Chapter 2 - Outline

2.1 Context-Free Grammars

- Formal definition of a context-free grammar
- Examples of context-free grammars
- Designing context-free grammars
- Ambiguity
- Chomsky normal form

2.2 Pushdown Automata

- Formal definition of a pushdown automaton
- Examples of pushdown automata
- Equivalence with context-free grammars

2.3 Non-Context-Free Languages

- The pumping lemma for context-free languages

2.4 Deterministic Context-Free Languages

- Properties of DCFLs
- Deterministic context-free grammars
- Relationship of DPDAs and DCFGs
- Parsing and LR(k) Grammars

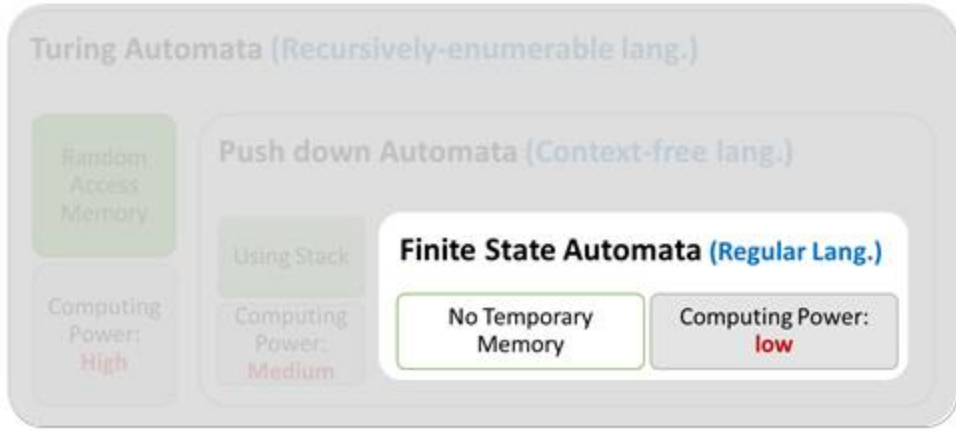
Chapter 2

Introduction

In chapter 1 :

- We learned that
  - we can describe many **languages** (regular languages) using two equivalent methods:
    - finite automata
    - regular expressions

- We found out
  - There are some **languages** that cannot be described using the methods above
    - Non-regular languages
    - Example: $\{0^n1^n \mid n \geq 0\}$



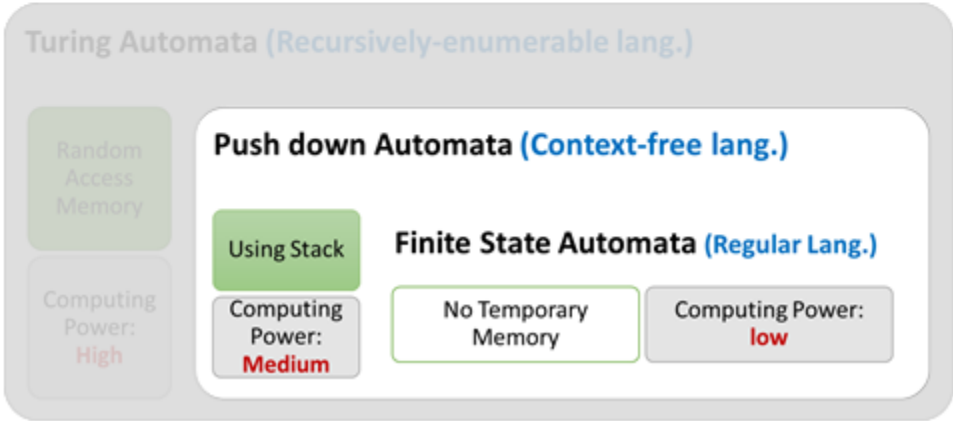
Chapter 2

Introduction

In chapter 2 :

- We present a **more powerful method** of describing languages:
  - called context-free grammars (CFG)

- the collection of **languages** associated with CFG are called the context-free languages (CFL).
  - They can describe languages with a recursive structure,

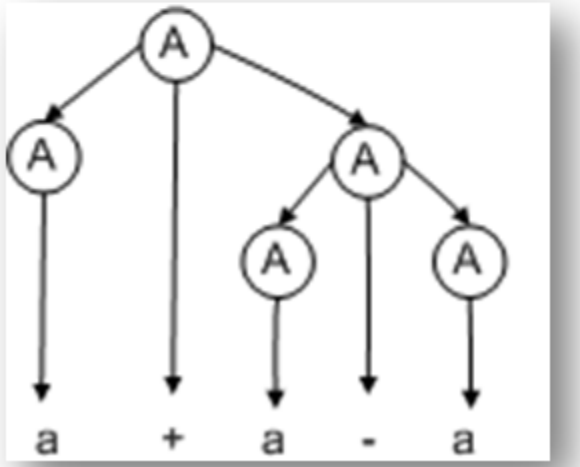


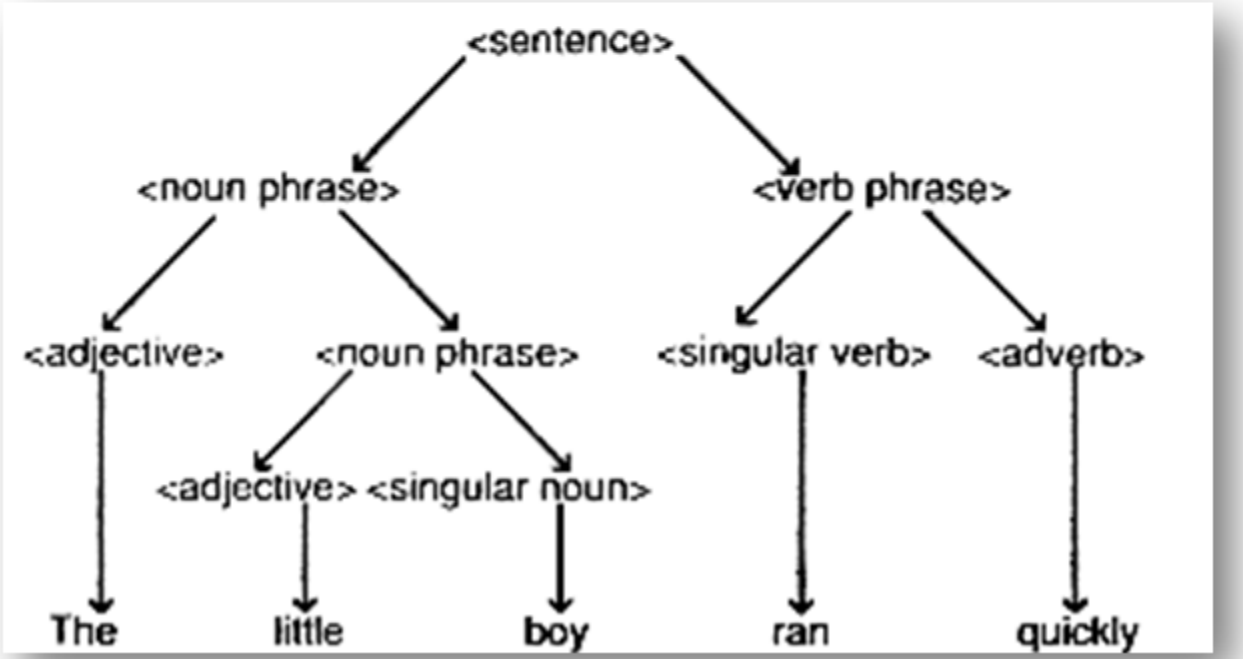
Introduction

context-free grammars:

- first **used** in the study of **human languages**
  - To organize and understand the **relationship of terms** such as noun, verb, and preposition and their respective phrases.

