L4:
Overview of Languages, Grammars and Probabilistic Context-free Grammars

Sudhir Aggarwal and Shiva Houshmand
Florida State University
Department of Computer Science
E-Crime Investigative Technologies Lab
Tallahassee, Florida 32306
August 5-7, 2015

Password Cracking
University of Jyväskylä
Summer School August 2015

Languages Definitions

- Any finite, nonempty set of symbols is an alphabet or vocabulary.

  Σ = {A, B, C, D, …, Z}

  Σ = {0, 1}

  Σ = { □, if, then, else}

  Σ = { **S**, L_1, …, L_{20}, D_1, …, D_{20}, S_1, …, S_{20}, alice, Bob, 427,….}

- A finite sequence of symbols from the alphabet is called a string or a word or a sentence.

  w = ALPHA

  w = 0100011101

Languages
Definitions

- Two strings can be concatenated to form another string:

  v = ALPHA, w = BETA

  Concat(v, w) = vw = ALPHABETA

- The length of a string w, denoted by $|w|$ is the number of symbols in the string.

  |ALPHA| = 5

- The empty string is denoted by λ or ε and its length is 0.

  $|\lambda| = 0$

Languages Definitions

- If Σ is the alphabet, Σ^* is the set of all strings over Σ, including the empty string.
- Σ^* is obtained by concatenating zero or more symbols from Σ.

$$\Sigma^+ = \Sigma^* - \{\lambda\}$$

Let $\Sigma = \{a, b, c, d\}$, what is Σ^* ?

Can you specify a procedure to generate Σ^* ?

What is $|\Sigma^*|$?

Languages
Definitions

- A language over Σ is a subset of Σ^*.

  $L \subseteq \Sigma^*$

Example: $\Sigma = \{a, b\}$

$L_1 = \{ a, aa, aba\}$ a finite language

$L_2 = \{ a^n b^n : n \geq 1\}$ an infinite language

Ways to represent languages

1. Recognition point of view

2. Generation point of view
   – Systematically generate (enumerate) all sentences of the language

Automata

- An automaton is an abstract model of a digital computer.
- Reads the input (string over the alphabet)
- Has a control unit which can be in any of the finite number of internal states and can change state in some defined manner.
- Given an input string, it outputs yes or no meaning that it either accepts the string or rejects it.

Grammars
Definitions

- A grammar is a method to describe and generate the sentences of a language.
- A **grammar** G is defined as a quadruple

$$G = (V, T, S, P)$$

V is a finite set of **variables**

T is a finite set of **terminal** symbols

S $\in$ V is a special variable called **start symbol**

P is a finite set of **production rules** of the form

$$x \rightarrow y$$

$$\text{where } x \in (V \cup T)^+ , \; y \in (V \cup T)^*$$

Grammars
Example

S → <noun phrase> <verb phrase>

<noun phrase> → <article> <noun>

<article> → the

<noun> → dog

<verb phrase> → is <adjective>

<adjective> → happy

S => <noun phrase><verb phrase> => <article><noun><verb phrase>
=> the <noun><verb phrase> => the <noun> is <adjective> =>
the dog is <adjective> => the dog is happy

Grammars
Definitions

- We say that *w* **derives** *z* if *w* = *uxv* , and *z* = *uyv*

and $x \rightarrow y \in P$

$$w \Rightarrow z$$

- If $w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n$ we say $w_1 \Rightarrow^* w_n$ (derives in zero or more steps)
- The set of **sentential forms** is

$$S(G) = \{\alpha \in (V \cup T)^* \mid S \Rightarrow^* \alpha\}$$

- The language generated by grammar G is

$$L(G) = \{ w \in T^* \mid S \Rightarrow^* w \}$$

Grammars
Example

$G = (V, T, P, S)$ $V = \{S, B, C\}$ $T = \{a, b, c\}$

P:

$S \rightarrow aSBC$ $bB \rightarrow bb$

$S \rightarrow aBC$ $bC \rightarrow bc$

$CB \rightarrow BC$ $cC \rightarrow cc$

$aB \rightarrow ab$

$S \Rightarrow^* aaBCBC$ sentential form

What is L(G) ? $L(G) = \{ a^n b^n c^n \mid n \geq 1\}$

Grammars
Example

$G = (\{S\}, \{a, b\}, S, P)$

Productions:

