University of Science

Computational Linguistics Center

Introduction to Natural Language Processing

Section 2: Formal Languages

Lecturer: Assoc.Prof. Dr. Dinh Dien

Introduction to Formal Languages

- Formal languages are defined with respect to a given ***alphabet***. The alphabet (Σ) is simply a finite set of symbols. E.g.:

- Σ (English) = [the, a, of, in, am, I, book, teacher, mother, new, read, carry out, fall in love, ...]
- Σ (Vietnamese) = [của, các, cái, một, là, trong, tôi, mới, sách, giáo viên, mẹ, đọc, thực hiện, cuốn, của đáng tội, ...]
- Σ (French) = [le, une, de, livre, mère, pomme de terre, ...]
- Σ (Chinese) = [的, 一,书,老师, 妈妈, 本, ...]
- Σ (Japanese) = [の,に,本, お母さん,...]
- Σ (Korean) = [안,이,그리고,책, 선생님,어머니, ...]

STRING

- A finite sequence of *symbols* is called a ***string***.
- The length of a string w is the number of symbols: $|w|$
- If w is a string ($w \in \Sigma^*$) and a is a symbol ($a \in \Sigma$), then $x.a$ (concatenation) will be a string on Σ
- If w,y are strings ($w,y \in \Sigma^*$), then $x.y$ (concatenation) will be a string on Σ with its length: $|xy| = |x| + |y|$.
- Ex: Σ (computer language) = {0,1}: x = "001" is a string on Σ and a = "0" is a symbol of Σ, then xa = "0010" is a new string on Σ. The length: $|x| = 3$ and $|xa| = 4$
- But: b= '2' $\notin \Sigma$, then: xb = "0012" is NOT a string on Σ.

SENTENCE

- x = “Tôi là một giáo_viên” is a sentence on Σ (Vietnamese), and a = “mới” is a word of Σ (Vietnamese), we have a new sentence in Vietnamese: xa = “Tôi là một giáo_viên mới”
- The length: $|x| = 4$ and $|xa| = 5$
- But: if b= “new” $\notin \Sigma$ (Vietnamese), then xb = “Tôi là một giáo_viên new” is NOT a Vietnamese sentence.

LANGUAGE

- Power set: from a given Σ=[0,1]), we have:
- Σ^0=[ε]: set of 0-symbol strings
- Σ^1=[0,1]: set of 1-symbol strings
- Σ^2=[00,01,10,11]: set of 2-symbol strings
- Σ^3=[000,001,010,011,100,101,110,111]
- Σ^n : set of n-symbol strings
- Σ^+ : set of n-symbol strings with $n \geq 1$
  $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots. \cup \Sigma^n$= [0,1,00,01,10,11,000,001, ...,111, ...]
- Σ^* : set of n-symbol strings with $n \geq 0$ => $\Sigma^* = \Sigma^0 \cup \Sigma^+$
- The order of strings in the power set: *normalization* or dictionary (alphabet).

LANGUAGE

- In NL, there is no Σ^0. Why?
- Σ^1= {"Cháy!", "Cướp!",…},{"Fire!", …} (limited)
- Σ^2 ={ "Tôi đi", "Anh ngủ", "Trời mưa", "Giáo_viên nghiên_cứu", "Sinh_viên thực_hiện",…}, {"It rains", "I go", "You sleep", "Teachers research", "Students carry_out",…} (many)
- Σ^3, Σ^4 ,... Σ^n : increase exponentially
- Ex: Σ (computer) = {0,1}
- => Σ^3 = [000, *001*, 010, *011*, *100*, 101, 110, 111]

✓ Only *italic* strings are valid.

➢ L = [*001*, *011*, *100*] $\subset \Sigma^3 \subset \Sigma^*$

LANGUAGE DEFINITIONS

- DEFINITION OF A LANGUAGE: is a ***subset*** of Σ^*
- How to determine that ***subset***?
- If the number is *limited*, we can *enumerate* them.
- How large is the subset ? Limited or Unlimited ?
- For Σ=[0,1] (its vocabulary has 2 words only)
- Σ^8: $2^8 = 256$ cases; Σ^{16} : $2^{16} = 65.536$; Σ^{32} : $2^{32} \sim 4,2$ billion. Σ^{64}: $2^{64} \sim 18$ million billion (quintillion) cases.
- The size of this "rice heap"(chess story)is 32 billion m^3, approx.12,000 times of the biggest pyramid (Khufu,Egypt)
- Luckily, the number of valid strings (instruction) is small/limited (opcode) => instruction set of CPU.
- But: in other languages ?

LANGUAGE PRINCIPLES

- Σ=[0,1], L_1= [11,011,101,0101,0110,...], $L_1 \subset \Sigma^*$
- L_1's principle? *The number of '1s' is 2*.
- L_2= [11,011,0101,0110,11011,110011,1011111,1101111,...]
- L_2's principle? *The number of '1s' is even*.
- L_3= [01,0101,0110, 010011, 011100, 01110010,10010101,...]
- L_3's principle? *The number of '1s' = '0s'*.

➢ L_1, L_2 , L_3: infinite countable sets

○ ***What is the principle of NL?***

➢ *The principles of phonetics/phonology, morphology,* ***grammar****(syntax), SEMANTICS, pragmatics,…*

