERARCHY

Eric Kunze

TU Dresden, January 27, 2020

CONTENTS

CONTENTS

SYNTAX VS. SEMANTIC

“Colorless green ideas sleep furiously.”

SYNTAX VS. SEMANTIC

"Colorless green ideas sleep furiously."

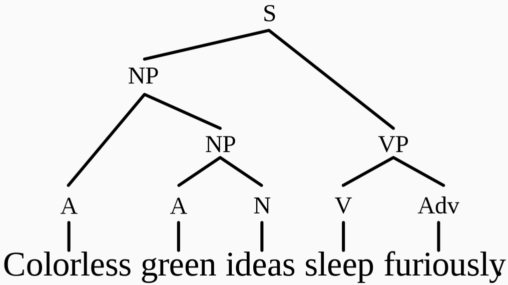

Figure 1: Tree representation of the sentence structure [4]

SYNTAX VS. SEMANTIC

“Colorless green ideas sleep furiously.”

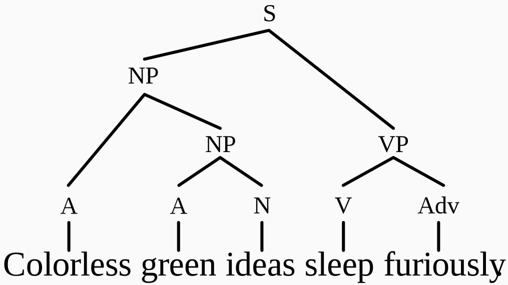

Figure 1: Tree representation of the sentence structure [4]

Noam Chomsky (* December 7, 1928)
“father of modern linguistics”

FORMAL LANGUAGE

i s o m o r p h i s m

FORMAL LANGUAGE

i s o m o r p h i s m

- **alphabet** Σ — set of symbols i s o m o r p h i s m

- **alphabet** Σ — set of symbols
- **word** w over Σ — finite sequence of symbols

FORMAL LANGUAGE

$$|\,\mathrm{i\ s\ o\ m\ o\ r\ p\ h\ i\ s\ m}\,| = 11$$

- **alphabet** Σ — set of symbols
- **word** w over Σ — finite sequence of symbols
- **length** $|w|$ — number of symbols of w

FORMAL LANGUAGE

i s o m o r p h i s m $\in \Sigma_{\text{lat}}^*$

- **alphabet** Σ — set of symbols
- **word** w over Σ — finite sequence of symbols
- **length** $|w|$ — number of symbols of w
- **Kleene-Star** Σ^* — set of all words over Σ

FORMAL LANGUAGE

$$\text{i s o m o r p h i s m} \in L_{\text{eng}} \subset \Sigma^*_{\text{lat}}$$

- **alphabet** Σ — set of symbols
- **word** w over Σ — finite sequence of symbols
- **length** $|w|$ — number of symbols of w
- **Kleene-Star** Σ^* — set of all words over Σ
- **(formal) language** $L \subseteq \Sigma^*$ — set of (selected) words over Σ

SIMPLE EXAMPLES

- $\mathbb{N}_0 = \{\varepsilon, 1, 11, 111, \ldots\}$ over $\Sigma = \{1\}$

SIMPLE EXAMPLES

- $\mathbb{N}_0 = \{\varepsilon, 1, 11, 111, \ldots\}$ over $\Sigma = \{1\}$
- $\mathsf{count}_2 = \{a^n b^n : n \in \mathbb{N}\}$ over $\Sigma = \{a, b\}$

SIMPLE EXAMPLES

- $\mathbb{N}_0 = \{\varepsilon, 1, 11, 111, \ldots\}$ over $\Sigma = \{1\}$
- $\text{count}_2 = \{a^n b^n : n \in \mathbb{N}\}$ over $\Sigma = \{a, b\}$
- $\text{count}_3 = \{a^n b^n c^n : n \in \mathbb{N}\}$ over $\Sigma = \{a, b, c\}$

SIMPLE EXAMPLES

- $\mathbb{N}_0 = \{\varepsilon, 1, 11, 111, \ldots\}$ over $\Sigma = \{1\}$
- $\text{count}_2 = \{a^n b^n : n \in \mathbb{N}\}$ over $\Sigma = \{a, b\}$
- $\text{count}_3 = \{a^n b^n c^n : n \in \mathbb{N}\}$ over $\Sigma = \{a, b, c\}$
- programming language *C*
  - alphabet: valid keywords and symbols
  - words: valid programs

CONTENTS

How to form strings over a given alphabet Σ?