- another **application**: **compilers** and **interpreters** for programming languages using **parser**

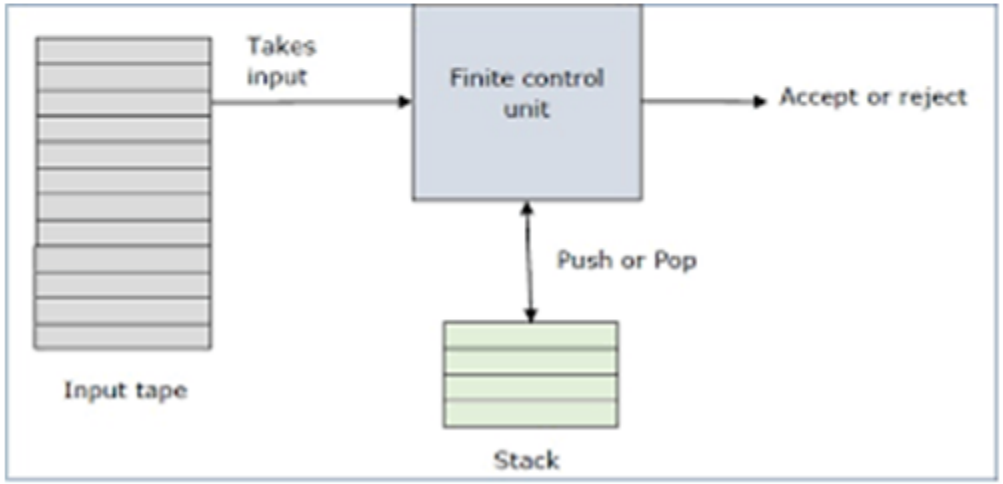




Introduction

pushdown automata:

- We also introduce pushdown automata, a class of machines recognizing the context-free languages



Chapter 2

2.1 Context-Free Grammars

CONTEXT-FREE GRAMMARS

Example : G_1 is a CFG.

$$A \rightarrow 0A1$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

CONTEXT-FREE GRAMMARS

Example : G_1 is a CFG.

substitution rules (productions)

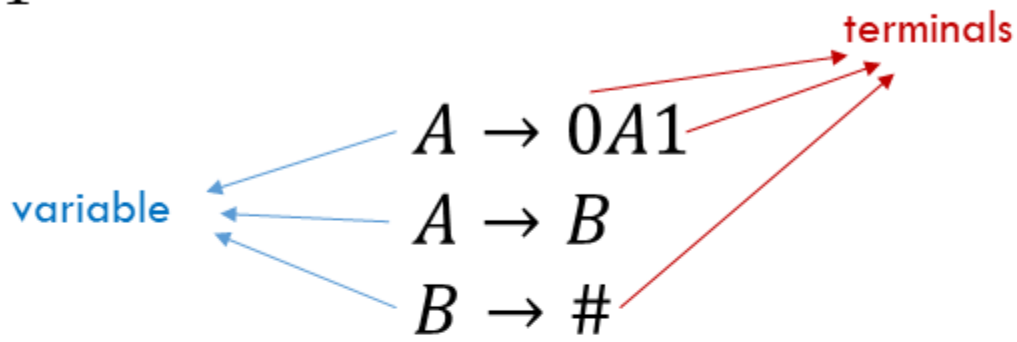
$$A \to 0A1$$
$$A \to B$$
$$B \to \#$$

$$A \to 0A1 \mid B$$

- A grammar consists of a collection of substitution rules, also called productions.
  - each rule appears as a line in the grammar

CONTEXT-FREE GRAMMARS

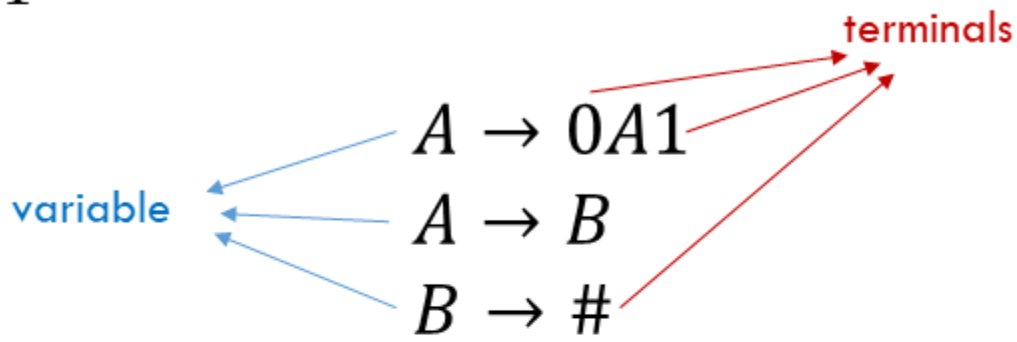
Example : G_1 is a CFG.



- A grammar consists of a collection of substitution rules, also called productions.
  - each rule appears as a line in the grammar
  - comprising a symbol (called a variable) and a string separated by an arrow. The **string** consists of variables and other symbols called terminals.

CONTEXT-FREE GRAMMARS

Example : G_1 is a CFG.



- A grammar consists of a collection of substitution rules, also called productions.
  - each rule appears as a line in the grammar
  - comprising a symbol (called a variable) and a string separated by an arrow. The **string** consists of variables and other symbols called terminals.
- **variable symbols:** often are represented by capital letters.
- **terminals** : are analogous to the input alphabet and often are represented by lowercase letters, numbers, or special symbols.

CONTEXT-FREE GRAMMARS

Example 1 : G_1 is a CFG.

start variable ← $A \rightarrow 0A1$

$A \rightarrow B$

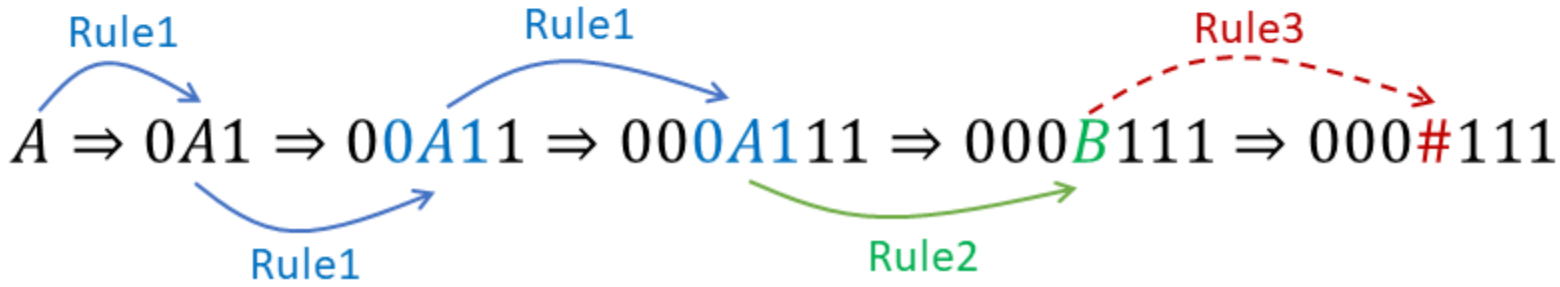
$B \rightarrow \#$

- A grammar consists of a collection of substitution rules, also called productions.
  - each rule appears as a line in the grammar
  - comprising a symbol (called a variable) and a string separated by an arrow. The **string** consists of variables and other symbols called terminals.
- **variable symbols:** often are represented by capital letters.
- **terminals** : are analogous to the input alphabet and often are represented by lowercase letters, numbers, or special symbols.
- One variable is designated as the start variable.
  - It usually occurs on the left-hand side of the topmost rule.

DERIVATION

A **grammar** describes a language by generating each string of that language in the following manner.

1. Write down the **start variable**.
2. Find a variable that is written down and a rule that starts with that variable. Replace the written down variable with the right-hand side of that rule.
3. Repeat step 2 until no variables remain.



G1

$A \rightarrow 0A1$

$A \rightarrow B$

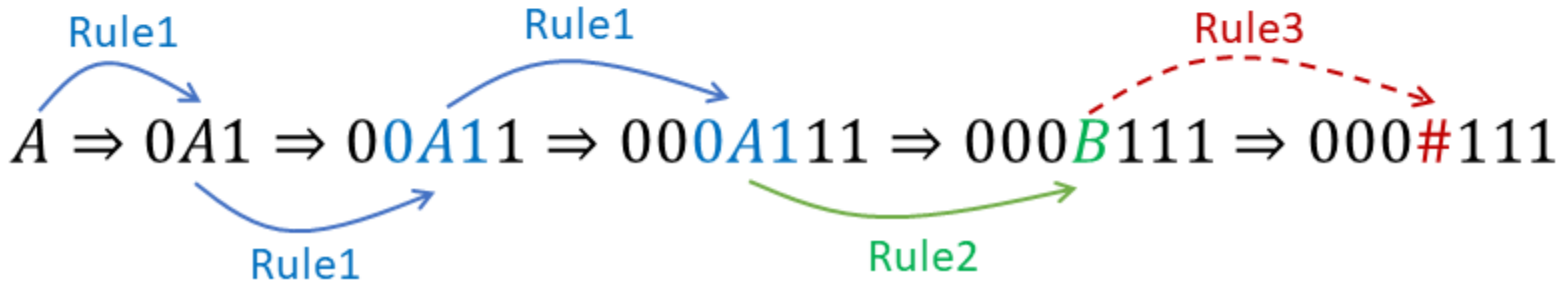
$B \rightarrow \#$

DERIVATION

A **grammar** describes a language by generating each string of that language in the following manner.