NATURAL LANGUAGES

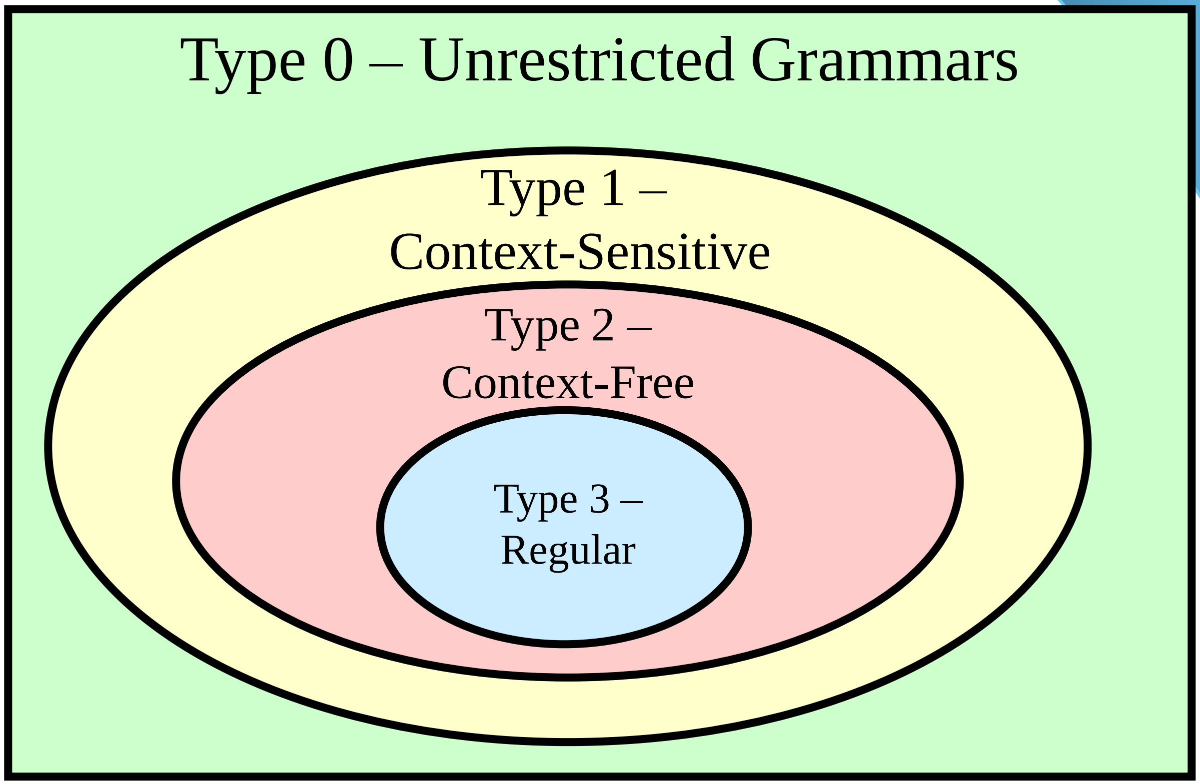
- Σ^+= [“Tôi là một”, “Tôi là một mới”, “là giáo_viên một mới sách”, “Tôi đọc giáo_viên sách mới”, “*Tôi đọc sách giáo_viên mới*”, “*Mẹ đọc sách giáo_viên mới*”, “Đọc mới giáo_viên sách là”,“*Mẹ mới đọc sách giáo_viên*”,“của sách một”,...]
- Italic sentences: grammatically & semantically correct.
- The number of correct sentences: huge!
- Σ (Vietnamese) ~ 40k words, the avg length~ 20w
- Candidate sentences: 40.000^{20}; 40.000 > 2^{15} (= 32.768) => $40.000^{20} > (2^{15})^{20} = 2^{300} = 2^{30 \times 10} = (10^9)^{10} = (10^{90})$ (90 ‘0s’ !)
- The number of sand grains on the Earth: 10^{21}
- The number of stars in the universe: 7×10^{22} (sextillion)

NATURAL LANGUAGES

- Possible Vietnamese sentences: 10^{40}
- English (Robert Mannell at Macquarie, Aus): 10^{570} http://clas.mq.edu.au/speech/infinite_sentences/index.html
- Extremely huge!
- We need another way to determine the subset of NL
- That is: GRAMMAR
- SUBSET = {collection of grammatically correct sentences}

Types of Grammars: Chomsky hierarchy

- 0. UG: $\alpha \rightarrow \beta$ with $\beta \in (N \cup \Sigma)^*$ and $\alpha \in (N \cup \Sigma)+$
- 1. CSG: $\gamma_1 X \gamma_2 \rightarrow \gamma_1 \alpha \gamma_2$ with $X, \alpha, \gamma_1, \gamma_2 \in (N \cup \Sigma)+$
- 2. CFG: $X \rightarrow \alpha$ with $X \in N$, $\alpha \in (N \cup \Sigma)+$
- 3. RG: $X \rightarrow a$ and $X \rightarrow aY$ or $X \rightarrow Yb$ with $X, Y \in N$ $a,b \in \Sigma$



DERIVATION

- Let G(V, Σ, P, S) be a CFG.
- Let w_0= αXβ (the concatenation of α,X and β) and w_1= αγβ be strings over Σ.
- If X → γ is a production of P we say that *w1* is directly derivable from *w0* and we write $w_o \Rightarrow w_1$
- If $w_0, w_1, \ldots, w_n$ are strings over V such that $w_0 \Rightarrow w_1$, $w_1 \Rightarrow w_2, \ldots, w_{n-1} \Rightarrow w_n$, then we say that w_n is derivable from w_0, and write $w_0 \Rightarrow^* w_n$
- The sequence of steps used to obtain w_n from w_o is called a derivation.

DEFINITION OF LANGUAGES

- Let G(V, Σ, P, S) be a CFG.
- The language generated by G denoted by L(G) , is the set of all strings of terminals that are derivable from the start symbol S.

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

Derivation Order

Consider the following example grammar with 5 productions:

$$1.\ S \rightarrow AB \qquad 2.\ A \rightarrow aaA \qquad 4.\ B \rightarrow Bb$$

$$3.\ A \rightarrow \lambda \qquad 5.\ B \rightarrow \lambda$$

$$1.\ S \to AB \qquad 2.\ A \to aaA \qquad 4.\ B \to Bb$$

$$3.\ A \to \lambda \qquad 5.\ B \to \lambda$$

Leftmost derivation order of string :

$$S \overset{1}{\Rightarrow} AB \overset{2}{\Rightarrow} aaAB \overset{3}{\Rightarrow} aaB \overset{4}{\Rightarrow} aaBb \overset{5}{\Rightarrow} aab$$

At each step, we substitute the leftmost variable

$$1.\ S \to AB \qquad 2.\ A \to aaA \qquad 4.\ B \to Bb$$

$$3.\ A \to \lambda \qquad 5.\ B \to \lambda$$

Rightmost derivation order of string : aab

$$S \overset{1}{\Rightarrow} AB \overset{4}{\Rightarrow} ABb \overset{5}{\Rightarrow} Ab \overset{2}{\Rightarrow} aaAb \overset{3}{\Rightarrow} aab$$

At each step, we substitute the rightmost variable

$$1.\ S \to AB \qquad 2.\ A \to aaA \qquad 4.\ B \to Bb$$
$$3.\ A \to \lambda \qquad 5.\ B \to \lambda$$

