Theory of Computation
CSC 339 – Spring 2021

Chapter-2: part1
Context-free Languages

King Saud University

Department of Computer Science

Dr. Azzam Alsudais

Outline

- Introduction
- Context-free grammar (CFG)

Introduction

- **Finite automata and regular expressions have some limitations**
  - **FA's main limitation is that they don't have enough memory to support some kinds of languages.**
    - **$\{0^n 1^n \mid n \geq 0\}$**
- **Now, we will study other (more powerful) methods of describing languages.**
  - **Context-free Grammars (CFG) → context-free languages (CFL)**

Context-free Grammar

Context-Free Languages

$$\{a^n b^n : n \geq 0\} \qquad \{ww^R\}$$

Regular Languages

$$a^*b^* \qquad (a+b)^*$$

Context-free Grammar

$$\boxed{\begin{array}{l} A \rightarrow 0A1 \\ A \rightarrow B \\ B \rightarrow \# \end{array}}$$

Context-free Grammar

$$\boxed{\begin{array}{l} A \rightarrow 0A1 \\ A \rightarrow B \\ B \rightarrow \# \end{array}}$$

- **A grammar consists of:**
  - **Substitution rules (productions)**

Context-free Grammar

$$\boxed{A \rightarrow 0A1}$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

- **A grammar consists of:**
  - **Substitution rules (productions)**

Context-free Grammar

$$\begin{array}{|l|}\hline A \rightarrow 0A1 \\ \boxed{A \rightarrow B} \\ B \rightarrow \# \\ \hline\end{array}$$

- **A grammar consists of:**
  - **Substitution rules (productions)**

Context-free Grammar

$$\begin{array}{|l|}\hline A \rightarrow 0A1 \\ A \rightarrow B \\ \hline B \rightarrow \# \\ \hline \end{array}$$

- **A grammar consists of:**
  - **Substitution rules (productions)**

Context-free Grammar

$$\begin{array}{|l|}\hline \boxed{\begin{array}{l} A \\ A \\ B \end{array}} \begin{array}{l} \rightarrow 0A1 \\ \rightarrow B \\ \rightarrow \# \end{array} \\ \hline \end{array}$$

- **A grammar consists of:**
  - **Substitution rules (productions)**
  - **Variables**

Context-free Grammar

$$\begin{aligned} \boxed{A} &\rightarrow 0A1 \\ A &\rightarrow B \\ B &\rightarrow \# \end{aligned}$$

- **A grammar consists of:**
  - **Substitution rules (productions)**
  - **Variables**
  - **Start variable**

Context-free Grammar

$$\begin{array}{l} A \to \boxed{0}A\boxed{1} \\ A \to B \\ B \to \boxed{\#} \end{array}$$

- A grammar consists of:
  - Substitution rules (productions)
  - Variables
  - Start variable
  - Terminals

Context-free Grammar

$$
\begin{aligned}
A &\rightarrow 0\boxed{A}1 \\
A &\rightarrow B \\
B &\rightarrow \#
\end{aligned}
$$

- **Recursion**
  - **One of the unique characteristics of CFG, which gives it more power over FA and Regex**

Context-free Grammar

$$\boxed{\begin{array}{l} A \rightarrow 0A1 \\ A \rightarrow B \\ B \rightarrow \# \end{array}}$$

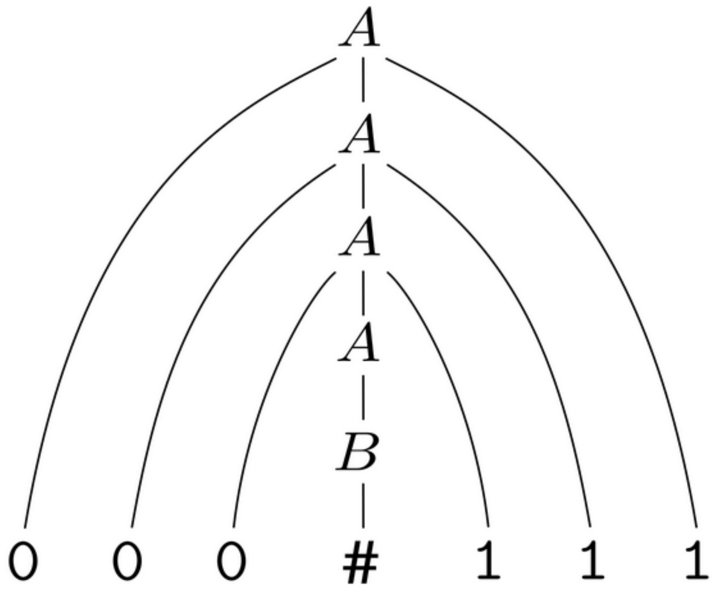
- **Derivation**
  - **The sequence of substitutions to obtain a string**
  - **e.g.,**

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

Context-free Grammar

$$
\begin{aligned}
&A \rightarrow 0A1\\
&A \rightarrow B\\
&B \rightarrow \#
\end{aligned}
$$

- **Parse Tree**
  - **Represent the same information via a tree**



Context-free Grammar

$$\boxed{\begin{array}{l} A \rightarrow 0A1 \\ A \rightarrow B \\ B \rightarrow \# \end{array}}$$

- **All strings generated using a grammar constitute the <u>language of the grammar</u>.**
    - **L(G) is the language for grammar G**

Context-free Grammar

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

Can be simplified

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

- **All strings generated using a grammar constitute the <u>language of the grammar</u>.**
  - **L(G) is the language for grammar G**

Context-free Grammar

$$\begin{aligned} A &\rightarrow 0A1 \\ A &\rightarrow B \\ B &\rightarrow \# \end{aligned}$$

Can be simplified

$$\begin{aligned} A &\rightarrow 0A1 \boxed{|} B \\ B &\rightarrow \# \end{aligned}$$

- **All strings generated using a grammar constitute the <u>language of the grammar</u>.**
  - **L(G) is the language for grammar G**

Context-free Grammar: Formal Definition (2.2)

