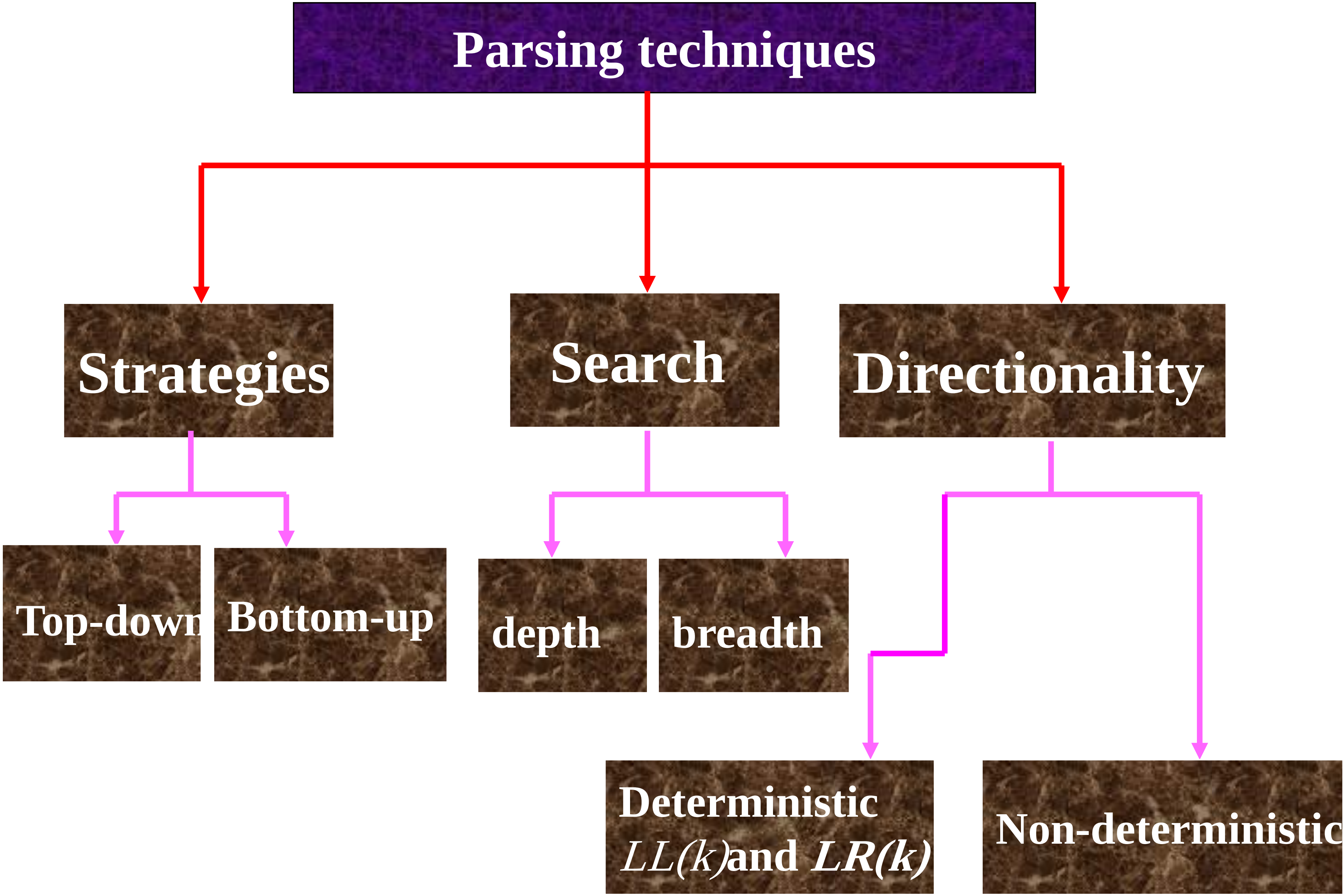
Compiler Constructions

Chapter 4(Parsing)

Part 2

Bottom-up Parsing


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

Bottom-up Parsing

Constructs parse tree for an input string beginning at the leaves (the bottom) and working towards the root (the top).