Leftmost derivation of : aab

$$S \overset{1}{\Rightarrow} AB \overset{2}{\Rightarrow} aaAB \overset{3}{\Rightarrow} aaB \overset{4}{\Rightarrow} aaBb \overset{5}{\Rightarrow} aab$$

Rightmost derivation of : aab

$$S \overset{1}{\Rightarrow} AB \overset{4}{\Rightarrow} ABb \overset{5}{\Rightarrow} Ab \overset{2}{\Rightarrow} aaAb \overset{3}{\Rightarrow} aab$$

Derivation Trees

Consider the same example grammar:

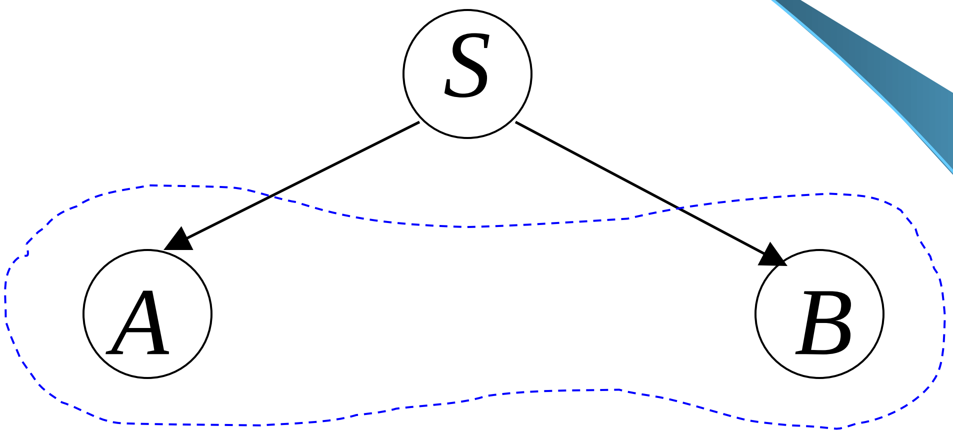
$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