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

  1) V is a finite set of variables,
  2) Σ is a finite set (disjoint from V), called terminals
  3) R is a finite set of rules
  4) S $\in$ V is the start variable

Context-free Grammar: Formal Definition (2.2)

- **If *u*, *v* and w are strings of variables and terminals, and A → *w* is a rule of the grammar, we say *u*A*v* yields *uwv*, written as:**

$$\boxed{uAv \Rightarrow uwv}$$

- **We say *u* derives *v*, written $u \overset{*}{\Rightarrow} v$, if *u* = *v*, or if a sequence u_1, u_2, …, u_k, for *k* ≥ 0 and**

$$\boxed{u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v}$$

Context-free Grammar: Formal Definition (2.2)

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

Strings in the CFG context may contain variables and terminals

- We say u derives v, written $u \overset{*}{\Rightarrow} v$, if $u = v$, or if a sequence u_1, u_2, …, u_k, for $k \geq 0$ and

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$$

Context-free Grammar: Formal Definition (2.2)

- If u, v and w are strings of variables and terminals, and A → w is a rule of the grammar, we say uAv yields uwv, written as:

Strings in the CFG context may contain variables and terminals

- We say u derives v, written $u \overset{*}{\Rightarrow} v$, if $u = v$, or if a sequence u_1, u_2, …, u_k, for $k \geq 0$ and

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$$

$$0A1 \overset{*}{\Rightarrow} 000\#111$$

Context-free Grammar

$$\langle \text{SENTENCE} \rangle \rightarrow \langle \text{NOUN - PHRASE} \rangle \langle \text{VERB - PHRASE} \rangle$$

$$\langle \text{NOUN - PHRASE} \rangle \rightarrow \langle \text{CMPLX - NOUN} \rangle \mid \langle \text{CMPLX - NOUN} \rangle \langle \text{PREP - PHRASE} \rangle$$

$$\langle \text{VERB - PHRASE} \rangle \rightarrow \langle \text{CMPLX - VERB} \rangle \mid \langle \text{CMPLX - VERB} \rangle \langle \text{PREP - PHRASE} \rangle$$

$$\langle \text{PREP - PHRASE} \rangle \rightarrow \langle \text{PREP} \rangle \langle \text{CMPLX - NOUN} \rangle$$

$$\langle \text{CMPLX - NOUN} \rangle \rightarrow \langle \text{ARTICLE} \rangle \langle \text{NOUN} \rangle$$

$$\langle \text{CMPLX - VERB} \rangle \rightarrow \langle \text{VERB} \rangle \mid \langle \text{VERB} \rangle \langle \text{NOUN - PHRASE} \rangle$$

$$\langle \text{ARTICLE} \rangle \rightarrow \text{a} \mid \text{the}$$

$$\langle \text{NOUN} \rangle \rightarrow \text{boy} \mid \text{girl} \mid \text{flower}$$

$$\langle \text{VERB} \rangle \rightarrow \text{touches} \mid \text{likes} \mid \text{sees}$$

$$\langle \text{PREP} \rangle \rightarrow \text{with}$$

Context-free Grammar

$\langle \text{SENTENCE} \rangle \rightarrow \langle \text{NOUN - PHRASE} \rangle \langle \text{VERB - PHRASE} \rangle$

$\langle \text{NOUN - PHRASE} \rangle \rightarrow \langle \text{CMPLX - NOUN} \rangle \mid \langle \text{CMPLX - NOUN} \rangle \langle \text{PREP - PHRASE} \rangle$

$\langle \text{VERB - PHRASE} \rangle \rightarrow \langle \text{CMPLX - VERB} \rangle \mid \langle \text{CMPLX - VERB} \rangle \langle \text{PREP - PHRASE} \rangle$

$\langle \text{PREP - PHRASE} \rangle \rightarrow \langle \text{PREP} \rangle \langle \text{CMPLX - NOUN} \rangle$

$\langle \text{CMPLX - NOUN} \rangle \rightarrow \langle \text{ARTICLE} \rangle \langle \text{NOUN} \rangle$

$\langle \text{CMPLX - VERB} \rangle \rightarrow \langle \text{VERB} \rangle \mid \langle \text{VERB} \rangle \langle \text{NOUN - PHRASE} \rangle$

$\langle \text{ARTICLE} \rangle \rightarrow \text{a} \mid \text{the}$

$\langle \text{NOUN} \rangle \rightarrow \text{boy} \mid \text{girl} \mid \text{flower}$

$\langle \text{VERB} \rangle \rightarrow \text{touches} \mid \text{likes} \mid \text{sees}$

$\langle \text{PREP} \rangle \rightarrow \text{with}$

This is a grammar that describes a fragment of the English language

Context-free Grammar

$\langle \text{SENTENCE} \rangle \rightarrow \langle \text{NOUN - PHRASE} \rangle \langle \text{VERB - PHRASE} \rangle$

$\langle \text{NOUN - PHRASE} \rangle \rightarrow \langle \text{CMPLX - NOUN} \rangle \mid \langle \text{CMPLX - NOUN} \rangle \langle \text{PREP - PHRASE} \rangle$

$\langle \text{VERB - PHRASE} \rangle \rightarrow \langle \text{CMPLX - VERB} \rangle \mid \langle \text{CMPLX - VERB} \rangle \langle \text{PREP - PHRASE} \rangle$

$\langle \text{PREP - PHRASE} \rangle \rightarrow \langle \text{PREP} \rangle \langle \text{CMPLX - NOUN} \rangle$

$\langle \text{CMPLX - NOUN} \rangle \rightarrow \langle \text{ARTICLE} \rangle \langle \text{NOUN} \rangle$

$\langle \text{CMPLX - VERB} \rangle \rightarrow \langle \text{VERB} \rangle \mid \langle \text{VERB} \rangle \langle \text{NOUN - PHRASE} \rangle$

$\langle \text{ARTICLE} \rangle \rightarrow \text{a} \mid \text{the}$

$\langle \text{NOUN} \rangle \rightarrow \text{boy} \mid \text{girl} \mid \text{flower}$