FORMAL GRAMMAR

How to form strings over a given alphabet Σ?

- accept words — automata theory
- generate words — theory of grammar

FORMAL GRAMMAR

How to form strings over a given alphabet Σ?

- accept words — automata theory
- generate words — theory of grammar

Definition: Grammar

A (formal) grammar G is a 4-tuple (N, Σ, P, S) where

- N and Σ are finite, disjoint sets of symbols (alphabets)
- elements of N are called non-terminal symbols,
  elements of Σ are called terminal symbols
- P is a set of productions $(N \cup \Sigma)^* N (N \cup \Sigma)^* \rightarrow (N \cup \Sigma)^*$
- $S \in N$ is the start symbol

EXAMPLE

Example

$G = (N, \Sigma, P, S)$ where $N = \{S\}$, $\Sigma = \{a, b\}$ and
$P = \{S \to aSb, S \to ab\}$.

EXAMPLE

Example

$G = (N, \Sigma, P, S)$ where $N = \{S\}$, $\Sigma = \{a, b\}$ and
$P = \{S \to aSb, S \to ab\}$.

Produced words over Σ:

$\{ab, aabb, aaabbb, \ldots\} = \mathrm{count}_2$

CONTENTS

CHOMSKY-HIERARCHY [1]

Let $G = (N, \Sigma, P, S)$ be a formal grammar. Then G is a **type-0-grammar**.

CHOMSKY-HIERARCHY [1]

Let $G = (N, \Sigma, P, S)$ be a formal grammar. Then G is a **type-0-grammar**. G is called a

- **type-1-grammar** (*context-sensitive*) if every production has the form
  - $u_1 A u_2 \to u_1 w u_2$ where $A \in N$, $u_1, u_2, w \in (N \cup \Sigma)^*$ and $|w| \geq 1$ or
  - $S \to \varepsilon$

  Is $(S \to \varepsilon) \in P$, so S never appears on a production's right side.

CHOMSKY-HIERARCHY [1]

Let $G = (N, \Sigma, P, S)$ be a formal grammar. Then G is a **type-0-grammar**. G is called a

- **type-1-grammar** (*context-sensitive*) if every production has the form
  - $u_1 A u_2 \to u_1 w u_2$ where $A \in N$, $u_1, u_2, w \in (N \cup \Sigma)^*$ and $|w| \geq 1$ or
  - $S \to \varepsilon$

  Is $(S \to \varepsilon) \in P$, so S never appears on a production's right side.

Example: $L(G) = \text{count}_3$

Let $N = \{S, A, B\}$, $\Sigma = \{a, b, c\}$ and P consist of the following rules:

- (1) $S \to abc$
- (2) $S \to aSBc$
- (3) $cB \to Bc$
- (4) $bB \to bb$

CHOMSKY-HIERARCHY [1]

Let $G = (N, \Sigma, P, S)$ be a formal grammar. G is called a

- **type-2-grammar** (*context-free*) if every production has the form $A \rightarrow w$ where $A \in N$ and $w \in (N \cup \Sigma)^*$

CHOMSKY-HIERARCHY [1]

Let $G = (N, \Sigma, P, S)$ be a formal grammar. G is called a

- **type-2-grammar** (*context-free*) if every production has the form $A \to w$ where $A \in N$ and $w \in (N \cup \Sigma)^*$

Example: $L(G) = \text{count}_2$

Let $N = \{S\}$, $\Sigma = \{a, b\}$ and

$$P = \{S \to aSb, S \to ab\}$$

CHOMSKY-HIERARCHY [1]

Let $G = (N, \Sigma, P, S)$ be a formal grammar. G is called a

- **type-3-grammar** (*regular*) if every production has the form $A \to uB$ oder $A \to u$ where $A, B \in N$ and $u \in \Sigma^*$

