Formal Languages

Philippe de Groote

2020-2021

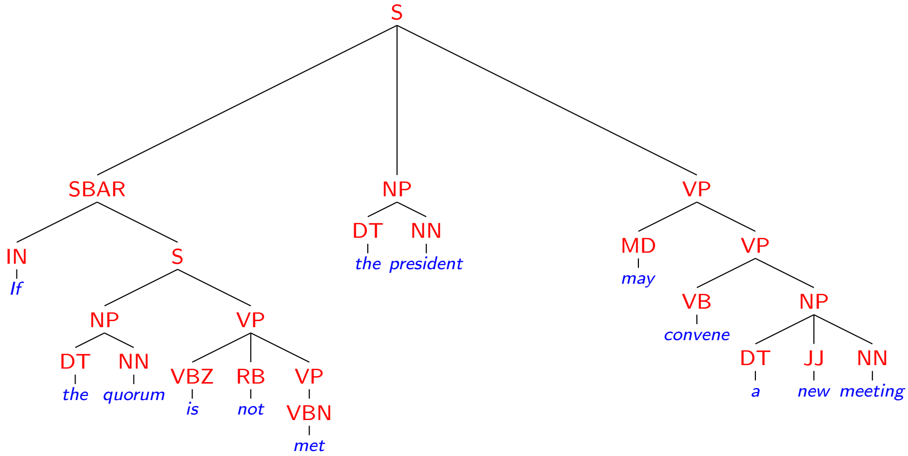
Outline

Introduction

If the quorum is not met, the president may convene a new meeting.

Introduction

If the quorum is not met, the president may convene a new meeting.



Introduction

$$
\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{NP\ VP} \\
\mathrm{S} &\rightarrow \mathrm{SBAR\ NP\ VP} \\
\mathrm{SBAR} &\rightarrow \mathrm{IN\ S} \\
\mathrm{VP} &\rightarrow \mathrm{VBN} \\
\mathrm{VP} &\rightarrow \mathrm{VB\ NP} \\
\mathrm{VP} &\rightarrow \mathrm{MD\ VP} \\
\mathrm{VP} &\rightarrow \mathrm{VBZ\ RB\ VP} \\
\mathrm{NP} &\rightarrow \mathrm{DT\ NN} \\
\mathrm{NP} &\rightarrow \mathrm{DT\ JJ\ NN} \\
\mathrm{VB} &\rightarrow \textit{convene} \\
\mathrm{VBN} &\rightarrow \textit{met} \\
\mathrm{VBZ} &\rightarrow \textit{is} \\
\mathrm{MD} &\rightarrow \textit{may} \\
\mathrm{NN} &\rightarrow \textit{quorum} \mid \textit{president} \mid \textit{meeting} \\
\mathrm{JJ} &\rightarrow \textit{new} \\
\mathrm{DT} &\rightarrow \textit{the} \mid \textit{a} \\
\mathrm{IN} &\rightarrow \textit{if} \\
\mathrm{RB} &\rightarrow \textit{not}
\end{aligned}
$$

Alphabets, words, and languages

- An *alphabet* is a finite, non-empty set of symbols.

Alphabets, words, and languages

- An *alphabet* is a finite, non-empty set of symbols.
- A *word* is a finite sequence of symbols chosen from some given alphabet.

Alphabets, words, and languages

- An *alphabet* is a finite, non-empty set of symbols.
- A *word* is a finite sequence of symbols chosen from some given alphabet.
- The *empty word*, denoted ϵ, is the empty sequence of symbols.

Alphabets, words, and languages

- An *alphabet* is a finite, non-empty set of symbols.
- A *word* is a finite sequence of symbols chosen from some given alphabet.
- The *empty word*, denoted ϵ, is the empty sequence of symbols.
- Let Σ be an alphabet. Σ^* denotes the set of all words over Σ.

Alphabets, words, and languages

- An *alphabet* is a finite, non-empty set of symbols.
- A *word* is a finite sequence of symbols chosen from some given alphabet.
- The *empty word*, denoted ϵ, is the empty sequence of symbols.
- Let Σ be an alphabet. Σ^* denotes the set of all words over Σ.
- Let $\alpha \in \Sigma^*$ be such that $\alpha = a_1 \ldots a_n$, with $a_i \in \Sigma$ for $1 \leq i \leq n$. The length of α, in notation $|\alpha|$, is the number of occurrences of symbols in α, i.e., $|\alpha| = n$.

Alphabets, words, and languages

- An *alphabet* is a finite, non-empty set of symbols.
- A *word* is a finite sequence of symbols chosen from some given alphabet.
- The *empty word*, denoted ϵ, is the empty sequence of symbols.
- Let Σ be an alphabet. Σ^* denotes the set of all words over Σ.
- Let $\alpha \in \Sigma^*$ be such that $\alpha = a_1 \dots a_n$, with $a_i \in \Sigma$ for $1 \leq i \leq n$. The length of α, in notation $|\alpha|$, is the number of occurrences of symbols in α, i.e., $|\alpha| = n$.
- In particular, $|\epsilon| = 0$.