$\langle \text{VERB} \rangle \rightarrow \text{touches} \mid \text{likes} \mid \text{sees}$

$\langle \text{PREP} \rangle \rightarrow \text{with}$

This is a grammar that describes a fragment of the English language

e.g.,

a boy sees

Context-free Grammar

Derivation of the string “a boy sees”

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

Context-free Grammar

Derivation of the string "a boy sees"

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

This shows how a <u>leftmost</u> derivation works

Context-free Grammar

Derivation of the string "a boy sees"

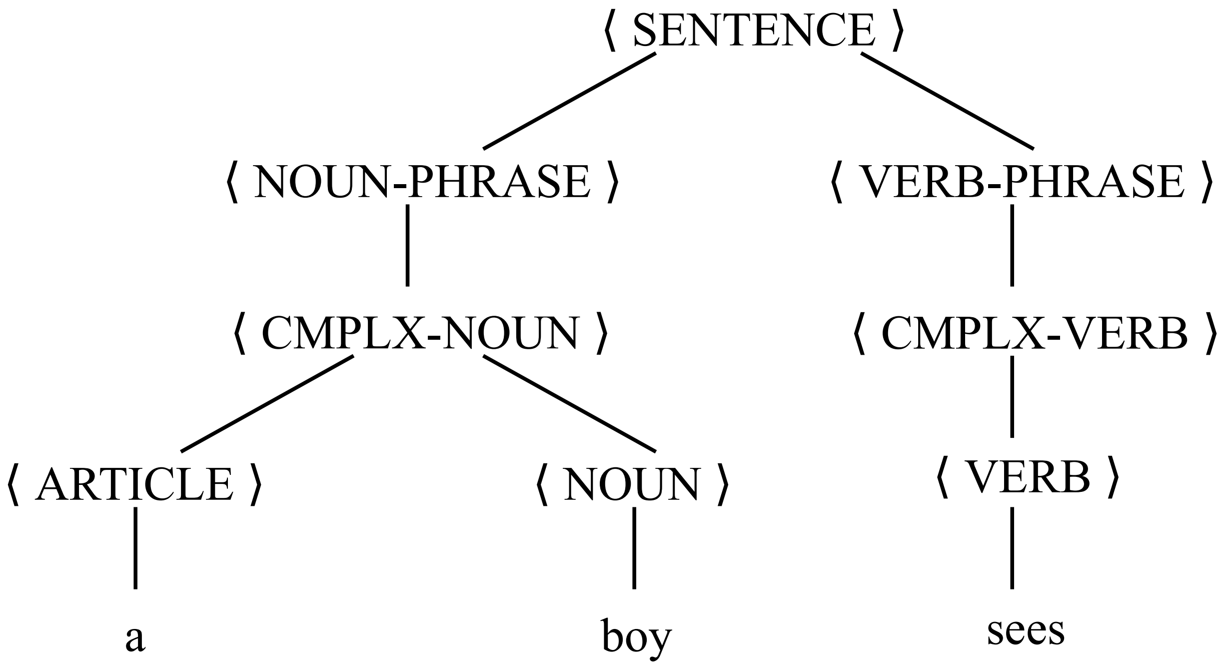
$$
\begin{aligned}
\langle \text{SENTENCE} \rangle \quad & \Rightarrow \langle \text{NOUN - PHRASE} \rangle \langle \text{VERB - PHRASE} \rangle \\
& \Rightarrow \langle \text{CMPLX - NOUN} \rangle \langle \text{VERB - PHRASE} \rangle \\
& \Rightarrow \langle \text{ARTICLE} \rangle \langle \text{NOUN}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

This shows how a <u>leftmost</u> derivation works

How would the <u>rightmost</u> derivation look like?

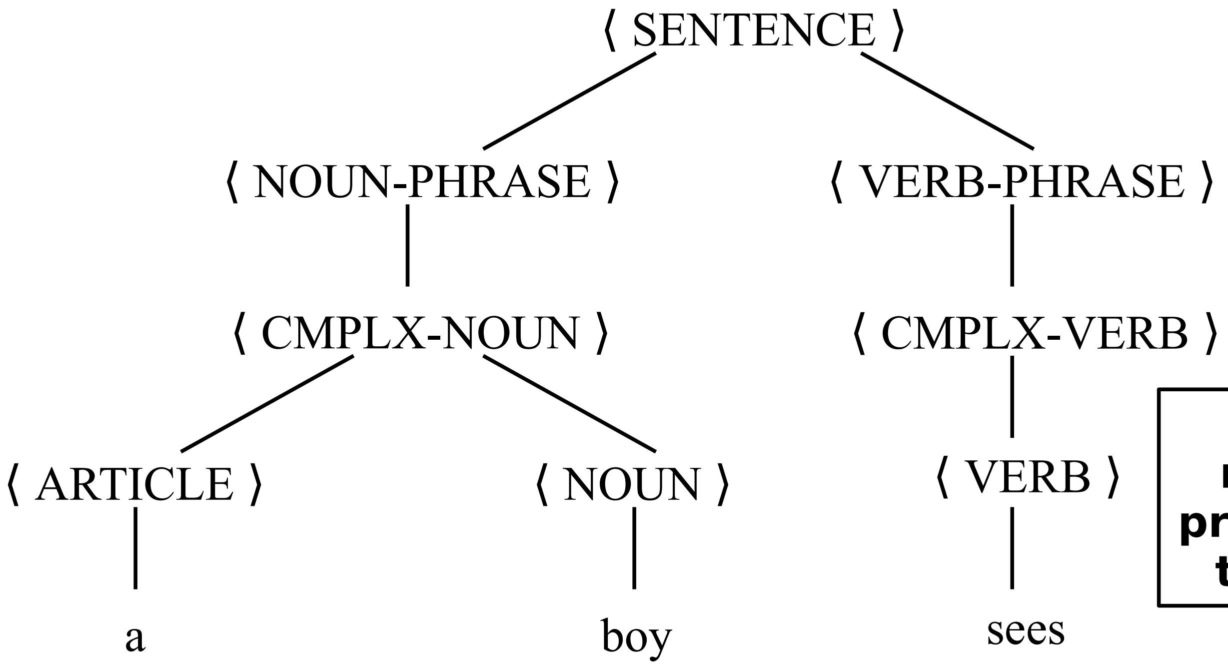
Context-free Grammar

Derivation (or parse) tree for the string “a boy sees”