$S \rightarrow aSb$

$S \rightarrow \lambda$

What is the L(G)? $L = \{a^n b^n : n \geq 0\}$

Grammars
Example

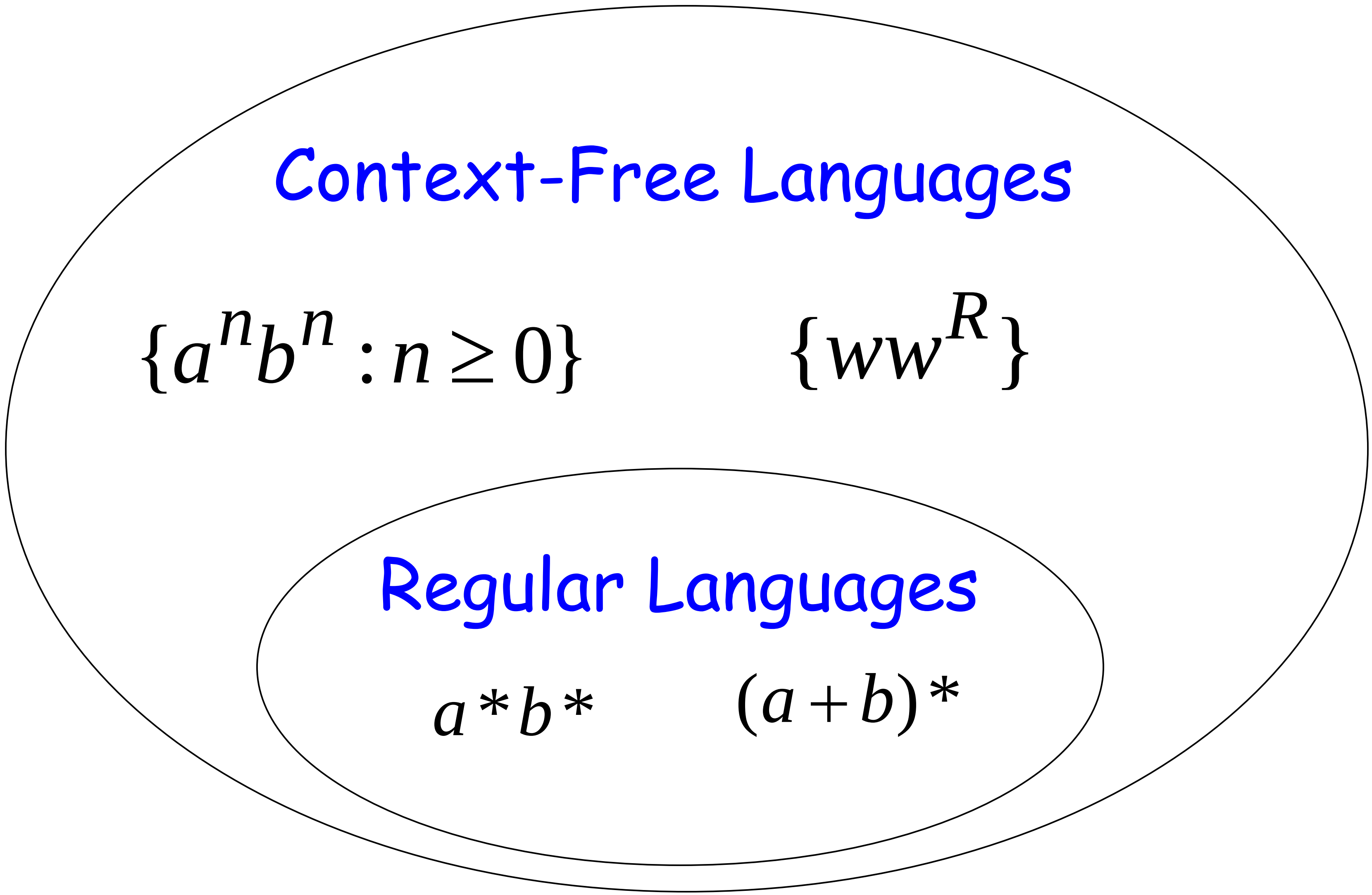
Find a grammar that generates

$$L = \{ a^n b^{2n} : n \geq 0 \}$$

$S \rightarrow aSbb \mid \lambda$

Summary

- An automaton recognizes (or accepts) a language
- A grammar generates a language
- For some grammars, it is possible to build an automaton M_G from the grammar G so that M_G recognizes the language L(G) generated by the grammar G.


Context-Free Languages
$\{a^n b^n : n \geq 0\}$
$\{ww^R\}$
Regular Languages
a^*b^*
$(a+b)^*$

Example 1

$G = (\{S\}, \{a, b\}, S, P)$

$$S \rightarrow aSb$$

$$S \rightarrow \lambda$$

Derivations:

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaabbbb$

Notation

We write: $S \overset{*}{\Rightarrow} aaabbb$

for zero or more derivation steps

Instead of:

S => aSb => aaSbb => aaaSbbb => aaabbb

Example

Grammar:

$S \rightarrow aSb$

$S \rightarrow \lambda$

Possible Derivations:

$S \overset{*}{\Rightarrow} \lambda$

$S \overset{*}{\Rightarrow} ab$

$S \overset{*}{\Rightarrow} aaaSbbb \overset{*}{\Rightarrow} aaaaabbbbb$

Language of a Grammar

- For a grammar G with start variable S

$$L(G) = \{ w : S \overset{*}{=>} w,\ \ w \in T^* \}$$

Example

Grammar:

$$S \rightarrow aSb$$

$$S \rightarrow \lambda$$

Language of the grammar:

$$L = \{a^n b^n : n \geq 0\}$$

Context-Free Grammar

- A grammar G=(V, T, S, P) is context-free if all productions in P have the form:

$$A \to \overbrace{x}^{\text{Sequence of terminals and variables}} \qquad \text{where } A \in V \text{ and } x \in (V \cup T)^*$$

- A language L is a context-free language iff there is a context-free grammar G such that L = L(G)

Context-Free Language

$L = \{a^n b^n : n \geq 0\}$ is a context-free language since the context-free grammar:

$S \rightarrow aSb \mid \lambda$ generates $L(G) = L$

Another Example

Context-free grammar G:

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

A derivation: S => aSa => abSba => abba

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

Another Example

Context-free grammar G:

$$S \rightarrow (S) \mid SS \mid \lambda$$

A derivation:

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ((S))S \Rightarrow (())(S) \Rightarrow (())()$$

L(G) : balanced parentheses

Example 2

$$L = \{ a^n b^m : n \neq m \}$$

$n > m$

aaaaaaaabbbbb

$S \rightarrow AS_1$

$S_1 \rightarrow aS_1b \mid \lambda$

$A \rightarrow aA \mid a$

$n < m$

aaaaabbbbbbbbb

$S \rightarrow S_1B$

$S_1 \rightarrow aS_1b \mid \lambda$

$B \rightarrow bB \mid b$

$S \rightarrow AS_1 \mid S_1B$

$S_1 \rightarrow aS_1b \mid \lambda$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

Derivations

Derivations

- When a sentential form has a number of variables, we can replace any one of them at any step.

- As a result, we have many different derivations of the same string of terminals.

Derivations

Example: 1. $S \rightarrow aAS$ 2. $S \rightarrow a$

3. $A \rightarrow SbA$ 4. $A \rightarrow SS$ 5. $A \rightarrow ba$

$\underline{S} \overset{1}{\Rightarrow} aA\underline{S} \overset{2}{\Rightarrow} a\underline{A}a \overset{3}{\Rightarrow} aSb\underline{A}a \overset{4}{\Rightarrow} aSbS\underline{S}a \overset{2}{\Rightarrow} aSb\underline{S}aa$

$\overset{2}{\Rightarrow} a\underline{S}baaa \overset{2}{\Rightarrow} aabaaa$

$\underline{S} \overset{1}{\Rightarrow} a\underline{A}S \overset{3}{\Rightarrow} aSbA\underline{S} \overset{2}{\Rightarrow} aSb\underline{A}a \overset{4}{\Rightarrow} a\underline{S}bSSa \overset{2}{\Rightarrow}$

$aabS\underline{S}a \overset{2}{\Rightarrow} aab\underline{S}aa \overset{2}{\Rightarrow} aabaaa$

Leftmost Derivation

A derivation is said to be leftmost if in each step the leftmost variable in the sentential form is replaced.

Example: $S \rightarrow aAS \mid a$

$A \rightarrow SbA \mid SS \mid ba$

S => aAS => aSbAS => aabAS => aabSSS => aabaSS => aabaaS => aabaaa

Leftmost

Rightmost Derivation

A derivation is said to be rightmost if in each step the rightmost variable is replaced.

