Computación y Estructuras Discretas III

Andrés A. Aristizábal P.
aaaristizabal@icesi.edu.co

Departamento de Computación y Sistemas Inteligentes

UNIVERSIDAD ICESI

2024-2

Agenda del día

1. **Languages and context-free grammars**
   - Introduction
   - CFG
   - Exercises

Languages and context-free grammars

Let's recap, what do automata do?

Languages and context-free grammars

Let's recap, what do automata do?

They process strings.

Languages and context-free grammars

Let's recap, what do automata do?

They process strings.

And now, what do grammars do?

Languages and context-free grammars

Let's recap, what do automata do?

They process strings.

And now, what do grammars do?

They generate strings from an initial symbol.

Languages and context-free grammars

Let's recap, what do automata do?

They process strings.

And now, what do grammars do?

They generate strings from an initial symbol.

What is a generative grammar?

Languages and context-free grammars

Let's recap, what do automata do?

They process strings.

And now, what do grammars do?

They generate strings from an initial symbol.

What is a generative grammar?

It is a model for the description of natural languages (Spanish, English, French, etc.).

Languages and context-free grammars

Let's recap, what do automata do?

They process strings.

And now, what do grammars do?

They generate strings from an initial symbol.

What is a generative grammar?

It is a model for the description of natural languages (Spanish, English, French, etc.).

And its formal definition?

Languages and context-free grammars

Let's recap, what do automata do?

They process strings.

And now, what do grammars do?

They generate strings from an initial symbol.

What is a generative grammar?

It is a model for the description of natural languages (Spanish, English, French, etc.).

And its formal definition?

Definition

A generative grammar is a quadruple, $G = (V, \Sigma, S, P)$ formed by two disjoint alphabets V (alphabet of variables or non-terminals) and Σ (alphabet of terminals), a special variable $S \in V$ (called the initial symbol) and a finite set $P \subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ of production or rewriting rules.

But, what is a production?

But, what is a production?

Definition

A production $(v, w) \in P$ *is denoted by* $v \to w$ *and read as "*v *produces* w*";* v *s called the head and* w *the body of the production. It is required that the head of the production has at least one variable.*

But, what is a production?

Definition

A production $(v, w) \in P$ *is denoted by* $v \to w$ *and read as "*v *produces* w*";* v *s called the head and* w *the body of the production. It is required that the head of the production has at least one variable.*

What is the meaning of the production $v \to w$?

But, what is a production?

Definition

A production $(v, w) \in P$ is denoted by $v \to w$ and read as "v produces w"; v s called the head and w the body of the production. It is required that the head of the production has at least one variable.

What is the meaning of the production $v \to w$?

The string v can be replaced by the string w. Starting with the initial symbol S and applying the productions of the grammar, in one or more steps, strings of terminals and/or non-terminals are obtained.

But, what is a production?

Definition

A production $(v, w) \in P$ *is denoted by* $v \to w$ *and read as "v produces w"; v s called the head and w the body of the production. It is required that the head of the production has at least one variable.*

What is the meaning of the production $v \to w$?

The string v can be replaced by the string w. Starting with the initial symbol S and applying the productions of the grammar, in one or more steps, strings of terminals and/or non-terminals are obtained. Those strings that only contain terminals form what is called the language generated by G.

Languages and context-free grammars

Grammars are classified according to the type of their productions.

Languages and context-free grammars

Grammars are classified according to the type of their productions.

Grammars are classified according to the type of their productions.

Languages and context-free grammars

Grammars are classified according to the type of their productions.

Grammars are classified according to the type of their productions.

Type 0 Grammars: They have no restrictions. They are also called unrestricted grammars.

Grammars are classified according to the type of their productions.

Grammars are classified according to the type of their productions.

Type 0 Grammars: They have no restrictions. They are also called unrestricted grammars.

Type 1 Grammars: The productions are of the form $u_1 A u_2 \to v_1 w v_2$, where A is a variable and $w \neq \lambda$. They are also called context-sensitive grammars.

Languages and context-free grammars

Grammars are classified according to the type of their productions.

Grammars are classified according to the type of their productions.

Type 0 Grammars: They have no restrictions. They are also called unrestricted grammars.

Type 1 Grammars: The productions are of the form $u_1 A u_2 \to v_1 w v_2$, where A is a variable and $w \neq \lambda$. They are also called context-sensitive grammars.

Type 2 Grammars: The productions are of the form $A \to w$ where A is a variable. They are also called context-free grammars.

Grammars are classified according to the type of their productions.

Grammars are classified according to the type of their productions.

Type 0 Grammars: They have no restrictions. They are also called unrestricted grammars.

Type 1 Grammars: The productions are of the form $u_1 A u_2 \to v_1 w v_2$, where A is a variable and $w \neq \lambda$. They are also called context-sensitive grammars.

Type 2 Grammars: The productions are of the form $A \to w$ where A is a variable. They are also called context-free grammars.

Type 3 Grammars: The productions are of the form $A \to a$ or of the form $A \to aB$, where A and B are variables and a is a terminal symbol. They are also called regular grammars.

Languages and context-free grammars

Grammars are classified according to the type of their productions.

Grammars are classified according to the type of their productions.

Type 0 Grammars: They have no restrictions. They are also called unrestricted grammars.

Type 1 Grammars: The productions are of the form $u_1 A u_2 \to v_1 w v_2$, where A is a variable and $w \neq \lambda$. They are also called context-sensitive grammars.

Type 2 Grammars: The productions are of the form $A \to w$ where A is a variable. They are also called context-free grammars.

Type 3 Grammars: The productions are of the form $A \to a$ or of the form $A \to aB$, where A and B are variables and a is a terminal symbol. They are also called regular grammars.

A language is said to be of type i f it is generated by a type i grammar.

Languages and context-free grammars

Grammars are classified according to the type of their productions.

Grammars are classified according to the type of their productions.

Type 0 Grammars: They have no restrictions. They are also called unrestricted grammars.

Type 1 Grammars: The productions are of the form $u_1 A u_2 \to v_1 w v_2$, where A is a variable and $w \neq \lambda$. They are also called context-sensitive grammars.

Type 2 Grammars: The productions are of the form $A \to w$ where A is a variable. They are also called context-free grammars.

Type 3 Grammars: The productions are of the form $A \to a$ or of the form $A \to aB$, where A and B are variables and a is a terminal symbol. They are also called regular grammars.

A language is said to be of type i f it is generated by a type i grammar. This classification of languages is known as Chomsky's hierarchy.

What is a context-free grammar?

What is a context-free grammar?

Definition

A context-free grammar (CFG) or type 2 grammar, is a quadruple, $G = (V, \Sigma, S, P)$ *formed by:*

What is a context-free grammar?

Definition

A context-free grammar (CFG) or type 2 grammar, is a quadruple, $G = (V, \Sigma, S, P)$ *formed by:*

1. *An alphabet* V *whose elements are called variables or non-terminal symbols.*

What is a context-free grammar?

Definition

A context-free grammar (CFG) or type 2 grammar, is a quadruple, $G = (V, \Sigma, S, P)$ formed by:

1. *An alphabet V whose elements are called variables or non-terminal symbols.*
2. *An alphabet Σ whose elements are called terminal symbols. It is required that the alphabets Σ and V be disjoint.*

What is a context-free grammar?

Definition

A context-free grammar (CFG) or type 2 grammar, is a quadruple, $G = (V, \Sigma, S, P)$ formed by:

1. *An alphabet V whose elements are called variables or non-terminal symbols.*
2. *An alphabet Σ whose elements are called terminal symbols. It is required that the alphabets Σ and V be disjoint.*
3. *A special variable $S \in V$, called the initial symbol of the grammar.*

Languages and context-free grammars

What is a context-free grammar?

Definition

A context-free grammar (CFG) or type 2 grammar, is a quadruple, $G = (V, \Sigma, S, P)$ formed by:

1. *An alphabet V whose elements are called variables or non-terminal symbols.*
2. *An alphabet Σ whose elements are called terminal symbols. It is required that the alphabets Σ and V be disjoint.*
3. *A special variable $S \in V$, called the initial symbol of the grammar.*
4. *A finite set $P \subseteq V \times (V \cup \Sigma)^*$ of productions or rewriting rules. A production $(A, w) \in P$ de G is denoted by $A \to w$ and read as "A produces w"; its meaning is: the variable A can be replaced (overwritten) by the string w. In the production $A \to w$, A is called the head and w the body of the production.*

Some important notations:

Some important notations:

1. Variables are denoted by uppercase letters A, B, C, ...

Some important notations:

1. Variables are denoted by uppercase letters A, B, C, ...
2. Elements of Σ or terminal symbols are denoted by lowercase letters a, b, c, ...

Some important notations:

1. Variables are denoted by uppercase letters A, B, C, ...
2. Elements of Σ or terminal symbols are denoted by lowercase letters a, b, c, ...
3. If $u, v \in (V \cup \Sigma)^*$ and $A \to w$ is a production, it is said that uwv is derived directly from uAv, which is denoted by $uAv \Longrightarrow uwv$.