Context-free Grammar

Derivation (or parse) tree for the string “a boy sees”



Both leftmost and rightmost derivation produce the same parse tree for this example

Context-free Grammar

$$\boxed{\begin{array}{l} A \rightarrow 0A1 \mid B \\ B \rightarrow \# \end{array}}$$

- **V = {A, B}**
- **Σ = {0, 1, #}**
- **S = A**
- **R is the collection of the three rules**

Context-free Grammar: Examples

$$\langle \text{ EXPR } \rangle \rightarrow \langle \text{ EXPR } \rangle + \langle \text{ TERM } \rangle \mid \langle \text{ TERM } \rangle$$
$$\langle \text{ TERM } \rangle \rightarrow \langle \text{ TERM } \rangle \text{x} \langle \text{ FACTOR } \rangle \mid \langle \text{ FACTOR } \rangle$$
$$\langle \text{ FACTOR } \rangle \rightarrow (\langle \text{ EXPR } \rangle) \mid \text{a}$$

- **V is {⟨ EXPR ⟩, ⟨ TERM ⟩, ⟨ FACTOR ⟩}**
- **Σ is {a, +, x, (,)}**

Context-free Grammar: Examples

$$\langle \text{ EXPR } \rangle \rightarrow \langle \text{ EXPR } \rangle + \langle \text{ TERM } \rangle \mid \langle \text{ TERM } \rangle$$
$$\langle \text{ TERM } \rangle \rightarrow \langle \text{ TERM } \rangle \text{x} \langle \text{ FACTOR } \rangle \mid \langle \text{ FACTOR } \rangle$$
$$\langle \text{ FACTOR } \rangle \rightarrow (\langle \text{ EXPR } \rangle) \mid \text{a}$$

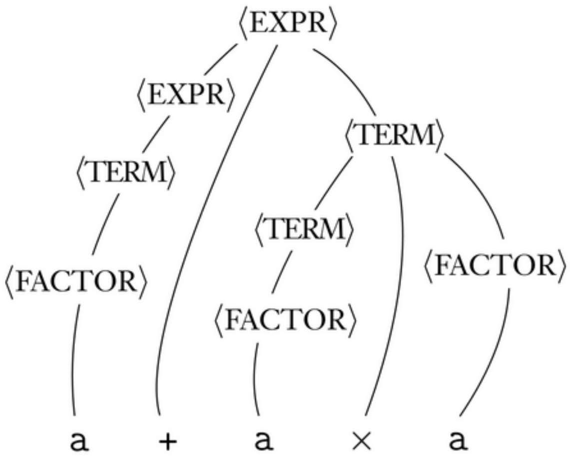
- **V is {⟨ EXPR ⟩, ⟨ TERM ⟩, ⟨ FACTOR ⟩}**
- **Σ is {a, +, x, (,)}**

Denotes a number

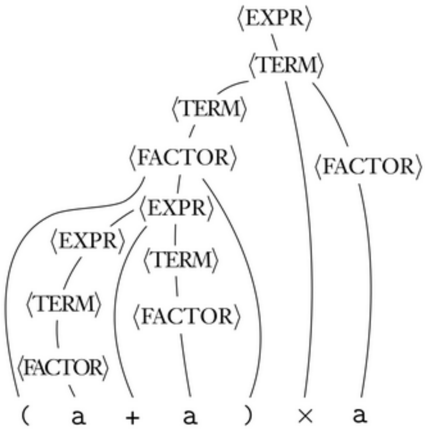
Context-free Grammar: Examples

$$\langle \text{ EXPR } \rangle \rightarrow \langle \text{ EXPR } \rangle + \langle \text{ TERM } \rangle \mid \langle \text{ TERM } \rangle$$
$$\langle \text{ TERM } \rangle \rightarrow \langle \text{ TERM } \rangle \text{x} \langle \text{ FACTOR } \rangle \mid \langle \text{ FACTOR } \rangle$$
$$\langle \text{ FACTOR } \rangle \rightarrow (\langle \text{ EXPR } \rangle) \mid \text{a}$$

a+a x a



(a+a) x a



Context-free Grammar: Designing CFG

- **Like we've seen with FA, designing a CFG requires creativity.**

Context-free Grammar: Designing CFG

- **Like we've seen with FA, designing a CFG requires creativity.**
- **But, there are some steps we can follow.**

Context-free Grammar: Designing CFG

- **Like we've seen with FA, designing a CFG requires creativity.**
- **But, there are some steps we can follow.**

  1)Start simple: many CFLs are unions of simpler CFLs.. So, we can design CFGs for individual CFLs

Context-free Grammar: Designing CFG

- **Like we've seen with FA, designing a CFG requires creativity.**

- **But, there are some steps we can follow.**

  1) **Start simple: many CFLs are unions of simpler CFLs.. So, we can design CFGs for individual CFLs**

  2) **If the language is regular (meaning there's an FA that describes it), we can convert the FA into a CFG as follows:**

Context-free Grammar: Designing CFG

- **Like we've seen with FA, designing a CFG requires creativity.**
- **But, there are some steps we can follow.**

  1) **Start simple: many CFLs are unions of simpler CFLs.. So, we can design CFGs for individual CFLs**
  2) **If the language is regular (meaning there's an FA that describes it), we can convert the FA into a CFG as follows:**
     1) **Make a variable R_i for each state q_i**