And a derivation of : *aab*

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

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

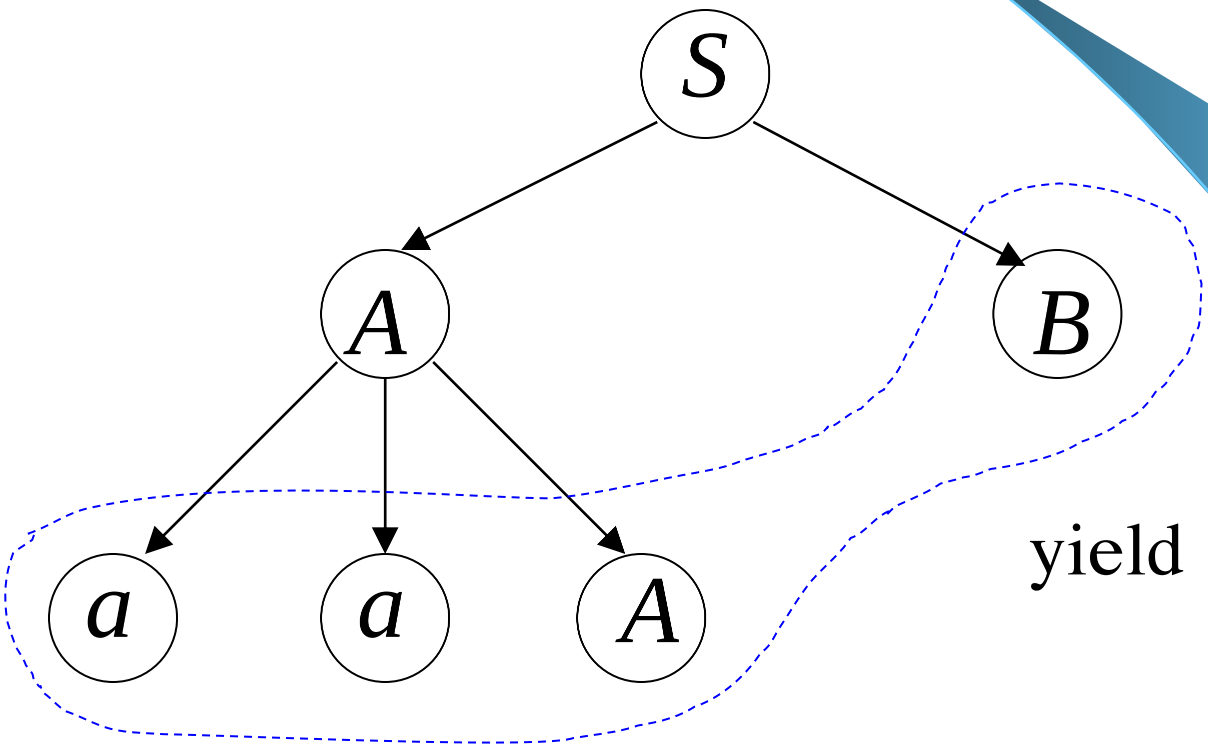
$$S \Rightarrow AB$$



yield AB

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

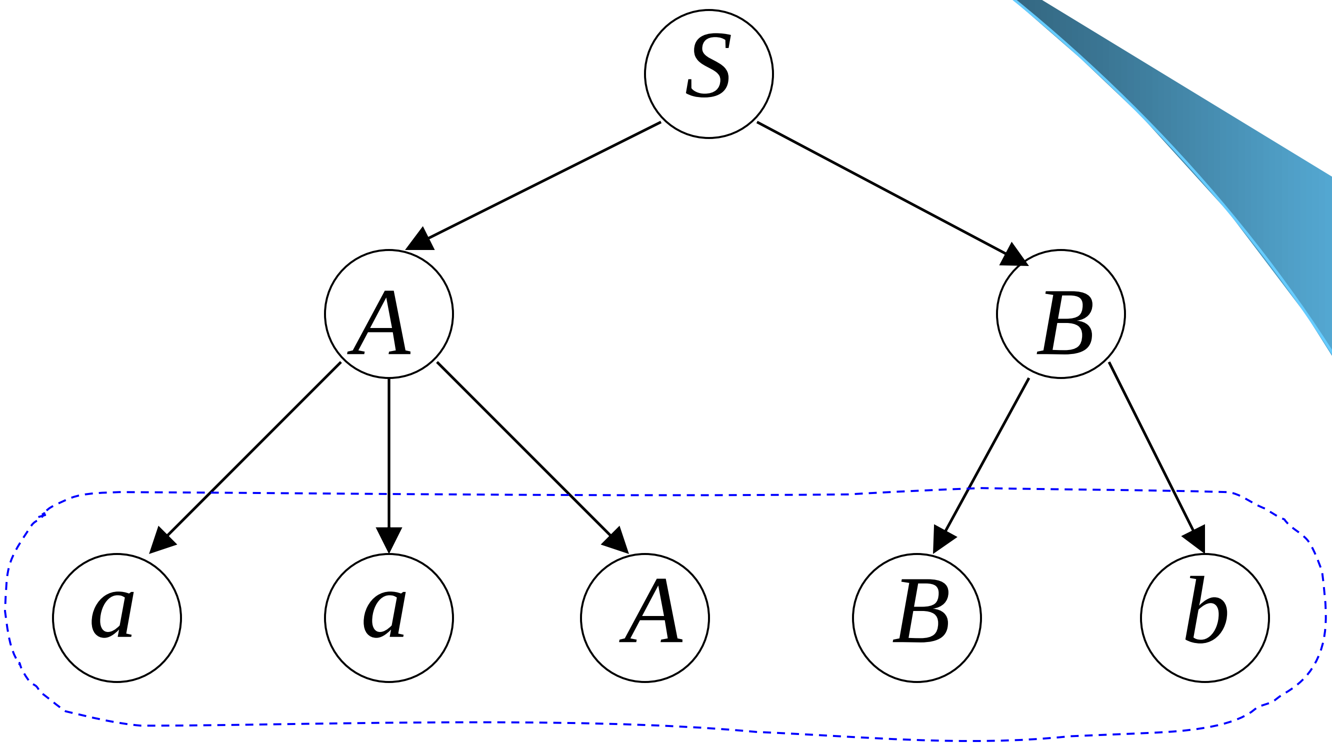
$$S \Rightarrow AB \Rightarrow aaAB$$



yield $aaAB$

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

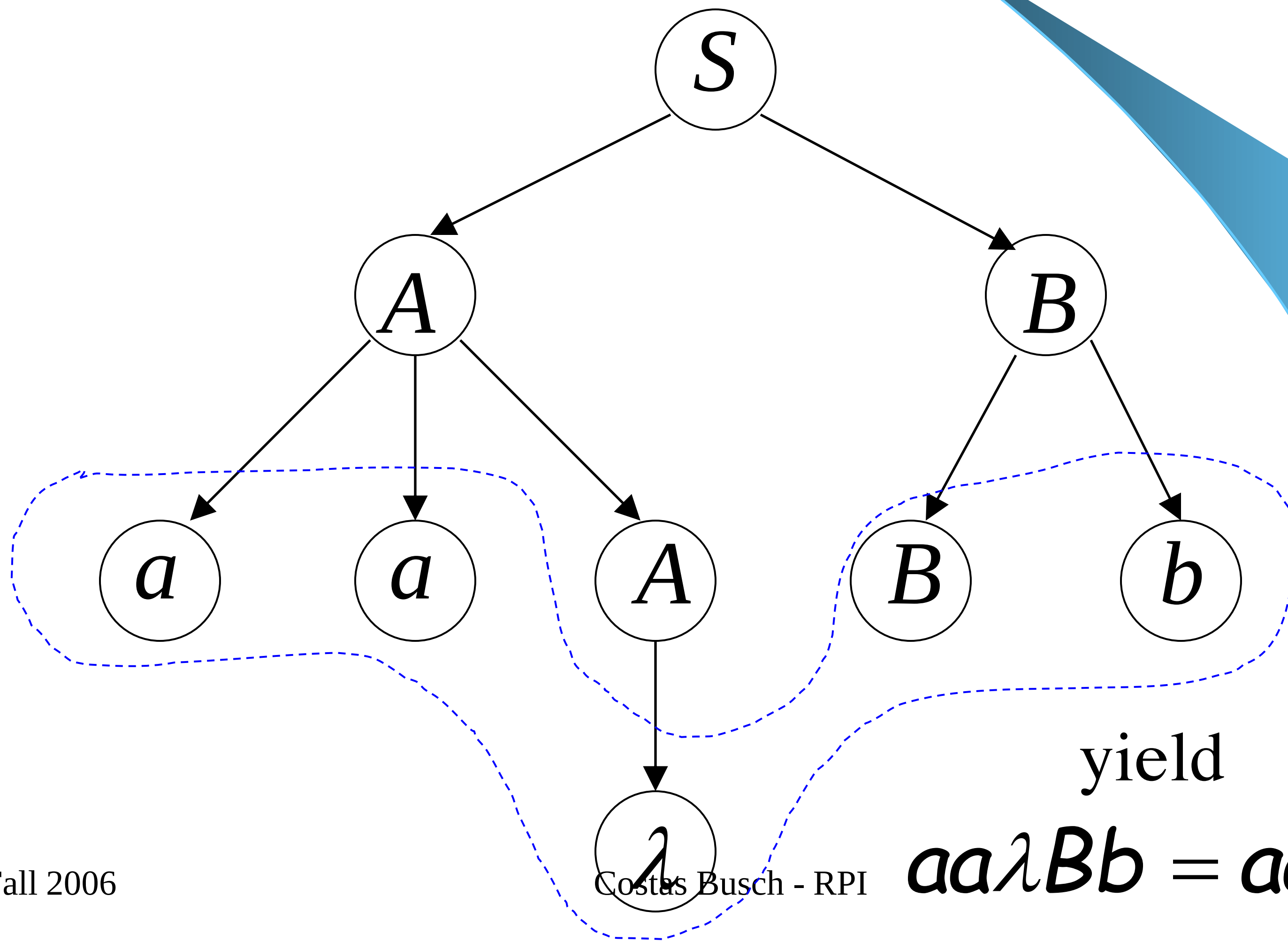
$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb$$