Languages and context-free grammars

Some important notations:

1. Variables are denoted by uppercase letters A, B, C, ...
2. Elements of Σ or terminal symbols are denoted by lowercase letters a, b, c, ...
3. If $u, v \in (V \cup \Sigma)^*$ and $A \to w$ is a production, it is said that uwv is derived directly from uAv, which is denoted by $uAv \Longrightarrow uwv$.
4. If reference is to be made to grammar G, it is written $uAv \overset{G}{\Longrightarrow} uwv$ or $uAv \Longrightarrow_G uwv$.

Languages and context-free grammars

Some important notations:

1. Variables are denoted by uppercase letters A, B, C, ...
2. Elements of Σ or terminal symbols are denoted by lowercase letters a, b, c, ...
3. If $u, v \in (V \cup \Sigma)^*$ and $A \to w$ is a production, it is said that uwv is derived directly from uAv, which is denoted by $uAv \Longrightarrow uwv$.
4. If reference is to be made to grammar G, it is written $uAv \overset{G}{\Longrightarrow} uwv$ or $uAv \Longrightarrow_G uwv$.
5. If $u_1, u_2, ..., u_n$ are strings in $(V \cup \Sigma)^*$ and there is a sequence of direct derivations $u_1 \overset{G}{\Longrightarrow} u_2$, $u_2 \overset{G}{\Longrightarrow} u_3$, ... , $u_{n-1} \overset{G}{\Longrightarrow} u_n$ it is said that u_n is derived from u_1 and written $u_1 \overset{*}{\Longrightarrow} u_n$. The aforementioned sequence of direct derivations is represented as $u_1 \Longrightarrow u_2 \Longrightarrow u_3 \Longrightarrow \cdots \Longrightarrow u_{n-1} \Longrightarrow u_n$ and is called a derivation or generation of u_n from u_1.

More notations:

Languages and context-free grammars

More notations:

1. For every string w it is assumed that $w \stackrel{*}{\Longrightarrow} w$; therefore, $u \stackrel{*}{\Longrightarrow} v$ means that v is obtained from u using zero, one, or more productions of the grammar.

Languages and context-free grammars

More notations:

1. For every string w it is assumed that $w \stackrel{*}{\Longrightarrow} w$; therefore, $u \stackrel{*}{\Longrightarrow} v$ means that v is obtained from u using zero, one, or more productions of the grammar.
2. $u \stackrel{+}{\Longrightarrow} v$ means tha v is obtained from u using one or more productions of the grammar.

More notations:

1. For every string w it is assumed that $w \stackrel{*}{\Longrightarrow} w$; therefore, $u \stackrel{*}{\Longrightarrow} v$ means that v is obtained from u using zero, one, or more productions of the grammar.
2. $u \stackrel{+}{\Longrightarrow} v$ means tha v is obtained from u using one or more productions of the grammar.
3. The language generated by a grammar G is denoted by $L(G)$ and is defined as $L(G) := \{w \in \Sigma \mid S \stackrel{+}{\Longrightarrow} w\}$.

Languages and context-free grammars

More notations:

1. For every string w it is assumed that $w \stackrel{*}{\Longrightarrow} w$; therefore, $u \stackrel{*}{\Longrightarrow} v$ means that v is obtained from u using zero, one, or more productions of the grammar.
2. $u \stackrel{+}{\Longrightarrow} v$ means tha v is obtained from u using one or more productions of the grammar.
3. The language generated by a grammar G is denoted by $L(G)$ and is defined as $L(G) := \{w \in \Sigma \,|\, S \stackrel{+}{\Longrightarrow} w\}$.
4. A langauge L over an alphabet Σ is said to be a context-free language (CFL) if there exists a CFG G such that $L(G) = L$.

Languages and context-free grammars

More notations:

1. For every string w it is assumed that $w \stackrel{*}{\Longrightarrow} w$; therefore, $u \stackrel{*}{\Longrightarrow} v$ means that v is obtained from u using zero, one, or more productions of the grammar.
2. $u \stackrel{+}{\Longrightarrow} v$ means tha v is obtained from u using one or more productions of the grammar.
3. The language generated by a grammar G is denoted by $L(G)$ and is defined as $L(G) := \{w \in \Sigma \,|\, S \stackrel{+}{\Longrightarrow} w\}$.
4. A langauge L over an alphabet Σ is said to be a context-free language (CFL) if there exists a CFG G such that $L(G) = L$.
5. Two CFGs G_1 and G_2 are equivalent if $L(G_1) = L(G_2)$.

Agenda del día

1. **Languages and context-free grammars**
   - Introduction
   - CFG
   - Exercises

Example

Let G be a grammar (V, Σ, S, P) given by:

Example

Let G be a grammar (V, Σ, S, P) *given by:*

$$V = \{S\}$$

Example

Let G be a grammar (V, Σ, S, P) *given by:*

$$V = \{S\}$$
$$\Sigma = \{a\}$$

Example

Let G be a grammar (V, Σ, S, P) *given by:*

$$V = \{S\}$$
$$\Sigma = \{a\}$$
$$P = \{S \to aS, S \to \lambda\}$$

Example

Let G be a grammar (V, Σ, S, P) given by:

$$V = \{S\}$$
$$\Sigma = \{a\}$$
$$P = \{S \rightarrow aS, S \rightarrow \lambda\}$$

We have $S \rightarrow \lambda$ and

Example

Let G be a grammar (V, Σ, S, P) *given by:*

$$\begin{aligned} V &= \{S\} \\ \Sigma &= \{a\} \\ P &= \{S \to aS, S \to \lambda\} \end{aligned}$$

We have $S \to \lambda$ *and*

$$S \Longrightarrow aS \stackrel{*}{\Longrightarrow} a \cdots aS \Longrightarrow a \cdots a.$$

Example

Let G be a grammar (V, Σ, S, P) given by:

$$\begin{aligned} V &= \{S\} \\ \Sigma &= \{a\} \\ P &= \{S \to aS, S \to \lambda\} \end{aligned}$$

We have $S \to \lambda$ and

$$S \Longrightarrow aS \stackrel{*}{\Longrightarrow} a \cdots aS \Longrightarrow a \cdots a.$$

Therefore, $L(G) = a^*$.

Languages and context-free grammars

Example

Find a CFG that generates the language $0^*10^*10^*$ *over* $\Sigma = \{0, 1\}$, *i.e., the language of all strings with exactly two ones.*

Example

Find a CFG that generates the language $0^*10^*10^*$ *over* $\Sigma = \{0, 1\}$, *i.e., the language of all strings with exactly two ones.*

Solution

$$G : \left\{ \begin{array}{l} S \to A1A1A \\ A \to 0A \mid \lambda \end{array} \right.$$

Example

Find a CFG that generates the language $0^*10^*10^*$ *over* $\Sigma = \{0, 1\}$, *i.e., the language of all strings with exactly two ones.*

Solution

$$G : \begin{cases} S \to A1A1A \\ A \to 0A \mid \lambda \end{cases}$$

An equivalent grammar is

Example

Find a CFG that generates the language $0^*10^*10^*$ *over* $\Sigma = \{0, 1\}$, *i.e., the language of all strings with exactly two ones.*

Solution

$$G : \begin{cases} S \to A1A1A \\ A \to 0A \mid \lambda \end{cases}$$

An equivalent grammar is

$$\begin{cases} S \to 0S \mid 1A \\ A \to 0A \mid 1B \\ B \to 0B \mid \lambda \end{cases}$$

Example

Find a CFG that generates the language $L = \{a^i b^i \mid i \geq 0\}$ *over* $\Sigma = \{a, b\}$, *which is not a regular language.*

Example

Find a CFG that generates the language $L = \{a^i b^i \mid i \geq 0\}$ *over* $\Sigma = \{a, b\}$, *which is not a regular language.*

Solution

$$S \rightarrow aSb \mid \lambda.$$

What is a rooted tree?

What is a rooted tree?

1. It is a very particular type of undirected graph.

What is a rooted tree?

1. It is a very particular type of undirected graph.
2. It consists of a set of vertices or nodes connected by edges with the following property: there is a special node, called the root of the tree, such that there is a unique path between any node and the root.

What is a rooted tree?

1. It is a very particular type of undirected graph.
2. It consists of a set of vertices or nodes connected by edges with the following property: there is a special node, called the root of the tree, such that there is a unique path between any node and the root.
3. Thus, the root branches out into nodes, called immediate descendants, each of which may have, in turn, immediate descendants, and so on.

What other characteristics should we know about a rooted tree?

What other characteristics should we know about a rooted tree?

1. The property that characterizes a tree guarantees that a node can have 0, 1, or more immediate descendants but only one immediate ancestor.

What other characteristics should we know about a rooted tree?

1. The property that characterizes a tree guarantees that a node can have 0, 1, or more immediate descendants but only one immediate ancestor.
2. The only node that has no ancestors is the root.

What other characteristics should we know about a rooted tree?

1. The property that characterizes a tree guarantees that a node can have 0, 1, or more immediate descendants but only one immediate ancestor.
2. The only node that has no ancestors is the root.
3. Nodes that have descendants, except the root, are called interior nodes.