Example: 1. $S \rightarrow aAS$ 2. $S \rightarrow a$

3. $A \rightarrow SbA$ 4. $A \rightarrow SS$ 5. $A \rightarrow ba$

$$S \overset{1}{\Rightarrow} aAS \overset{2}{\Rightarrow} aAa \overset{3}{\Rightarrow} aSbAa \overset{4}{\Rightarrow} aSbSSa \overset{2}{\Rightarrow} aSbSaa$$

$$\overset{2}{\Rightarrow} aSbaaa \overset{2}{\Rightarrow} aabaaa$$

Rightmost

Leftmost and Rightmost Derivation

Example: 1. $S \rightarrow aAS$ 2. $S \rightarrow a$

3. $A \rightarrow SbA$ 4. $A \rightarrow SS$ 5. $A \rightarrow ba$

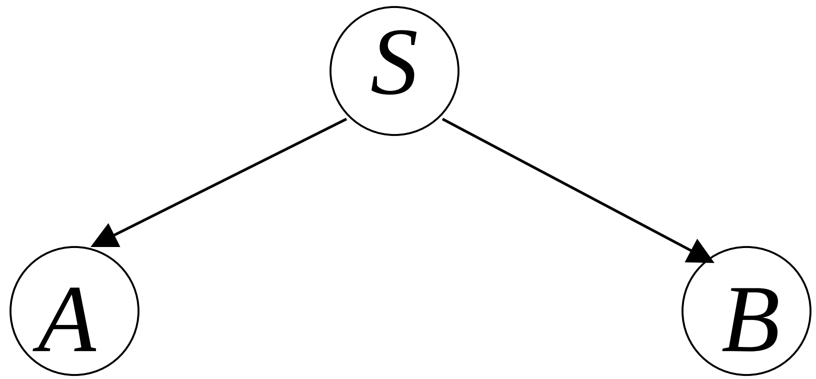
$\underline{S} \Rightarrow a\underline{A}S \Rightarrow aSbA\underline{S} \Rightarrow aSb\underline{A}a \Rightarrow a\underline{S}bSSa \Rightarrow$
$aabS\underline{S}a \Rightarrow aab\underline{S}aa \Rightarrow aabaaa$

Neither

Derivation Trees

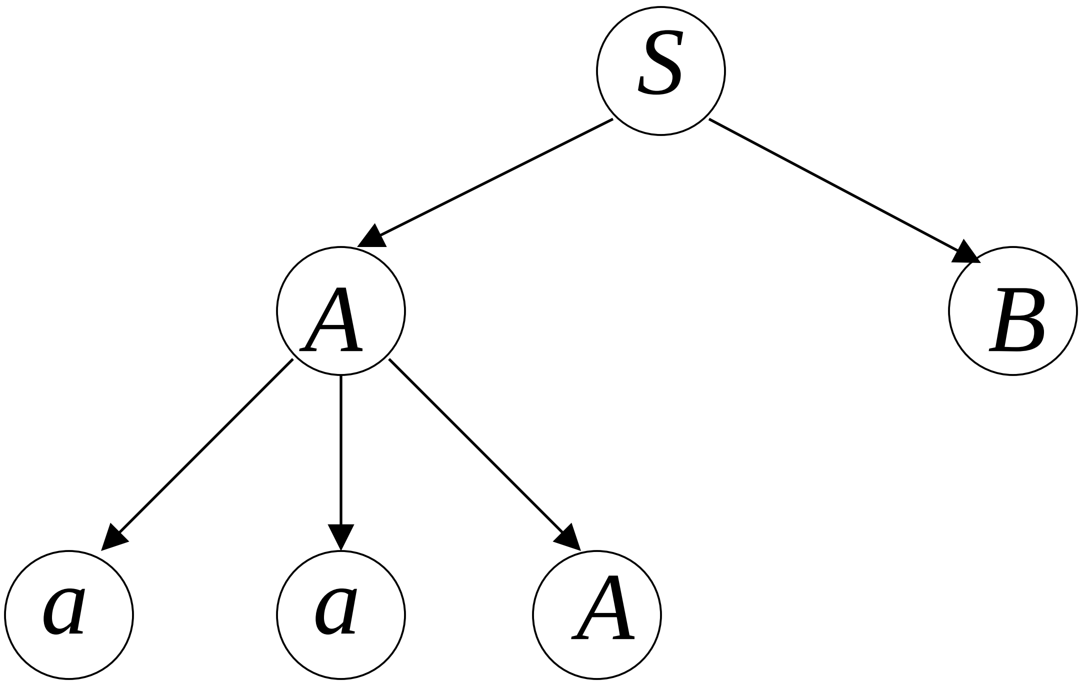
$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

$$S \Rightarrow AB$$



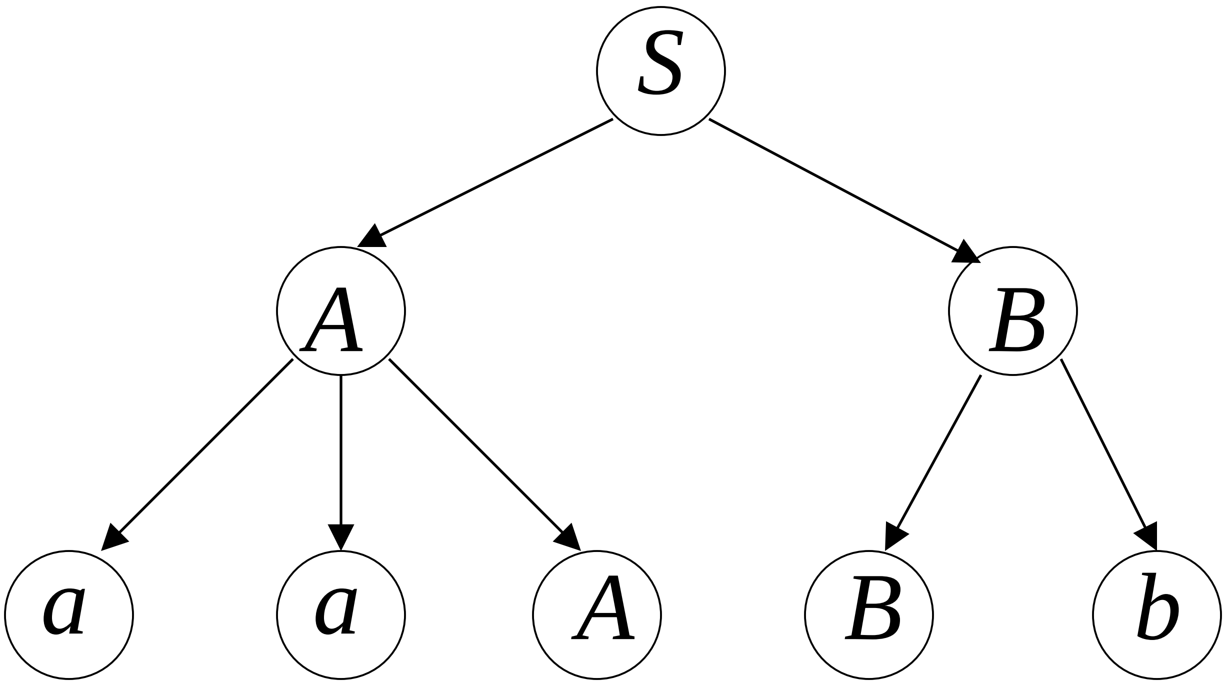
$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