yield $aaABb$

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

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



yield

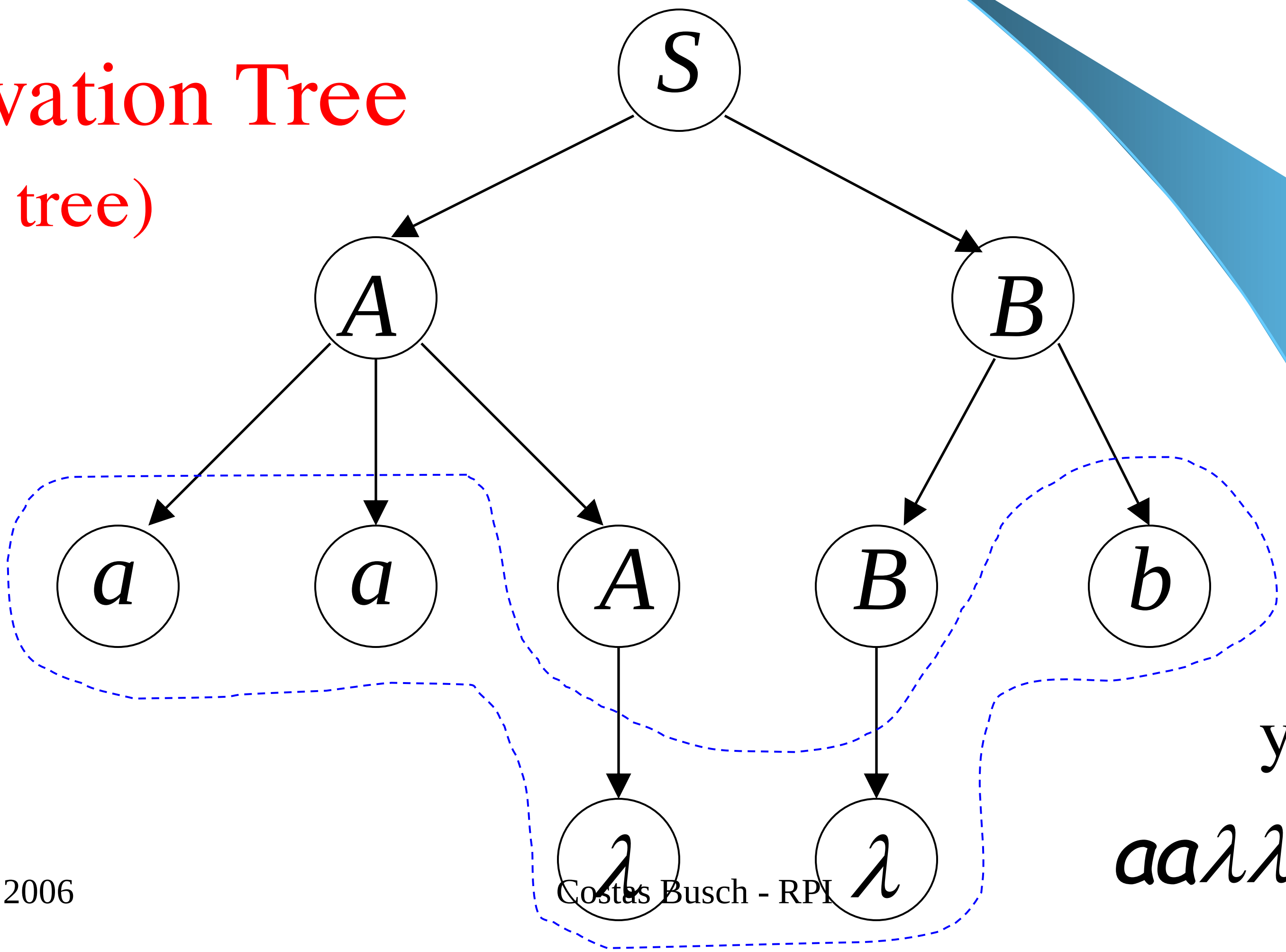
$$aa\lambda Bb = aaBb$$

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

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

Derivation Tree

(parse tree)



yield

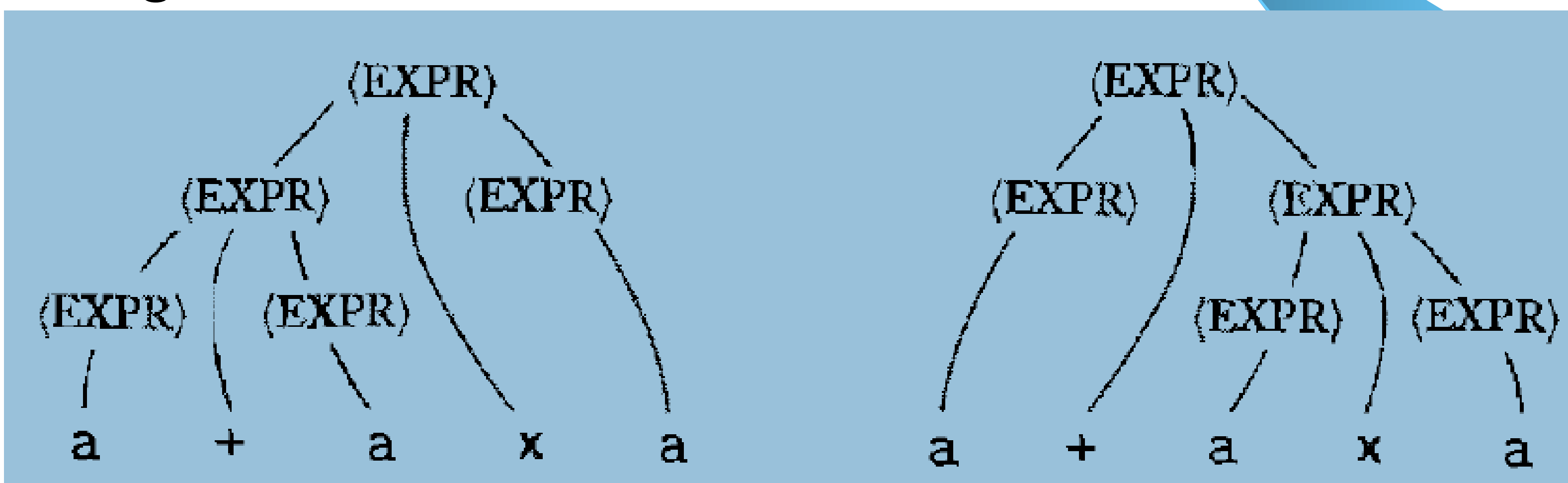
$aa\lambda\lambda b = aab$

AMBIGUOUS

- A sentence which can derive more than one parse *tree* is **ambiguous**.
- Example: a grammar for arithmetic expressions:

‹EXPR› →‹EXPR› + ‹EXPR› |‹EXPR› . ‹EXPR› |(‹EXPR›)| a

✓ String: a + a.a



- A language in which contains at least one ambiguous sentence is **ambiguous.**

➢ Give examples of ambiguous sentences in NL?

