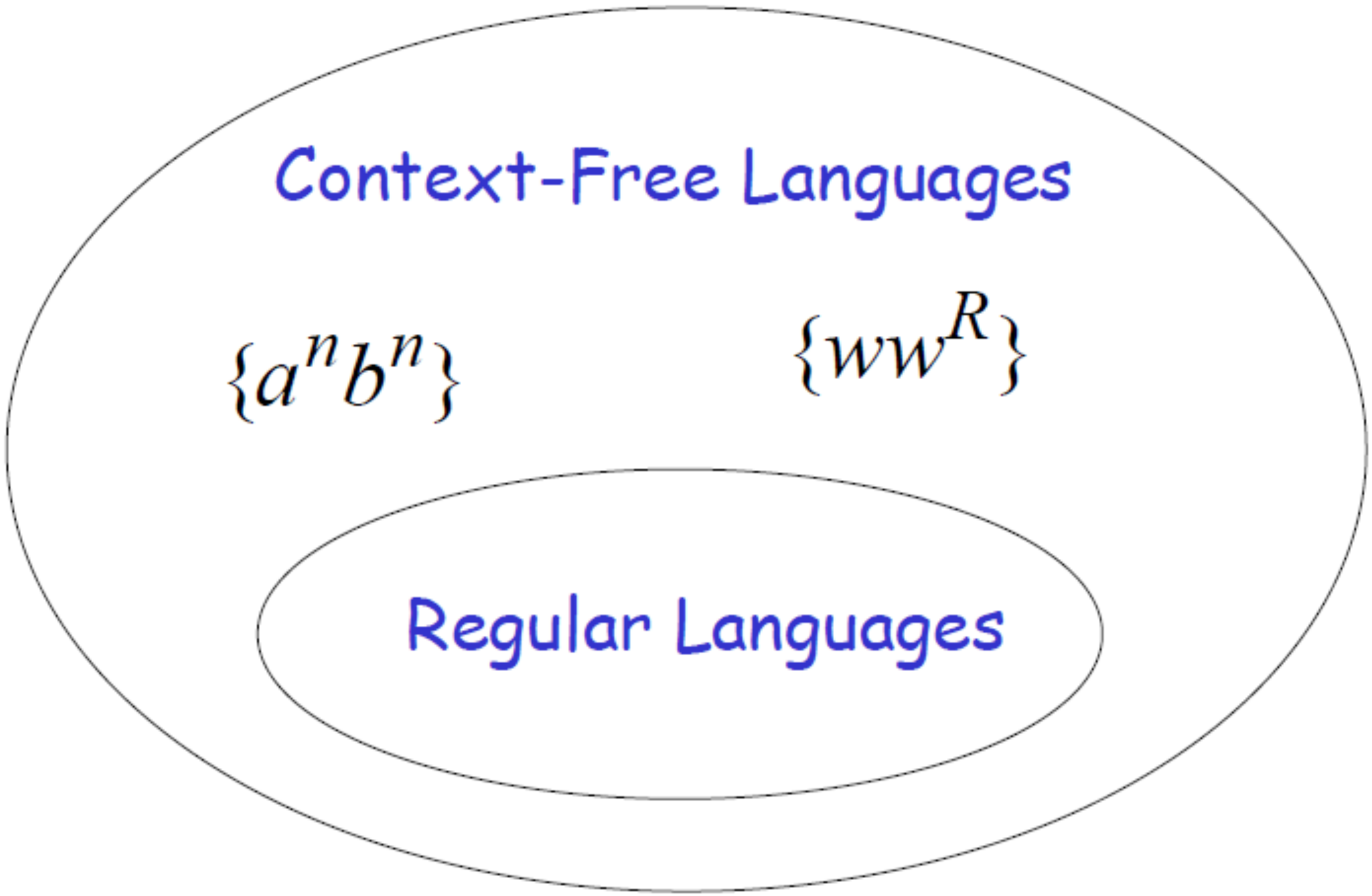
CS3110 Formal Language and Automata

Tingting Chen
Computer Science
Cal Poly Pomona



Context-Free Grammar

Pushdown Automata

Context-free Grammars

Definition:

A grammar $G = (V, T, S, P)$ is said to be context-free if all production rules in P have the form

$$A \rightarrow x$$

where $A \in V$ and $x \in (V \cup T)^*$

A language is said to be context-free iff there is a context free grammar G such that $L = L(G)$.

Context-free Grammars

Some grammars are not context-free grammars. For example:

$$1Z1 \rightarrow 101$$

In this production rule, the variable Z goes to 0 only in the *context* of a 1 on its left and a 1 to the right. This is a *context-sensitive* rule.