Context-free Grammar: Designing CFG

- **Like we've seen with FA, designing a CFG requires creativity.**
- **But, there are some steps we can follow.**
  1) **Start simple: many CFLs are unions of simpler CFLs.. So, we can design CFGs for individual CFLs**
  2) **If the language is regular (meaning there's an FA that describes it), we can convert the FA into a CFG as follows:**
     1) **Make a variable R_i for each state q_i**
     2) **Add rule $R_i \rightarrow aR_j$ if $\delta(q_i, a) = q_j$**

Context-free Grammar: Designing CFG

- **Like we've seen with FA, designing a CFG requires creativity.**
- **But, there are some steps we can follow.**

  1) **Start simple: many CFLs are unions of simpler CFLs.. So, we can design CFGs for individual CFLs**
  2) **If the language is regular (meaning there's an FA that describes it), we can convert the FA into a CFG as follows:**
     1) **Make a variable R_i for each state q_i**
     2) **Add rule $R_i \rightarrow aR_j$ if $\delta(q_i, a) = q_j$**
     3) **Add rule $R_i \rightarrow \varepsilon$ if q_i is an accept state**

Context-free Grammar: Designing CFG

- **Like we've seen with FA, designing a CFG requires creativity.**

- **But, there are some steps we can follow.**

  1) **Start simple: many CFLs are unions of simpler CFLs.. So, we can design CFGs for individual CFLs**
  2) **If the language is regular (meaning there's an FA that describes it), we can convert the FA into a CFG as follows:**
     1) **Make a variable R_i for each state q_i**
     2) **Add rule $R_i \rightarrow aR_j$ if $\delta(q_i, a) = q_j$**
     3) **Add rule $R_i \rightarrow \varepsilon$ if q_i is an accept state**
     4) **Make R_0 the start variable where q_0 is the start state.**

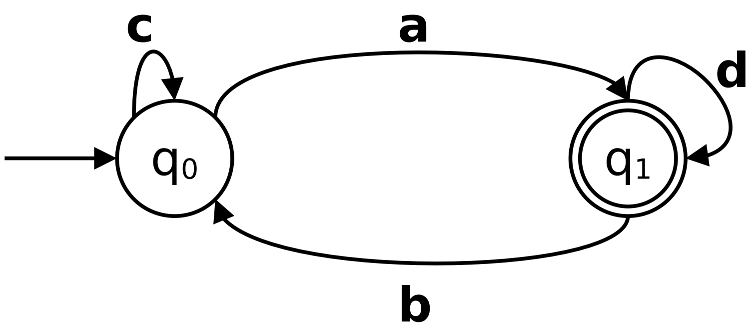
Context-free Grammar: Designing CFG

- Like we’ve seen with FA, designing a CFG requires creativity.
- But, there are some steps we can follow.
  1) Start simple: many CFLs are unions of simpler CFLs. So, we can des
  2) If the t describ ws:
     1) Ma
     2) Add rule $R_i \to aR_j$ if $\delta(q_i, a) = q_j$
     3) Add rule $R_i \to \varepsilon$ if q_i is an accept state
     4) Make R_0 the start variable where q_0 is the start state.

For more complex languages, we need to be more creative, and use techniques such as recursion to provide more power to the CFG

Context-free Grammar: Designing CFG

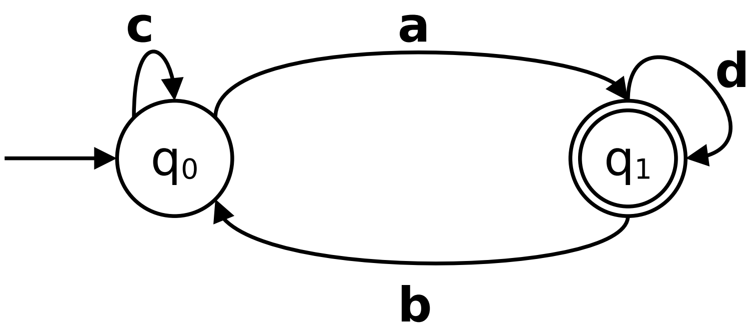
- **Design a CFG that describes the language of this NFA**