+*Equivalent* grammar:

G1 and G2 are equivalent iff $L(G1) = L(G2)$

+ *R*ecursive:

Given a $G = (N, \Sigma, P, S)$, a non-term X of G is recursive if there are following production rules:

$X \Rightarrow \alpha X\beta$ with $\alpha, \beta \in (\Sigma \cup N)^*$

If $\alpha = \varepsilon$: left recursive

If $\beta = \varepsilon$: right recursive

If $\alpha, \beta \neq \varepsilon$: middle recursive

Indirect recursive ?

Give examples of recursive structures in NL?

+*C*ategorized grammar

Given a grammar $G = (N, \Sigma, T, P, S)$.

T is set of categories of terminals.

$\Sigma = \cup T_i$

With $i \neq j$: $T_i \cap T_i = \phi$?

Give examples in English, Vietnamese, etc.

+ *CNF* (Chomsky Normal Form):if every production rules in P have following forms:

$X \rightarrow YZ$ with $X, Y, Z \in N$, or

$X \rightarrow a$ with $a \in \Sigma$

Can CFG be converted into CNF ?

Backus-Naur Form

- ⟨sentence⟩ ::= ⟨noun phrase⟩ ⟨verb phrase⟩
- ⟨noun phrase⟩ ::= ⟨article⟩ [⟨adjective⟩] ⟨noun⟩
- ⟨verb phrase⟩ ::= ⟨verb⟩ [⟨adverb⟩]
- ⟨article⟩ ::= a | the
- ⟨adjective⟩ ::= large | hungry
- ⟨noun⟩ ::= rabbit | mathematician
- ⟨verb⟩ ::= eats | hops
- ⟨adverb⟩ ::= quickly | wildly

Square brackets [] mean “optional”

Vertical bars mean “alternatives”

Given a grammar:

S ::= NP VP

NP ::= det noun

NP ::= det adj noun

NP ::= det noun PP

VP ::= verb

PP ::= prep NP

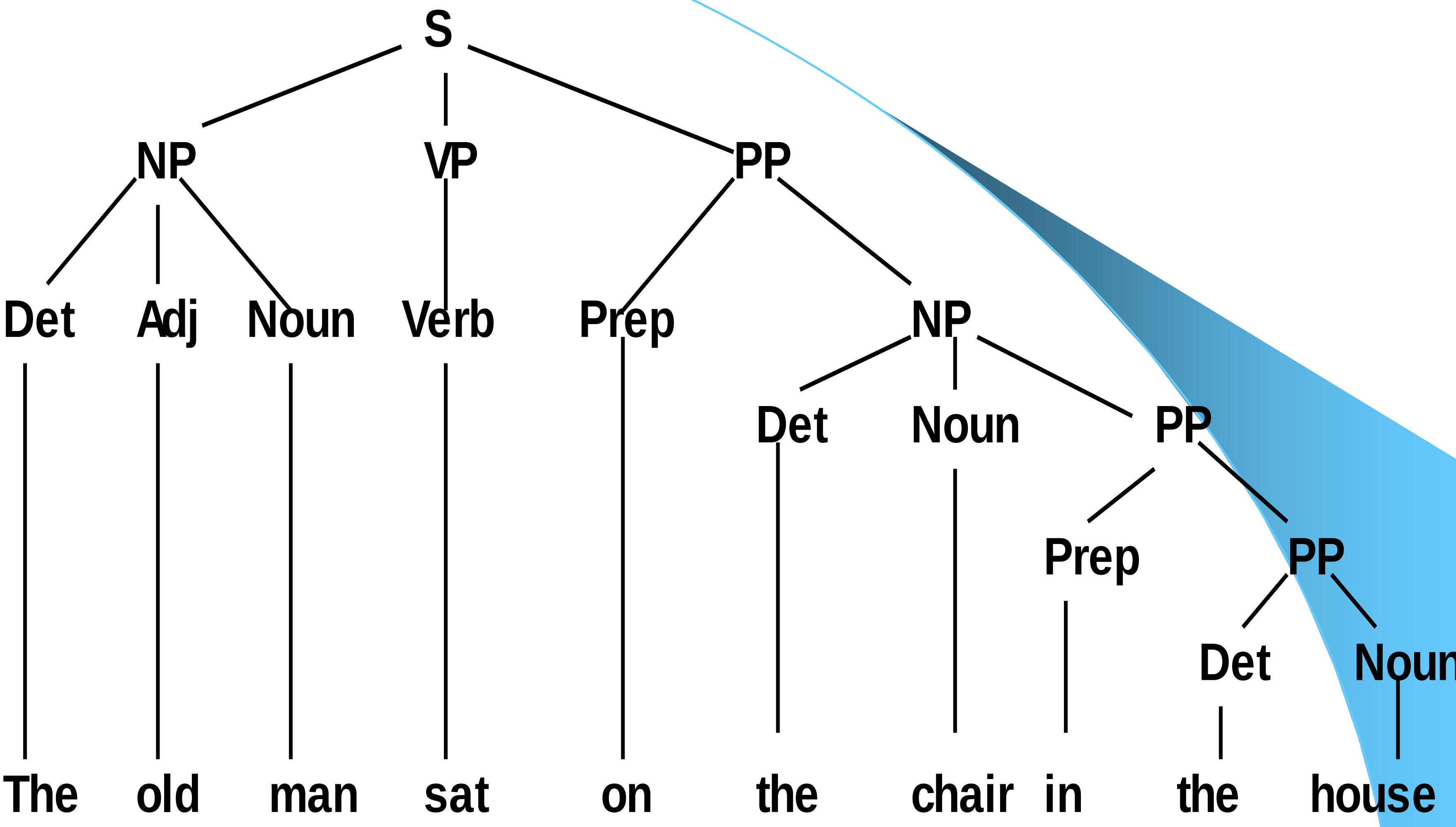
PP ::= prep NP PP

Is that CFG ?