1. Write down the **start variable**.
2. Find a variable that is written down and a rule that starts with that variable. Replace the written down variable with the right-hand side of that rule.
3. Repeat step 2 until no variables remain.

- The **sequence of substitutions** to obtain a string is called a derivation.
- A derivation of string 000#111 in grammar G1 is



G1

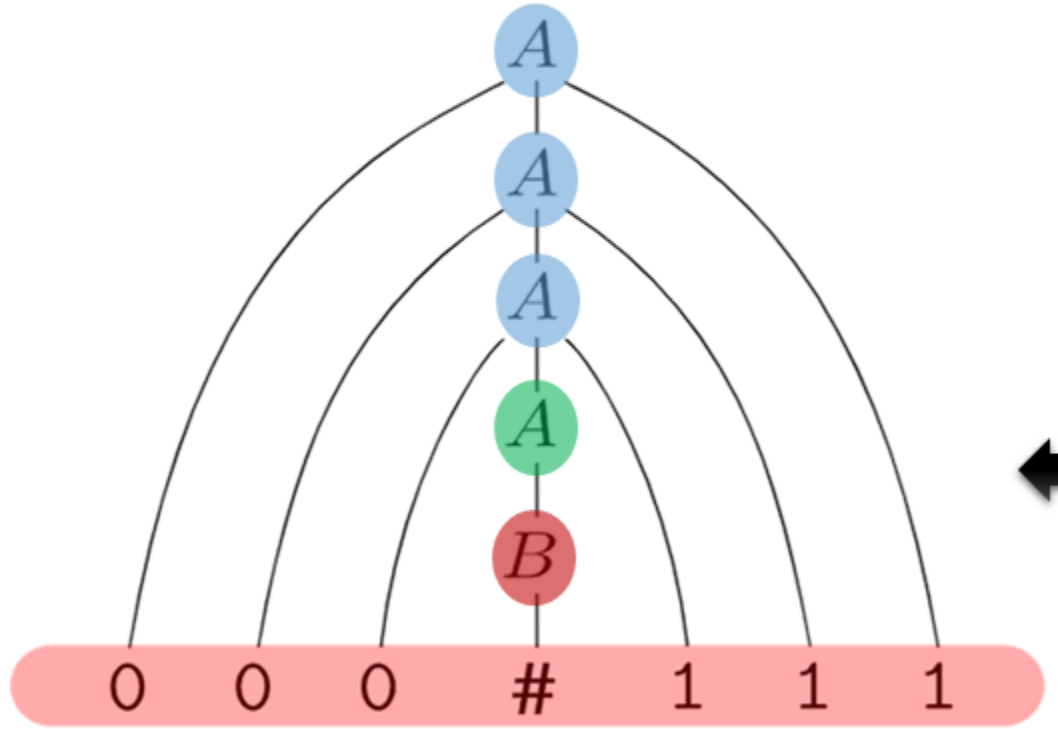
$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

PARSE TREE

You may also represent the same information pictorially with a parse tree.



G1

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

Rule1 Rule1 Rule3

$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

Rule1 Rule2

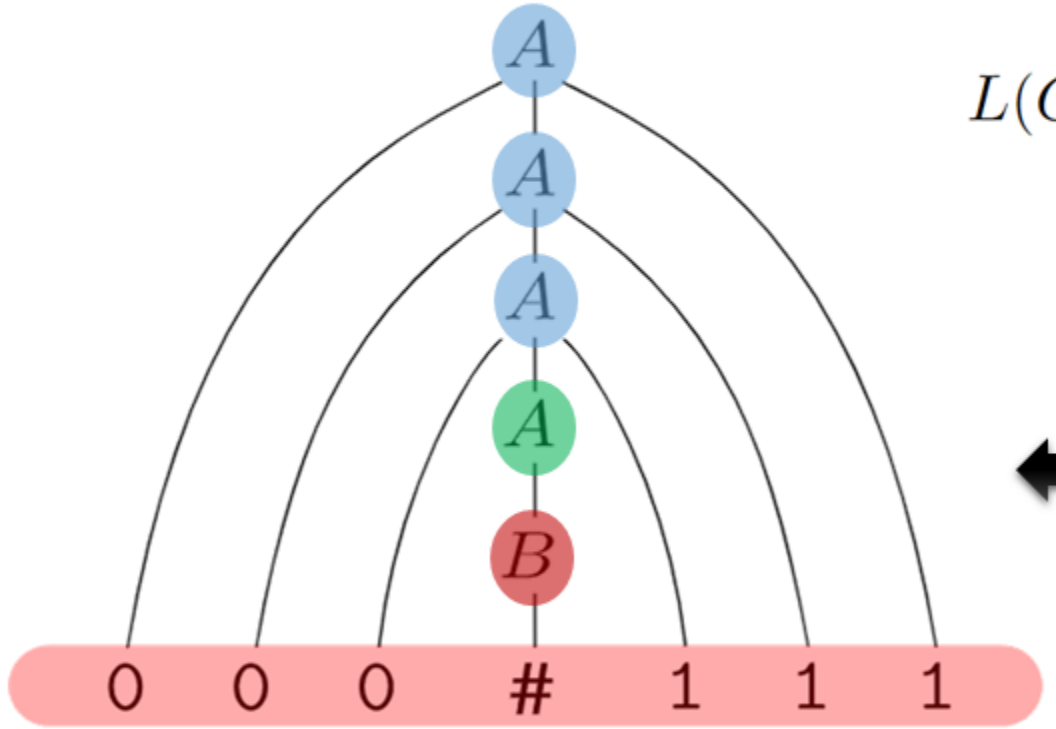
Chapter 2

PARSE TREE

You may also represent the same information pictorially with a parse tree.

Example Sentences

- A
- 0A1
- 00A11
- 000A111
- 0000A1111
- 00000A11111
- 000000A111111
- 0000000A1111111
- 00000000A11111111
- 000000000A111111111



$L(G_1)$ is $\{0^n\#1^n \mid n \geq 0\}$

G1

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

Rule1 Rule1 Rule3

$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$

Rule1 Rule2

DERIVATION

Example 2 : G_2 which describes a fragment of the English language.

$$
\begin{aligned}
\langle \text{SENTENCE} \rangle &\rightarrow \langle \text{NOUN-PHRASE} \rangle \langle \text{VERB-PHRASE} \rangle \\
\langle \text{NOUN-PHRASE} \rangle &\rightarrow \langle \text{CMPLX-NOUN} \rangle \mid \langle \text{CMPLX-NOUN} \rangle \langle \text{PREP-PHRASE} \rangle \\
\langle \text{VERB-PHRASE} \rangle &\rightarrow \langle \text{CMPLX-VERB} \rangle \mid \langle \text{CMPLX-VERB} \rangle \langle \text{PREP-PHRASE} \rangle \\
\langle \text{PREP-PHRASE} \rangle &\rightarrow \langle \text{PREP} \rangle \langle \text{CMPLX-NOUN} \rangle \\
\langle \text{CMPLX-NOUN} \rangle &\rightarrow \langle \text{ARTICLE} \rangle \langle \text{NOUN} \rangle \\
\langle \text{CMPLX-VERB} \rangle &\rightarrow \langle \text{VERB} \rangle \mid \langle \text{VERB} \rangle \langle \text{NOUN-PHRASE} \rangle \\
\langle \text{ARTICLE} \rangle &\rightarrow \texttt{a} \mid \texttt{the} \\
\langle \text{NOUN} \rangle &\rightarrow \texttt{boy} \mid \texttt{girl} \mid \texttt{flower} \\
\langle \text{VERB} \rangle &\rightarrow \texttt{touches} \mid \texttt{likes} \mid \texttt{sees} \\
\langle \text{PREP} \rangle &\rightarrow \texttt{with}
\end{aligned}
$$

DERIVATION

Example 2 : G_2 which describes a fragment of the English language.