Example of Context-free Grammars

Given the grammar G = ({S}, {a, b}, S, P), with production rules:

$$S \rightarrow aSa \mid bSb \mid \lambda$$

It is a context-free grammar.

Is it a regular grammar?

The language it generates is not regular.

Example of Context-free Grammars

Given the grammar G = ({S}, {a, b}, S, P), with production rules:

$$S \rightarrow aSa \mid bSb \mid \lambda$$

a typical derivation in this grammar might be:

$$S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabSbaa \Rightarrow aabbaa$$

The language generated by this grammar is:

$L(G) = \{ww^R : w \in \{a,b\}^*\}$

Derivation

Given the grammar,

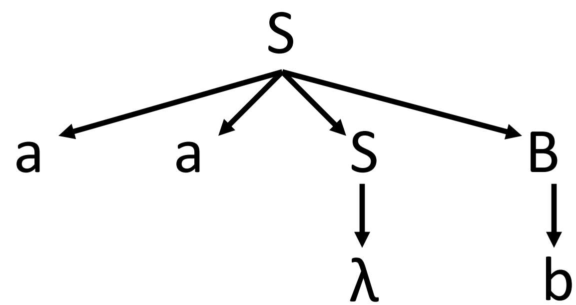
$$S \to aaSB \mid \lambda$$
$$B \to bB \mid b$$

the string *aab* can be derived in different ways.

$$S \Rightarrow aaSB \Rightarrow aaB \Rightarrow aab$$
$$S \Rightarrow aaSB \Rightarrow aaSb \Rightarrow aab$$

Parse tree

Both derivations on the previous slide correspond to the following parse (or derivation) tree.



• Each internal node of the tree corresponds to a nonterminal, and the leaves of the derivation tree represent the string of terminals.

• The tree structure shows the rule that is applied to each nonterminal, without showing the order of rule applications.

Leftmost/Rightmost Derivation

In the derivation

$$S \Rightarrow aaSB \Rightarrow aaB \Rightarrow aab$$

From aaSB, we replace S with λ, and then to replace B with b.

We moved from left to right, replacing the leftmost variable at each step. This is called a leftmost derivation.

Similarly, the derivation

$$S \Rightarrow aaSB \Rightarrow aaSb \Rightarrow aab$$

is called a rightmost derivation.

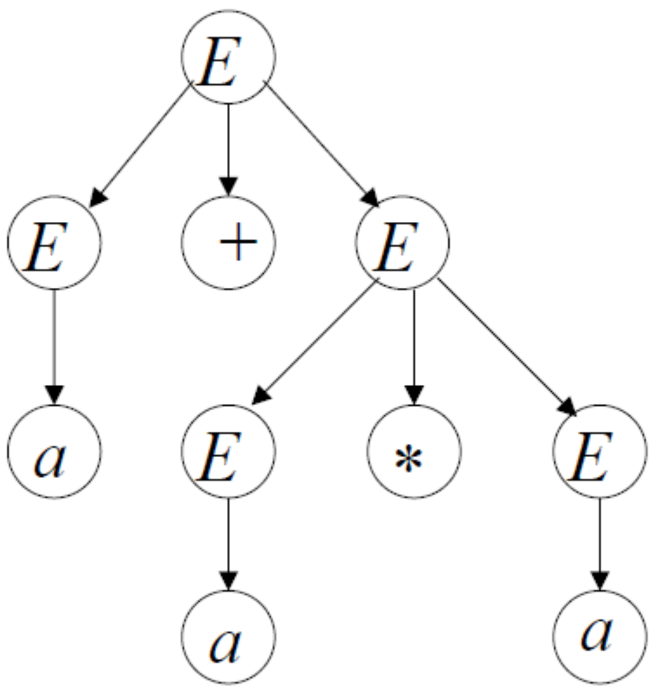
For a single derivation tree, there is a single leftmost derivation. For a string, there may be more than one leftmost derivation.

Example

Production Rules:

$$E \rightarrow E+E \ \mid E*E \mid (E) \mid a$$

What is the derivation tree for string $a+a*a$?