$$S \Rightarrow AB \Rightarrow aaAB$$



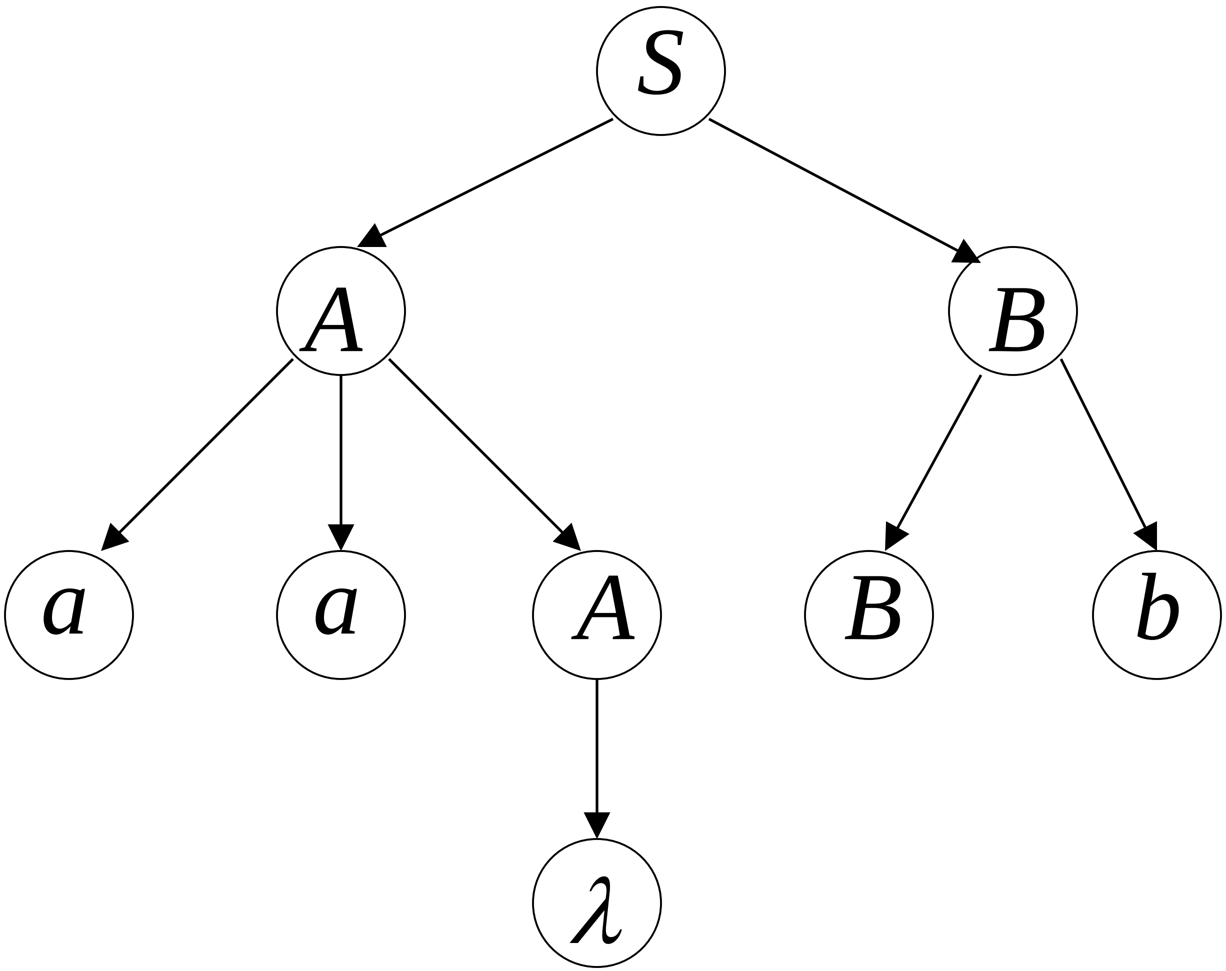
$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb$$



$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

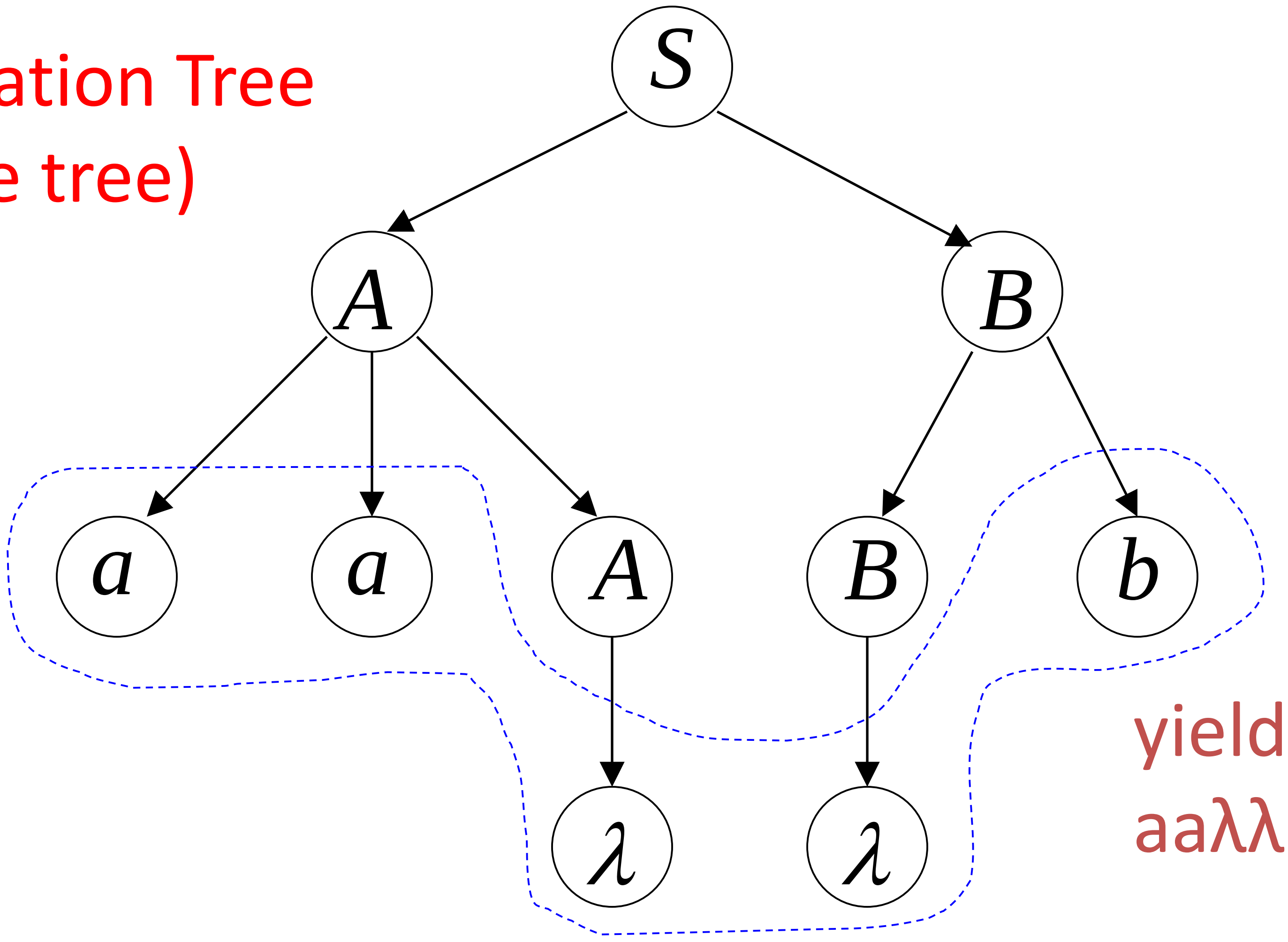
$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb$$