Bottom-up Parsing

Example:

A reduction of the string $(b) + b$ to S is given using the rules of the grammar AE.

Reduction	Rule
$(b) + b$	
$(T) + b$	$T \to b$
$(A) + b$	$A \to T$
$T + b$	$T \to (A)$
$A + b$	$A \to T$
$A + T$	$T \to b$
A	$A \to A + T$
S	$S \to A$

$V = \{S, A, T\}$

$\Sigma = \{b, +, (,)\}$

P: 1. $S \to A$

2. $A \to T$

3. $A \to A + T$

4. $T \to b$

5. $T \to (A)$

Bottom-up Parsing

Reversing the order of the sentential forms that constitute the reduction of w to S produces the rightmost derivation

$$
\begin{aligned}
S &\Rightarrow A \\
&\Rightarrow A + T \\
&\Rightarrow A + b \\
&\Rightarrow T + b \\
&\Rightarrow (A) + b \\
&\Rightarrow (T) + b \\
&\Rightarrow (b) + b.
\end{aligned}
$$

For this reason, bottom-up parsers are often said to construct rightmost derivations in reverse.

Bottom-up Parsing

Reduction	Rule
$(b) + b$	
$(T) + b$	$T \rightarrow b$
$(A) + b$	$A \rightarrow T$
$T + b$	$T \rightarrow (A)$
$A + b$	$A \rightarrow T$
$A + T$	$T \rightarrow b$
A	$A \rightarrow A + T$
S	$S \rightarrow A$

$$\begin{aligned} S &\Rightarrow A \\ &\Rightarrow A + T \\ &\Rightarrow A + b \\ &\Rightarrow T + b \\ &\Rightarrow (A) + b \\ &\Rightarrow (T) + b \\ &\Rightarrow (b) + b. \end{aligned}$$