Left-most derivation:

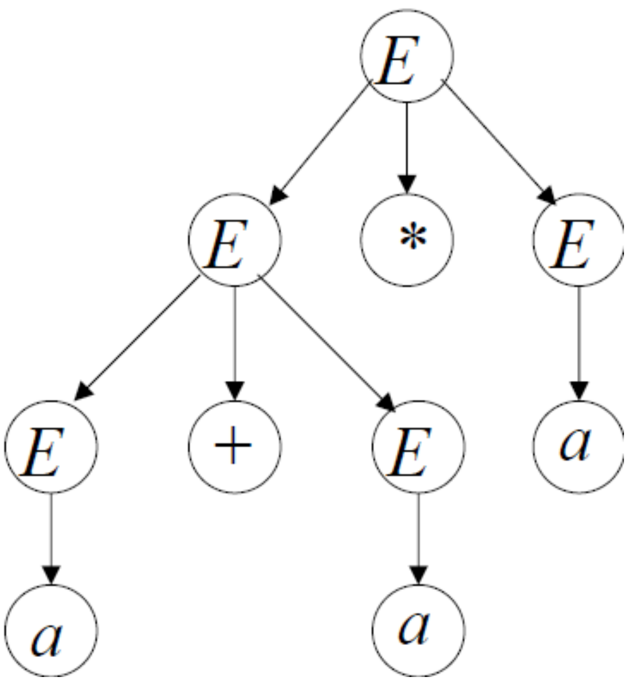
$$E \Rightarrow E+E \Rightarrow a+E \Rightarrow a+E*E$$
$$\Rightarrow a+a*E \Rightarrow a+a*a$$

Example

Production Rules:

$$E \rightarrow E + E \;\; | \;\; E * E \;|\; (E) \;|\; a$$

What is the derivation tree for string $a + a * a$?



Left-most derivation:

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

Ambiguity

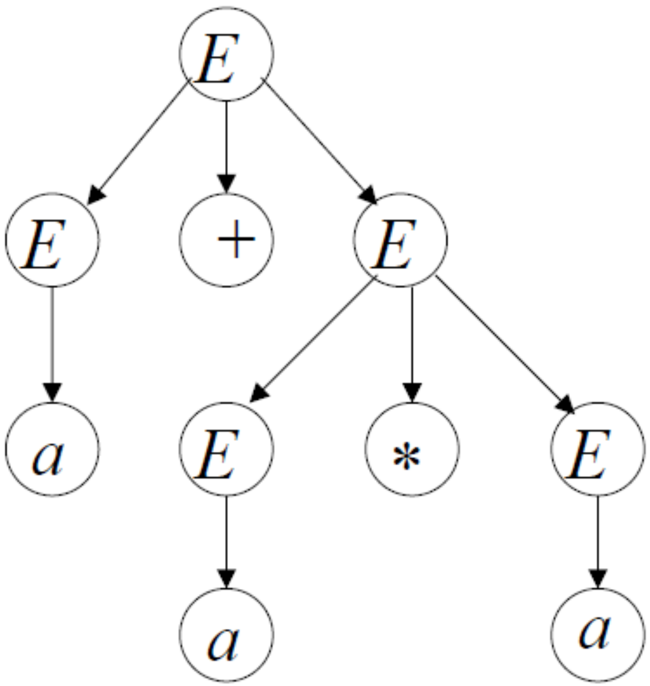
A grammar G is ambiguous if there is a string $w \in L(G)$ has two or more possible derivation trees.

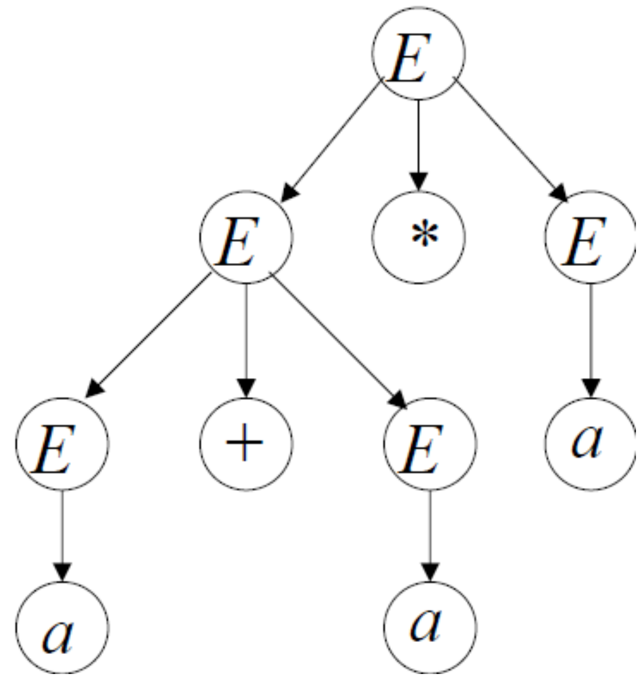
In other words,
A grammar G is ambiguous if there is a string $w \in L(G)$ has two or more possible leftmost (or rightmost) derivations.