Languages and context-free grammars

What other characteristics should we know about a rooted tree?

1. The property that characterizes a tree guarantees that a node can have 0, 1, or more immediate descendants but only one immediate ancestor.
2. The only node that has no ancestors is the root.
3. Nodes that have descendants, except the root, are called interior nodes.
4. Nodes that have no descendants are called leaves of the tree.

What other characteristics should we know about a rooted tree?

1. The property that characterizes a tree guarantees that a node can have 0, 1, or more immediate descendants but only one immediate ancestor.
2. The only node that has no ancestors is the root.
3. Nodes that have descendants, except the root, are called interior nodes.
4. Nodes that have no descendants are called leaves of the tree.
5. The number of vertices and the number of edges of a tree can be infinite.

Languages and context-free grammars

How is the tree of a derivation $S \overset{*}{\Longrightarrow} w,\ w \in \Sigma^*$, constructed in a CFG?

How is the tree of a derivation $S \overset{*}{\Longrightarrow} w,\ w \in \Sigma^*$, constructed in a CFG?

The rooted tree is formed as follows:

Languages and context-free grammars

How is the tree of a derivation $S \overset{*}{\Longrightarrow} w,\ w \in \Sigma^*$, constructed in a CFG?

The rooted tree is formed as follows:

1. The root is labeled with the start symbol S.

Languages and context-free grammars

How is the tree of a derivation $S \overset{*}{\Longrightarrow} w,\ w \in \Sigma^*$, constructed in a CFG?

The rooted tree is formed as follows:

1. The root is labeled with the start symbol S.
2. Each interior node is labeled with a non-terminal symbol.

Languages and context-free grammars

How is the tree of a derivation $S \overset{*}{\Longrightarrow} w,\ w \in \Sigma^*$, constructed in a CFG?

The rooted tree is formed as follows:

1. The root is labeled with the start symbol S.
2. Each interior node is labeled with a non-terminal symbol.
3. Each leaf is labeled with a terminal symbol or with λ.

Languages and context-free grammars

How is the tree of a derivation $S \stackrel{*}{\Longrightarrow} w,\ w \in \Sigma^*$, constructed in a CFG?

The rooted tree is formed as follows:

1. The root is labeled with the start symbol S.
2. Each interior node is labeled with a non-terminal symbol.
3. Each leaf is labeled with a terminal symbol or with λ.
4. If in the derivation the production $A \rightarrow s_1 s_2 \cdots s_k$, is used where $s_i \in (V \cup \Sigma)^*$, the node A has k immediate descendants: s_1, s_2, ..., s_k, written form left to right.

Languages and context-free grammars

How is the tree of a derivation $S \stackrel{*}{\Longrightarrow} w$, $w \in \Sigma^*$, constructed in a CFG?

The rooted tree is formed as follows:

1. The root is labeled with the start symbol S.
2. Each interior node is labeled with a non-terminal symbol.
3. Each leaf is labeled with a terminal symbol or with λ.
4. If in the derivation the production $A \to s_1 s_2 \cdots s_k$, is used where $s_i \in (V \cup \Sigma)^*$, the node A has k immediate descendants: s_1, s_2, ..., s_k, written form left to right.

From this construction, what can we say about the leaves of the derivation tree $S \stackrel{*}{\Longrightarrow} w$?

Languages and context-free grammars

How is the tree of a derivation $S \stackrel{*}{\Longrightarrow} w$, $w \in \Sigma^*$, constructed in a CFG?

The rooted tree is formed as follows:

1. The root is labeled with the start symbol S.
2. Each interior node is labeled with a non-terminal symbol.
3. Each leaf is labeled with a terminal symbol or with λ.
4. If in the derivation the production $A \to s_1 s_2 \cdots s_k$, is used where $s_i \in (V \cup \Sigma)^*$, the node A has k immediate descendants: $s_1, s_2, \ldots, s_k$, written form left to right.

From this construction, what can we say about the leaves of the derivation tree $S \stackrel{*}{\Longrightarrow} w$?

They are precisely the symbols of the string w, read from left to right, and possibly some λ.

Languages and context-free grammars

How is the tree of a derivation $S \stackrel{*}{\Longrightarrow} w$, $w \in \Sigma^*$, constructed in a CFG?

The rooted tree is formed as follows:

1. The root is labeled with the start symbol S.
2. Each interior node is labeled with a non-terminal symbol.
3. Each leaf is labeled with a terminal symbol or with λ.
4. If in the derivation the production $A \to s_1 s_2 \cdots s_k$, is used where $s_i \in (V \cup \Sigma)^*$, the node A has k immediate descendants: $s_1, s_2, ..., s_k$, written form left to right.

From this construction, what can we say about the leaves of the derivation tree $S \stackrel{*}{\Longrightarrow} w$?

They are precisely the symbols of the string w, read from left to right, and possibly some λ.

What is a syntactic analysis of w?

Languages and context-free grammars

How is the tree of a derivation $S \stackrel{*}{\Longrightarrow} w,\ w \in \Sigma^*$, constructed in a CFG?

The rooted tree is formed as follows:

1. The root is labeled with the start symbol S.
2. Each interior node is labeled with a non-terminal symbol.
3. Each leaf is labeled with a terminal symbol or with λ.
4. If in the derivation the production $A \rightarrow s_1 s_2 \cdots s_k$, is used where $s_i \in (V \cup \Sigma)^*$, the node A has k immediate descendants: $s_1, s_2, \ldots, s_k$, written form left to right.

From this construction, what can we say about the leaves of the derivation tree $S \stackrel{*}{\Longrightarrow} w$?

They are precisely the symbols of the string w, read from left to right, and possibly some λ.

What is a syntactic analysis of w?

It is the search for a derivation tree for a string $w \in \Sigma^*$

Example

Let G be the grammar:

Languages and context-free grammars

Example

Let G be the grammar:

$$\begin{cases} S \to AB \mid AaB \\ A \to aA \mid a \\ B \to bBa \mid b \end{cases}$$

Languages and context-free grammars

Example

Let G be the grammar:

$$\begin{cases} S \to AB \mid AaB \\ A \to aA \mid a \\ B \to bBa \mid b \end{cases}$$

The derivation tree

Example

Let G be the grammar:

$$\begin{cases} S \to AB \mid AaB \\ A \to aA \mid a \\ B \to bBa \mid b \end{cases}$$

The derivation tree

$$S \Longrightarrow AB \Longrightarrow AbBa \Longrightarrow abBa \Longrightarrow abba$$

Example

Let G be the grammar:

$$\begin{cases} S \to AB \mid AaB \\ A \to aA \mid a \\ B \to bBa \mid b \end{cases}$$

The derivation tree

$$S \Longrightarrow AB \Longrightarrow AbBa \Longrightarrow abBa \Longrightarrow abba$$

is

Example

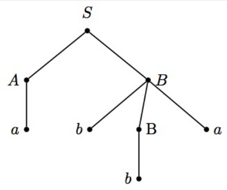
Let G be the grammar:

$$\begin{cases} S \to AB \mid AaB \\ A \to aA \mid a \\ B \to bBa \mid b \end{cases}$$

The derivation tree

$$S \Longrightarrow AB \Longrightarrow AbBa \Longrightarrow abBa \Longrightarrow abba$$

is



When does the notion of ambiguity arise in grammars?

When does the notion of ambiguity arise in grammars?

The notion of ambiguity can arise in terms of syntactic trees or in terms of certain “standard” derivations (called leftmost derivations).

When does the notion of ambiguity arise in grammars?

The notion of ambiguity can arise in terms of syntactic trees or in terms of certain "standard" derivations (called leftmost derivations).

What is a leftmost derivation?

When does the notion of ambiguity arise in grammars?

The notion of ambiguity can arise in terms of syntactic trees or in terms of certain "standard" derivations (called leftmost derivations).

What is a leftmost derivation?

Definition

A derivation is called a leftmost derivation if, at each step, a production is applied to the variable that is furthest to the left.

How is any derivation transformed into a leftmost derivation?

How is any derivation transformed into a leftmost derivation?

By applying, at each step, productions to the variable that is furthest to the left.

How is any derivation transformed into a leftmost derivation?

By applying, at each step, productions to the variable that is furthest to the left.

Now, what is an ambiguous grammar?

How is any derivation transformed into a leftmost derivation?

By applying, at each step, productions to the variable that is furthest to the left.

Now, what is an ambiguous grammar?

Definition

A CFG G is ambiguous if there exists a string $w \in \Sigma^*$ for which there are two different leftmost derivations. Equivalently, a CFG G is ambiguous if there exists a string $w \in \Sigma^*$ with two different derivation trees.

