Theoretical computer science

Tutorial - week 11

April 8, 2021

Innopolis University

Agenda

- Generative Grammars
- Chomsky Hierarchy.
- Context-Free Grammars: Backus-Naur Form
- Kahoot!

Models

- **Automata**

Models

- **Automata** (operational models):
  models suitable to recognize/accept, translate, compute
  language: receive an input string and process it.

Models

- **Automata** (operational models):
  models suitable to recognize/accept, translate, compute
  language: receive an input string and process it.

Models

- **Automata** (operational models):
  models suitable to recognize/accept, translate, compute
  language: receive an input string and process it.

- **Grammars**

Models

- **Automata** (operational models):
  models suitable to recognize/accept, translate, compute language: receive an input string and process it.

- **Grammars** (generative models):
  Models suitable to describe how to generate a language: set of rules to build phrases of a language.

Grammars

A grammar is

Grammars

A grammar is a set of rules to produce strings

Grammar: definition

A grammar is a tuple

$$\langle V_N, V_T, P, S \rangle$$

where

- V_N is the non-terminal alphabet;
- V_T is the terminal alphabet;
- $V = V_N \cup V_T$ the alphabet;
- $P \subseteq (V^* \cdot V_N \cdot V^*) \times V^*$ is the (finite) set of rewriting rules of production;
- $S \in V_N$ is a particular element called axiom or initial symbol.

Grammar: definition

A grammar is a tuple

$$\langle V_N, V_T, P, S \rangle$$

where

- V_N is the non-terminal alphabet;
- V_T is the terminal alphabet;
- $V = V_N \cup V_T$ the alphabet;
- $P \subseteq (V^* \cdot V_N \cdot V^*) \times V^*$ is the (finite) set of rewriting rules of production;
- $S \in V_N$ is a particular element called axiom or initial symbol.

A grammar $\langle V_N, V_T, P, S \rangle$ generates a language on V_T.

Production Rule

Let $G = \langle V_N, V_T, P, S \rangle$ be a grammar.

A **production rule** $\alpha \to \beta$ is an element of P where

- $\alpha \in V^* \cdot V_N \cdot V^*$ is a sequence of symbols including at least one non-terminal symbol;
- $\beta \in V^*$ is a (potentially empty) sequence of (terminal or non-terminal) symbols.

$V = V_N \cup V_T$

Noam Chomsky

Google Scholar

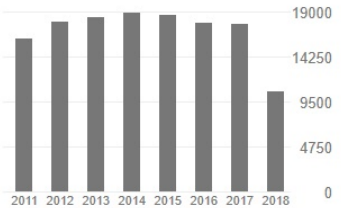
Noam Chomsky FOLLOW

Professor of Linguistics (Emeritus), MIT
No verified email - Homepage
Linguistic Theory Syntax Semantics Philosophy of Language

TITLE	CITED BY	YEAR
Aspects of the Theory of Syntax N Chomsky MIT press	34199	2014
The minimalist program N Chomsky MIT press	25189	2014
Lectures on government and binding: The Pisa lectures N Chomsky Walter de Gruyter	22678	1993
Syntactic structures N Chomsky, DW Lightfoot Walter de Gruyter	21624	2002
The sound pattern of English. N Chomsky, M Halle	15137	1968
Knowledge of language: Its nature, origin, and use N Chomsky Greenwood Publishing Group	12903	1986
The biological foundations of language EH Lenneberg Hospital Practice 2 (12), 59-67	11158	1967

Cited by VIEW ALL

	All	Since 2013
Citations	370302	103483
h-index	173	110
i10-index	966	491



is an American linguist, philosopher, cognitive scientist, historian, political activist, and social critic. Sometimes described as "the father of modern linguistics", Chomsky is also a major figure in analytic philosophy and one of the founders of the field of cognitive science.

Chomsky Hierarchy

Chomsky has classified grammars in four types according to the form of their productions.

(type 3) Regular grammars

(type 2) Context-Free grammars

(type 1) Context-Sensitive grammars

(type 0) Unrestricted grammars

Regular grammars (type 3)

production rules restricted to a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal, possibly

1. followed by a single non-terminal - right grammar
2. preceded by a single non-terminal - left grammar

but **NOT** both in the same grammar

Regular grammars (type 3)

Right regular grammar

A right regular grammar is a formal grammar $\langle V_N, V_T, P, S \rangle$ such that all the production rules in P are of one of the following forms:

1. $A \to b$, where $A \in V_N$ and $b \in V_T$;
2. $A \to bB$, where $A, B \in V_N$ and $b \in V_T$;
3. $A \to \epsilon$, where $A \in V_N$ and ϵ denotes the empty string.