Why care about Ambiguity

$$a + a * a$$

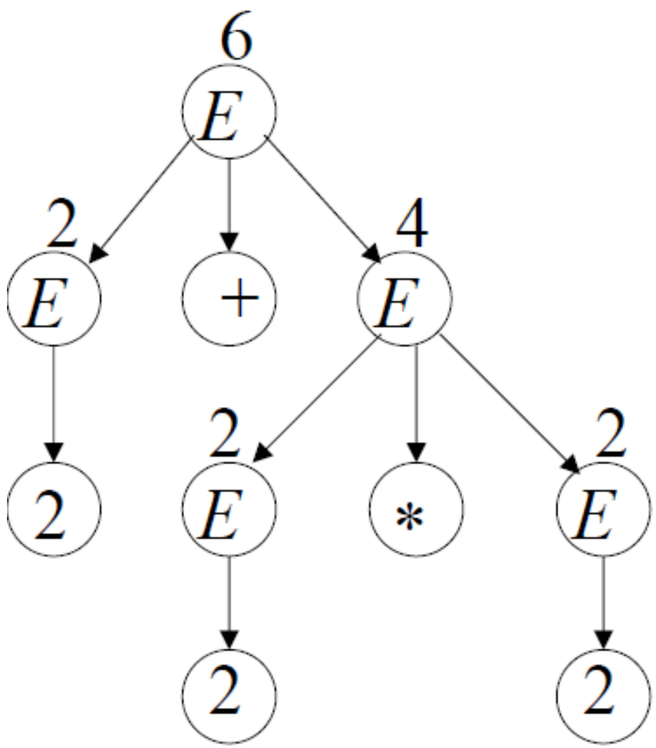
Take a =2



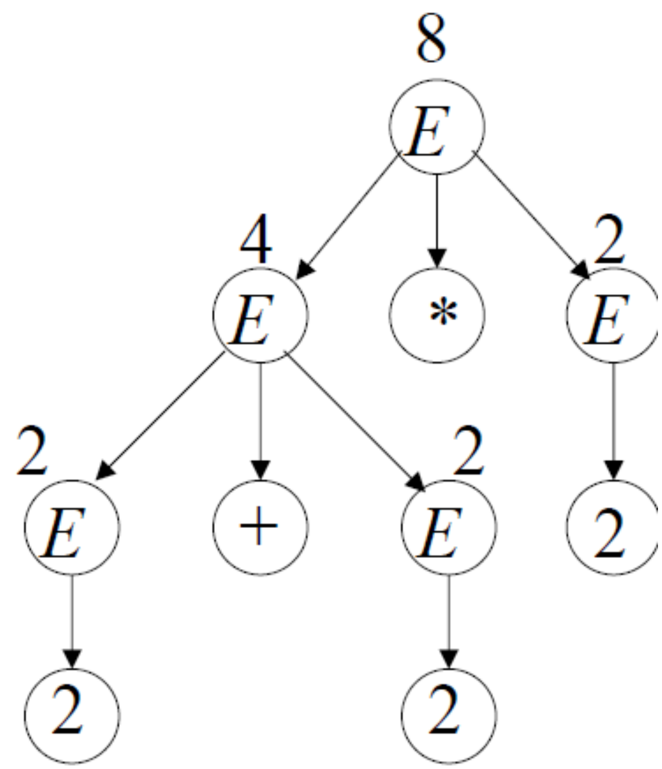


Why care about Ambiguity

2+2*2=6



2+2*2=8



Equivalent grammars

To make parsing easier, we prefer grammars that are not ambiguous.

Here is a non-ambiguous grammar that generates the same language.

