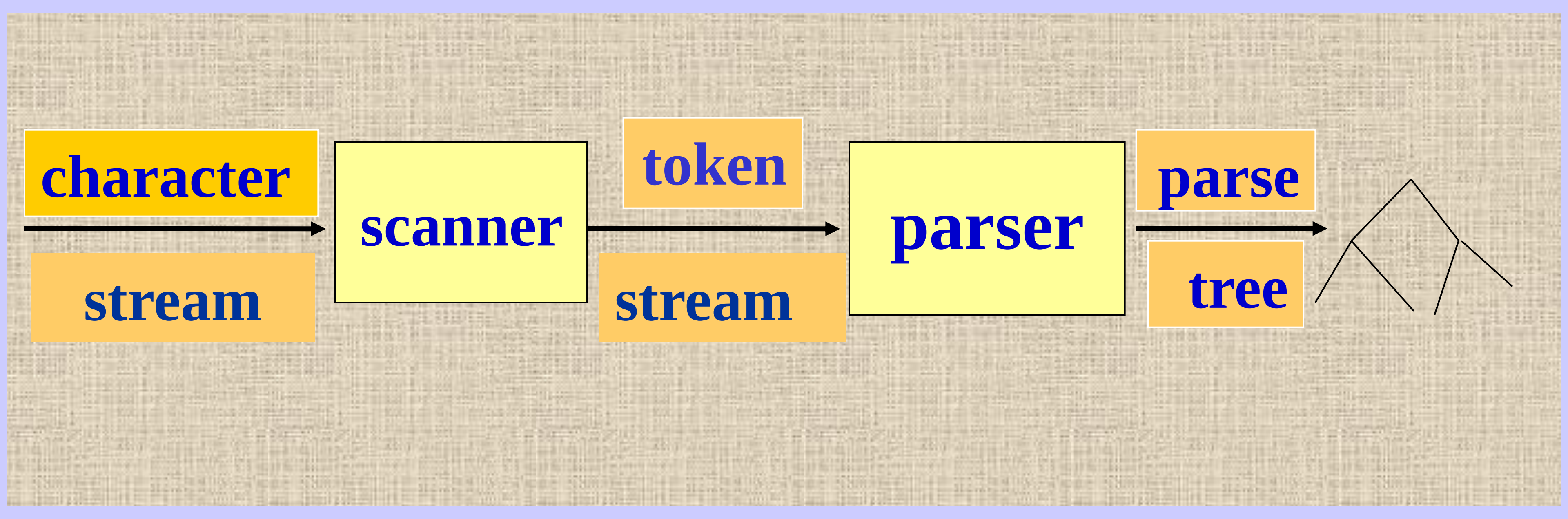
Compiler Constructions

Chapter 4(Parsing)

Part 1

Dr. Doaa Shebl
Faculty of Computers and Artificial Intelligence
Beni-Suef University