Example

Consider the alphabet $\Sigma = 0, 1, +, *, (,)$. *The following grammar* G_{sp} *for generating natural numbers, addition, and multiplication in binary notation is ambiguous:*

Example

Consider the alphabet $\Sigma = 0, 1, +, *, (,)$. *The following grammar* G_{sp} *for generating natural numbers, addition, and multiplication in binary notation is ambiguous:*

$$S \to S + S \mid S * S \mid (S) \mid 0S \mid 1S \mid 0 \mid 1$$

Example

Consider the alphabet $\Sigma = 0, 1, +, *, (,)$. *The following grammar* G_{sp} *for generating natural numbers, addition, and multiplication in binary notation is ambiguous:*

$$S \to S + S \mid S * S \mid (S) \mid 0S \mid 1S \mid 0 \mid 1$$

The string $1 + 1 * 0$ *has two different leftmost derivations:*

Languages and free-context grammars

Example

Consider the alphabet $\Sigma = 0, 1, +, *, (,)$. *The following grammar* G_{sp} *for generating natural numbers, addition, and multiplication in binary notation is ambiguous:*

$$S \to S + S \mid S * S \mid (S) \mid 0S \mid 1S \mid 0 \mid 1$$

The string $1 + 1 * 0$ *has two different leftmost derivations:*

$$S \Longrightarrow S + S \Longrightarrow 1 + S \Longrightarrow 1 + S * S \Longrightarrow 1 + 1 * S \Longrightarrow 1 + 1 * 0.$$

Example

Consider the alphabet $\Sigma = 0, 1, +, *, (,)$*. The following grammar* G_{sp} *for generating natural numbers, addition, and multiplication in binary notation is ambiguous:*

$$S \to S + S \mid S * S \mid (S) \mid 0S \mid 1S \mid 0 \mid 1$$

The string $1 + 1 * 0$ *has two different leftmost derivations:*

$$S \Longrightarrow S + S \Longrightarrow 1 + S \Longrightarrow 1 + S * S \Longrightarrow 1 + 1 * S \Longrightarrow 1 + 1 * 0.$$

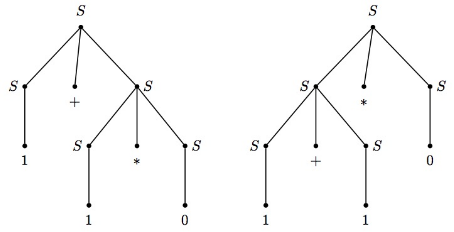
$$S \Longrightarrow S * S \Longrightarrow S + S * S \Longrightarrow 1 + S * S \Longrightarrow 1 + 1 * S \Longrightarrow 1 + 1 * 0.$$

Example

The derivation trees corresponding to the previous derivations are:

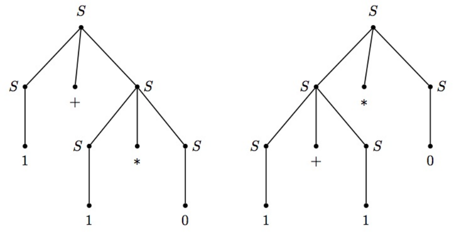
Example

The derivation trees corresponding to the previous derivations are:



Example

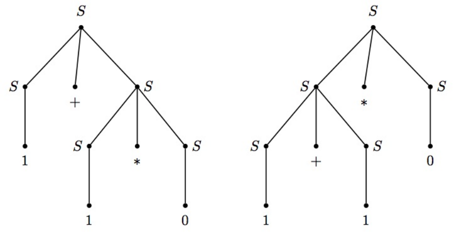
The derivation trees corresponding to the previous derivations are:



In grammar G_{sp}, ambiguity can be eliminated by introducing parentheses:

Example

The derivation trees corresponding to the previous derivations are:



In grammar G_{sp}, ambiguity can be eliminated by introducing parentheses:

$$S \to (S + S) \mid (S * S) \mid (S) \mid 0S \mid 1S \mid 0 \mid 1$$

Example

The following grammar G is ambiguous:

Languages and free-context grammars

Example

The following grammar G is ambiguous:

$$G : \begin{cases} S \to aSA \mid \lambda \\ A \to bA \mid \lambda \end{cases}$$

Example

The following grammar G is ambiguous:

$$G : \begin{cases} S \rightarrow aSA \mid \lambda \\ A \rightarrow bA \mid \lambda \end{cases}$$

G is ambiguous because for the string aab there are two different leftmost derivations.

Example

The following grammar G is ambiguous:

$$G : \begin{cases} S \to aSA \,|\, \lambda \\ A \to bA \,|\, \lambda \end{cases}$$

G is ambiguous because for the string aab there are two different leftmost derivations.

$$S \Longrightarrow aSA \Longrightarrow aaSAA \Longrightarrow aaAA \Longrightarrow aaA \Longrightarrow aabA \Longrightarrow aab.$$

Example

The following grammar G is ambiguous:

$$G : \begin{cases} S \to aSA \mid \lambda \\ A \to bA \mid \lambda \end{cases}$$

G is ambiguous because for the string aab there are two different leftmost derivations.

$$S \Longrightarrow aSA \Longrightarrow aaSAA \Longrightarrow aaAA \Longrightarrow aaA \Longrightarrow aabA \Longrightarrow aab.$$

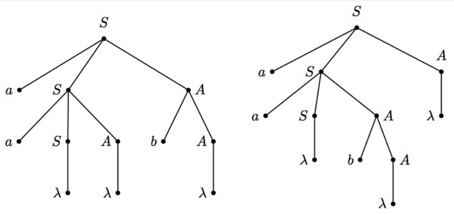
$$S \Longrightarrow aSA \Longrightarrow aaSAA \Longrightarrow aaAA \Longrightarrow aabAA \Longrightarrow aabA \Longrightarrow aab.$$

Example

The derivation trees for these two derivations are:

Example

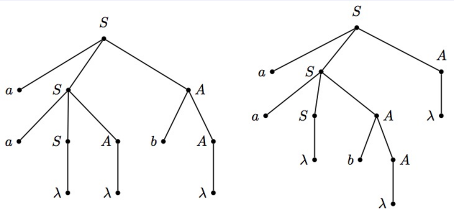
The derivation trees for these two derivations are:



Languages and free-context grammars

Example

The derivation trees for these two derivations are:

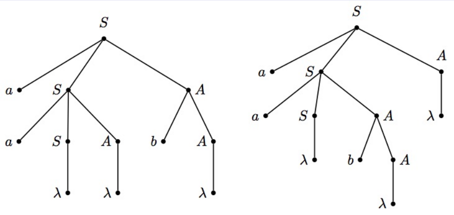


The language generated by this grammar is $a^+ b^* \cup \lambda$. *We can construct a non-ambiguous grammar that generates the same language:*

Languages and free-context grammars

Example

The derivation trees for these two derivations are:



The language generated by this grammar is $a^+b^* \cup \lambda$. We can construct a non-ambiguous grammar that generates the same language:

$$G : \begin{cases} S \to AB \mid \lambda \\ A \to aA \mid a \\ B \to bB \mid \lambda \end{cases}$$

Languages and free-context grammars

What is the relationship between CFGs and programming languages?

What is the relationship between CFGs and programming languages?

The syntax of programming languages, or at least a large portion of it, is typically presented using CFGs. In those cases, the language is said to be in Backus-Naur Form or simply in BNF.

Languages and free-context grammars

What is the relationship between CFGs and programming languages?

The syntax of programming languages, or at least a large portion of it, is typically presented using CFGs. In those cases, the language is said to be in Backus-Naur Form or simply in BNF.

What is the name of those languages where their syntax is presented by a CFG?

What is the relationship between CFGs and programming languages?

The syntax of programming languages, or at least a large portion of it, is typically presented using CFGs. In those cases, the language is said to be in Backus-Naur Form or simply in BNF.

What is the name of those languages where their syntax is presented by a CFG?

These languages are said to be in Backus-Naur Form or simply in BNF.

Languages and free-context grammars

What is the relationship between CFGs and programming languages?

The syntax of programming languages, or at least a large portion of it, is typically presented using CFGs. In those cases, the language is said to be in Backus-Naur Form or simply in BNF.

What is the name of those languages where their syntax is presented by a CFG?

These languages are said to be in Backus-Naur Form or simply in BNF. Languages that are in BNF offer significant advantages for the design of syntactic analyzers in compilers.

Example

Next, we present a grammar that generates unsigned real numbers, similar to the one used in many programming languages.

Languages and free-context grammars

Example

Next, we present a grammar that generates unsigned real numbers, similar to the one used in many programming languages.

- *Variables are enclosed in angle brackets* $\langle , \rangle$, *and* $\langle$*real*$\rangle$ *is the starting variable of the grammar.*

Languages and free-context grammars

Example

Next, we present a grammar that generates unsigned real numbers, similar to the one used in many programming languages.

- *Variables are enclosed in angle brackets* $\langle , \rangle$, *and* $\langle$*real*$\rangle$ *is the starting variable of the grammar.*
- *The terminal alphabet is* $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., +, -,$ *E. In the context of programming languages, terminals are also referred to as lexical components, lexemes, or tokens.*

Languages and free-context grammars

Example

Next, we present a grammar that generates unsigned real numbers, similar to the one used in many programming languages.

- *Variables are enclosed in angle brackets* $\langle, \rangle$*, and* $\langle real \rangle$ *is the starting variable of the grammar.*
- *The terminal alphabet is* $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., +, -, E$*. In the context of programming languages, terminals are also referred to as lexical components, lexemes, or tokens.*

$$\langle real \rangle \quad \to \quad \langle digits \rangle \, \langle decimal \rangle \, \langle exp \rangle$$

Languages and free-context grammars

Example

Next, we present a grammar that generates unsigned real numbers, similar to the one used in many programming languages.