Bottom-up Parsing

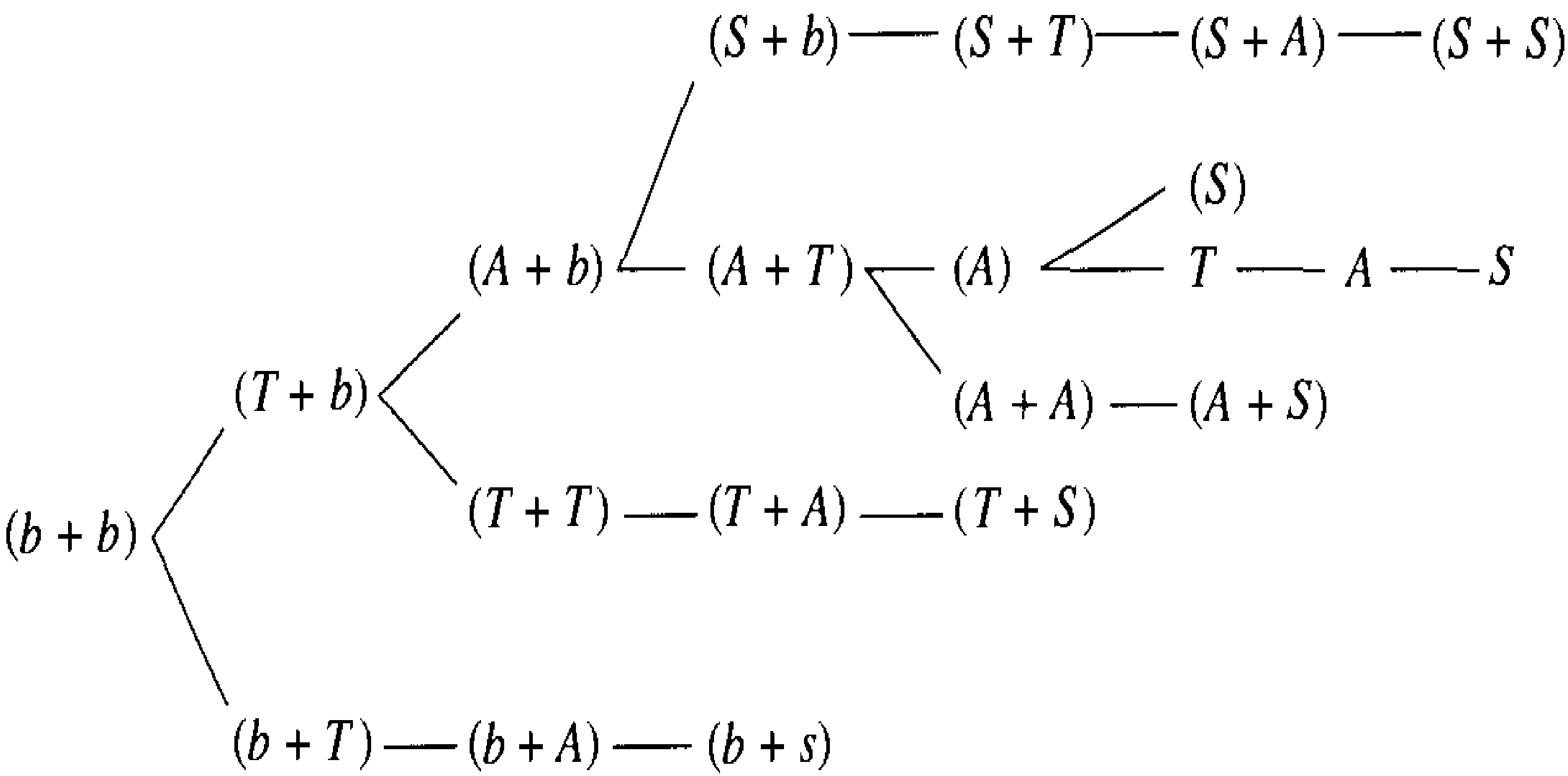
Breadth-First Bottom-Up Parser

Algorithm 4.5.1
Breadth-First Bottom-Up Parser

input: context-free grammar $G = (V, \Sigma, P, S)$
 string $p \in \Sigma^*$
 queue $\mathbf{Q}$

1. initialize T with root p
 $INSERT(p, \mathbf{Q})$
2. **repeat**
 $q := REMOVE(\mathbf{Q})$
 2.1. **for** each rule $A \rightarrow w$ in P **do**
  2.1.1. **for** each decomposition uwv of q with $v \in \Sigma^*$ **do**
   2.1.1.1. $INSERT(uAv, \mathbf{Q})$
   2.1.1.2. Add node uAv to T. Set a pointer from uAv to q.
  end for
 end for
 until $q = S$ **or** $EMPTY(\mathbf{Q})$
3. **if** $q = S$ **then** accept **else** reject

Bottom-up Parsing



Breadth-first bottom-up parse of $(b + b)$.

Bottom-up Parsing

A Depth-First Bottom-Up Parser

Algorithm 4.6.1
Depth-First Bottom-Up Parsing Algorithm

input: context-free grammar $G = (V, \Sigma, P, S)$ with nonrecursive start symbol
string $p \in \Sigma^*$
stack $\mathbf{S}$

1. $PUSH([\lambda, 0, p], \mathbf{S})$
2. **repeat**
   - 2.1. $[u, i, v] := POP(\mathbf{S})$
   - 2.2. dead-end $:= false$
   - 2.3. **repeat**
     - Find the first $j > i$ with rule number j that satisfies
     - i) $A \rightarrow w$ with $u = qw$ and $A \neq S$ or
     - ii) $S \rightarrow w$ with $u = w$ and $v = \lambda$
     - 2.3.1. **if** there is such a j **then**
       - 2.3.1.1. $PUSH([u, j, v], \mathbf{S})$
       - 2.3.1.2. $u := qA$
       - 2.3.1.3. $i := 0$
     - **end if**
     - 2.3.2. **if** there is no such j **and** $v \neq \lambda$ **then**
       - 2.3.2.1. $shift(u, v)$
       - 2.3.2.2. $i := 0$
     - **end if**
     - 2.3.3. **if** there is no such j **and** $v = \lambda$ **then** dead-end $:= true$
   - **until** $(u = S)$ **or** dead-end
   - **until** $(u = S)$ **or** $EMPTY(\mathbf{S})$
3. **if** $EMPTY(\mathbf{S})$ **then** reject **else** accept