Context-free Grammar: Designing CFG

- **Design a CFG that describes the language of this NFA**

1 **Variable for every state**

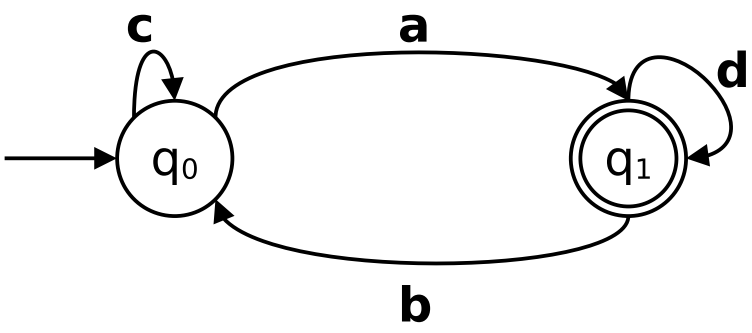


Context-free Grammar: Designing CFG

- **Design a CFG that describes the language of this NFA**

1

Variable for every state



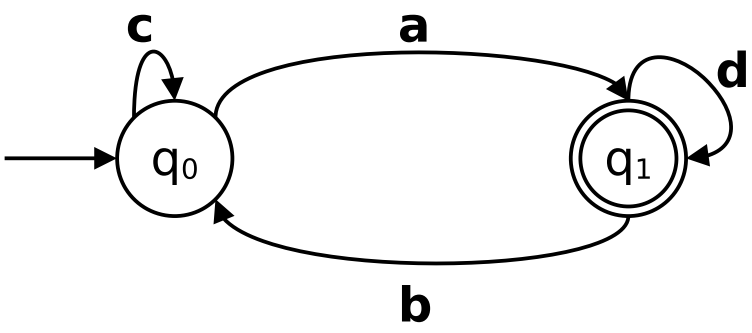
V

$V = \{R_0, R_1\}$

Context-free Grammar: Designing CFG

- **Design a CFG that describes the language of this NFA**

1 Variable for every state



2 Adding rules according to transition functions

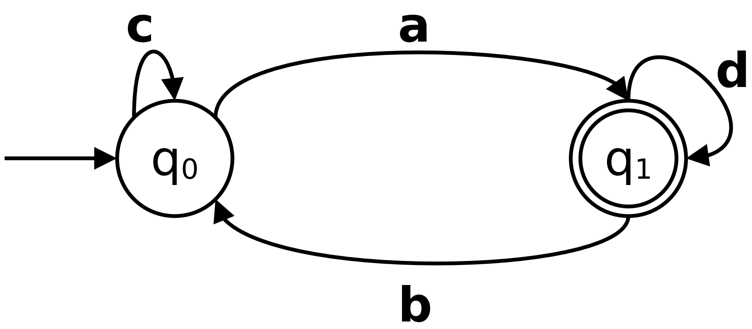
V $V = \{R_0, R_1\}$

Context-free Grammar: Designing CFG

➢ **Design a CFG that describes the language of this NFA**

1

Variable for every state



2

Adding rules according to transition functions

V

$V = \{R_0, R_1\}$

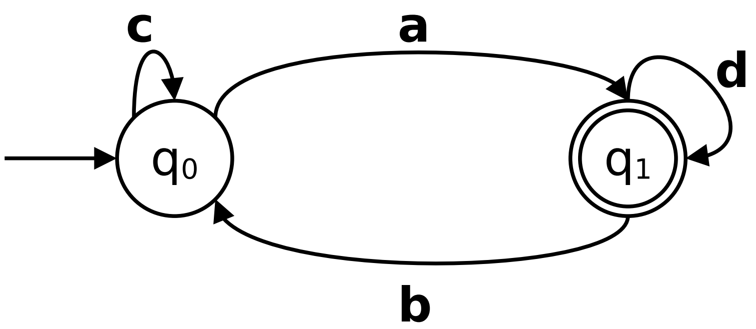
R

$R_0 \rightarrow cR_0 \mid aR_1$

$R_1 \rightarrow dR_1 \mid bR_0$

Context-free Grammar: Designing CFG

- **Design a CFG that describes the language of this NFA**



1

Variable for every state

2

Adding rules according to transition functions

3

Add rule $R_i \rightarrow \varepsilon$ if q_i is an accept state

V

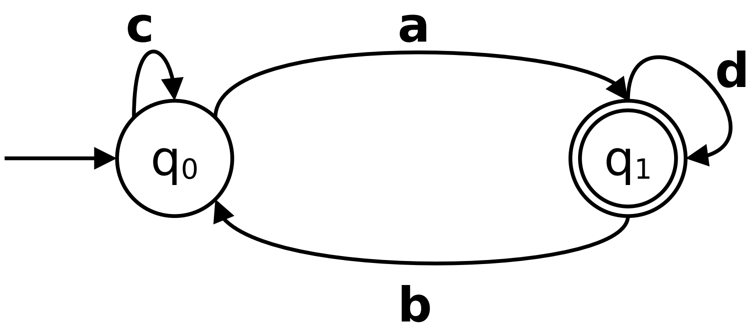
$V = \{R_0, R_1\}$

R

$R_0 \rightarrow cR_0 \mid aR_1$
$R_1 \rightarrow dR_1 \mid bR_0$

Context-free Grammar: Designing CFG

- **Design a CFG that describes the language of this NFA**



1

Variable for every state

2

Adding rules according to transition functions

3

Add rule $R_i \rightarrow \varepsilon$ if q_i is an accept state

V

$V = \{R_0, R_1\}$

R

$R_0 \rightarrow cR_0 \mid aR_1$

$R_1 \rightarrow dR_1 \mid bR_0 \mid \varepsilon$

Context-free Grammar: Ambiguity

- **Some CFGs may produce more than one parse tree for the same string.. We call those CFGs ambiguous.**
- **e.g., G1 =** $\boxed{\mathbf{E \rightarrow E + E \mid E * E \mid (E) \mid a}}$

Context-free Grammar: Ambiguity

- **Some CFGs may produce more than one parse tree for the same string.. We call those CFGs ambiguous.**
- **e.g., G1 =** $\boxed{\mathbf{E \rightarrow E + E \mid E * E \mid (E) \mid a}}$ w = a + a * a

Context-free Grammar: Ambiguity

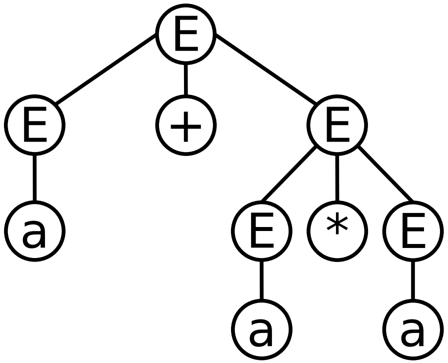
- **Some CFGs may produce more than one parse tree for the same string.. We call those CFGs ambiguous.**
- **e.g., G1 =** $\boxed{\mathbf{E \rightarrow E + E \mid E * E \mid (E) \mid a}}$ w = a + a * a

$$\boxed{\begin{array}{l} E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E \\ \Rightarrow a + a * E \Rightarrow a + a * a \end{array}}$$

Context-free Grammar: Ambiguity

- **Some CFGs may produce more than one parse tree for the same string.. We call those CFGs ambiguous.**
- **e.g., G1 =** $\boxed{\mathbf{E \rightarrow E + E \mid E * E \mid (E) \mid a}}$ w = a + a * a

$$\boxed{\begin{array}{l} E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E \\ \Rightarrow a + a * E \Rightarrow a + a * a \end{array}}$$

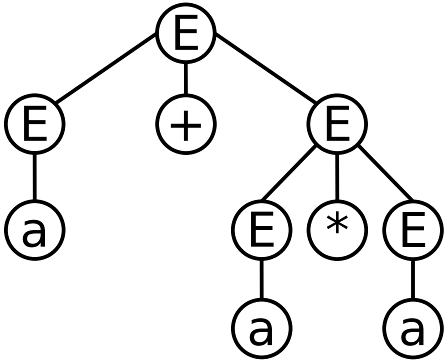


Context-free Grammar: Ambiguity

- **Some CFGs may produce more than one parse tree for the same string.. We call those CFGs ambiguous.**
- **e.g., G1 =** $\boxed{\mathbf{E \rightarrow E + E \mid E * E \mid (E) \mid a}}$