- *Variables are enclosed in angle brackets* $\langle , \rangle$, *and* $\langle real \rangle$ *is the starting variable of the grammar.*
- *The terminal alphabet is* $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., +, -, E$. *In the context of programming languages, terminals are also referred to as lexical components, lexemes, or tokens.*

$$
\begin{array}{lcl}
\langle real \rangle & \rightarrow & \langle digits \rangle\, \langle decimal \rangle\, \langle exp \rangle \\
\langle digits \rangle & \rightarrow & \langle digits \rangle\, \langle digits \rangle \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{array}
$$

Languages and free-context grammars

Example

Next, we present a grammar that generates unsigned real numbers, similar to the one used in many programming languages.

- *Variables are enclosed in angle brackets* $\langle , \rangle$*, and* $\langle real \rangle$ *is the starting variable of the grammar.*
- *The terminal alphabet is* $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., +, -, E$*. In the context of programming languages, terminals are also referred to as lexical components, lexemes, or tokens.*

$$
\begin{array}{lcl}
\langle real \rangle & \rightarrow & \langle digits \rangle\, \langle decimal \rangle\, \langle exp \rangle \\
\langle digits \rangle & \rightarrow & \langle digits \rangle\, \langle digits \rangle \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
\langle decimal \rangle & \rightarrow & .\langle digits \rangle \mid \lambda
\end{array}
$$

Languages and free-context grammars

Example

Next, we present a grammar that generates unsigned real numbers, similar to the one used in many programming languages.

- *Variables are enclosed in angle brackets* $\langle , \rangle$, *and* $\langle real \rangle$ *is the starting variable of the grammar.*
- *The terminal alphabet is* $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., +, -, E$. *In the context of programming languages, terminals are also referred to as lexical components, lexemes, or tokens.*

$$
\begin{array}{lcl}
\langle real \rangle & \rightarrow & \langle digits \rangle \, \langle decimal \rangle \, \langle exp \rangle \\
\langle digits \rangle & \rightarrow & \langle digits \rangle \, \langle digits \rangle \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
\langle decimal \rangle & \rightarrow & .\langle digits \rangle \mid \lambda \\
\langle exp \rangle & \rightarrow & E\langle digits \rangle \mid E{+}\langle digits \rangle \mid E{-}\langle digits \rangle \mid \lambda
\end{array}
$$

Languages and free-context grammars

Example

Next, we present a grammar that generates unsigned real numbers, similar to the one used in many programming languages.

- *Variables are enclosed in angle brackets* $\langle,\rangle$*, and* $\langle$*real*$\rangle$ *is the starting variable of the grammar.*
- *The terminal alphabet is* $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., +, -, E$*. In the context of programming languages, terminals are also referred to as lexical components, lexemes, or tokens.*

$$
\begin{array}{lcl}
\langle \textit{real} \rangle & \to & \langle \textit{digits} \rangle\, \langle \textit{decimal} \rangle\, \langle \textit{exp} \rangle \\
\langle \textit{digits} \rangle & \to & \langle \textit{digits} \rangle\, \langle \textit{digits} \rangle \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
\langle \textit{decimal} \rangle & \to & .\langle \textit{digits} \rangle \mid \lambda \\
\langle \textit{exp} \rangle & \to & E\langle \textit{digits} \rangle \mid E{+}\langle \textit{digits} \rangle \mid E{-}\langle \textit{digits} \rangle \mid \lambda
\end{array}
$$

- *This grammar generates expressions like 47.236, 321.25E+35, 0.8E9, and 0.8E+9.*

Example

Next, we present a grammar that generates unsigned real numbers, similar to the one used in many programming languages.

- *Variables are enclosed in angle brackets* $\langle , \rangle$, *and* $\langle real \rangle$ *is the starting variable of the grammar.*
- *The terminal alphabet is* $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., +, -, E$. *In the context of programming languages, terminals are also referred to as lexical components, lexemes, or tokens.*

$$
\begin{array}{lcl}
\langle real \rangle & \to & \langle digits \rangle \, \langle decimal \rangle \, \langle exp \rangle \\
\langle digits \rangle & \to & \langle digits \rangle \, \langle digits \rangle \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
\langle decimal \rangle & \to & .\langle digits \rangle \mid \lambda \\
\langle exp \rangle & \to & E\langle digits \rangle \mid E{+}\langle digits \rangle \mid E{-}\langle digits \rangle \mid \lambda
\end{array}
$$

- *This grammar generates expressions like 47.236, 321.25E+35, 0.8E9, and 0.8E+9.*
- *The decimal and exponential parts are "optional" due to the productions* $\langle decimal \rangle \to \lambda$ *and* $\langle exp \rangle \to \lambda$, *but expressions like .325, E125, 42.5E, and 0.1E+ are not generated.*

Example

Grammar for generating identifiers in programming languages, i.e., strings whose first symbol is a letter followed by letters and/or digits.

Example

Grammar for generating identifiers in programming languages, i.e., strings whose first symbol is a letter followed by letters and/or digits.

- *Variables are enclosed in angle brackets* $\langle, \rangle$, *and* $\langle$*identifier*$\rangle$ *is the initial variable of the grammar.*

Example

Grammar for generating identifiers in programming languages, i.e., strings whose first symbol is a letter followed by letters and/or digits.

- *Variables are enclosed in angle brackets* $\langle , \rangle$, *and* $\langle$*identifier*$\rangle$ *is the initial variable of the grammar.*
- *The variable* $\langle$*lsds*$\rangle$ *represents "letters or digits".*

Example

Grammar for generating identifiers in programming languages, i.e., strings whose first symbol is a letter followed by letters and/or digits.

- *Variables are enclosed in angle brackets $\langle, \rangle$, and $\langle$identifier$\rangle$ is the initial variable of the grammar.*
- *The variable $\langle$lsds$\rangle$ represents "letters or digits".*
- *The terminals in this grammar are letters, lowercase or uppercase, and digits.*

Example

Grammar for generating identifiers in programming languages, i.e., strings whose first symbol is a letter followed by letters and/or digits.

- *Variables are enclosed in angle brackets* $\langle , \rangle$, *and* $\langle$*identifier*$\rangle$ *is the initial variable of the grammar.*
- *The variable* $\langle$*lsds*$\rangle$ *represents "letters or digits".*
- *The terminals in this grammar are letters, lowercase or uppercase, and digits.*

$$\langle \textit{identifier} \rangle \quad \to \quad \langle \textit{letter} \rangle \, \langle \textit{lsds} \rangle$$

Languages and free-context grammars

Example

Grammar for generating identifiers in programming languages, i.e., strings whose first symbol is a letter followed by letters and/or digits.

- *Variables are enclosed in angle brackets* $\langle , \rangle$, *and* $\langle identifier \rangle$ *is the initial variable of the grammar.*
- *The variable* $\langle lsds \rangle$ *represents "letters or digits".*
- *The terminals in this grammar are letters, lowercase or uppercase, and digits.*

$$\begin{array}{lcl} \langle identifier \rangle & \to & \langle letter \rangle \, \langle lsds \rangle \\ \langle lsds \rangle & \to & \langle letter \rangle \, \langle lsds \rangle \mid \langle digit \rangle \, \langle lsds \rangle \mid \lambda \end{array}$$

Example

Grammar for generating identifiers in programming languages, i.e., strings whose first symbol is a letter followed by letters and/or digits.

- *Variables are enclosed in angle brackets* $\langle , \rangle$, *and* $\langle identifier \rangle$ *is the initial variable of the grammar.*
- *The variable* $\langle lsds \rangle$ *represents "letters or digits".*
- *The terminals in this grammar are letters, lowercase or uppercase, and digits.*

$$
\begin{array}{lcl}
\langle identifier \rangle & \rightarrow & \langle letter \rangle\, \langle lsds \rangle \\
\langle lsds \rangle & \rightarrow & \langle letter \rangle\, \langle lsds \rangle \mid \langle digit \rangle\, \langle lsds \rangle \mid \lambda \\
\langle letter \rangle & \rightarrow & a \mid b \mid c \mid \cdots \mid x \mid y \mid z \mid A \mid B \mid C \mid \cdots \mid X \mid Y \mid Z
\end{array}
$$

Example

Grammar for generating identifiers in programming languages, i.e., strings whose first symbol is a letter followed by letters and/or digits.