10 variables

$$
\begin{aligned}
\langle \text{SENTENCE} \rangle &\rightarrow \langle \text{NOUN-PHRASE} \rangle \langle \text{VERB-PHRASE} \rangle \\
\langle \text{NOUN-PHRASE} \rangle &\rightarrow \langle \text{CMPLX-NOUN} \rangle \mid \langle \text{CMPLX-NOUN} \rangle \langle \text{PREP-PHRASE} \rangle \\
\langle \text{VERB-PHRASE} \rangle &\rightarrow \langle \text{CMPLX-VERB} \rangle \mid \langle \text{CMPLX-VERB} \rangle \langle \text{PREP-PHRASE} \rangle \\
\langle \text{PREP-PHRASE} \rangle &\rightarrow \langle \text{PREP} \rangle \langle \text{CMPLX-NOUN} \rangle \\
\langle \text{CMPLX-NOUN} \rangle &\rightarrow \langle \text{ARTICLE} \rangle \langle \text{NOUN} \rangle \\
\langle \text{CMPLX-VERB} \rangle &\rightarrow \langle \text{VERB} \rangle \mid \langle \text{VERB} \rangle \langle \text{NOUN-PHRASE} \rangle \\
\langle \text{ARTICLE} \rangle &\rightarrow \texttt{a} \mid \texttt{the} \\
\langle \text{NOUN} \rangle &\rightarrow \texttt{boy} \mid \texttt{girl} \mid \texttt{flower} \\
\langle \text{VERB} \rangle &\rightarrow \texttt{touches} \mid \texttt{likes} \mid \texttt{sees} \\
\langle \text{PREP} \rangle &\rightarrow \texttt{with}
\end{aligned}
$$

- 27 terminals
- the standard **English alphabet** plus a **space character**.

18 rules

DERIVATION

Example 2 : G_2 which describes a fragment of the English language.

- Strings in L(G2) include:
  - a boy sees
  - the boy sees a flower
  - a girl with a flower likes the boy

⟨SENTENCE⟩ → ⟨NOUN-PHRASE⟩⟨VERB-PHRASE⟩
⟨NOUN-PHRASE⟩ → ⟨CMPLX-NOUN⟩ | ⟨CMPLX-NOUN⟩⟨PREP-PHRASE⟩
⟨VERB-PHRASE⟩ → ⟨CMPLX-VERB⟩ | ⟨CMPLX-VERB⟩⟨PREP-PHRASE⟩
⟨PREP-PHRASE⟩ → ⟨PREP⟩⟨CMPLX-NOUN⟩
⟨CMPLX-NOUN⟩ → ⟨ARTICLE⟩⟨NOUN⟩
⟨CMPLX-VERB⟩ → ⟨VERB⟩ | ⟨VERB⟩⟨NOUN-PHRASE⟩
⟨ARTICLE⟩ → a | the
⟨NOUN⟩ → boy | girl | flower
⟨VERB⟩ → touches | likes | sees
⟨PREP⟩ → with

$CFG\ G_2$

DERIVATION

Example 2 : G_2 which describes a fragment of the English language.

- Strings in L(G2) include:
  - a boy sees
  - the boy sees a flower
  - a girl with a flower likes the boy

(a derivation)

$$
\begin{aligned}
\langle \text{SENTENCE} \rangle &\Rightarrow \langle \text{NOUN-PHRASE} \rangle \langle \text{VERB-PHRASE} \rangle \\
&\Rightarrow \langle \text{CMPLX-NOUN} \rangle \langle \text{VERB-PHRASE} \rangle \\
&\Rightarrow \langle \text{ARTICLE} \rangle \langle \text{NOUN} \rangle \langle \text{VERB-PHRASE} \rangle \\
&\Rightarrow \texttt{a}\ \langle \text{NOUN} \rangle \langle \text{VERB-PHRASE} \rangle \\
&\Rightarrow \texttt{a boy}\ \langle \text{VERB-PHRASE} \rangle \\
&\Rightarrow \texttt{a boy}\ \langle \text{CMPLX-VERB} \rangle \\
&\Rightarrow \texttt{a boy}\ \langle \text{VERB} \rangle \\
&\Rightarrow \texttt{a boy sees}
\end{aligned}
$$

$$
\begin{aligned}
\langle \text{SENTENCE} \rangle &\rightarrow \langle \text{NOUN-PHRASE} \rangle \langle \text{VERB-PHRASE} \rangle \\
\langle \text{NOUN-PHRASE} \rangle &\rightarrow \langle \text{CMPLX-NOUN} \rangle \mid \langle \text{CMPLX-NOUN} \rangle \langle \text{PREP-PHRASE} \rangle \\
\langle \text{VERB-PHRASE} \rangle &\rightarrow \langle \text{CMPLX-VERB} \rangle \mid \langle \text{CMPLX-VERB} \rangle \langle \text{PREP-PHRASE} \rangle \\
\langle \text{PREP-PHRASE} \rangle &\rightarrow \langle \text{PREP} \rangle \langle \text{CMPLX-NOUN} \rangle \\
\langle \text{CMPLX-NOUN} \rangle &\rightarrow \langle \text{ARTICLE} \rangle \langle \text{NOUN} \rangle \\
\langle \text{CMPLX-VERB} \rangle &\rightarrow \langle \text{VERB} \rangle \mid \langle \text{VERB} \rangle \langle \text{NOUN-PHRASE} \rangle \\
\langle \text{ARTICLE} \rangle &\rightarrow \texttt{a} \mid \texttt{the} \\
\langle \text{NOUN} \rangle &\rightarrow \texttt{boy} \mid \texttt{girl} \mid \texttt{flower} \\
\langle \text{VERB} \rangle &\rightarrow \texttt{touches} \mid \texttt{likes} \mid \texttt{sees} \\
\langle \text{PREP} \rangle &\rightarrow \texttt{with}
\end{aligned}
$$

$CFG\ G_2$

FORMAL DEFINITION OF A CONTEXT-FREE GRAMMAR

A ***context-free grammar*** is a 4-tuple (V, Σ, R, S), where

1. V is a finite set called the ***variables***,
2. Σ is a finite set, disjoint from V, called the ***terminals***,
3. R is a finite set of ***rules***, with each rule being a variable and a string of variables and terminals, and
4. $S \in V$ is the start variable.

G1

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

In grammar G1

- $V = \{A, B\}$
- $\Sigma = \{0, 1, \#\}$
- $S = A$

DERIVATION

- If $u, v,$ and w are strings of **variables and terminals**, and $A \rightarrow w$ is a rule of the grammar, we say that uAv yields uwv ,written $uAv \Rightarrow uwv$.

DERIVATION

- If $u, v,$ and w are strings of **variables and terminals**, and $A \rightarrow w$ is a rule of the grammar, we say that uAv yields uwv ,written $uAv \Rightarrow uwv$.

- Say that u derives v, written $u \overset{*}{\Rightarrow} v$,
  - if $u = v$ or
  - if a sequence $u_1, u_2, \ldots, u_k$ exists for k≥0 and

$$u \Rightarrow u1 \Rightarrow u2 \Rightarrow \cdots \Rightarrow uk \Rightarrow v.$$

- The **language of the grammar** is $\{w \in \Sigma^* | S \overset{*}{\Rightarrow} w\}$.

EXAMPLES OF CONTEXT-FREE GRAMMARS

- Consider grammar $G_3 = (\{S\}, \{a, b\}, R, S)$.
- The set of rules, R, is $S \rightarrow aSb|SS|\varepsilon$.

EXAMPLES OF CONTEXT-FREE GRAMMARS

- Consider grammar $G_3 = (\{S\}, \{a, b\}, R, S)$.
- The set of rules, R, is $S \rightarrow aSb|SS|\varepsilon$.

- This grammar generates :
  - strings such as $abab$, $aaabbb$, and $aababb$.

EXAMPLES OF CONTEXT-FREE GRAMMARS

- Consider grammar $G_3 = (\{S\}, \{a, b\}, R, S)$.
- The set of rules, R, is $S \rightarrow aSb|SS|\varepsilon$.

- This grammar generates :
  - strings such as $abab$, $aaabbb$, and $aababb$.
    - $S \Rightarrow SS \Rightarrow aSbS \Rightarrow aSbaSb \Rightarrow a\varepsilon baSb \Rightarrow aba\varepsilon b \Rightarrow abab$
    - $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaa\varepsilon bbb \Rightarrow aaabbb$
    - $S \Rightarrow aSb \Rightarrow aSSb \Rightarrow aaSbSb \Rightarrow aaSbaSbb \Rightarrow aa\varepsilon baSbb \Rightarrow aaba\varepsilon bb \Rightarrow aababb$

EXAMPLES OF CONTEXT-FREE GRAMMARS

- Consider grammar $G_3 = (\{S\}, \{a, b\}, R, S)$.
- The set of rules, R, is $S \rightarrow aSb|SS|\varepsilon$.