$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb \Rightarrow aab$$

Derivation Tree
(parse tree)



yield
aaλλb = aab

Derivation Trees

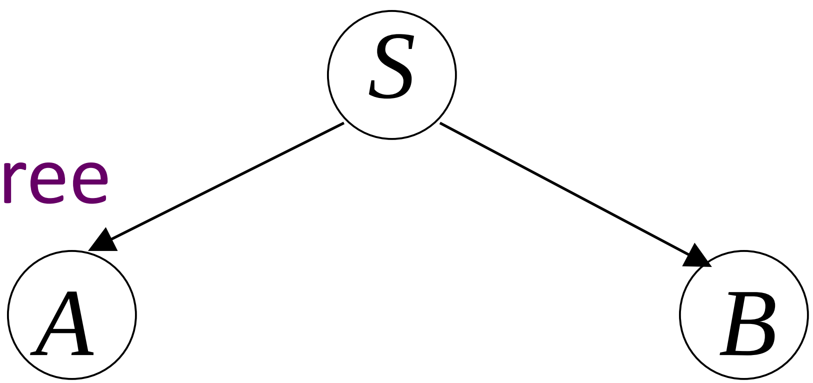
- Derivation trees are trees whose nodes are labeled by symbols of a CFG.
- Root is labeled by S (start symbol).
- Leaves are labeled by terminals $T \cup \{\lambda\}$
- Interior nodes are labeled by non-terminals V.
- If a node has label $A \in V$, and there is a production rule $A \rightarrow \alpha_1\alpha_2\ldots\alpha_n$ then its children are labeled from left to right $\alpha_1, \alpha_2, \ldots, \alpha_n$.
- The string of symbols obtained by reading the leaves from left to right is said to be the yield.

Partial Derivation Tree

A partial derivation tree is a subset of the derivation tree (the leaves can be non-terminals or terminals.

$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

Partial derivation tree

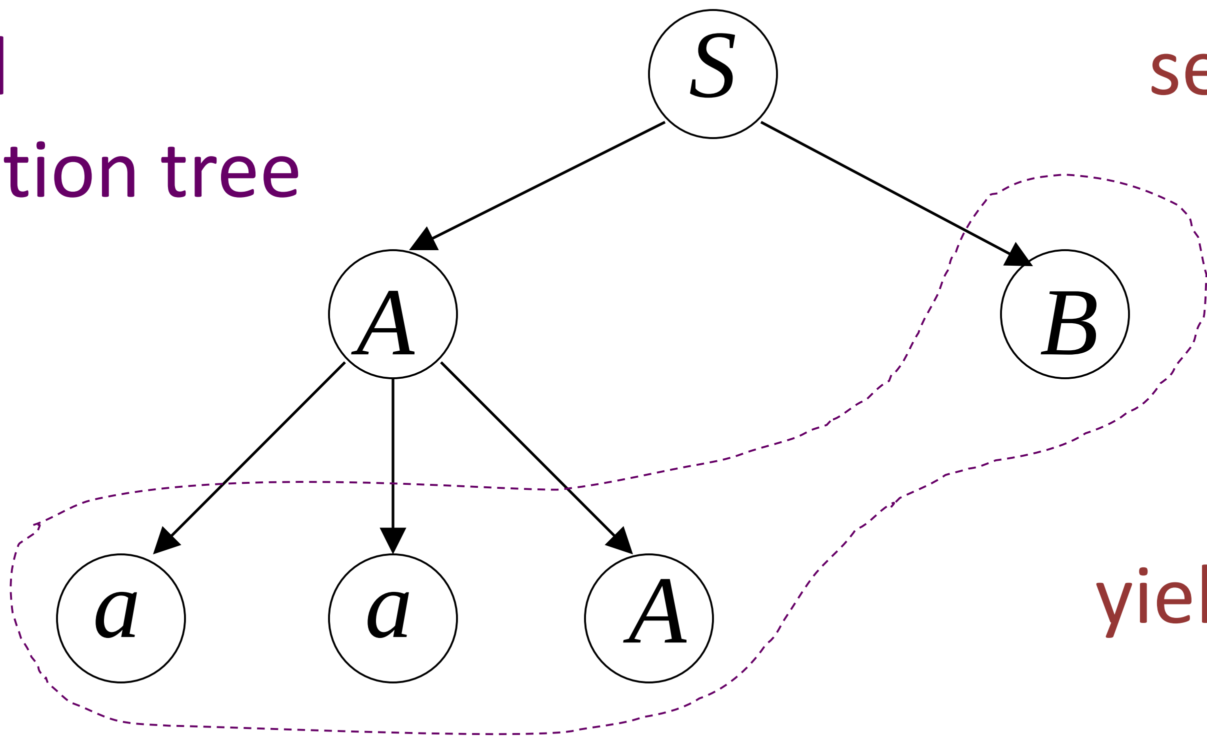


Partial Derivation Tree

$$S \Rightarrow AB \Rightarrow aaAB$$

sentential form

Partial
derivation tree



yield aaAB

CFG Theorem

1) If there is a derivation tree with root labeled A that yields w, then $A \Rightarrow^*_{lm} w$.

2) If $A \Rightarrow^*_{lm} w$, then there is a derivation tree with root A that yields w.