Example: $L(G) = \mathbb{N}_0$

Let $N = \{S\}$, $\Sigma = \{1\}$,

$$P = \{S \to \varepsilon,\ S \to 1S\}$$

CONTENTS

APPLICATIONS

- **syntax of programming languages**
  e.g. `if`-condition in *C*

$$ifStat \rightarrow \texttt{if}\ (\ BoolExp\ )\ Stat$$
$$ifStat \rightarrow \texttt{if}\ (\ BoolExp\ )\ Stat\ \texttt{else}\ Stat$$

APPLICATIONS

- **syntax of programming languages**
  e.g. `if`-condition in *C*

$$\mathit{ifStat} \to \texttt{if}\ (\ \mathit{BoolExp}\ )\ \mathit{Stat}$$

$$\mathit{ifStat} \to \texttt{if}\ (\ \mathit{BoolExp}\ )\ \mathit{Stat}\ \texttt{else}\ \mathit{Stat}$$

- **mathematical logic**
  - (propositional) logic as formal language L_{log} over alphabet $\Sigma = \{p_1, p_2, \ldots\} \cup \{\neg, \wedge, \vee, (,)\}$
  - words: valid formulae — e.g. $(\neg(p_1 \vee p_2) \wedge p_3) \in L_{\text{log}}$

APPLICATIONS

- **syntax of programming languages**
  e.g. `if`-condition in *C*

$$ifStat \rightarrow \texttt{if} \; (\; BoolExp \;) \; Stat$$

$$ifStat \rightarrow \texttt{if} \; (\; BoolExp \;) \; Stat \; \texttt{else} \; Stat$$

- **mathematical logic**
  - (propositional) logic as formal language L_{log} over alphabet $\Sigma = \{p_1, p_2, \ldots\} \cup \{\neg, \wedge, \vee, (,)\}$
  - words: valid formulae — e.g. $(\neg(p_1 \vee p_2) \wedge p_3) \in L_{\text{log}}$
- **Computability Theory**
  - Which problem is computable with which machine?

REFERENCES

F. Baader.
Formale Systeme – Automatentheorie und Formale Sprachen.
Lecture notes, TU Dresden, Institut für theoretische Informatik, Dresden, Germany, 2016.

E. Börger.
***Computability, complexity, logic*.**
North Holland, Amsterdam, 1989.

N. Chomsky.
Three models for the description of language.
IRE Transactions on Information Theory, 2(3):113–124, Sep. 1956.

N. Chomsky.
***Syntactic structures*.**
Mouton, The Hague [u.a.], 11. pr. edition, 1975.

J. E. Hopcroft, R. Motwani, and J. D. Ullman.
***Introduction to Automata Theory, Languages, and Computation (3rd Edition)*.**
Addison-Wesley Longman Publishing Co., Inc., USA, 2006.

H. Rogers.
***Theory of recursive functions and effective computability*.**
MIT Press, Cambridge, Mass. [u.a.], 3. print. edition, 1992.

FORMAL LANGUAGE

- **alphabet** Σ — set of symbols
  - latin alphabet $\Sigma_{\text{lat}} = \{a, b, \ldots, z\}$
  - binary alphabet $\Sigma_{\text{bin}} = \{0, 1\}$

FORMAL LANGUAGE

- **alphabet** Σ — set of symbols
  - latin alphabet $\Sigma_{\text{lat}} = \{a, b, \ldots, z\}$
  - binary alphabet $\Sigma_{\text{bin}} = \{0, 1\}$
- **word** w over Σ — finite sequence of symbols
  - *english* and *sdlfkhui* are words over Σ_{lat}
  - 01101 is a word over Σ_{bin}, but not over Σ_{lat}

FORMAL LANGUAGE

- **alphabet** Σ — set of symbols
  - latin alphabet $\Sigma_{\text{lat}} = \{a, b, \dots, z\}$
  - binary alphabet $\Sigma_{\text{bin}} = \{0, 1\}$
- **word** w over Σ — finite sequence of symbols
  - *english* and *sdlfkhui* are words over Σ_{lat}
  - 01101 is a word over Σ_{bin}, but not over Σ_{lat}
- **length** $|w|$ — number of symbols of w
  - word with length zero: ε
  - $|english| = 7$