Compiler Construction



$$G = (V, \Sigma, P, S)$$

$$L(G) = \{w \,/\, w \in \Sigma^* \mid S \Rightarrow^* w\}.$$

$$\text{G: } S \to a\,S\,b \,/\, ab$$

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

$$\text{L} = \{ab, aabb, \ldots\}$$

$$L \subseteq \Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$$

Derivation

$S \Rightarrow ab$

$S \Rightarrow aSb$
$\Rightarrow aabb$

$S \Rightarrow aSb$
$\Rightarrow aaSbb$
$\Rightarrow \cdots aa...bb$

G: $S \rightarrow a\,S\,b \,/\, SS \,/ab$

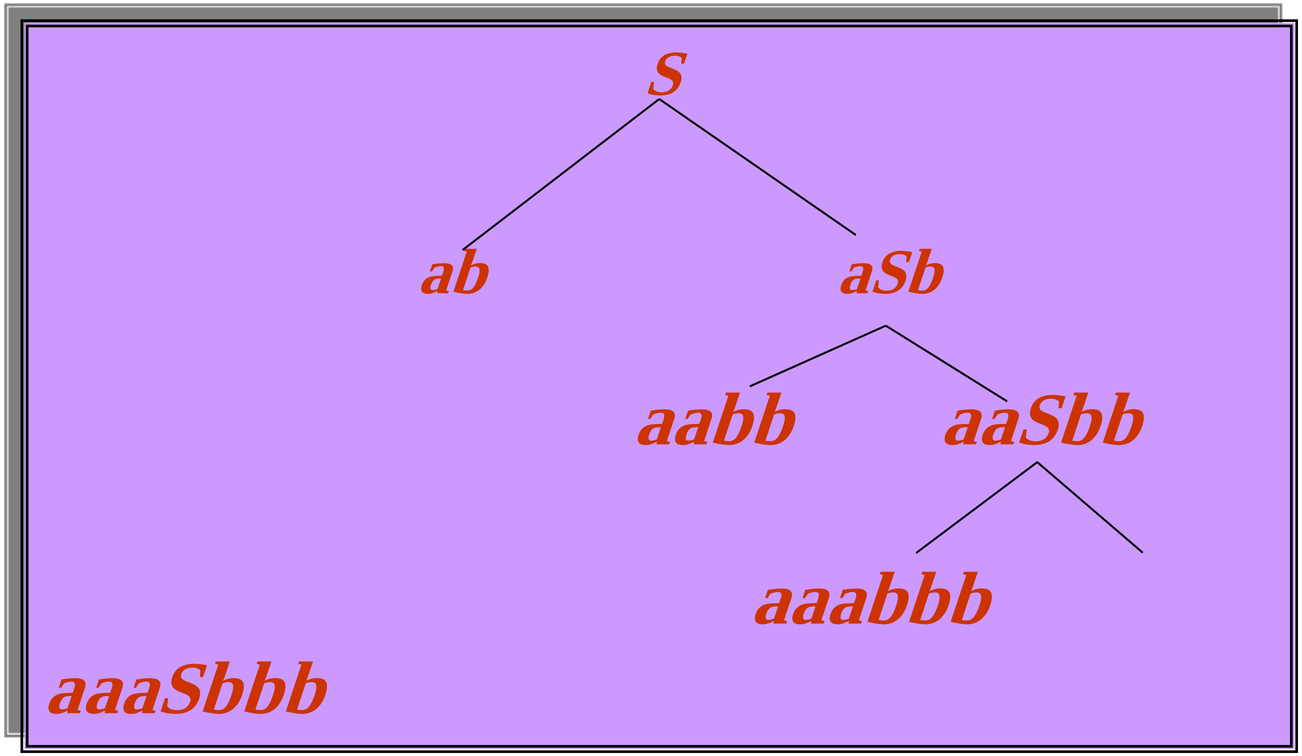
The role of the parser

Yes (syntactically correct)