Left regular grammar

A left regular grammar is a formal grammar $\langle V_N, V_T, P, S \rangle$ such that all the production rules in P are of one of the following forms:

1. $A \to b$, where $A \in V_N$ and $b \in V_T$;
2. $A \to B\mathrm{b}$, where $A, B \in V_N$ and $b \in V_T$;
3. $A \to \epsilon$, where $A \in V_N$ and ϵ denotes the empty string.

Context-Free grammars (type 2)

Defined by rules of the form $A \rightarrow \gamma$ where A is a non-terminal and γ is a string of terminals and non-terminals.

Example

Generate language $A^n B^n$

- $S \rightarrow \mathrm{a} S \mathrm{b}$
- $S \rightarrow \epsilon$

Context-Free grammars (type 2)

Defined by rules of the form $A \rightarrow \gamma$ where A is a non-terminal and γ is a string of terminals and non-terminals.

Example

Consider grammar $\langle V_N, V_T, P, S \rangle$. The production rules in P are of the following forms:

1. $A \rightarrow$b, where $A \in V_N$ and $b \in V_T$

Context-Free grammars (type 2)

Defined by rules of the form $A \to \gamma$ where A is a non-terminal and γ is a string of terminals and non-terminals.

Example

Consider grammar $\langle V_N, V_T, P, S \rangle$. The production rules in P are of the following forms:

1. $A \to$b, where $A \in V_N$ and $b \in V_T$ ✓
2. $A \to B$b, where $A, B \in V_N$ and $b \in V_T$

Context-Free grammars (type 2)

Defined by rules of the form $A \to \gamma$ where A is a non-terminal and γ is a string of terminals and non-terminals.

Example

Consider grammar $\langle V_N, V_T, P, S \rangle$. The production rules in P are of the following forms:

1. $A \to$b, where $A \in V_N$ and $b \in V_T$ ✓
2. $A \to B$b, where $A, B \in V_N$ and $b \in V_T$ ✓
3. $A \to B$bB, where $A, B \in V_N$ and $b \in V_T$

Context-Free grammars (type 2)

Defined by rules of the form $A \to \gamma$ where A is a non-terminal and γ is a string of terminals and non-terminals.

Example

Consider grammar $\langle V_N, V_T, P, S \rangle$. The production rules in P are of the following forms:

1. $A \to$b, where $A \in V_N$ and $b \in V_T$ ✓
2. $A \to B$b, where $A, B \in V_N$ and $b \in V_T$ ✓
3. $A \to B$bB, where $A, B \in V_N$ and $b \in V_T$ ✓
4. $A \to$bBb, where $A, B \in V_N$ and $b \in V_T$

Context-Free grammars (type 2)

Defined by rules of the form $A \to \gamma$ where A is a non-terminal and γ is a string of terminals and non-terminals.

Example

Consider grammar $\langle V_N, V_T, P, S \rangle$. The production rules in P are of the following forms:

1. $A \to$b, where $A \in V_N$ and $b \in V_T$ ✓
2. $A \to B$b, where $A, B \in V_N$ and $b \in V_T$ ✓
3. $A \to B$bB, where $A, B \in V_N$ and $b \in V_T$ ✓
4. $A \to$bBb, where $A, B \in V_N$ and $b \in V_T$ ✓
5. $AB \to B$b, where $A, B \in V_N$ and $b \in V_T$

Context-Free grammars (type 2)

Defined by rules of the form $A \to \gamma$ where A is a non-terminal and γ is a string of terminals and non-terminals.

Example

Consider grammar $\langle V_N, V_T, P, S \rangle$. The production rules in P are of the following forms:

1. $A \to$b, where $A \in V_N$ and $b \in V_T$ ✓
2. $A \to B$b, where $A, B \in V_N$ and $b \in V_T$ ✓
3. $A \to B$bB, where $A, B \in V_N$ and $b \in V_T$ ✓
4. $A \to$bBb, where $A, B \in V_N$ and $b \in V_T$ ✓
5. ~~$AB \to B$b, where $A, B \in V_N$ and $b \in V_T$~~

Context-Sensitive grammars (type 1)

The rules of the form $\alpha A \beta \rightarrow \alpha \gamma \beta$, where A is a non-terminal and α, β and γ are strings of terminals and non-terminals.