Bottom-up Parsing

A Depth-First Bottom-Up Parser

Example

Using Algorithm 4.6.1 and the grammar AE, we can construct a derivation of the string $(b + b)$. The stack is given in the second column, with the stack top being the top triple. The decomposition of the string and current rule numbers are in the columns labeled u, v, and i. The operation that produced the new configuration is given on the left. At the beginning of the computation the stack contains the single element $[\lambda, 0, (b + b)]$. The configuration consisting of an empty stack and $u = \lambda$, $i = 0$, and $v = (b + b)$ is obtained by popping the stack.

Bottom-up Parsing

A Depth-First Bottom-Up Parser

Operation	Stack	u	i	v
	$[\lambda, 0, (b+b)]$			
pop		λ	0	$(b+b)$
shift		$($	0	$b+b)$
shift		$(b$	0	$+b)$
reduction	$[(b, 4, +b)]$	$(T$	0	$+b)$
reduction	$[(T, 2, +b)]$ $[(b, 4, +b)]$	$(A$	0	$+b)$
shift	$[(T, 2, +b)]$ $[(b, 4, +b)]$	$(A+$	0	$b)$
shift	$[(T, 2, +b)]$ $[(b, 4, +b)]$	$(A+b$	0	$)$

Continued

1. $S \to A$
2. $A \to T$
3. $A \to A + T$
4. $T \to b$
5. $T \to (A)$

Bottom-up Parsing

A Depth-First Bottom-Up Parser

Operation	Stack	u	i	v
reduction	$[(A+b,4,))]$ $[(T,2,+b)]$ $[(b,4,+b)]$	$(A+T$	0	$)$
reduction	$[(A+T,2,))]$ $[(A+b,4,))]$ $[(T,2,+b)]$ $[(b,4,+b)]$	$(A+A$	0	$)$
shift	$[(A+T,2,))]$ $[(A+b,4,))]$ $[(T,2,+b)]$ $[(b,4,+b)]$	$(A+A)$	0	λ
pop	$[(A+b,4,))]$ $[(T,2,+b)]$ $[(b,4,+b)]$	$(A+T$	2	$)$

1. $S \rightarrow A$
2. $A \rightarrow T$
3. $A \rightarrow A + T$
4. $T \rightarrow b$
5. $T \rightarrow (A)$

Bottom-up Parsing

A Depth-First Bottom-Up Parser

1. $S \rightarrow A$
2. $A \rightarrow T$
3. $A \rightarrow A + T$
4. $T \rightarrow b$
5. $T \rightarrow (A)$

reduction	$[(A + T, 3,)]$ $[(A + b, 4,)]$ $[(T, 2, + b)]$ $[(b, 4, + b)]$	$(A$	0	$)$
shift	$[(A + T, 3,)]$ $[(A + b, 4,)]$ $[(T, 2, + b)]$ $[(b, 4, + b)]$	(A)	0	λ
reduction	$[(A), 5, \lambda]$ $[(A + T, 3,)]$ $[(A + b, 4,)]$ $[(T, 2, + b)]$ $[(b, 4, + b)]$	T	0	λ

Continued

Bottom-up Parsing

A Depth-First Bottom-Up Parser

1. $S \to A$
2. $A \to T$
3. $A \to A + T$
4. $T \to b$
5. $T \to (A)$

Operation	Stack	u	i	v
	$[T, 2, \lambda]$			
	$[(A), 5, \lambda]$			
	$[(A + T, 3,)]$			
	$[(A + b, 4,)]$			
	$[(T, 2, + b)]$			
reduction	$[(b, 4, + b)]$	A	0	λ
	$[A, 1, \lambda]$			
	$[T, 2, \lambda]$			
	$[(A), 5, \lambda]$			
	$[(A + T, 3,)]$			
	$[(A + b, 4,)]$			
	$[(T, 2, + b)]$			
reduction	$[(b, 4, + b)]$	S	0	λ