Alphabets, words, and languages

- Let $\alpha, \beta \in \Sigma^*$ be such that $\alpha = a_1 \ldots a_n$, with $a_i \in \Sigma$ for $1 \leq i \leq n$, and $\beta = b_1 \ldots b_m$, with $b_i \in \Sigma$ for $1 \leq i \leq m$. $\alpha \cdot \beta$ is defined to be the *concatenation* of α and β, i.e., $\alpha \cdot \beta = a_1 \ldots a_n b_1 \ldots b_m$.

Alphabets, words, and languages

- Let $\alpha, \beta \in \Sigma^*$ be such that $\alpha = a_1 \ldots a_n$, with $a_i \in \Sigma$ for $1 \leq i \leq n$, and $\beta = b_1 \ldots b_m$, with $b_i \in \Sigma$ for $1 \leq i \leq m$. $\alpha \cdot \beta$ is defined to be the *concatenation* of α and β, i.e., $\alpha \cdot \beta = a_1 \ldots a_n b_1 \ldots b_m$.
- In particular, $\epsilon \cdot \alpha = \alpha = \alpha \cdot \epsilon$.

Alphabets, words, and languages

- Let $\alpha, \beta \in \Sigma^*$ be such that $\alpha = a_1 \ldots a_n$, with $a_i \in \Sigma$ for $1 \leq i \leq n$, and $\beta = b_1 \ldots b_m$, with $b_i \in \Sigma$ for $1 \leq i \leq m$. $\alpha \cdot \beta$ is defined to be the *concatenation* of α and β, i.e., $\alpha \cdot \beta = a_1 \ldots a_n b_1 \ldots b_m$.
- In particular, $\epsilon \cdot \alpha = \alpha = \alpha \cdot \epsilon$.
- $\langle \Sigma^*, \cdot, \epsilon \rangle$ is a monoid.

Alphabets, words, and languages

- Let $\alpha, \beta \in \Sigma^*$ be such that $\alpha = a_1 \ldots a_n$, with $a_i \in \Sigma$ for $1 \leq i \leq n$, and $\beta = b_1 \ldots b_m$, with $b_i \in \Sigma$ for $1 \leq i \leq m$. $\alpha \cdot \beta$ is defined to be the *concatenation* of α and β, i.e., $\alpha \cdot \beta = a_1 \ldots a_n b_1 \ldots b_m$.
- In particular, $\epsilon \cdot \alpha = \alpha = \alpha \cdot \epsilon$.
- $\langle \Sigma^*, \cdot, \epsilon \rangle$ is a monoid.
- A *language* over Σ is a subset of Σ^*.

Alphabets, words, and languages

Operation on languages:

Alphabets, words, and languages

Operation on languages:

- Usual set-theoretic operations: *union, intersection, complement* (w.r.t. Σ^*), ...

Alphabets, words, and languages

Operation on languages:

- Usual set-theoretic operations: *union, intersection, complement* (w.r.t. Σ^*), ...
- *Concatenation*:

$$L_1 \cdot L_2 = \{\omega : \exists\alpha \in L_1.\exists\beta \in L_2.\omega = \alpha \cdot \beta\}$$

Alphabets, words, and languages

Operation on languages:

- Usual set-theoretic operations: *union, intersection, complement* (w.r.t. Σ^*), ...
- *Concatenation*:

$$L_1 \cdot L_2 = \{\omega : \exists \alpha \in L_1 . \exists \beta \in L_2 . \omega = \alpha \cdot \beta\}$$

- *Exponentiation*:

$$\begin{aligned} L^0 &= \{\epsilon\} \\ L^{n+1} &= L \cdot L^n \end{aligned}$$

Alphabets, words, and languages

Operation on languages:

- Usual set-theoretic operations: *union, intersection, complement* (w.r.t. Σ^*), ...
- *Concatenation*:

$$L_1 \cdot L_2 = \{\omega : \exists\alpha \in L_1.\exists\beta \in L_2.\omega = \alpha \cdot \beta\}$$

- *Exponentiation*:

$$\begin{aligned} L^0 &= \{\epsilon\} \\ L^{n+1} &= L \cdot L^n \end{aligned}$$

- *Closure*:

$$L^* = \bigcup_{i \in \mathbb{N}} L^i$$

Definition of a phrase structure grammar

A *phrase structure grammar* is a 4-tuple $G = \langle N, \Sigma, P, S\rangle$, where

- N is an alphabet, the elements of which are called *non-terminal symbols*;
- Σ is an alphabet disjoint from N, the elements of which are called *terminal symbols*;
- $P \subset ((N \cup \Sigma)^* N (N \cup \Sigma)^*) \times (N \cup \Sigma)^*$ is a set of *production rules*;
- $S \in N$ is called the *start symbol*.