Example

Generate language $\{A^n B^n C^n | n > 0\}$

Context-Sensitive grammars (type 1)

The rules of the form $\alpha A \beta \rightarrow \alpha \gamma \beta$, where A is a non-terminal and α, β and γ are strings of terminals and non-terminals.

Example

Generate language $\{A^n B^n C^n | n > 0\}$

1. $S \rightarrow \mathrm{a}BC$
2. $S \rightarrow \mathrm{a}SBC$
3. $CB \rightarrow CZ$
4. $CZ \rightarrow WZ$
5. $WZ \rightarrow WC$
6. $WC \rightarrow BC$
7. $\mathrm{a}B \rightarrow \mathrm{ab}$
8. $\mathrm{b}B \rightarrow \mathrm{bb}$
9. $\mathrm{b}C \rightarrow \mathrm{bc}$
10. $\mathrm{c}C \rightarrow \mathrm{cc}$

Unrestricted grammars (type 0)

The rules of the form $\alpha \to \beta$, where α and β are strings of non-terminals and terminals.

The grammars without any limitation on production rules.

Production Rule

Let $G = \langle V_N, V_T, P, S \rangle$ be a grammar.

A **production rule** $\alpha \to \beta$ is an element of P where

- $\alpha \in V^* \cdot V_N \cdot V^*$ is a sequence of symbols including at least one non-terminal symbol;
- $\beta \in V^*$ is a (potentially empty) sequence of (terminal or non-terminal) symbols.

$V = V_N \cup V_T$

Context-Free grammars (type 2)

Defined by rules of the form $A \to \gamma$ where A is a non-terminal and γ is a string of terminals and non-terminals.

Backus Naur Form (BNF)

The BNF Notation

- BNF (Backus Normal Form or Backus–Naur Form) is a notation techniques for context-free grammars.
- It is often used to describe the syntax of programming languages.
- Non-terminals are words in $\langle \ldots \rangle$
  Example: $<\textit{statement}>$, and represents non-terminal symbols.
- Terminal symbols are often multicharacter strings indicated by single quotation marks. Example: 'while'.

The BNF Notation

- Symbol ::= is often used for $\rightarrow$ (from the production rules).
- Symbol | is used as a shorthand for a list of productions with the same left side.
  - Example: $\{S \rightarrow 0S1 \mid 01\}$ is shorthand for $\{S \rightarrow 0S1, S \rightarrow 01\}$.
- Symbol [...] is used to represent optional.
  - Example: a [b] can produce: *ab* or *a*.
- Symbol $\{\ldots\}$ is used to represent zero or more times.
  - Example: a$\{$b$\}$ can produce: *ab* or *a* or *abbb*.

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{`a'`a'} \langle X \rangle$$
$$\langle X \rangle ::= \text{`a'} \langle X \rangle \mid \text{`b'} \langle X \rangle \mid \epsilon$$

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$
$$\langle X \rangle ::= \text{‘a’} \langle X \rangle \mid \text{‘b’} \langle X \rangle \mid \epsilon$$

Can the string *bbaab* be produced by the Grammar?

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$
$$\langle X \rangle ::= \text{‘a’} \langle X \rangle \mid \text{‘b’} \langle X \rangle \mid \epsilon$$

Can the string *bbaab* be produced by the Grammar? We start by the initial symbol $\langle S \rangle$ and then replacing

$$\langle S \rangle = \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$
$$\langle X \rangle ::= \text{‘a’} \langle X \rangle \mid \text{‘b’} \langle X \rangle \mid \epsilon$$

Can the string *bbaab* be produced by the Grammar? We start by the initial symbol $\langle S \rangle$ and then replacing

$$\langle S \rangle = \underline{\langle X \rangle} \text{‘a’‘a’} \langle X \rangle$$

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$
$$\langle X \rangle ::= \text{‘a’} \langle X \rangle \mid \text{‘b’} \langle X \rangle \mid \epsilon$$

Can the string *bbaab* be produced by the Grammar? We start by the initial symbol $\langle S \rangle$ and then replacing

$$\langle S \rangle = \text{‘b’} \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$
$$\langle X \rangle ::= \text{‘a’} \langle X \rangle \mid \text{‘b’} \langle X \rangle \mid \epsilon$$

Can the string *bbaab* be produced by the Grammar? We start by the initial symbol $\langle S \rangle$ and then replacing

$$\langle S \rangle = \text{‘b’} \underline{\langle X \rangle} \text{‘a’‘a’} \langle X \rangle$$

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$
$$\langle X \rangle ::= \text{‘a’} \langle X \rangle \mid \text{‘b’} \langle X \rangle \mid \epsilon$$