Ambiguity

Ambiguous grammars
Example

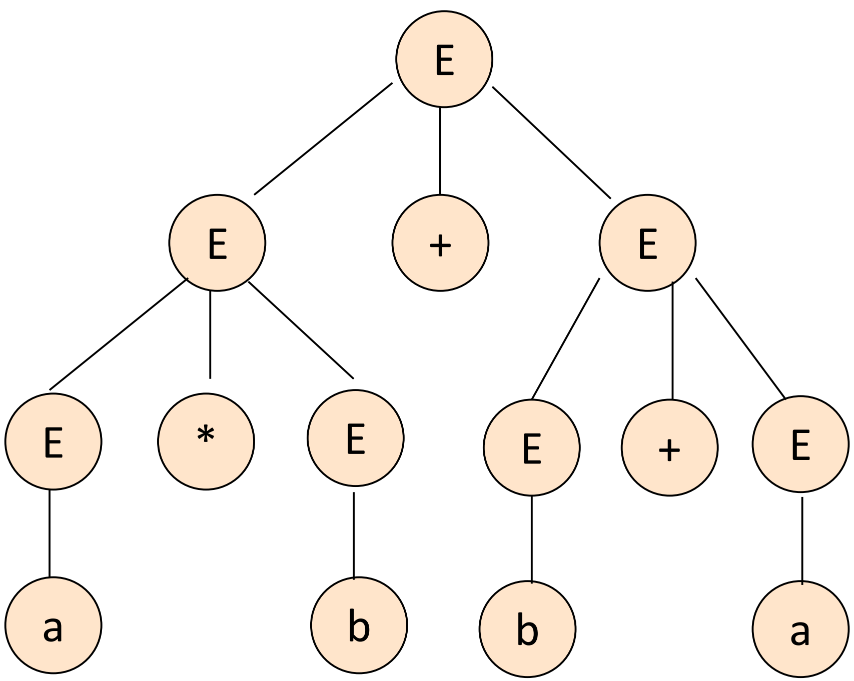
$E \rightarrow E + E$

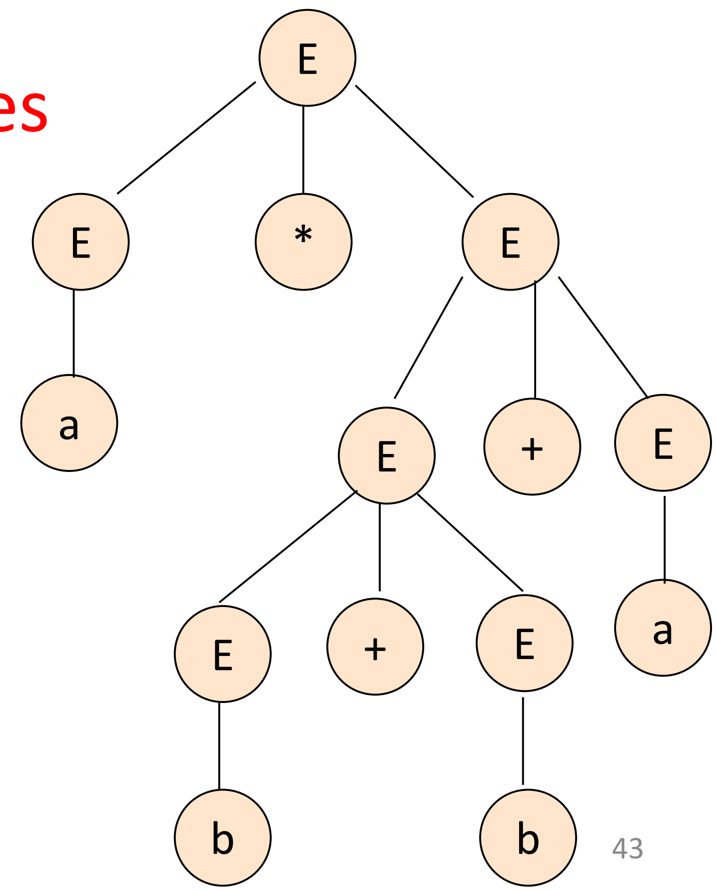
$E \rightarrow E * E$

$E \rightarrow a \mid b$

a * b + b + a

Two derivation trees





Example

$E \rightarrow E + E$ $E \rightarrow E * E$ $E \rightarrow a \mid b$

$\underline{E} \Rightarrow \underline{E} + E \Rightarrow \underline{E} * E + E \Rightarrow a * \underline{E} + E \Rightarrow a * b + \underline{E}$
$\Rightarrow$
$a * b + \underline{E} + E \Rightarrow a * b + b + \underline{E} \Rightarrow a * b + b + a$

Leftmost derivation

$\underline{E} \Rightarrow \underline{E} * E \Rightarrow a * \underline{E} \Rightarrow a * \underline{E} + E \Rightarrow a * \underline{E} + E + E$
$\Rightarrow a * b + \underline{E} + E \Rightarrow a * b + b + E \Rightarrow a * b + b + a$

Leftmost derivation

Ambiguous grammars

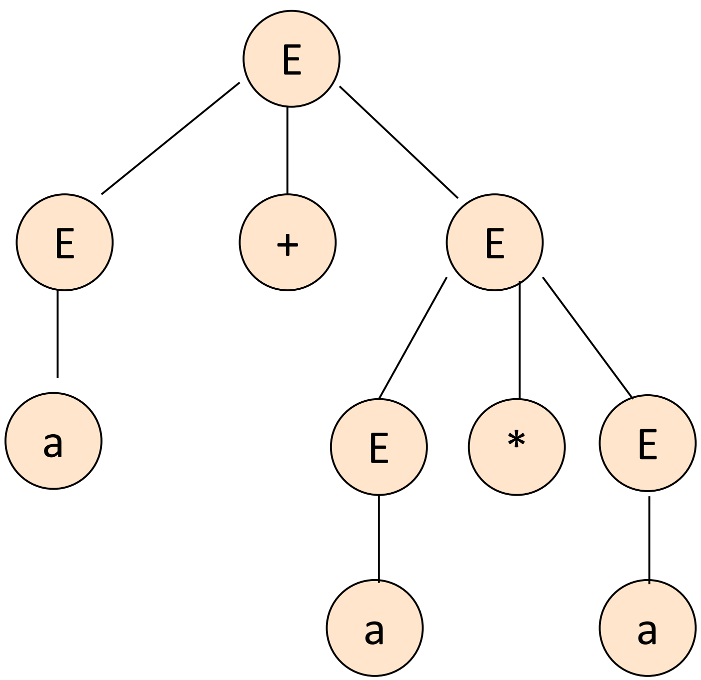
- A context-free grammar G is ambiguous if there exist some $w \in L(G)$ that has at least two distinct derivation trees.

- Or if there exists two or more leftmost derivations (or rightmost).