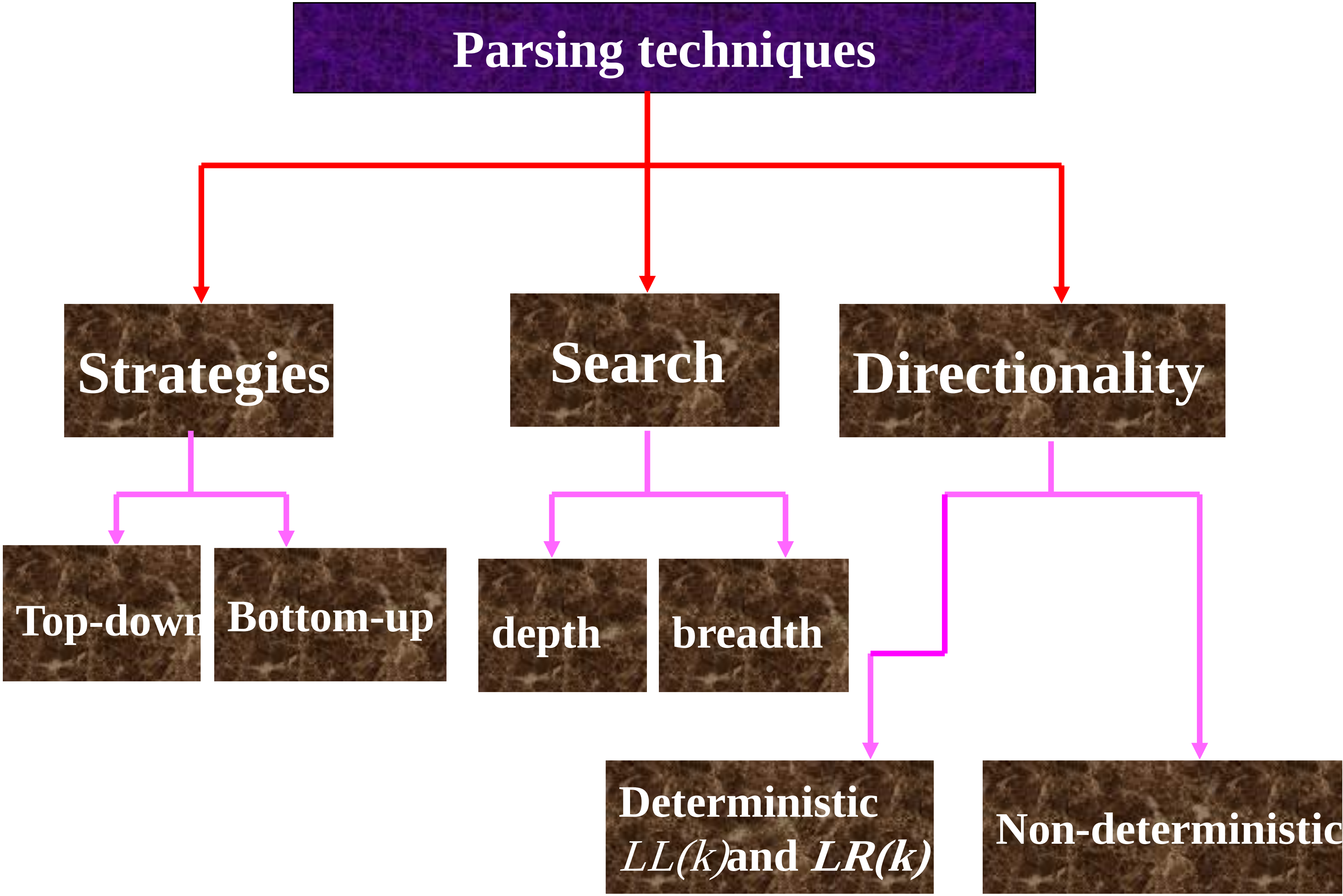
Is $w \in L(G)$?

No (syntax error)

G: $S \rightarrow a\,S\,b \,/\, ab$



Directed graph of the grammar


Parsing techniques
Strategies
Search
Directionality
Top-down
Bottom-up
depth
breadth
Deterministic
LL(k) and *LR(k)*
Non-deterministic

1- Breadth-First Top-down Parsing Algorithm

Input: context –free grammar G (V, Σ, *P*, *S*)
String $p \in \Sigma^*$
queue **Q**

1. initialize T with root *S*
   INSERT (S, Q)
2. **repeat**
   2.1. q:=*REMOVE*(Q)
   2.2. $i = 0$
   2.3. done = false
   let q = uAv where A is the first variable in q.
   2.4. **repeat**
   - 2.4.1. if there is no A rule numbered greater than i
     then done:=true
   - 2.4.2. if not done **then**
     Let $A \rightarrow w$ be the first A rule with numbered greater than i.
     Let j be the number of this rule.
     2.4.2.1 if $uwv \notin \Sigma^*$ and the terminal prefix of uwv matches a prefix of p then
     2.4.2.1.1 *INSERT* (uwv, Q)
     2.4.2.1.2 Add node uwv to T. Set a pointer from uwv to q.
     end if
     end if
   - 2.4.3. $i = j$