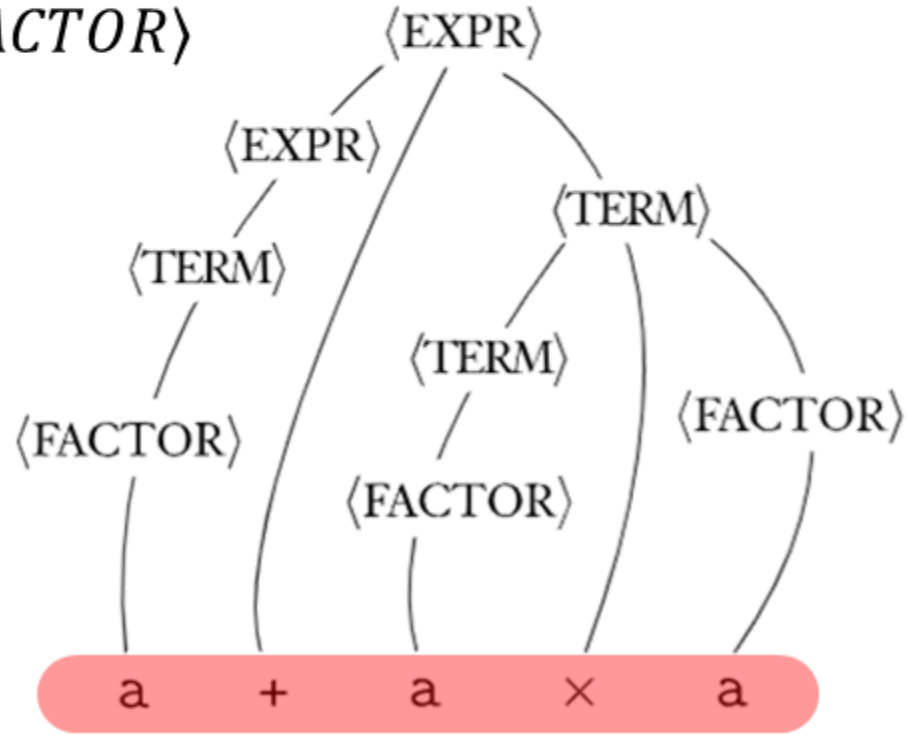
- This grammar generates :
  - strings such as $abab$, $aaabbb$, and $aababb$.
    - $S \Rightarrow SS \Rightarrow aSbS \Rightarrow aSbaSb \Rightarrow a\varepsilon ba\varepsilon b \Rightarrow abab$
    - $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaa\varepsilon bbb \Rightarrow aaabbb$
    - $S \Rightarrow aSb \Rightarrow aSSb \Rightarrow aaSbSb \Rightarrow aaSbaSbb \Rightarrow aa\varepsilon baSbb \Rightarrow aaba\varepsilon bb \Rightarrow aababb$

  - if you think of a as a left parenthes is "("and b as a right parenthes is ")"
    - Then L(G_3) is the language of all strings of properly nested parentheses.

EXAMPLES OF CONTEXT-FREE GRAMMARS

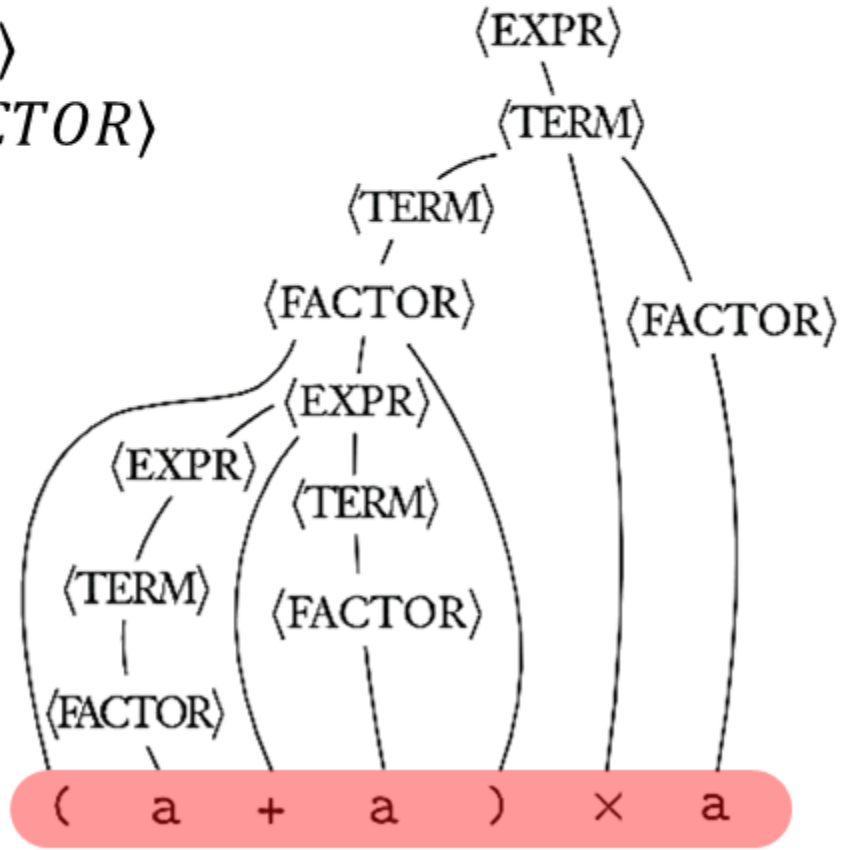
- Consider grammar $G_4 = (V, \Sigma, R, \langle EXPR \rangle)$
- $V = \{\langle EXPR \rangle, \langle TERM \rangle, \langle FACTOR \rangle\}$
- $\Sigma = \{a, +, \times, (,)\}$.
- The **rules** are
  - $\langle \boldsymbol{EXPR} \rangle \rightarrow \langle EXPR \rangle + \langle TERM \rangle | \langle TERM \rangle$
  - $\langle \boldsymbol{TERM} \rangle \rightarrow \langle TERM \rangle \times \langle FACTOR \rangle | \langle FACTOR \rangle$
  - $\langle \boldsymbol{FACTOR} \rangle \rightarrow (\langle EXPR \rangle) | a$

- **Grammar G_4** describes :
  - a fragment of a programming language concerned with arithmetic expressions.



EXAMPLES OF CONTEXT-FREE GRAMMARS

- Consider grammar $G_4 = (V, \Sigma, R, \langle EXPR \rangle)$
- $V = \{\langle EXPR \rangle, \langle TERM \rangle, \langle FACTOR \rangle\}$
- $\Sigma = \{a, +, \times, (,)\}$.
- The **rules** are
  - $\langle \boldsymbol{EXPR} \rangle \rightarrow \langle EXPR \rangle + \langle TERM \rangle | \langle TERM \rangle$
  - $\langle \boldsymbol{TERM} \rangle \rightarrow \langle TERM \rangle \times \langle FACTOR \rangle | \langle FACTOR \rangle$
  - $\langle \boldsymbol{FACTOR} \rangle \rightarrow (\langle EXPR \rangle) | a$



DESIGNING CONTEXT-FREE GRAMMARS

- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

DESIGNING CONTEXT-FREE GRAMMARS

- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 1: Based on this Fact: Many CFLs are the union of simpler CFLs.

DESIGNING CONTEXT-FREE GRAMMARS

- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 1: Based on this Fact: Many CFLs are the union of simpler CFLs.

- break a CFL into simpler pieces
- and then construct individual grammars for each piece
- merge the individual grammars into a grammar for the original language
  - By combining their rules and then adding the new rule

$$S \to S_1 | S_2 | \cdots | S_k,$$

Where the variables S_i are the start variables for the individual grammars.

DESIGNING CONTEXT-FREE GRAMMARS

- Example: Construct a grammar for the language

$$\{0^n 1^n | n \geq 0\} \cup \{1^n 0^n | n \geq 0\}.$$

DESIGNING CONTEXT-FREE GRAMMARS

- Example: Construct a grammar for the language

$$\{0^n 1^n | n \geq 0\} \cup \{1^n 0^n | n \geq 0\}.$$

- $L_1 = \{0^n 1^n | n \geq 0\}$ Rules: $S_1 \rightarrow 0S_1 1 | \varepsilon$

DESIGNING CONTEXT-FREE GRAMMARS

- Example: Construct a grammar for the language

$$\{0^n 1^n | n \geq 0\} \cup \{1^n 0^n | n \geq 0\}.$$

- $L_1 = \{0^n 1^n | n \geq 0\}$ Rules: $S_1 \rightarrow 0S_1 1 | \varepsilon$
- $L_2 = \{1^n 0^n | n \geq 0\}$ Rules: $S_2 \rightarrow 1S_2 0 | \varepsilon$

DESIGNING CONTEXT-FREE GRAMMARS

- Example: Construct a grammar for the language

$$\{0^n 1^n | n \geq 0\} \cup \{1^n 0^n | n \geq 0\}.$$

- $L_1 = \{0^n 1^n | n \geq 0\}$ Rules: $S_1 \rightarrow 0S_1 1 | \varepsilon$
- $L_2 = \{1^n 0^n | n \geq 0\}$ Rules: $S_2 \rightarrow 1S_2 0 | \varepsilon$

- $L_1 \cup L_2$ Rules: $S \rightarrow S1 | S2$
  $S_1 \rightarrow 0S_1 1 | \varepsilon$
  $S_2 \rightarrow 1S_2 0 | \varepsilon$

DESIGNING CONTEXT-FREE GRAMMARS

- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 2: constructing a CFG for a regular language :

DESIGNING CONTEXT-FREE GRAMMARS

- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 2: constructing a CFG for a regular language :

- construct a DFA for that language.