- *Variables are enclosed in angle brackets* $\langle , \rangle$, *and* $\langle$*identifier*$\rangle$ *is the initial variable of the grammar.*
- *The variable* $\langle$*lsds*$\rangle$ *represents "letters or digits".*
- *The terminals in this grammar are letters, lowercase or uppercase, and digits.*

$$\begin{array}{lcl} \langle \mathit{identifier} \rangle & \to & \langle \mathit{letter} \rangle \, \langle \mathit{lsds} \rangle \\ \langle \mathit{lsds} \rangle & \to & \langle \mathit{letter} \rangle \, \langle \mathit{lsds} \rangle \mid \langle \mathit{digit} \rangle \, \langle \mathit{lsds} \rangle \mid \lambda \\ \langle \mathit{letter} \rangle & \to & a \mid b \mid c \mid \cdots \mid x \mid y \mid z \mid A \mid B \mid C \mid \cdots \mid X \mid Y \mid Z \\ \langle \mathit{digit} \rangle & \to & 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{array}$$

Languages and free-context grammars

What is the relationship between CFGs and natural languages?

What is the relationship between CFGs and natural languages?

For natural languages, such as Spanish, CFGs can be used to generate the permitted phrases or sentences in spoken or written communication.

Languages and free-context grammars

What is the relationship between CFGs and natural languages?

For natural languages, such as Spanish, CFGs can be used to generate the permitted phrases or sentences in spoken or written communication.

What would a CFG look like that allows generating a portion of Spanish language sentences?

What is the relationship between CFGs and natural languages?

For natural languages, such as Spanish, CFGs can be used to generate the permitted phrases or sentences in spoken or written communication.

What would a CFG look like that allows generating a portion of Spanish language sentences?

- Variables will be enclosed in angle brackets $\langle, \rangle$, and $\langle$Sentence$\rangle$ is the initial variable of the grammar.

Languages and free-context grammars

What is the relationship between CFGs and natural languages?

For natural languages, such as Spanish, CFGs can be used to generate the permitted phrases or sentences in spoken or written communication.

What would a CFG look like that allows generating a portion of Spanish language sentences?

- Variables will be enclosed in angle brackets $\langle, \rangle$, and $\langle$Sentence$\rangle$ is the initial variable of the grammar.
- Terminals will be the words specific to the language.

Languages and free-context grammars

$$
\begin{array}{lcl}
\langle \text{Oración} \rangle & \rightarrow & \langle \text{Sujeto} \rangle \ \langle \text{Verbo} \rangle \ \langle \text{Compl. Directo} \rangle \mid \\
& & \langle \text{Sujeto} \rangle \ \langle \text{Verbo} \rangle \ \langle \text{Compl. Directo} \rangle \ \langle \text{Compl. Circunst.} \rangle \mid \\
& & \langle \text{Sujeto} \rangle \ \langle \text{Verbo} \rangle \ \langle \text{Compl. Indirecto} \rangle \ \langle \text{Compl. Circunst.} \rangle
\end{array}
$$

Languages and free-context grammars

$$
\begin{array}{lcl}
\langle \text{Oración} \rangle & \rightarrow & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Directo} \rangle \mid \\
 & & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Directo} \rangle\ \langle \text{Compl. Circunst.} \rangle \mid \\
 & & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Indirecto} \rangle\ \langle \text{Compl. Circunst.} \rangle \\
\langle \text{Sujeto} \rangle & \rightarrow & \langle \text{Sustant.} \rangle \mid \text{Juan} \mid \text{Pedro} \mid \text{María} \mid \cdots
\end{array}
$$

Languages and free-context grammars

$$
\begin{array}{lcl}
\langle \text{Oración} \rangle & \rightarrow & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Directo} \rangle \mid \\
 & & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Directo} \rangle\ \langle \text{Compl. Circunst.} \rangle \mid \\
 & & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Indirecto} \rangle\ \langle \text{Compl. Circunst.} \rangle \\
\langle \text{Sujeto} \rangle & \rightarrow & \langle \text{Sustant.} \rangle \mid \text{Juan} \mid \text{Pedro} \mid \text{María} \mid \cdots \\
\langle \text{Compl. Directo} \rangle & \rightarrow & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \langle \text{Prepos.} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Adjetivo} \rangle
\end{array}
$$

Languages and free-context grammars

$$
\begin{array}{lcl}
\langle \text{Oración} \rangle & \to & \langle \text{Sujeto} \rangle \ \langle \text{Verbo} \rangle \ \langle \text{Compl. Directo} \rangle \mid \\
 & & \langle \text{Sujeto} \rangle \ \langle \text{Verbo} \rangle \ \langle \text{Compl. Directo} \rangle \ \langle \text{Compl. Circunst.} \rangle \mid \\
 & & \langle \text{Sujeto} \rangle \ \langle \text{Verbo} \rangle \ \langle \text{Compl. Indirecto} \rangle \ \langle \text{Compl. Circunst.} \rangle \\
\langle \text{Sujeto} \rangle & \to & \langle \text{Sustant.} \rangle \mid \text{Juan} \mid \text{Pedro} \mid \text{María} \mid \cdots \\
\langle \text{Compl. Directo} \rangle & \to & \langle \text{Prepos.} \rangle \ \langle \text{Artículo} \rangle \ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle \ \langle \text{Artículo} \rangle \ \langle \text{Sustant.} \rangle \mid \langle \text{Prepos.} \rangle \ \langle \text{Artículo} \rangle \ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle \ \langle \text{Artículo} \rangle \ \langle \text{Sustant.} \rangle \mid \langle \text{Prepos.} \rangle \ \langle \text{Sustant.} \rangle \ \langle \text{Adjetivo} \rangle \\
\langle \text{Compl. Indirecto} \rangle & \to & \langle \text{Prepos.} \rangle \ \langle \text{Artículo} \rangle \ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle \ \langle \text{Artículo} \rangle \ \langle \text{Sustant.} \rangle \ \langle \text{Adjetivo} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle \ \langle \text{Sustant.} \rangle \ \langle \text{Prepos.} \rangle \mid \langle \text{Sustant.} \rangle
\end{array}
$$

Languages and free-context grammars

$$
\begin{array}{lcl}
\langle \text{Oración} \rangle & \rightarrow & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Directo} \rangle \mid \\
 & & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Directo} \rangle\ \langle \text{Compl. Circunst.} \rangle \mid \\
 & & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Indirecto} \rangle\ \langle \text{Compl. Circunst.} \rangle \\
\langle \text{Sujeto} \rangle & \rightarrow & \langle \text{Sustant.} \rangle \mid \text{Juan} \mid \text{Pedro} \mid \text{María} \mid \cdots \\
\langle \text{Compl. Directo} \rangle & \rightarrow & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \langle \text{Prepos.} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Adjetivo} \rangle \\
\langle \text{Compl. Indirecto} \rangle & \rightarrow & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Adjetivo} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Prepos.} \rangle \mid \langle \text{Sustant.} \rangle \\
\langle \text{Compl. Circust.} \rangle & \rightarrow & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Adjetivo} \rangle \mid \langle \text{Adverbio} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle
\end{array}
$$

Languages and free-context grammars

$$
\begin{array}{lcl}
\langle \text{Oración} \rangle & \rightarrow & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Directo} \rangle \mid \\
 & & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Directo} \rangle\ \langle \text{Compl. Circunst.} \rangle \mid \\
 & & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Indirecto} \rangle\ \langle \text{Compl. Circunst.} \rangle \\
\langle \text{Sujeto} \rangle & \rightarrow & \langle \text{Sustant.} \rangle \mid \text{Juan} \mid \text{Pedro} \mid \text{María} \mid \cdots \\
\langle \text{Compl. Directo} \rangle & \rightarrow & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \langle \text{Prepos.} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Adjetivo} \rangle \\
\langle \text{Compl. Indirecto} \rangle & \rightarrow & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Adjetivo} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Prepos.} \rangle \mid \langle \text{Sustant.} \rangle \\
\langle \text{Compl. Circust.} \rangle & \rightarrow & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Adjetivo} \rangle \mid \langle \text{Adverbio} \rangle \mid \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \\
\langle \text{Sustant.} \rangle & \rightarrow & \text{casa} \mid \text{perro} \mid \text{libro} \mid \text{lápiz} \mid \text{mesa} \mid \lambda \mid \cdots
\end{array}
$$

Languages and free-context grammars

$$
\begin{array}{lcl}
\langle \text{Oración} \rangle & \rightarrow & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Directo} \rangle\ | \\
 & & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Directo} \rangle\ \langle \text{Compl. Circunst.} \rangle\ | \\
 & & \langle \text{Sujeto} \rangle\ \langle \text{Verbo} \rangle\ \langle \text{Compl. Indirecto} \rangle\ \langle \text{Compl. Circunst.} \rangle \\
\langle \text{Sujeto} \rangle & \rightarrow & \langle \text{Sustant.} \rangle\ |\ \text{Juan}\ |\ \text{Pedro}\ |\ \text{María}\ |\ \cdots \\
\langle \text{Compl. Directo} \rangle & \rightarrow & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ | \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ |\ \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ | \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ |\ \langle \text{Prepos.} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Adjetivo} \rangle \\
\langle \text{Compl. Indirecto} \rangle & \rightarrow & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ | \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Adjetivo} \rangle\ | \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Prepos.} \rangle\ |\ \langle \text{Sustant.} \rangle \\
\langle \text{Compl. Circust.} \rangle & \rightarrow & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ | \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Adjetivo} \rangle\ |\ \langle \text{Adverbio} \rangle\ | \\
 & & \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle\ \langle \text{Prepos.} \rangle\ \langle \text{Artículo} \rangle\ \langle \text{Sustant.} \rangle \\
\langle \text{Sustant.} \rangle & \rightarrow & \text{casa}\ |\ \text{perro}\ |\ \text{libro}\ |\ \text{lápiz}\ |\ \text{mesa}\ |\ \lambda\ |\ \cdots \\
\langle \text{Adjetivo.} \rangle & \rightarrow & \text{rojo}\ |\ \text{azul}\ |\ \text{inteligente}\ |\ \text{málvado}\ |\ \text{útil}\ |\ \lambda\ |\ \cdots
\end{array}
$$