Can the string *bbaab* be produced by the Grammar? We start by the initial symbol $\langle S \rangle$ and then replacing

$$\langle S \rangle = \text{‘b’‘b’} \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$
$$\langle X \rangle ::= \text{‘a’} \langle X \rangle \mid \text{‘b’} \langle X \rangle \mid \epsilon$$

Can the string *bbaab* be produced by the Grammar? We start by the initial symbol $\langle S \rangle$ and then replacing

$$\langle S \rangle = \text{‘b’‘b’} \underline{\langle X \rangle} \text{‘a’‘a’} \langle X \rangle$$

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$
$$\langle X \rangle ::= \text{‘a’} \langle X \rangle \mid \text{‘b’} \langle X \rangle \mid \epsilon$$

Can the string *bbaab* be produced by the Grammar? We start by the initial symbol $\langle S \rangle$ and then replacing

$$\langle S \rangle = \text{‘b’‘b’‘a’‘a’} \langle X \rangle$$

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$
$$\langle X \rangle ::= \text{‘a’} \langle X \rangle \mid \text{‘b’} \langle X \rangle \mid \epsilon$$

Can the string *bbaab* be produced by the Grammar? We start by the initial symbol $\langle S \rangle$ and then replacing

$$\langle S \rangle = \text{‘b’‘b’‘a’‘a’} \underline{\langle X \rangle}$$

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$
$$\langle X \rangle ::= \text{‘a’} \langle X \rangle \mid \text{‘b’} \langle X \rangle \mid \epsilon$$

Can the string *bbaab* be produced by the Grammar? We start by the initial symbol $\langle S \rangle$ and then replacing

$$\langle S \rangle = \text{‘b’‘b’‘a’‘a’‘b’} \langle X \rangle$$

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$
$$\langle X \rangle ::= \text{‘a’} \langle X \rangle \mid \text{‘b’} \langle X \rangle \mid \epsilon$$

Can the string *bbaab* be produced by the Grammar? We start by the initial symbol $\langle S \rangle$ and then replacing

$$\langle S \rangle = \text{‘b’‘b’‘a’‘a’‘b’} \underline{\langle X \rangle}$$

BNF: Example

$$\langle S \rangle ::= \langle X \rangle \text{‘a’‘a’} \langle X \rangle$$
$$\langle X \rangle ::= \text{‘a’} \langle X \rangle \mid \text{‘b’} \langle X \rangle \mid \epsilon$$

Can the string *bbaab* be produced by the Grammar? We start by the initial symbol $\langle S \rangle$ and then replacing

$$\langle S \rangle = \text{‘b’‘b’‘a’‘a’‘b’}$$

BNF: Example

- BNF rules of JAVA:
  `http://cui.unige.ch/isi/bnf/JAVA/AJAVA.html`
- BNF rules of Python:
  `https://docs.python.org/3/reference/grammar.html`
- BNF rules of Eiffel:
  `https://www.eiffel.org/doc/eiffel/Eiffel_programming_language_syntax#Eiffel_BNF-E_Syntax`

Regular Expressions: Definition (type 3)

Inductive definition of RegExps over an alphabet Σ:

Basis.

- Symbol $\emptyset$ is a regular expression (denoting the language $\emptyset$);
- Symbol ϵ is a RegExp (denoting the language $\{\epsilon\}$);
- Each symbol σ of Σ is a RegExp
  $L(\sigma) = \{\sigma\}$ for each $\sigma \in \Sigma$.

Induction. Let r and s be two RegExps, then

- $(r.s)$ is a RegExp (for simplicity, the dot is often omitted);
  $L((r.s)) = \{ww' \mid (w \in L(r) \wedge w' \in L(s))\}$,
- $(r|s)$ is a RegExp
  $L((r \mid s)) = L(r) \cup L(s)$;
- r^* is a RegExp
  $L(r^*) = L(r)^*$

RegExp: Precedence order

Parentheses in an expression may be omitted. If they are, evaluation is done in the precedence order:

1. Kleene star *
2. Concatenation .
3. Union |

RegExp: Examples

Consider the alphabet $\Sigma = \{0, 1\}$. Give English descriptions of the languages of the following regular expressions:

1. $1^*\emptyset$;
2. $\emptyset^*$;
3. $(0^*1^*)^*000(0 \mid 1)^*$;
4. $(1 \mid \epsilon)(00^*1)^*0^*$;

Wrap up

- What have you learnt today?

Wrap up

- What have you learnt today?
- What for this could be useful?