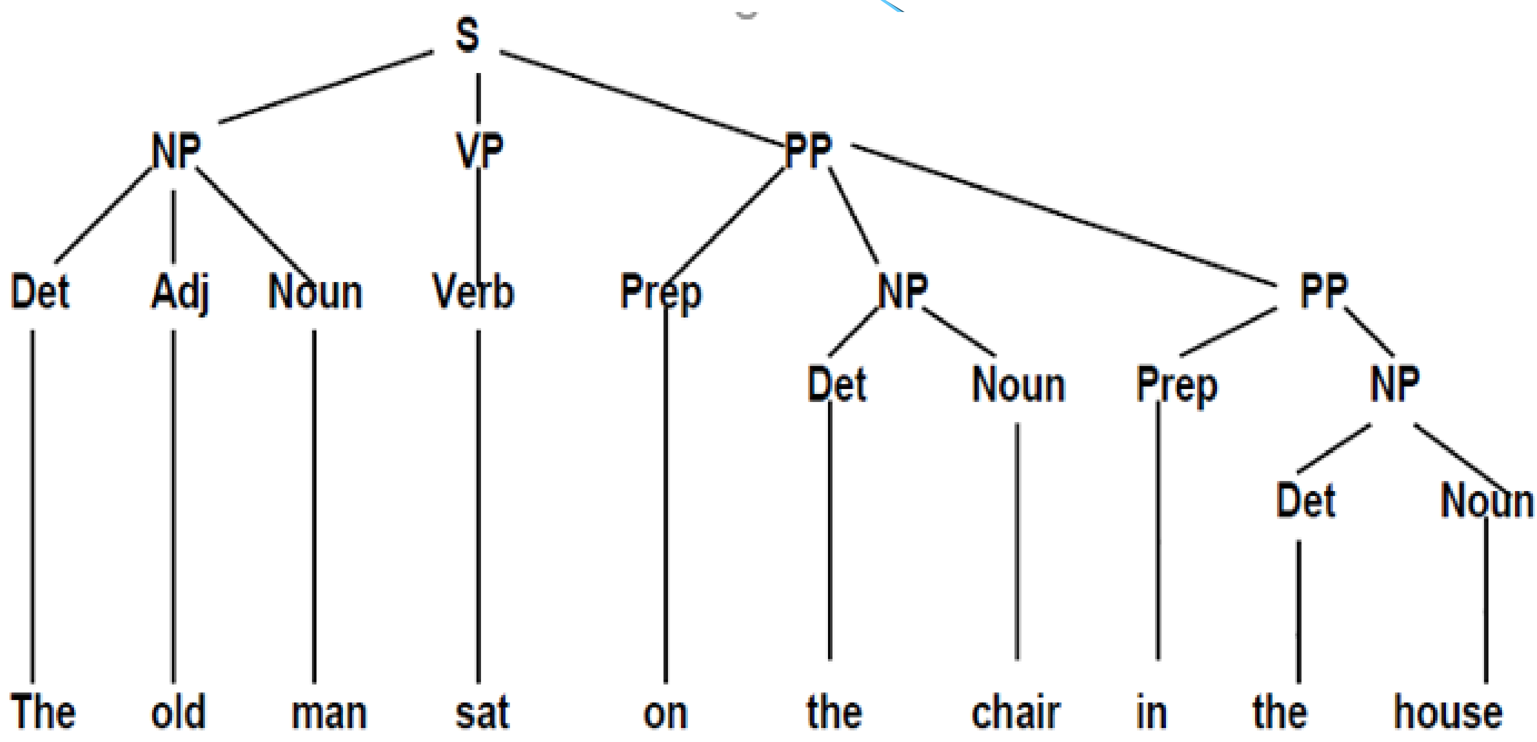
Parse: “The old man sat on the chair in the house”

Is that sentence ambiguous?

Derivation tree #1



Derivation tree #2



EARLEY PARSING ALGORITHM

Given a CFG grammar:

$S \rightarrow NP\ VP$

$S \rightarrow NP\ VP\ PP$

$NP \rightarrow d\ NP3$

$NP3 \rightarrow a\ NP3$

$NP3 \rightarrow n$

$NP3 \rightarrow n\ PP$

$PP \rightarrow p\ NP2$

$NP2 \rightarrow d\ NP3$

$VP \rightarrow v$

- In which: d (det), a (adj), p (prep), n (noun), v (verb): are terminals
- S, NP, NP3, NP2, VP, PREPS : non-terminal symbols
- Parse the sentence: “ The young student sat in the class”
- POS: “The/d young/a student/n sat/v in/p the/d class/n”
- Formally: $w = a_1a_2a_3a_4a_5a_6a_7$
- With: a_1=d; a_2=a; a_3=n; a_4=v; a_5=p; a_6=d; a_7=n;

Building the parsing list

- (1) If $S \rightarrow \alpha \in P$, then add $[S \rightarrow \bullet\ \alpha, 0]$ into I_0
- Repeat (2) and (3) until no new added-item in I_0
- (2) If $[B \rightarrow \gamma\ \bullet, 0] \in I_0$, then add:
  $[A \rightarrow \alpha B \bullet \beta, 0]$ for all items $[A \rightarrow \alpha \bullet B\ \beta, 0]$ in $I_{0.}$
- (3) If $[A \rightarrow \alpha \bullet B\ \beta, 0] \in I_0$, add into I_0, all items $[B \rightarrow \bullet\ \gamma, 0]$ (if no exist in I_0).

Building the parsing list

- Building the next table: I_j
- (4) For each item $[B \rightarrow \alpha \bullet a\ \beta, i]$ in table I_{j-1} with $a = a_j$, we add item $[B \rightarrow \alpha\ a \bullet\ \beta, i]$ into table I_j
- Repeat (5) and (6) until no new added-item in I_j.
- (5) If $[A \rightarrow \alpha \bullet B\beta, i] \in I_j$, we add into I_j items $[B \rightarrow \bullet\ \gamma, j]$.
- (6)If $[A \rightarrow \alpha \bullet, i] \in I_j$, search in I_i for items $[B \rightarrow \alpha \bullet A\beta, k]$
- If found, we add item $[B \rightarrow \alpha A \bullet \beta, k]$ into table I_j.