$w = a + a * a$

$$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * a$$



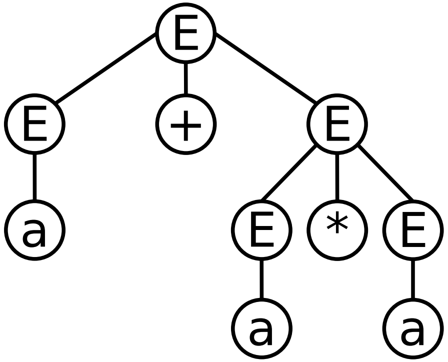
$$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * a$$

Context-free Grammar: Ambiguity

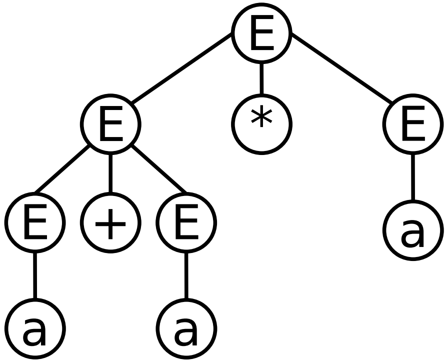
- **Some CFGs may produce more than one parse tree for the same string.. We call those CFGs ambiguous.**
- **e.g., G1 =** $\boxed{\mathbf{E \rightarrow E + E \mid E * E \mid (E) \mid a}}$

w = a + a * a

$$\boxed{\begin{array}{l} E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E \\ \Rightarrow a + a * E \Rightarrow a + a * a \end{array}}$$



$$\boxed{\begin{array}{l} E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow a + E * E \\ \Rightarrow a + a * E \Rightarrow a + a * a \end{array}}$$



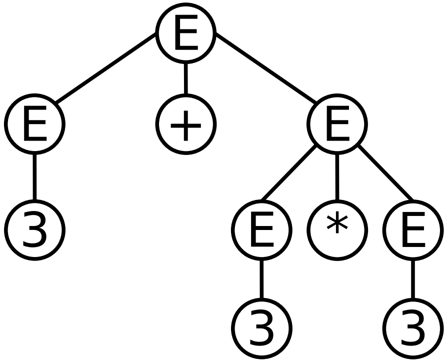
Context-free Grammar: Ambiguity

- **Some CFGs may produce more than one parse tree for the same string.. We call those CFGs ambiguous.**
- **e.g., G1 =** $\boxed{\mathbf{E \to E + E \mid E * E \mid (E) \mid a}}$

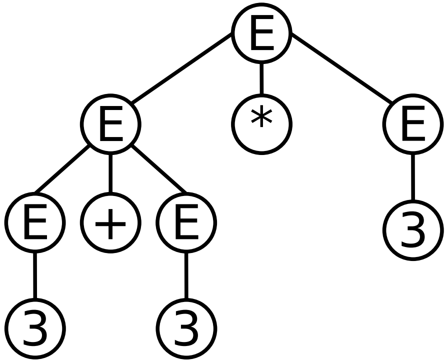
$w = a + a * a$

$$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * a$$

$$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * a$$



Let's see what happens when $a = 3$



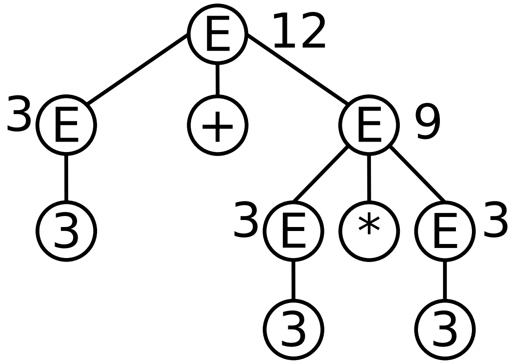
Context-free Grammar: Ambiguity

- **Some CFGs may produce more than one parse tree for the same string.. We call those CFGs ambiguous.**
- **e.g., G1 =** $\boxed{\mathbf{E \rightarrow E + E \mid E * E \mid (E) \mid a}}$

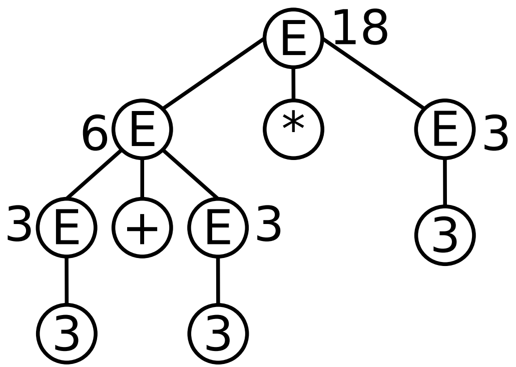
w = a + a * a

$$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * a$$

$$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow a + E * E \Rightarrow a + a * E \Rightarrow a + a * a$$



Let’s see what happens when a = 3



Context-free Grammar: Ambiguity

- **Ambiguity in CFG causes undesirable outcomes in some applications.**
  - **In programming languages (PL), an ambiguous CFG used by a PL compiler could mistakenly compile different executable programs for the same source code.**
  - **In arithmetic operations, ambiguity can lead to wrong answers.**

Context-free Grammar: Ambiguity

- A context-free grammar G is ambiguous if there is a string w such that w ∈ L(G) that has:
  - Two different parse trees, or
  - Two different leftmost derivations

Context-free Grammar: Ambiguity

- **Sometimes, we can convert an ambiguous CFG into an unambiguous version.**

Context-free Grammar: Ambiguity

- **Sometimes, we can convert an ambiguous CFG into an unambiguous version.**

Ambiguous CFG

$$\begin{aligned} &\mathbf{E} \rightarrow \mathbf{E} + \mathbf{E} \\ &\mathbf{E} \rightarrow \mathbf{E} * \mathbf{E} \\ &\mathbf{E} \rightarrow (\mathbf{E}) \\ &\mathbf{E} \rightarrow \mathbf{a} \end{aligned}$$