Languages and free-context grammars

⟨Oración⟩	$\rightarrow$	⟨Sujeto⟩ ⟨Verbo⟩ ⟨Compl. Directo⟩ |
		⟨Sujeto⟩ ⟨Verbo⟩ ⟨Compl. Directo⟩ ⟨Compl. Circunst.⟩ |
		⟨Sujeto⟩ ⟨Verbo⟩ ⟨Compl. Indirecto⟩ ⟨Compl. Circunst.⟩
⟨Sujeto⟩	$\rightarrow$	⟨Sustant.⟩ | Juan | Pedro | María | $\cdots$
⟨Compl. Directo⟩	$\rightarrow$	⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ | ⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ | ⟨Prepos.⟩ ⟨Sustant.⟩ ⟨Adjetivo⟩
⟨Compl. Indirecto⟩	$\rightarrow$	⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ ⟨Adjetivo⟩ |
		⟨Prepos.⟩ ⟨Sustant.⟩ ⟨Prepos.⟩ | ⟨Sustant.⟩
⟨Compl. Circust.⟩	$\rightarrow$	⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ ⟨Adjetivo⟩ | ⟨Adverbio⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ ⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩
⟨Sustant.⟩	$\rightarrow$	casa | perro | libro | lápiz | mesa | λ | $\cdots$
⟨Adjetivo.⟩	$\rightarrow$	rojo | azul | inteligente | málvado | útil | λ | $\cdots$
⟨Prepos.⟩	$\rightarrow$	a | ante | bajo | cabe | con | λ | $\cdots$

Languages and free-context grammars

$$
\begin{array}{lcl}
\langle\text{Oración}\rangle & \rightarrow & \langle\text{Sujeto}\rangle\ \langle\text{Verbo}\rangle\ \langle\text{Compl. Directo}\rangle \mid \\
 & & \langle\text{Sujeto}\rangle\ \langle\text{Verbo}\rangle\ \langle\text{Compl. Directo}\rangle\ \langle\text{Compl. Circunst.}\rangle \mid \\
 & & \langle\text{Sujeto}\rangle\ \langle\text{Verbo}\rangle\ \langle\text{Compl. Indirecto}\rangle\ \langle\text{Compl. Circunst.}\rangle \\
\langle\text{Sujeto}\rangle & \rightarrow & \langle\text{Sustant.}\rangle \mid \text{Juan} \mid \text{Pedro} \mid \text{María} \mid \cdots \\
\langle\text{Compl. Directo}\rangle & \rightarrow & \langle\text{Prepos.}\rangle\ \langle\text{Artículo}\rangle\ \langle\text{Sustant.}\rangle \mid \\
 & & \langle\text{Prepos.}\rangle\ \langle\text{Artículo}\rangle\ \langle\text{Sustant.}\rangle \mid \langle\text{Prepos.}\rangle\ \langle\text{Artículo}\rangle\ \langle\text{Sustant.}\rangle \mid \\
 & & \langle\text{Prepos.}\rangle\ \langle\text{Artículo}\rangle\ \langle\text{Sustant.}\rangle \mid \langle\text{Prepos.}\rangle\ \langle\text{Sustant.}\rangle\ \langle\text{Adjetivo}\rangle \\
\langle\text{Compl. Indirecto}\rangle & \rightarrow & \langle\text{Prepos.}\rangle\ \langle\text{Artículo}\rangle\ \langle\text{Sustant.}\rangle \mid \\
 & & \langle\text{Prepos.}\rangle\ \langle\text{Artículo}\rangle\ \langle\text{Sustant.}\rangle\ \langle\text{Adjetivo}\rangle \mid \\
 & & \langle\text{Prepos.}\rangle\ \langle\text{Sustant.}\rangle\ \langle\text{Prepos.}\rangle \mid \langle\text{Sustant.}\rangle \\
\langle\text{Compl. Circust.}\rangle & \rightarrow & \langle\text{Prepos.}\rangle\ \langle\text{Artículo}\rangle\ \langle\text{Sustant.}\rangle \mid \\
 & & \langle\text{Prepos.}\rangle\ \langle\text{Artículo}\rangle\ \langle\text{Sustant.}\rangle\ \langle\text{Adjetivo}\rangle \mid \langle\text{Adverbio}\rangle \mid \\
 & & \langle\text{Prepos.}\rangle\ \langle\text{Artículo}\rangle\ \langle\text{Sustant.}\rangle\ \langle\text{Prepos.}\rangle\ \langle\text{Artículo}\rangle\ \langle\text{Sustant.}\rangle \\
\langle\text{Sustant.}\rangle & \rightarrow & \text{casa} \mid \text{perro} \mid \text{libro} \mid \text{lápiz} \mid \text{mesa} \mid \lambda \mid \cdots \\
\langle\text{Adjetivo.}\rangle & \rightarrow & \text{rojo} \mid \text{azul} \mid \text{inteligente} \mid \text{málvado} \mid \text{útil} \mid \lambda \mid \cdots \\
\langle\text{Prepos.}\rangle & \rightarrow & \text{a} \mid \text{ante} \mid \text{bajo} \mid \text{cabe} \mid \text{con} \mid \lambda \mid \cdots \\
\langle\text{Artículo}\rangle & \rightarrow & \text{el} \mid \text{la} \mid \text{lo} \mid \text{las} \mid \text{los} \mid \text{un} \mid \text{uno} \mid \text{una} \mid \text{unas} \mid \text{unos} \mid \lambda
\end{array}
$$

Languages and free-context grammars

⟨Oración⟩	→	⟨Sujeto⟩ ⟨Verbo⟩ ⟨Compl. Directo⟩ |
		⟨Sujeto⟩ ⟨Verbo⟩ ⟨Compl. Directo⟩ ⟨Compl. Circunst.⟩ |
		⟨Sujeto⟩ ⟨Verbo⟩ ⟨Compl. Indirecto⟩ ⟨Compl. Circunst.⟩
⟨Sujeto⟩	→	⟨Sustant.⟩ | Juan | Pedro | María | ···
⟨Compl. Directo⟩	→	⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ | ⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ | ⟨Prepos.⟩ ⟨Sustant.⟩ ⟨Adjetivo⟩
⟨Compl. Indirecto⟩	→	⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ ⟨Adjetivo⟩ |
		⟨Prepos.⟩ ⟨Sustant.⟩ ⟨Prepos.⟩ | ⟨Sustant.⟩
⟨Compl. Circust.⟩	→	⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ ⟨Adjetivo⟩ | ⟨Adverbio⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ ⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩
⟨Sustant.⟩	→	casa | perro | libro | lápiz | mesa | λ | ···
⟨Adjetivo.⟩	→	rojo | azul | inteligente | málvado | útil | λ | ···
⟨Prepos.⟩	→	a | ante | bajo | cabe | con | λ | ···
⟨Artículo⟩	→	el | la | lo | las | los | un | uno | una | unas | unos | λ
⟨Adverbio⟩	→	muy | bastante | poco | demasiado | lento | λ | ···

Languages and free-context grammars

⟨Oración⟩	$\to$	⟨Sujeto⟩ ⟨Verbo⟩ ⟨Compl. Directo⟩ |
		⟨Sujeto⟩ ⟨Verbo⟩ ⟨Compl. Directo⟩ ⟨Compl. Circunst.⟩ |
		⟨Sujeto⟩ ⟨Verbo⟩ ⟨Compl. Indirecto⟩ ⟨Compl. Circunst.⟩
⟨Sujeto⟩	$\to$	⟨Sustant.⟩ | Juan | Pedro | María | $\cdots$
⟨Compl. Directo⟩	$\to$	⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ | ⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ | ⟨Prepos.⟩ ⟨Sustant.⟩ ⟨Adjetivo⟩
⟨Compl. Indirecto⟩	$\to$	⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ ⟨Adjetivo⟩ |
		⟨Prepos.⟩ ⟨Sustant.⟩ ⟨Prepos.⟩ | ⟨Sustant.⟩
⟨Compl. Circust.⟩	$\to$	⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ ⟨Adjetivo⟩ | ⟨Adverbio⟩ |
		⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩ ⟨Prepos.⟩ ⟨Artículo⟩ ⟨Sustant.⟩
⟨Sustant.⟩	$\to$	casa | perro | libro | lápiz | mesa | λ | $\cdots$
⟨Adjetivo.⟩	$\to$	rojo | azul | inteligente | málvado | útil | λ | $\cdots$
⟨Prepos.⟩	$\to$	a | ante | bajo | cabe | con | λ | $\cdots$
⟨Artículo⟩	$\to$	el | la | lo | las | los | un | uno | una | unas | unos | λ
⟨Adverbio⟩	$\to$	muy | bastante | poco | demasiado | lento | λ | $\cdots$
⟨Verbo⟩	$\to$	escribir | escribo | escribe | escribes | escriben | λ | $\cdots$

Languages and free-context grammars

What can we say about natural languages?