- Table I0:
- $S \rightarrow \bullet NP\ VP, 0$
- $S \rightarrow \bullet NP\ VP\ PP, 0$
- $NP \rightarrow \bullet d\ NP3, 0$
- Table I1: (The/a_1=d)
- $NP \rightarrow d \bullet NP3, 0$
- $NP3 \rightarrow \bullet a\ NP3, 1$
- $NP3 \rightarrow \bullet n, 1$
- $NP3 \rightarrow \bullet n\ PP, 1$
- Table I2 : (young/ a_2=a)
- $NP3 \rightarrow a \bullet NP3, 1$
- $NP3 \rightarrow \bullet a\ NP3, 2$
- $NP3 \rightarrow \bullet n, 2$
- $NP3 \rightarrow \bullet n\ PP, 2$

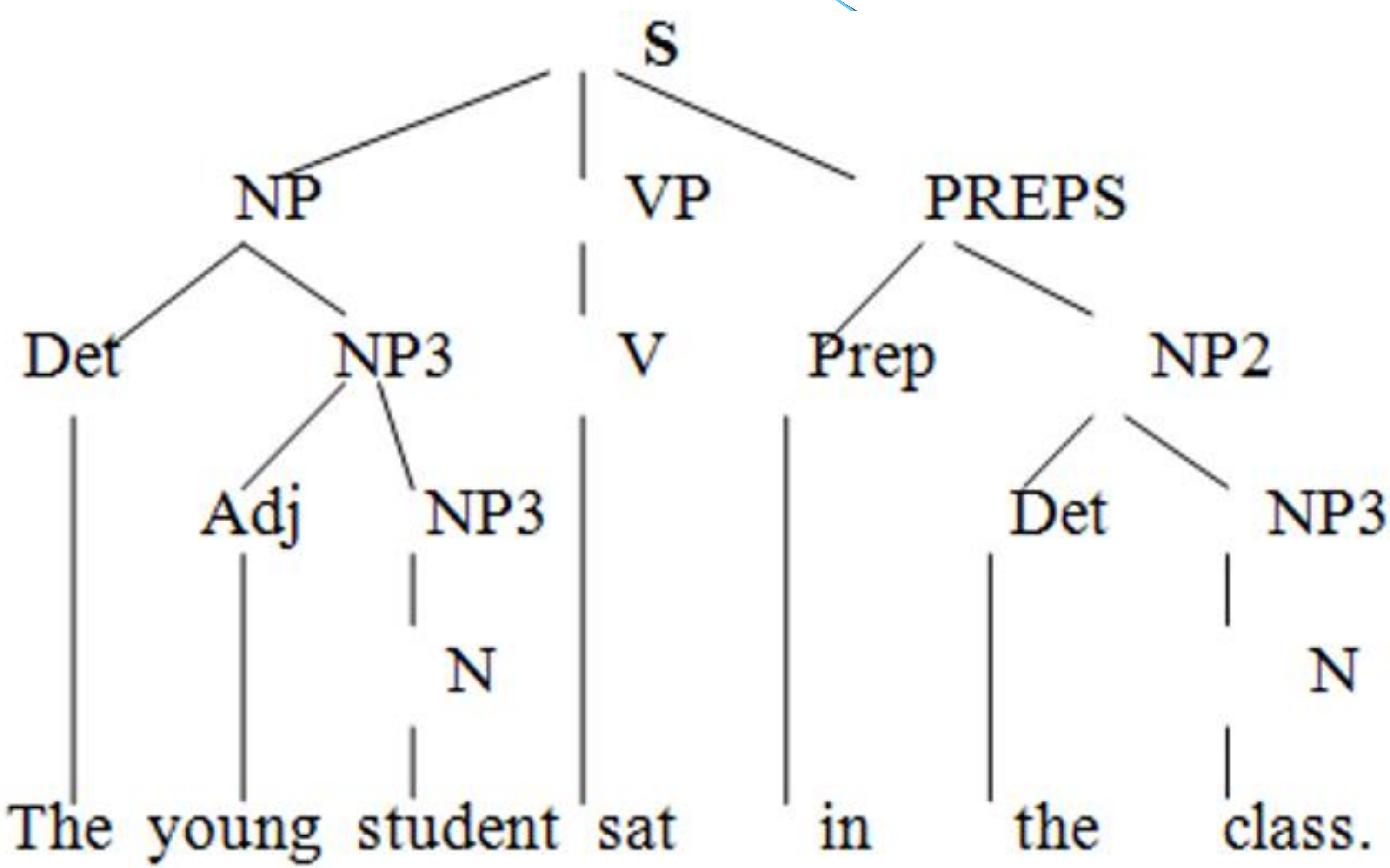
- Table I3 : (student/ a_3=n)
- NP3 → N •, 2
- NP3 → N •PP, 2
- PP → •p NP2, 3
- NP3 → a NP3• , 1
- NP → d NP3• , 0
- S → NP • VP, 0
- S → NP • VP PP, 0
- VP → • v, 3
- Table I4 : (sat/ a_4=v)
- VP → V •, 3
- S → NP VP •, 0 *
- S → NP VP• PP, 0
- PP → •p NP2, 4

- Table I5 : (in/a_5=p)
- PP → p • NP2, 4
- NP2 →• d NP3, 5
- Table I6 : (the/a_6=d)
- NP2 → d • NP3, 5
- NP3 → •a NP3, 6
- NP3 → •n, 6
- NP3 → •n PP, 6
- Table I7 : (class/a_7=n)
- NP3 → n•, 6
- NP3 → n• PP, 6
- PP → •p NP2, 7
- NP2 → d NP3• , 5
- PP → p NP2 •, 4
- S → NP VP PP •, 0 **

ALGORITHM FOR BUILDING THE PARSING TREE

- From list of tables => building the parsing tree
- If $[S \rightarrow \alpha \bullet, 0] \notin I_n$ (last table)=>wrong sentence => no tree
- Else: correct sentence => building the tree:
- Initialize a global variable: k = n (length of w)
- (1) Assign: A = S and $\alpha = X_1X_2\ldots X_m$. From R->L:
- (2) If $X_i \in \Sigma$, k--
- (3) If $X_i \in N$, find item $[X_i \rightarrow \gamma \bullet, r]$ in table I_k
- (4) Assign $A=X_i$ and $\gamma = X_1X_2\ldots X_m$.
- (5) Repeat from (2) until k=0

SYNTAX TREE


S
NP
VP
PREPS
Det
NP3
V
Prep
NP2
Adj
NP3
Det
NP3
N
N
The young student sat in the class.

- **Homework:** using above-mentioned grammar to:
- parse the sentence: " An old man sat on the new chair in the house"
- If Ok => draw the parsing tree.
- Else: modify the grammar to accept that sentence.
- Note: avoid "flip-flop" phenomenon.