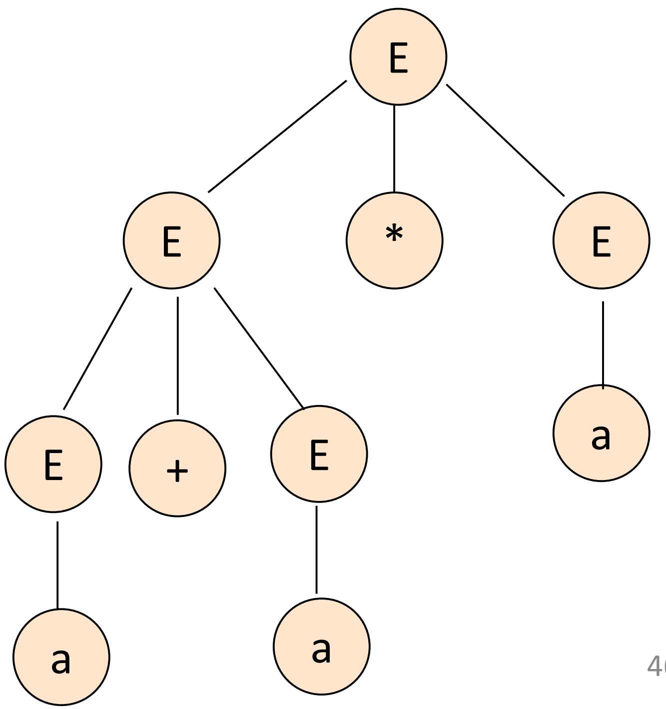
Why do we care about ambiguity?

Grammar for mathematical expressions:

$E \rightarrow E + E$ $\qquad$ $E \rightarrow E * E$ $\qquad$ $E \rightarrow a$

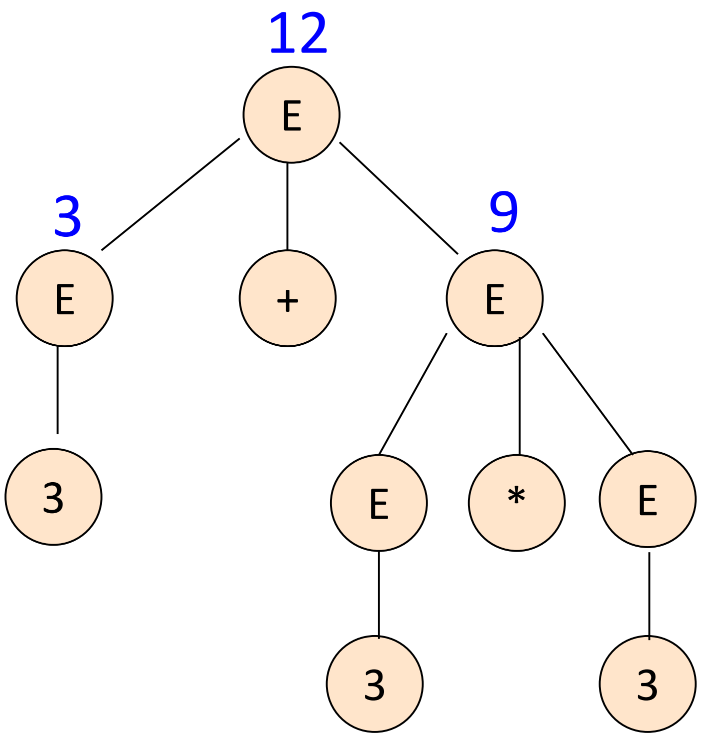


a + a * a

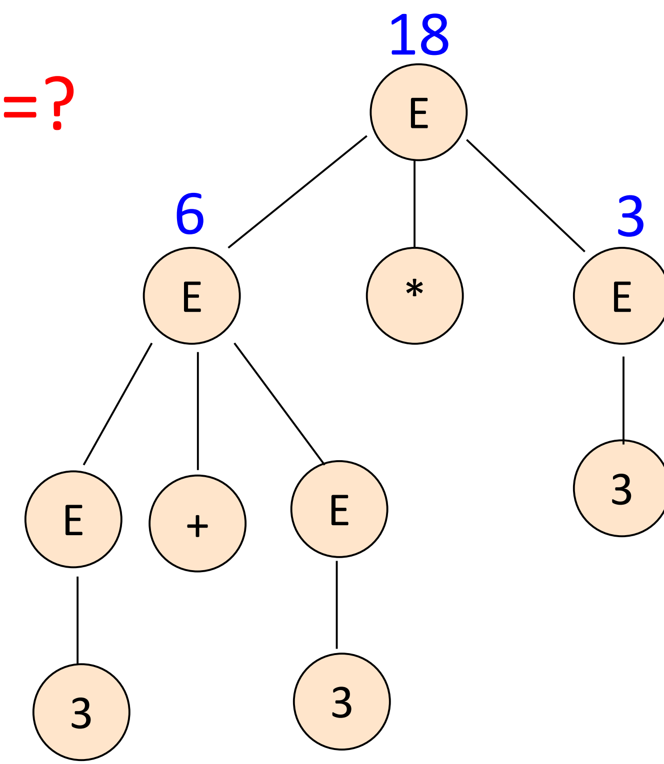


Why do we care about ambiguity?

Compute expressions result using the tree

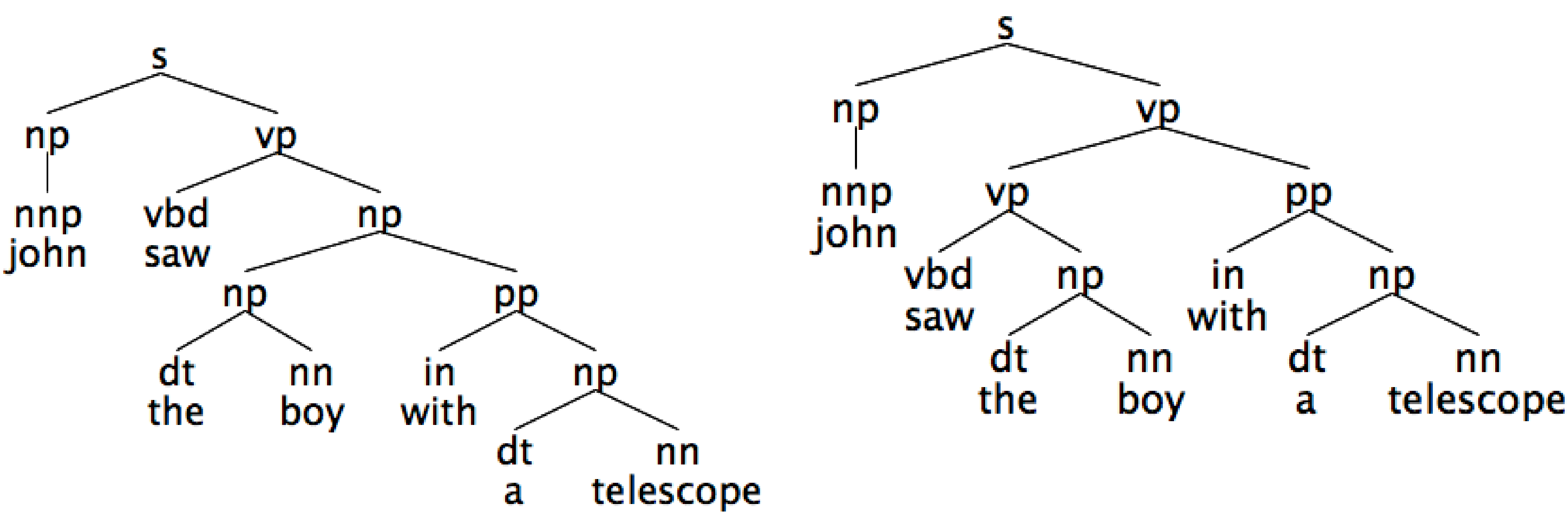


3 + 3 * 3 =?