Languages and free-context grammars

What can we say about natural languages?

Natural languages are almost always ambiguous because there are many production rules, which result in multiple derivation trees for certain sentences.

Languages and free-context grammars

What can we say about natural languages?

Natural languages are almost always ambiguous because there are many production rules, which result in multiple derivation trees for certain sentences.

Example

The sentence

What can we say about natural languages?

Natural languages are almost always ambiguous because there are many production rules, which result in multiple derivation trees for certain sentences.

Example

The sentence

Juan mira a una persona con un telescopio

What can we say about natural languages?

Natural languages are almost always ambiguous because there are many production rules, which result in multiple derivation trees for certain sentences.

Example

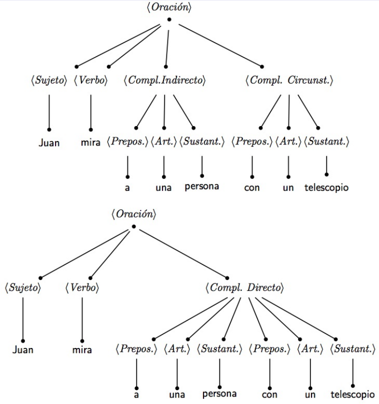
The sentence

Juan mira a una persona con un telescopio

is ambiguous. Ambiguity arises because the productions allow for two derivation trees.

Languages and free-context grammars

Example



Languages and free-context grammars

What is a regular grammar?

What is a regular grammar?

Definition

A CFG is called regular if its productions are of the form

$$G : \begin{cases} A \to aB, \ a \in \Sigma, B \in V. \\ A \to \lambda \end{cases}$$

What is a regular grammar?

Definition

A CFG is called regular if its productions are of the form

$$G : \begin{cases} A \to aB, \ a \in \Sigma, B \in V. \\ A \to \lambda \end{cases}$$

How can a connection be established between regular languages and regular grammars using DFAs?

Languages and free-context grammars

What is a regular grammar?

Definition

A CFG is called regular if its productions are of the form

$$G : \begin{cases} A \to aB, \ a \in \Sigma, \ B \in V. \\ A \to \lambda \end{cases}$$

How can a connection be established between regular languages and regular grammars using DFAs?

Teorema

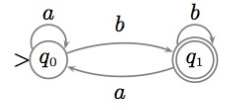
Given a DFA $M = (Q, \Sigma, q_0, F, \delta)$*, there exists a regular CFG* $G = (V, \Sigma, S, P)$ *such that* $L(M) = L(G)$.

Example

The following DFA M accepts strings that end with b:

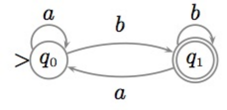
Example

The following DFA M accepts strings that end with b:



Example

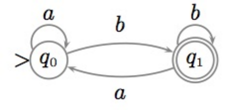
The following DFA M accepts strings that end with b:



M induces the regular grammar:

Example

The following DFA M accepts strings that end with b:

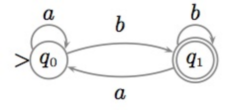


M induces the regular grammar:

$$G : \begin{cases} q_0 \rightarrow aq_0 \mid bq_1 \\ q_1 \rightarrow bq_1 \mid aq_0 \mid \lambda \end{cases}$$

Example

The following DFA M accepts strings that end with b:



M induces the regular grammar:

$$G : \begin{cases} q_0 \to aq_0 \mid bq_1 \\ q_1 \to bq_1 \mid aq_0 \mid \lambda \end{cases}$$

which satisfies $L(M) = L(G)$. The variables of G are q_0 and q_1, with q_0 being the initial variable.

Example

For the regular language $0^*10^*10^*$,*over* $\Sigma = 0, 1$, *we have the following grammar that generates it:*

Example

For the regular language $0^*10^*10^*$,*over* $\Sigma = 0, 1$*, we have the following grammar that generates it:*

$$\begin{cases} S \rightarrow A1A1A \\ A \rightarrow 0A \,|\, \lambda \end{cases}$$

Example

For the regular language $0^*10^*10^*$,*over* $\Sigma = 0, 1$, *we have the following grammar that generates it:*

$$\begin{cases} S \to A1A1A \\ A \to 0A \mid \lambda \end{cases}$$

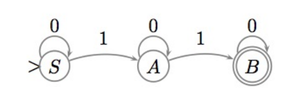
This grammar is not regular, but with the DFA

Example

For the regular language $0^*10^*10^*$,*over* $\Sigma = 0, 1$, *we have the following grammar that generates it:*

$$\begin{cases} S \to A1A1A \\ A \to 0A \mid \lambda \end{cases}$$

This grammar is not regular, but with the DFA

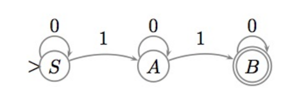


Example

For the regular language $0^*10^*10^*$*,over* $\Sigma = 0, 1$*, we have the following grammar that generates it:*

$$\begin{cases} S \to A1A1A \\ A \to 0A \mid \lambda \end{cases}$$

This grammar is not regular, but with the DFA



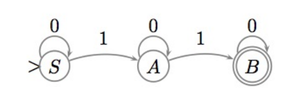
and the previous theorem, we can obtain a regular CFG that generates $0^*10^*10^*$:

Example

For the regular language $0^*10^*10^*$,*over* $\Sigma = 0, 1$, *we have the following grammar that generates it:*

$$\begin{cases} S \to A1A1A \\ A \to 0A \,|\, \lambda \end{cases}$$

This grammar is not regular, but with the DFA



and the previous theorem, we can obtain a regular CFG that generates $0^*10^*10^*$:

$$\begin{cases} S \to 0S \,|\, 1A \\ A \to 0A \,|\, 1B \\ B \to 0B \,|\, \lambda \end{cases}$$

Languages and free-context grammars

How can a connection be established between regular languages and regular grammars using NFAs?

How can a connection be established between regular languages and regular grammars using NFAs?

Teorema

Given a regular CFG $G = (V, \Sigma, S, P)$, there exists an NFA $M = (Q, \Sigma, q_0, F, \Delta)$ such that $L(M) = L(G)$.

How can a connection be established between regular languages and regular grammars using NFAs?

Teorema

Given a regular CFG $G = (V, \Sigma, S, P)$*, there exists an NFA* $M = (Q, \Sigma, q_0, F, \Delta)$ *such that* $L(M) = L(G)$.

From this, we can deduce:

How can a connection be established between regular languages and regular grammars using NFAs?

Teorema

Given a regular CFG $G = (V, \Sigma, S, P)$, *there exists an NFA* $M = (Q, \Sigma, q_0, F, \Delta)$ *such that* $L(M) = L(G)$.

From this, we can deduce:

Corollary

1. *A language is regular if and only if it is generated by a regular grammar.*

How can a connection be established between regular languages and regular grammars using NFAs?

Teorema

Given a regular CFG $G = (V, \Sigma, S, P)$, there exists an NFA $M = (Q, \Sigma, q_0, F, \Delta)$ such that $L(M) = L(G)$.

From this, we can deduce:

Corollary

1. *A language is regular if and only if it is generated by a regular grammar.*
2. *Every regular language is a CFL (but not vice versa).*

What is a right-regular grammar?

What is a right-regular grammar?

Definition

A CFG is called right-regular if its productions are of the form:

What is a right-regular grammar?

Definition

A CFG is called right-regular if its productions are of the form:

$$\begin{cases} A \to wB, \; w \in \Sigma^*, \, B \in V. \\ A \to \lambda \end{cases}$$

What is a right-regular grammar?

Definition

A CFG is called right-regular if its productions are of the form:

$$\begin{cases} A \to wB, \; w \in \Sigma^*, \; B \in V. \\ A \to \lambda \end{cases}$$

What relationship exists between regular grammars and right-regular grammars?

Languages and free-context grammars

What is a right-regular grammar?

Definition

A CFG is called right-regular if its productions are of the form:

$$\begin{cases} A \to wB, \; w \in \Sigma^*, \; B \in V. \\ A \to \lambda \end{cases}$$

What relationship exists between regular grammars and right-regular grammars?

Teorema

Regular grammars and right-regular grammars generate the same languages, i.e., regular languages.

What is a right-regular grammar?

Definition

A CFG is called right-regular if its productions are of the form:

$$\begin{cases} A \to wB, \; w \in \Sigma^*, B \in V. \\ A \to \lambda \end{cases}$$

What relationship exists between regular grammars and right-regular grammars?

Teorema

Regular grammars and right-regular grammars generate the same languages, i.e., regular languages. The definition of a regular grammar is equivalent to the definition of a right-regular grammar.

Agenda del día

1. **Languages and context-free grammars**
   - Introduction
   - CFG
   - Exercises