Context-free Grammar: Ambiguity

- **Sometimes, we can convert an ambiguous CFG into an unambiguous version.**

Ambiguous CFG

$$\begin{aligned} E &\rightarrow E + E \\ E &\rightarrow E * E \\ E &\rightarrow (E) \\ E &\rightarrow a \end{aligned}$$

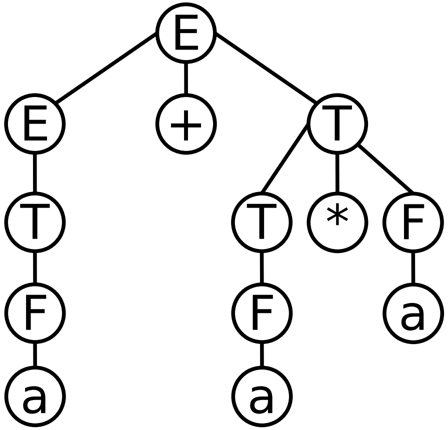
Unambiguous CFG

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T * F \mid F \\ F &\rightarrow (E) \mid a \end{aligned}$$

Context-free Grammar: Ambiguity

- **Sometimes, we can convert an ambiguous CFG into an unambiguous version.**

$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T \Rightarrow a + T * F$
$\Rightarrow a + F * F \Rightarrow a + a * F \Rightarrow a + a * a$



a + a * a

Always has the same parse tree

Unambiguous CFG

$$\mathbf{E \rightarrow E + T \mid T}$$
$$\mathbf{T \rightarrow T * F \mid F}$$
$$\mathbf{F \rightarrow (E) \mid a}$$

Context-free Grammar: Ambiguity

- **However, we cannot always convert ambiguous CFGs into unambiguous ones.**

Context-free Grammar: Ambiguity

- **However, we cannot always convert ambiguous CFGs into unambiguous ones.**

- **Determining if a grammar is ambiguous cannot be done for all grammars.**

Context-free Grammar: Ambiguity

- **However, we cannot always convert ambiguous CFGs into unambiguous ones.**
- **Determining if a grammar is ambiguous cannot be done for all grammars.**
  - **This is an undecidable problem → there is no algorithm that can tell us if a given grammar is ambiguous for all grammars.**

Context-free Grammar: Ambiguity

- **However, we cannot always convert ambiguous CFGs into unambiguous ones.**

- **Determining if a grammar is ambiguous cannot be done for all grammars.**
  - **This is an undecidable problem → there is no algorithm that can tell us if a given grammar is ambiguous for all grammars.**
  - **However, for some grammars we can provide an example string that shows the grammar is, in fact, ambiguous.**

Context-free Grammar: Ambiguity

- However, we cannot always convert ambiguous CFGs into unambiguous ones.

- Determining if a grammar is ambiguous cannot be done for all grammars.
  - This is an undecidable problem → there is no algorithm that can tell us if a given grammar is ambiguous for all grammars.
  - However, for some grammars we can provide an example string that shows the grammar is, in fact, ambiguous.
  - But, failing to provide an example string does not say anything about the CFG.

Context-free Grammar: Ambiguity

- **Ambiguity is a property of grammars not languages.**

Context-free Grammar: Ambiguity

- **Ambiguity is a property of grammars not languages.**
- **Some languages are inherently ambiguous**

Context-free Grammar: Ambiguity

- **Ambiguity is a property of grammars not languages.**
- **Some languages are inherently ambiguous**
  - **Language L is inherently ambiguous if it can be generated only by ambiguous grammars.**

Context-free Grammar: Ambiguity

- **Ambiguity is a property of grammars not languages.**
- **Some languages are inherently ambiguous**
  - **Language L is inherently ambiguous if it can be generated only by ambiguous grammars.**
  - **e.g.,** $\mathrm{L} = \{\mathrm{a}^i \mathrm{b}^j \mathrm{c}^k \mid i = j \text{ or } j = k\}$

Context-free Grammar: Ambiguity

- **Ambiguity is a property of grammars not languages.**
- **Some languages are inherently ambiguous**
  - **Language L is inherently ambiguous if it can be generated only by ambiguous grammars.**
  - **e.g.,** $L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$ note: $i, j, k \geq 0$

Context-free Grammar: Ambiguity

- **Ambiguity is a property of grammars not languages.**
- **Some languages are inherently ambiguous**
  - **Language L is inherently ambiguous if it can be generated only by ambiguous grammars.**
  - **e.g.,** $L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$

$$\downarrow$$

$$L = \{a^i b^i c^k\} \cup \{a^i b^k c^k\}$$

Context-free Grammar: Ambiguity

- **Ambiguity is a property of grammars not languages.**
- **Some languages are inherently ambiguous**
  - **Language L is inherently ambiguous if it can be generated only by ambiguous grammars.**
  - **e.g.,** $L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$

$$\downarrow$$

$$L = \{a^i b^i c^k\} \cup \{a^i b^k c^k\}$$

$$S \rightarrow S1 \mid S2$$

Context-free Grammar: Ambiguity

- **Ambiguity is a property of grammars not languages.**
- **Some languages are inherently ambiguous**
  - **Language L is inherently ambiguous if it can be generated only by ambiguous grammars.**
  - **e.g.,** $L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$

$$\downarrow$$

$$L = \{a^i b^i c^k\} \cup \{a^i b^k c^k\}$$

$$S \rightarrow S1 \mid S2$$

$$\boxed{\begin{array}{ll} S1 & \rightarrow S1c \mid A \\ A & \rightarrow aAb \mid \varepsilon \end{array}}$$

$$\boxed{\begin{array}{ll} S2 & \rightarrow aS2 \mid B \\ B & \rightarrow bBc \mid \varepsilon \end{array}}$$