DESIGNING CONTEXT-FREE GRAMMARS

- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 2: constructing a CFG for a regular language :

- construct a DFA for that language.
- convert the DFA into an equivalent CFG as follows.

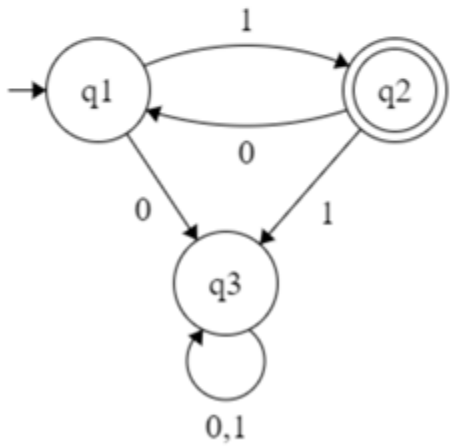
DESIGNING CONTEXT-FREE GRAMMARS

- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 2: constructing a CFG for a regular language :

- construct a DFA for that language.
- convert the DFA into an equivalent CFG as follows.



DESIGNING CONTEXT-FREE GRAMMARS

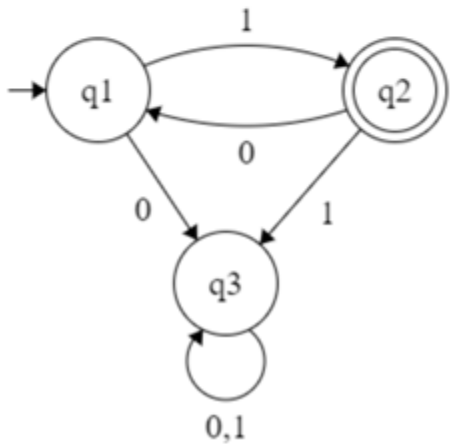
- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 2: constructing a CFG for a regular language :

- construct a DFA for that language.
- convert the DFA into an equivalent CFG as follows.
  - Make a variable R_i for each state q_i of the DFA.

- $V = \{R1, R2, R3\}$



DESIGNING CONTEXT-FREE GRAMMARS

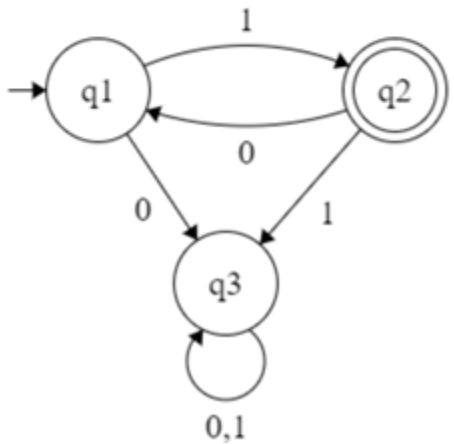
- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 2: constructing a CFG for a regular language :

- construct a DFA for that language.
- convert the DFA into an equivalent CFG as follows.
  - Make a variable R_i for each state q_i of the DFA.
  - Add the rule $R_i \rightarrow aR_j$ to the CFG if $\delta(q_i,a)=q_j$.

- $V = \{R1, R2, R3\}$
- $R_1 \rightarrow 0R_3 \mid 1R_2 \quad R_2 \rightarrow 0R_1 \mid 1R_3 \quad R_3 \rightarrow 1R_3 \mid 1R_3$



DESIGNING CONTEXT-FREE GRAMMARS

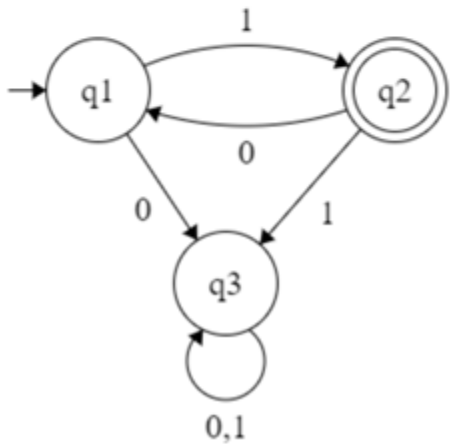
- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 2: constructing a CFG for a regular language :

- construct a DFA for that language.
- convert the DFA into an equivalent CFG as follows.
  - Make a variable R_i for each state q_i of the DFA.
  - Add the rule $R_i \rightarrow aR_j$ to the CFG if $\delta(q_i,a)=q_j$.
  - Add the rule $R_i \rightarrow \varepsilon$ if q_i is an accept state of the DFA.

- $V = \{R1, R2, R3\}$
- $\boldsymbol{R_1} \rightarrow 0R_3 \mid 1R_2 \quad R_2 \rightarrow 0R_1 \mid 1R_3 \quad R_3 \rightarrow 1R_3 \mid 1R_3$
- $R_2 \rightarrow \varepsilon$



DESIGNING CONTEXT-FREE GRAMMARS

- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 2: constructing a CFG for a regular language :

- construct a DFA for that language.
- convert the DFA into an equivalent CFG as follows.
  - Make a variable R_i for each state q_i of the DFA.
  - Add the rule $R_i \rightarrow aR_j$ to the CFG if $\delta(q_i,a)=q_j$.
  - Add the rule $R_i \rightarrow \varepsilon$ if q_i is an accept state of the DFA.
  - Make R_0 the start variable of the grammar, where q_0 is the start state of the machine.
  - the resulting CFG generates the same language that the DFA recognizes. (Verify it on your own)

DESIGNING CONTEXT-FREE GRAMMARS

- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 3: constructing a CFG for a Certain context-free languages :

DESIGNING CONTEXT-FREE GRAMMARS

- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 3: constructing a CFG for a Certain context-free languages :

- strings with two substrings that are "linked"
  - need to remember an unbounded amount of information about one of the substrings
    - to verify that it corresponds properly to the other substring.
  - **Example**: $\{0^n 1^n | n \geq 0\}$

DESIGNING CONTEXT-FREE GRAMMARS

- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 3: constructing a CFG for a Certain context-free languages :

- strings with two substrings that are "linked"
  - need to remember an unbounded amount of information about one of the substrings
    - to verify that it corresponds properly to the other substring.
  - **Example**: $\{0^n 1^n | n \geq 0\}$

- **construct a CFG:** by using a rule of the form $R \rightarrow uRv$

DESIGNING CONTEXT-FREE GRAMMARS

- The design of context-free grammars requires creativity.
- Trickier to construct than finite automata

Helpful techniques

Method 4: constructing a CFG for a language with a recursive structure:

- **Example**:

- Consider grammar $G_4 = (V, \Sigma, R, \langle EXPR \rangle)$
- $V = \{\langle EXPR \rangle, \langle TERM \rangle, \langle FACTOR \rangle\}$
- $\Sigma = \{a, +, \times, (,)\}$.
- The **rules** are
  - $\langle \boldsymbol{EXPR} \rangle \rightarrow \langle EXPR \rangle + \langle TERM \rangle | \langle TERM \rangle$
  - $\langle \boldsymbol{TERM} \rangle \rightarrow \langle TERM \rangle \times \langle FACTOR \rangle | \langle FACTOR \rangle$
  - $\langle \boldsymbol{FACTOR} \rangle \rightarrow (\langle EXPR \rangle) | a$

AMBIGUITY

- Sometimes a grammar can generate the same string in several different ways.
  - We say, the string is derived ambiguously in that grammar.

AMBIGUITY

- Sometimes a grammar can generate the same string in several different ways.
  - We say, the string is derived ambiguously in that grammar.

- If a grammar generates some string ambiguously,
  - then the grammar is ambiguous.

AMBIGUITY

- Sometimes a grammar can generate the same string in several different ways.
  - We say, the string is derived ambiguously in that grammar.

- If a grammar generates some string ambiguously,
  - then the grammar is ambiguous.

- Ambiguity is because of
  - several different parse trees (not two different derivations.)

AMBIGUITY

- Sometimes a grammar can generate the same string in several different ways.
  - We say, the string is derived ambiguously in that grammar.