FORMAL LANGUAGE

- **alphabet** Σ — set of symbols
  - latin alphabet $\Sigma_{\text{lat}} = \{a, b, \ldots, z\}$
  - binary alphabet $\Sigma_{\text{bin}} = \{0, 1\}$
- **word** w over Σ — finite sequence of symbols
  - *english* and *sdlfkhui* are words over Σ_{lat}
  - 01101 is a word over Σ_{bin}, but not over Σ_{lat}
- **length** $|w|$ — number of symbols of w
  - word with length zero: ε
  - $|english| = 7$
- **Kleene-Star** Σ^* — set of all words over Σ

FORMAL LANGUAGE

- **alphabet** Σ — set of symbols
  - latin alphabet $\Sigma_{\text{lat}} = \{a, b, \ldots, z\}$
  - binary alphabet $\Sigma_{\text{bin}} = \{0, 1\}$
- **word** w over Σ — finite sequence of symbols
  - *english* and *sdlfkhui* are words over Σ_{lat}
  - 01101 is a word over Σ_{bin}, but not over Σ_{lat}
- **length** $|w|$ — number of symbols of w
  - word with length zero: ε
  - $|english| = 7$
- **Kleene-Star** Σ^* — set of all words over Σ
- **(formal) language** $L \subseteq \Sigma^*$ — set of (selected) words over Σ

EXAMPLE — count$_2$

Theorem

Let $G = (N, \Sigma, P, S)$ *where* $N = \{S\}$, $\Sigma = \{a, b\}$ *and*

$$P = \{ \underbrace{S \to aSb}_{(1)}, \underbrace{S \to ab}_{(2)} \}$$

Then $L(G) = \{a^n b^n : n \in \mathbb{N}\} = \mathrm{count}_2$.

Proof. We just prove that $\mathrm{count}_2 \subseteq L(G)$, i.e. $a^n b^n \in L(G)$ for all $n \in \mathbb{N}$.

- consideration:
  - $S \vdash_{(1)} aSb \vdash_{(2)} aabb = a^2b^2$
  - $S \vdash^2_{(1)} a^2Sb^2 \vdash_{(2)} a^2abb^2 = a^3b^3 \quad \ldots$
- in general:

$$S \vdash^{n-1}_{(1)} a^{n-1}Sb^{n-1} \vdash_{(2)} a^{n-1}abb^{n-1} = a^n b^n$$

EXAMPLE — count_3

Is there a grammer G with $L(G) = \mathrm{count}_3 = \{a^n b^n c^n : n \in \mathbb{N}\}$?

Theorem

Let $G = (N, \Sigma, P, S)$ be a formal grammar with $N = \{S, A, B\}$, $\Sigma = \{a, b, c\}$ and P consists of the following rules:

- (1) $S \to abc$
- (2) $S \to aSBc$
- (3) $cB \to Bc$
- (4) $bB \to bb$

Then $L(G) = \mathrm{count}_3$.

Example: $a^3 b^3 c^3 \in L(G)$?

$$
\begin{aligned}
S \quad &\vdash^2_{(2)} aaSBcBc && \vdash_{(1)} aaabcBcBc \\
&\vdash^2_{(3)} aaabBcBcc && \vdash^1_{(3)} aaabBBccc \\
&\vdash^2_{(4)} aaabbbccc
\end{aligned}
$$

MATHEMATICAL LOGIC

- (propositional) Logic as formal language L_{log}
- alphabet: $\Sigma = \{p_1, p_2, \ldots\} \cup \{\neg, \wedge, \vee, (,)\}$
- words:
  - $p_1, p_2, \cdots \in L_{\text{log}}$
  - $\neg f \in L_{\text{log}}$ if $f \in L_{\text{log}}$
  - $(f_1 \wedge f_2) \in L_{\text{log}}$ if $f_1, f_2 \in L_{\text{log}}$
  - $(f_1 \vee f_2) \in L_{\text{log}}$ if $f_1, f_2 \in L_{\text{log}}$
- examples:
  - $(p_1 \wedge p_2) \in L_{\text{log}}$
  - $((\neg(p_1 \vee p_2) \wedge p_3) \vee \neg p_1) \in L_{\text{log}}$

COMPUTABILITY THEORY

Question: Which problem is computable with which machine?

CHURCH-TURING thesis

Any real-world computation can be translated into an equivalent computation involving a Turing machine.

Halting Problem: Decide whether a given (arbitrary) program will finish running or continue to run forever