Definition of a phrase structure grammar

A *phrase structure grammar* is a 4-tuple $G = \langle N, \Sigma, P, S \rangle$, where

- N is an alphabet, the elements of which are called *non-terminal symbols*;
- Σ is an alphabet disjoint from N, the elements of which are called *terminal symbols*;
- $P \subset ((N \cup \Sigma)^* N (N \cup \Sigma)^*) \times (N \cup \Sigma)^*$ is a set of *production rules*;
- $S \in N$ is called the *start symbol*.

A production rule $(\alpha, \beta) \in P$ will be written as $\alpha \to \beta$.

Definition of a phrase structure grammar

Let $G = \langle N, \Sigma, P, S \rangle$ be a phrase structure grammar, and let $\alpha, \beta \in (N \cup \Sigma)^*$. We say that α *directly generates* β, and we write $\alpha \Rightarrow \beta$, if and only if there exist $\alpha_0, \beta_0, \gamma, \delta \in (N \cup \Sigma)^*$ such that:

$$\alpha = \gamma\alpha_0\delta \text{ and } \beta = \gamma\beta_0\delta \text{ and } (\alpha_0 \rightarrow \beta_0) \in P$$

Definition of a phrase structure grammar

Let $G = \langle N, \Sigma, P, S \rangle$ be a phrase structure grammar, and let $\alpha, \beta \in (N \cup \Sigma)^*$. We say that α *directly generates* β, and we write $\alpha \Rightarrow \beta$, if and only if there exist $\alpha_0, \beta_0, \gamma, \delta \in (N \cup \Sigma)^*$ such that:

$$\alpha = \gamma\alpha_0\delta \text{ and } \beta = \gamma\beta_0\delta \text{ and } (\alpha_0 \to \beta_0) \in P$$

We write $\Rightarrow^*$ for the reflexive-transitive closure of $\Rightarrow$.

Definition of a phrase structure grammar

Let $G = \langle N, \Sigma, P, S\rangle$ be a phrase structure grammar, and let $\alpha, \beta \in (N \cup \Sigma)^*$. We say that α *directly generates* β, and we write $\alpha \Rightarrow \beta$, if and only if there exist $\alpha_0, \beta_0, \gamma, \delta \in (N \cup \Sigma)^*$ such that:

$$\alpha = \gamma\alpha_0\delta \text{ and } \beta = \gamma\beta_0\delta \text{ and } (\alpha_0 \to \beta_0) \in P$$

We write $\Rightarrow^*$ for the reflexive-transitive closure of $\Rightarrow$.

Let $G = \langle N, \Sigma, P, S\rangle$ be a phrase structure grammar. The *language generated by* G, in notation $L(G)$, is defined as follows:

$$L(G) = \{\alpha \in \Sigma^* : S \Rightarrow^* \alpha\}$$

Chomsky hierarchy

type	*rules*	*languages*	*automata*
0	$\alpha A \beta \rightarrow \gamma$	*recursively enumerable*	Turing machines
1	$\alpha A \beta \rightarrow \alpha \delta \beta$	*context-sensitive*	linear bounded Turing machines
2	$A \rightarrow \alpha$	*context-free*	pushdown automata
3	$A \rightarrow aB$ or $A \rightarrow b$	*regular*	finite state automata

where:

$\alpha, \beta, \gamma, \delta \in (N \cup \Sigma)^*$;

$\delta \neq \epsilon$;

$A, B \in N$;

$a \in \Sigma$;

$b \in \Sigma \cup \{\epsilon\}$;

Decision problems

Let $G = \langle N, \Sigma, P, S \rangle$ be a phrase structure grammar.

Decision problems

Let $G = \langle N, \Sigma, P, S \rangle$ be a phrase structure grammar.

- Emptiness:

  Is $L(G)$ different from the empty set?

Decision problems

Let $G = \langle N, \Sigma, P, S \rangle$ be a phrase structure grammar.

- Emptiness:

  Is $L(G)$ different from the empty set?

- Membership:

  Let $\alpha \in \Sigma^*$. Does α belong $L(G)$?

Exercices

1. Write a phrase structure grammar that generates $\{(ab)^n : n \geq 0\}$
2. Write a phrase structure grammar that generates $\{(ab)^n : n \geq 1\}$
3. Write a phrase structure grammar that generates $\{a^n b^n : n \geq 1\}$
4. Write a phrase structure grammar that generates $\{a^n b^n c^n : n \geq 1\}$
5. Write a type-3 grammar that generates $\{(ab)^n : n \geq 1\}$
6. Write a type-2 grammar that generates $\{a^n b^n : n \geq 1\}$
7. Write a type-1 grammar that generates $\{a^n b^n c^n : n \geq 1\}$