- If a grammar generates some string ambiguously,
  - then the grammar is ambiguous.

- Ambiguity is because of
  - several different parse trees (not two different derivations.)
  - thus several different meanings
  - **undesirable** for certain applications, such as programming languages
    - They need a unique interpretation.

AMBIGUITY

Example: consider grammarG5:

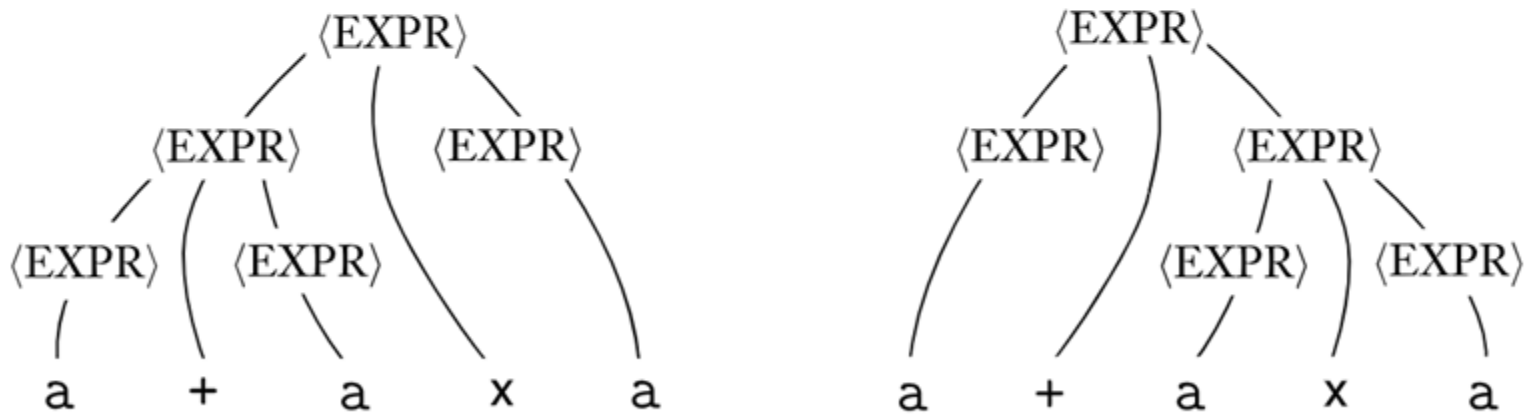
$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle | \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle | (\langle \text{EXPR} \rangle) | \text{a}$$

AMBIGUITY

Example: consider grammarG5:

$$\langle \text{EXPR} \rangle \rightarrow \langle \text{EXPR} \rangle + \langle \text{EXPR} \rangle | \langle \text{EXPR} \rangle \times \langle \text{EXPR} \rangle | (\langle \text{EXPR} \rangle) | \text{a}$$

This grammar generates the string a + a × a ambiguously.



LEFTMOST AND RIGHTMOST DERIVATION

- A derivation of a string w in a grammar G is a leftmost derivation
  - if **at every step** the leftmost remaining variable is the one replaced.

- A derivation of a string w in a grammar G is a rightmost derivation
  - if **at every step** the rightmost remaining variable is the one replaced.

LEFTMOST AND RIGHTMOST DERIVATION

Example: Let any set of production rules in a CFG be

$$X \rightarrow X + X \mid X \times X \mid |X| \; a$$

over an alphabet $\{a\}$. We want to generate $a + a \times a$.

LEFTMOST AND RIGHTMOST DERIVATION

Example: Let any set of production rules in a CFG be

$$X \rightarrow X + X \mid X \times X \mid |X| \; a$$

over an alphabet $\{a\}$. We want to generate $a + a \times a$.

- leftmost derivation :
  - $X \rightarrow X + X \rightarrow a + X \rightarrow a + X \times X \rightarrow a + a \times X \rightarrow a + a \times a$

LEFTMOST AND RIGHTMOST DERIVATION

Example: Let any set of production rules in a CFG be

$$X \rightarrow X + X \mid X \times X \mid |X| \; a$$

over an alphabet $\{a\}$. We want to generate $a + a \times a$.

- leftmost derivation :
  - $X \rightarrow X + X \rightarrow a + X \rightarrow a + X \times X \rightarrow a + a \times X \rightarrow a + a \times a$
- rightmost derivation
  - $X \rightarrow X \times X \rightarrow X \times a \rightarrow X + X \times a \rightarrow X + a \times a \rightarrow a + a \times a$

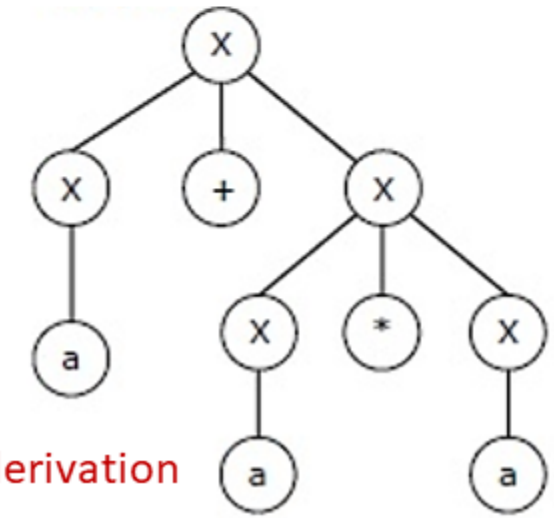
LEFTMOST AND RIGHTMOST DERIVATION

Example: Let any set of production rules in a CFG be

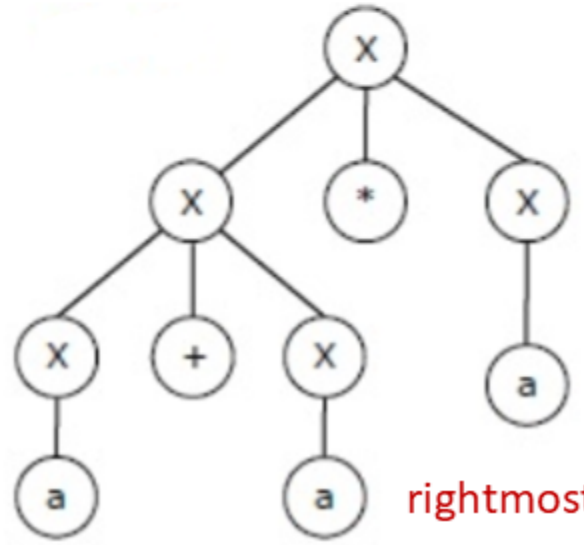
$$X \rightarrow X + X \mid X \times X \mid |X| \mid a$$

over an alphabet $\{a\}$. We want to generate $a + a \times a$.

- leftmost derivation :
  - $X \rightarrow X + X \rightarrow a + X \rightarrow a + X \times X \rightarrow a + a \times X \rightarrow a + a \times a$
- rightmost derivation
  - $X \rightarrow X \times X \rightarrow X \times a \rightarrow X + X \times a \rightarrow X + a \times a \rightarrow a + a \times a$



leftmost derivation



rightmost derivation

AMBIGUITY AND LEFTMOST AND RIGHTMOST DERIVATION

- A string w is derived ambiguously in context-free grammar G if it has two or more different leftmost derivations.

- Grammar G is ambiguous if it generates some string ambiguously.

AMBIGUITY AND LEFTMOST AND RIGHTMOST DERIVATION

- A string w is derived ambiguously in context-free grammar G if it has two or more different leftmost derivations.
- Grammar G is ambiguous if it generates some string ambiguously.
- Some context-free languages, can be generated
  - **using an ambiguous grammar and an unambiguous grammar**.
- Some context-free languages, however, can be generated
  - only by ambiguous grammars.
  - Such languages are called inherently ambiguous.

CHOMSKY NORMAL FORM

- One of the simplest and most useful forms of context-free grammars is called Chomsky normal form.

CHOMSKY NORMAL FORM

- One of the simplest and most useful forms of context-free grammars is called Chomsky normal form.

A context-free grammar is in ***Chomsky normal form*** if every rule is of the form

$$A \to BC$$
$$A \to a$$

where a is any terminal and A, B, and C are any variables—except that B and C may not be the start variable. In addition, we permit the rule $S \to \varepsilon$, where S is the start variable.

CHOMSKY NORMAL FORM

- Theorem: Any context-free language is generated by a context-free grammar in Chomsky normal form.

CHOMSKY NORMAL FORM

Theorem: Any context-free language is generated by a context-free grammar in Chomsky normal form.

PROOF:

1. **Eliminate the start symbol from right-hand sides**
   - we add a new start variable S_0 and a new rule $S_0 \rightarrow S$,
   - where S was the original start variable.