   until done or $p = uwv$

   until *EMPTY* or $p = uwv$
3. if $p = uwv$ **then** accept **else** reject

Breadth-First Top-down Parsing Algorithm

Example: To illustrate how to construct a string (b+b) by using the above algorithm Let G be the grammar, where

$$G = (\{S, A, T\}, \{b, +, (,)\}, P, S)$$

P: 1. $S \rightarrow A$

2. $A \rightarrow T$

3. $A \rightarrow A+T$

4. $T \rightarrow b$

5. $T \rightarrow (A)$

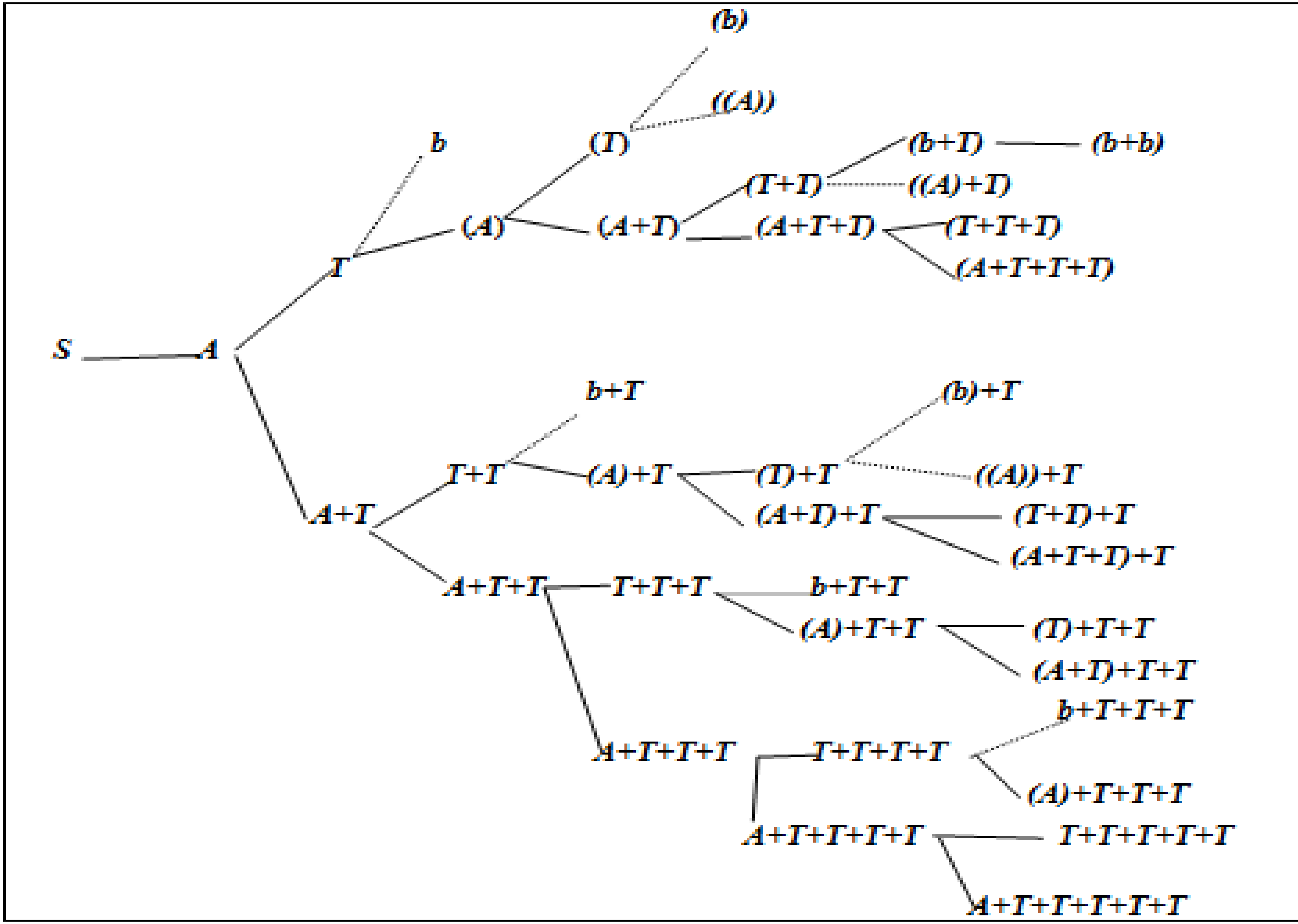


Figure 1. Path and derivation generated by Breadth-first search for (b+b).