Why do we care about ambiguity?

John saw the boy with a telescope.



Ambiguity

- In general, ambiguity is bad for programming languages and we want to remove it

- Sometimes it is possible to find a non-ambiguous grammar for a language

- But in general it is difficult to achieve this

Ambiguous Grammars

- If L is a context-free language for which there exists an unambiguous grammar, then L is said to be unambiguous. If every grammar that generates L is ambiguous, then the language is called inherently ambiguous.

- In general it is very difficult to show whether or not a language is inherently ambiguous.

Probabilistic Context-free Grammars

Assigning Probabilities

- Example:

  S $\rightarrow$ NP VP

  NP $\rightarrow$ the man

  NP $\rightarrow$ the book

  VP $\rightarrow$ Verb NP

  Verb$\rightarrow$ took

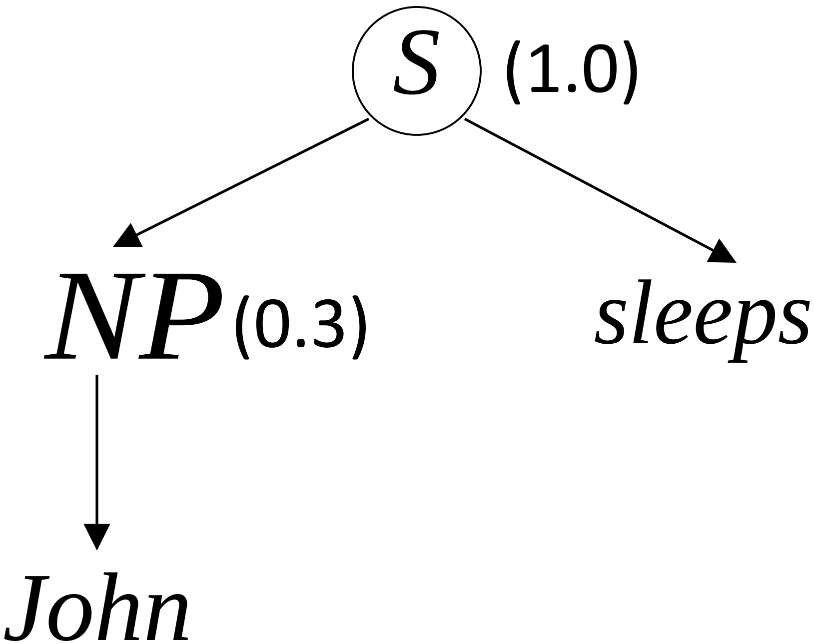
- Where can we assign probabilities?
  - Rules are nondeterministic
  - We have parse trees that are generated
  - We have terminals (strings) of a language that are generated

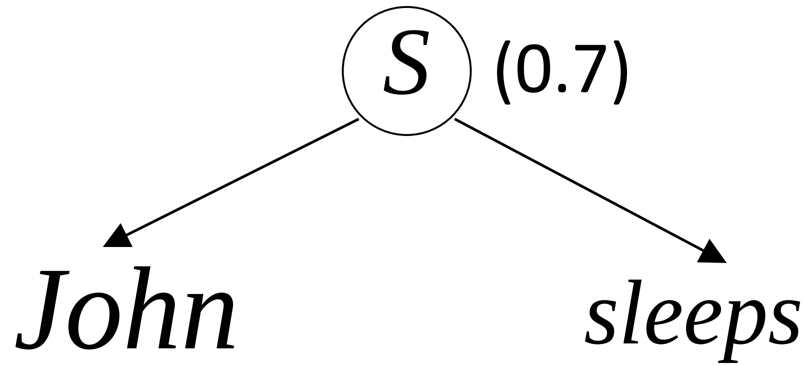
Example 1

- Example:
  S $\rightarrow$ NP sleeps (1.0)
  S $\rightarrow$ John sleeps (0.7)
  NP $\rightarrow$ John (0.3)

- What can be derived?

Tree probabilities ok but rules probabilities not ok





Tree probability:
0.7

Tree probability:
0.3

Fixing the probabilities

- Example:

  S $\rightarrow$ NP sleeps (0.3)

  S $\rightarrow$ John sleeps (0.7)

  NP $\rightarrow$ John (1.0)

Example 2

- Example:
  $\mathbf{S} \rightarrow \mathbf{S}\ \mathbf{S}$ (0.7) = p
  $\mathbf{S} \rightarrow$ a (0.3) = (1 – p)

- What can be derived? Let x_h be total probability of all parses of height ≤ x_h.
  a: $h_1 = (0.3)$
  a + aa: $h_2 = (0.3 + .7 \times .3 \times .3)$
  a + aa + aaa: $h_3 = (0.3 + .7 \times h_2 \times h_2)$

- It can be shown that if p > ½ then the total probability of all parses is less than 1.

Probability of a Parse Tree

- Let P be the set of production rules in the grammar, and let A → α be a rule. Let τ be a parse tree. For a rule A → α, let f(A → α ; τ) be the frequency of the rule in τ.
- Then:

$$p(\tau) = \prod_{(A \to \alpha) \in P} p(A \to \alpha)^{f(A \to \alpha;\tau)}$$

Note that for a proper distribution the sum of all parse trees should sum to one.

Example 3

S $\rightarrow L_5\ D_3\ S_1$ 0.8
S $\rightarrow D_3\ L_5$ 0.2
$L_5 \rightarrow$ alice (0.5) | carol (0.5)
$D_3 \rightarrow$ 111 (.3) | 123 (.3) | 999 (.25) | 007 (.15)
$S_1 \rightarrow$! 0.6
$S_1 \rightarrow$ # 0.4

- Maximum likelihood estimation assigns proper probabilities to PCFGs if one uses the full observations case (knows all the parse trees). Also termed relative frequency estimation. See Chi and Geman (1998)