CHOMSKY NORMAL FORM

- Theorem: Any context-free language is generated by a context-free grammar in Chomsky normal form.

PROOF:

1. **Eliminate the start symbol from right-hand sides**
   - we add a new start variable S_0 and a new rule $S_0 \rightarrow S$,
   - where S was the original start variable.

$$S \rightarrow ASA \mid aB$$
$$A \rightarrow B \mid S$$
$$B \rightarrow b \mid \varepsilon$$

$$S0 \rightarrow S$$
$$S \rightarrow ASA \mid aB$$
$$A \rightarrow B \mid S$$
$$B \rightarrow b \mid \varepsilon$$

CHOMSKY NORMAL FORM

$$A \rightarrow BC$$
$$A \rightarrow a$$

CHOMSKY NORMAL FORM

PROOF: (cont.)

2. Removing ε-rules.

- We remove an ε-rule $A \to \varepsilon$, where ***A* is not the start variable**.
- Then for each occurrence of an A on the right-hand side of a rule,
  - we add a new rule with that occurrence deleted.
  - In other words, if $R \to uAv$ is a rule in which u and v are strings of variables and terminals, we add rule $R \to uv$.

$$
\begin{aligned}
S0 &\to S \\
S &\to ASA|aB \\
A &\to B|S \\
B &\to b|\varepsilon
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
S0 &\to S \\
S &\to ASA|aB|a \\
A &\to B|S|\varepsilon \\
B &\to b|\varepsilon
\end{aligned}
$$

CHOMSKY NORMAL FORM

PROOF: (cont.)

2. **Removing ε-rules.** (cont.)
   - We do so **for each occurrence of an A**,
   - **Example**:
     - $R \to \varepsilon,\ R \to uAvAw \quad \Rightarrow\ R \to uvAw \mid uAvw \mid uvw.$
     - $R \to \varepsilon,\ R \to A \quad \Rightarrow\ R \to A \mid \varepsilon$ (unless we had previously removed it)

   - We **repeat these steps** until we eliminate all ε-rules not involving the start variable.

CHOMSKY NORMAL FORM

PROOF: (cont.)

3. we handle all unit rules.

- We remove a unit rule $A \to B$.
- Then, whenever a rule $B \to u$ appears,
  - we add the rule $A \to u$ unless this was a unit rule previously removed.
  - As before, u is a string of variables and terminals.
- We repeat these steps until we eliminate all unit rules.

$$\begin{matrix} A \to B \\ B \to CDEF \end{matrix} \quad \Rightarrow \quad A \to CDEF$$

CHOMSKY NORMAL FORM

PROOF: (cont.)

4. **Finally, we convert all remaining rules into the proper form.**
   - (Eliminate right-hand sides with more than 2 nonterminals)

CHOMSKY NORMAL FORM

PROOF: (cont.)

4. Finally, we convert all remaining rules into the proper form.

- (Eliminate right-hand sides with more than 2 nonterminals)
- We replace each rule $A \to u_1 u_2 \cdots u_k$,
  - where k≥3 and each u_i is a variable or terminal symbol,
- with the rules $A \to u_1 A_1$, $A_1 \to u_2 A_2$, $A_2 \to u_3 A_3$,…,and $A_{k-2} \to u_{k-1} u_k$.
  - The A_i's are new variables.

CHOMSKY NORMAL FORM

PROOF: (cont.)

4. Finally, we convert all remaining rules into the proper form.

- (Eliminate right-hand sides with more than 2 nonterminals)
- We replace each rule $A \rightarrow u_1 u_2 \cdots u_k$,
  - where k≥3 and each u_i is a variable or terminal symbol,
- with the rules $A \rightarrow u_1 A_1$, $A_1 \rightarrow u_2 A_2$, $A_2 \rightarrow u_3 A_3$,...,and $A_{k-2} \rightarrow u_{k-1} u_k$.
  - The A_i's are new variables.
- We replace any terminal u_i in the preceding rule(s) with the new variable U_i and add the rule $U_i \rightarrow u_i$.

CHOMSKY NORMAL FORM

PROOF: (cont.)

4. Finally, we convert all remaining rules into the proper form.

- (Eliminate right-hand sides with more than 2 nonterminals)
- We replace each rule $A \to u_1 u_2 \cdots u_k$,
  - where k≥3 and each u_i is a variable or terminal symbol,
- with the rules $A \to u_1 A_1$, $A_1 \to u_2 A_2$, $A_2 \to u_3 A_3$,...,and $A_{k-2} \to u_{k-1} u_k$.
  - The A_i's are new variables.
- We replace any terminal u_i in the preceding rule(s) with the new variable U_i and add the rule $U_i \to u_i$.

$$S0 \to ASA|aB|a|SA|AS$$
$$S \to ASA|aB|a|SA|AS$$
$$A \to S|b|ASA|aB|a|SA|AS$$
$$B \to b$$

$$S0 \to AA_1|UB|a|SA|AS$$
$$S \to AA_1|UB|a|SA|AS$$
$$A \to b|AA_1|UB|a|SA|AS$$
$$A_1 \to SA$$
$$U \to a$$
$$B \to b$$

CHOMSKY NORMAL FORM

Example: Convert the following CFG to CNF

$$S \rightarrow ASA|aB$$
$$A \rightarrow B|S$$
$$B \rightarrow b|\varepsilon$$

- **Step1:** Eliminate the start symbol from right-hand sides and add **a new start symbol**

$$S0 \rightarrow S$$
$$S \rightarrow ASA|aB$$
$$A \rightarrow B|S$$
$$B \rightarrow b|\varepsilon$$

CHOMSKY NORMAL FORM

Example: (Cont.)

- **Step2:** Eliminate all ε-rules

$$S0 \rightarrow S$$
$$S \rightarrow ASA|aB$$
$$A \rightarrow B|S$$
$$B \rightarrow b|\varepsilon$$

Previous rules

$$S0 \rightarrow S$$
$$S \rightarrow ASA|aB|a$$
$$A \rightarrow B|S|\varepsilon$$
$$B \rightarrow b|\varepsilon$$

$$S0 \rightarrow S$$
$$S \rightarrow ASA|aB|a|SA|AS|S$$
$$A \rightarrow B|S|\varepsilon$$
$$B \rightarrow b$$

CHOMSKY NORMAL FORM

Example: (Cont.)

- **Step3a:** Eliminate all unit rules; $S \rightarrow S$ and $S0 \rightarrow S$

$S0 \rightarrow S$
$S \rightarrow ASA|aB|a|SA|AS|S$
$A \rightarrow B|S|\varepsilon$
$B \rightarrow b$

Previous rules

$S0 \rightarrow S$
$S \rightarrow ASA|aB|a|SA|AS|S$
$A \rightarrow B|S$
$B \rightarrow b$

$S0 \rightarrow ASA|aB|a|SA|AS$
$S \rightarrow ASA|aB|a|SA|AS$
$A \rightarrow B|S$
$B \rightarrow b$

CHOMSKY NORMAL FORM

Example: (Cont.)

- **Step3b:** Eliminate all unit rules; $A \rightarrow B$ and $A \rightarrow S$

$$S0 \rightarrow S|ASA|aB|a|SA|AS$$
$$S \rightarrow ASA|aB|a|SA|AS$$
$$A \rightarrow B|S$$
$$B \rightarrow b$$

Previous rules

$$S0 \rightarrow ASA|aB|a|SA|AS$$
$$S \rightarrow ASA|aB|a|SA|AS$$
$$A \rightarrow B|S|b$$
$$B \rightarrow b$$

$$S0 \rightarrow ASA|aB|a|SA|AS$$
$$S \rightarrow ASA|aB|a|SA|AS$$
$$A \rightarrow S|b|ASA|aB|a|SA|AS$$
$$B \rightarrow b$$

CHOMSKY NORMAL FORM

Example: (Cont.)

- **Step4:** Convert the remaining rules into the proper form by adding additional variables and rules.

$$S0 \rightarrow ASA|aB|a|SA|AS$$
$$S \rightarrow ASA|aB|a|SA|AS$$
$$A \rightarrow S|b|ASA|aB|a|SA|AS$$
$$B \rightarrow b$$

Previous rules

$$S0 \rightarrow AA_1|UB|a|SA|AS$$
$$S \rightarrow AA_1|UB|a|SA|AS$$
$$A \rightarrow b|AA_1|UB|a|SA|AS$$
$$A_1 \rightarrow SA$$
$$U \rightarrow a$$
$$B \rightarrow b$$

$$A \rightarrow BC$$
$$A \rightarrow a$$