2- Top-down parsing: Depth first Search

```
Input : context –free grammar G ( V, Σ, P, S)
                    string p ∈ Σ*
                    stack S
       1. PUSH([S, 0, 0], S)
       2. repeat
             2.1. [q, i] := POP(S)
             2.2. dead-end =false
             2.3. repeat
                      let q = uAv where A is the left most variable  in q.
                      2.3.1. if u is not a prefix  of p then dead-end = true
                      2.3.2. if there is no A rule numbered  greater  than  i
    then
                                  dead-end = true
                2.3.3.  if not dead-end  then
                              Let A→w be the  first A rule  with number
    grater  than  i
                              Let j be the number  of this rule
                              2.3.3.1. PUSH ([q, j],S)
                              2.3.3.2. q = uwv
                              2.3.3.3. i:= 0
                         end if
              until dead-end or q∈ Σ*
        until  q = p or EMPTY(S)
```

Depth-first top-down parsing algorithm

Example: To illustrate how to construct a string (b+b) by using the above algorithm Let G be the grammar, where

$$G = (\{S, A, T\}, \{b, +, (,)\}, P, S)$$

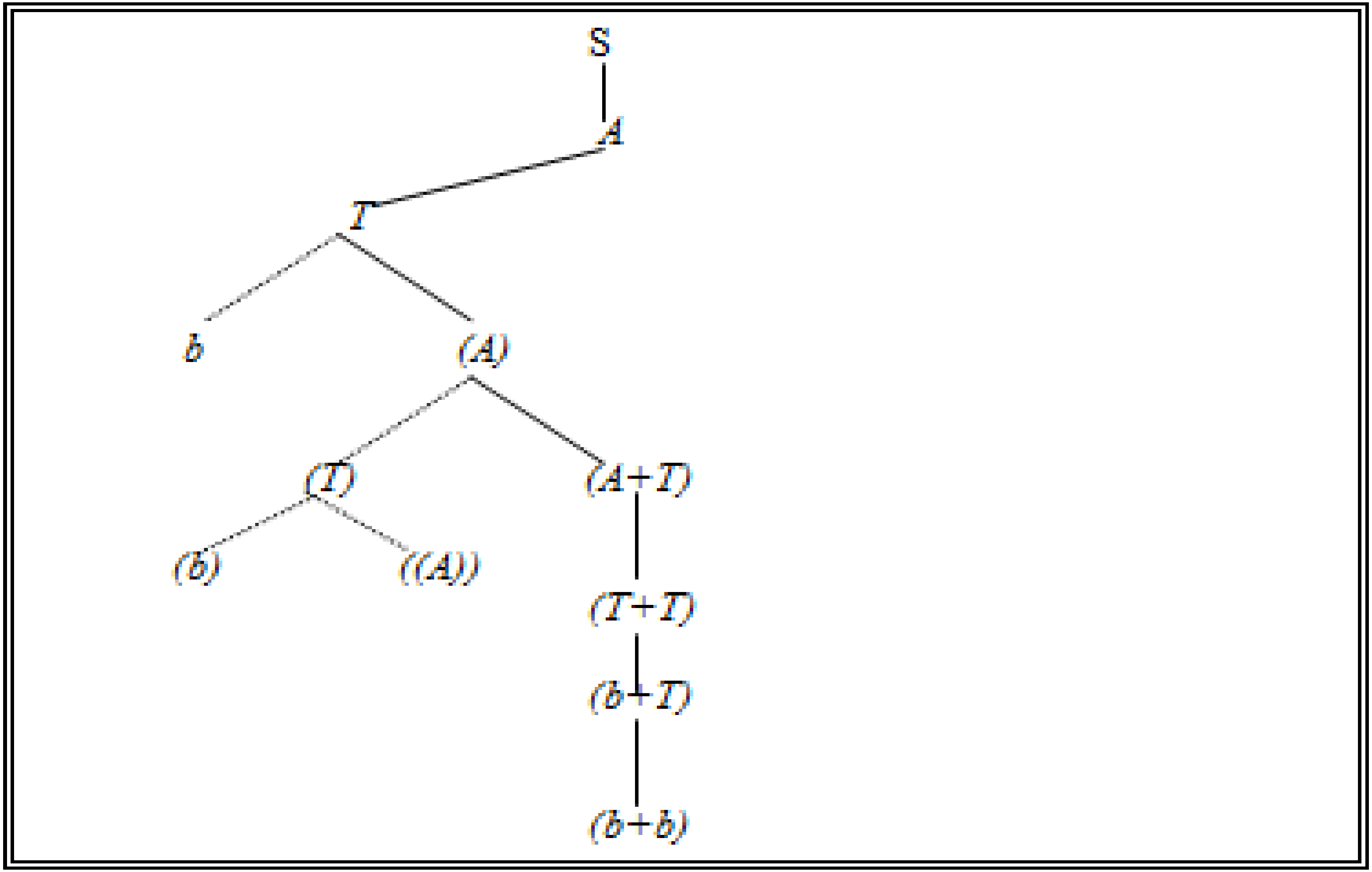
P: 1. $S \rightarrow A$