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

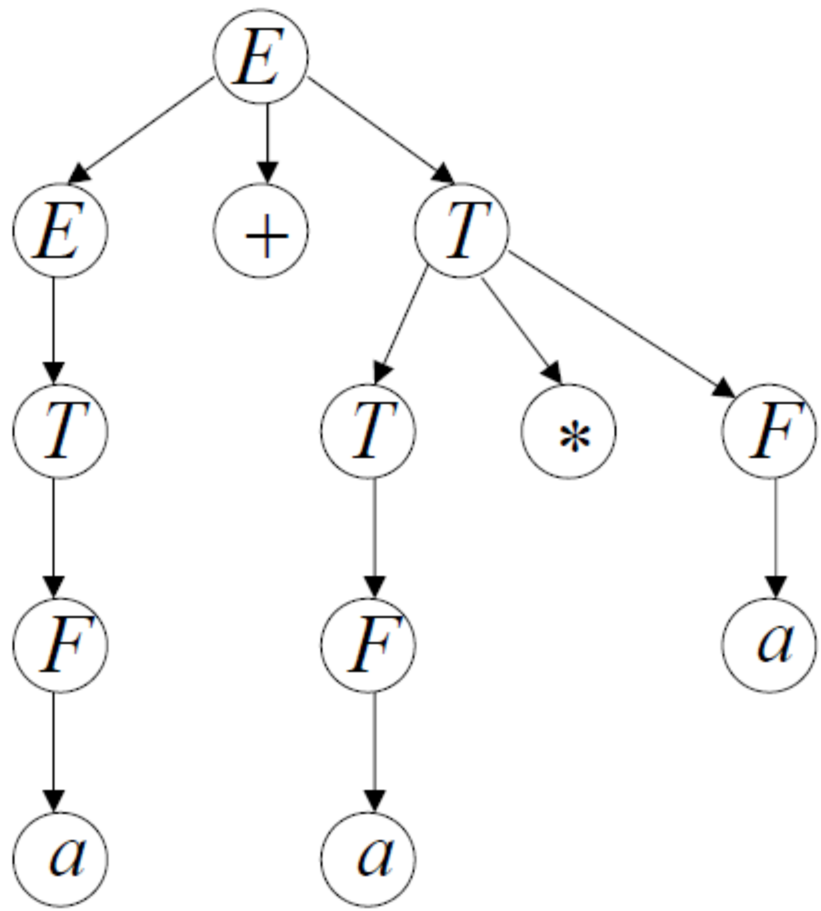
$F \rightarrow (E) \mid a$

Two grammars that generate the same language are said to be equivalent.

Equivalent grammars

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

$$E \rightarrow E+T$$
$$E \rightarrow T$$
$$T \rightarrow T*F$$
$$T \rightarrow F$$
$$F \rightarrow (E)$$
$$F \rightarrow a$$



Ambiguous grammars & equivalent grammars

- There is no general algorithm for determining whether a given CFG is ambiguous.

- There is no general algorithm for determining whether a given CFG is equivalent to another CFG.

Inherent Ambiguity

- Some context free languages have only ambiguous grammars.

- If every grammar that generates L is ambiguous, then the language is called *inherently ambiguous*.

- Example: $L=\{a^n b^n c^m\} \cup \{a^n b^m c^m\}$

$S \to S_1 | S_2$

$S1 \to S1c | A$

$A \to aAb | \lambda$

$S2 \to aS2 | B$

$B \to bBc | \lambda$

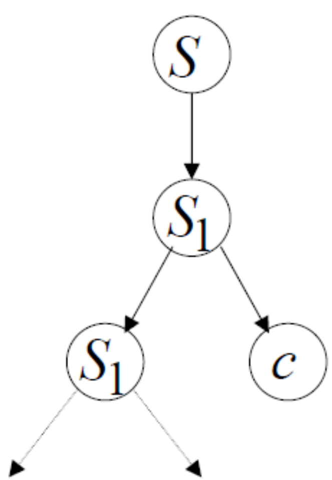
Inherent Ambiguity

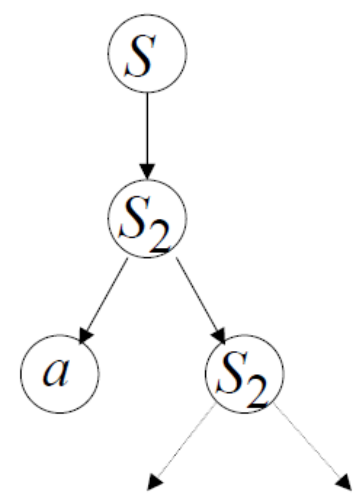
- Example: $L=\{a^n b^n c^m\} \cup \{a^n b^m c^m\}$

$S \rightarrow S_1 | S_2$ $\quad$ $S1 \rightarrow S1c | A$ $\quad$ $S2 \rightarrow aS2 | B$

$\quad$ $A \rightarrow aAb | \lambda$ $\quad$ $B \rightarrow bBc | \lambda$

The string $a^n b^n c^n$ has two derivation tree.





Exercise

Show that the following grammar is ambiguous.

$$S \to AB \mid aaB$$
$$A \to a \mid Aa$$
$$B \to b$$

Construct an equivalent grammar that is unambiguous.