2. $A \rightarrow T$

3. $A \rightarrow A+T$

4. $T \rightarrow b$

5. $T \rightarrow (A)$



path and derivation generated by depth-first search for (*b*+*b*)

Example 3:

1- $S \rightarrow A$

2- $A \rightarrow T$

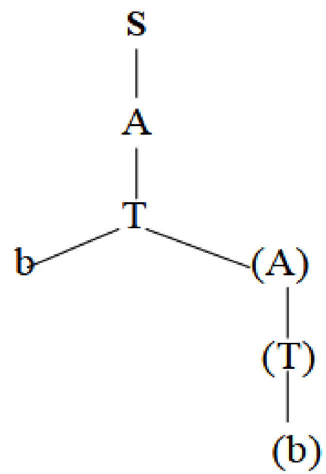
3- $A \rightarrow A+T$

4- $T \rightarrow b$

5- $T \rightarrow (A)$

string : (b)

i) parse tree using the Depth- First Top-Down Parsing Algorithm.



---

ii) **Stack**

[S,0]	[S, 1]	
	[A, 2]	
	[T, 4]	[T, 5]
	b	[(A), 2]
		[(T), 4]
		(b)

iii) $q = S$, $i=0$

$1-\quad S \rightarrow A \quad j=1$

$q = A$

$2-\quad A \rightarrow T \quad j=2$

$q = T$

$4-\quad T \rightarrow b \quad j=4$

$q = T$

$5-\quad T \rightarrow (A) \quad j=5$

$q = (A)$

$2-\quad A \rightarrow T \quad j=2$

$q = (T)$

$4-\quad T \rightarrow b$

$1-\quad S \rightarrow A$

$2-\quad A \rightarrow T$

$3-\quad A \rightarrow A+T$

$4-\quad T \rightarrow b$

$5-\quad T \rightarrow (A)$

Example 4:

1 $S \rightarrow AA$
2 $A \rightarrow aa$
3 $A \rightarrow bb$

String : bbbb

i) Depth- First Top-Down Parsing Algorithm.

[q, i] = [S, 0]

q= S , i = 0

$S \rightarrow AA$ i =1

q= AA

$A \rightarrow aa$ i =2

q= *aa* A

×

$A \rightarrow bb$ i =3

q= *bb* A

ii) Trace the stack

[S,0]   [S,1]

[AA, 2]   [AA, 3]

aaA   [bbA, 2]   [bbA, 3]

bbaa   bbbb

iii) tree