Parsing

Part V

Bottom-Up Parsing

- A **bottom-up parser** creates the parse tree of the given input starting from leaves towards the root

- A bottom-up parser tries to find the **right-most derivation** of the given input in the reverse order.

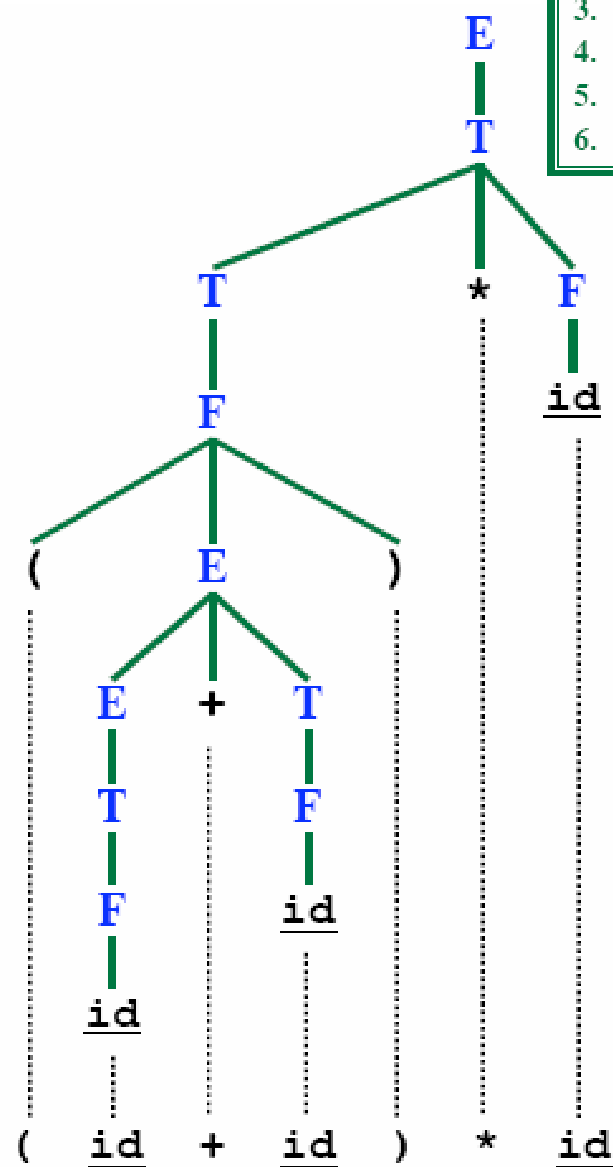
$S \Rightarrow \ldots \Rightarrow \omega$ (the right-most derivation of ω)

$\leftarrow$ (the bottom-up parser finds the right-most derivation in the reverse order)

Rightmost Derivation

1. E → E + T
2. E → T
3. T → T * F
4. T → F
5. F → (E)
6. F → id

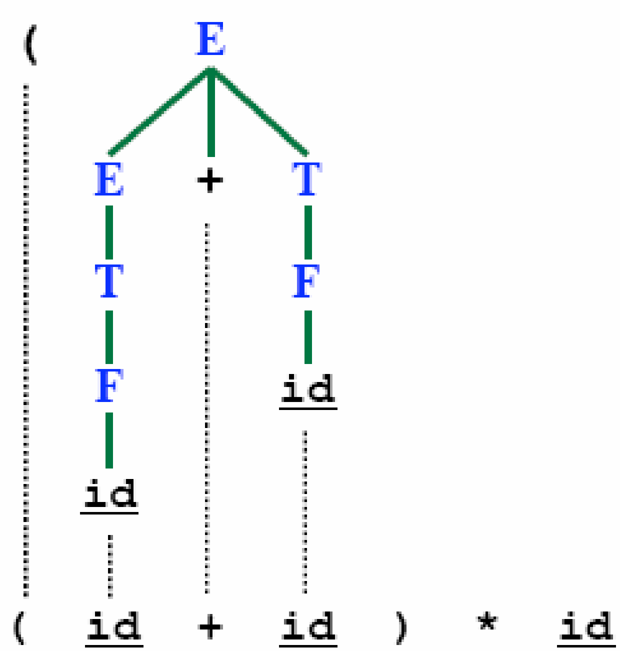
Rules Used:	***Right-Sentential Forms:***
	E
E → T	T
T → T * F	T * F
F → id	T * id
T → F	F * id
F → (E)	(E) * id
E → E + T	(E + T) * id
T → F	(E + F) * id
F → id	(E + id) * id
E → T	(T + id) * id
T → F	(F + id) * id
F → id	(id + id) * id



Rightmost Derivation In reverse

1. E → E + T
2. E → T
3. T → T * F
4. T → F
5. F → (E)
6. F → id

Rules Used:	*Right-Sentential Forms:*
	(id + id) * id
F → id	(F + id) * id
T → F	(T + id) * id
E → T	(E + id) * id
F → id	(E + F) * id
T → F	(E + T) * id
E → E + T	(E) * id



Rightmost Derivation In reverse

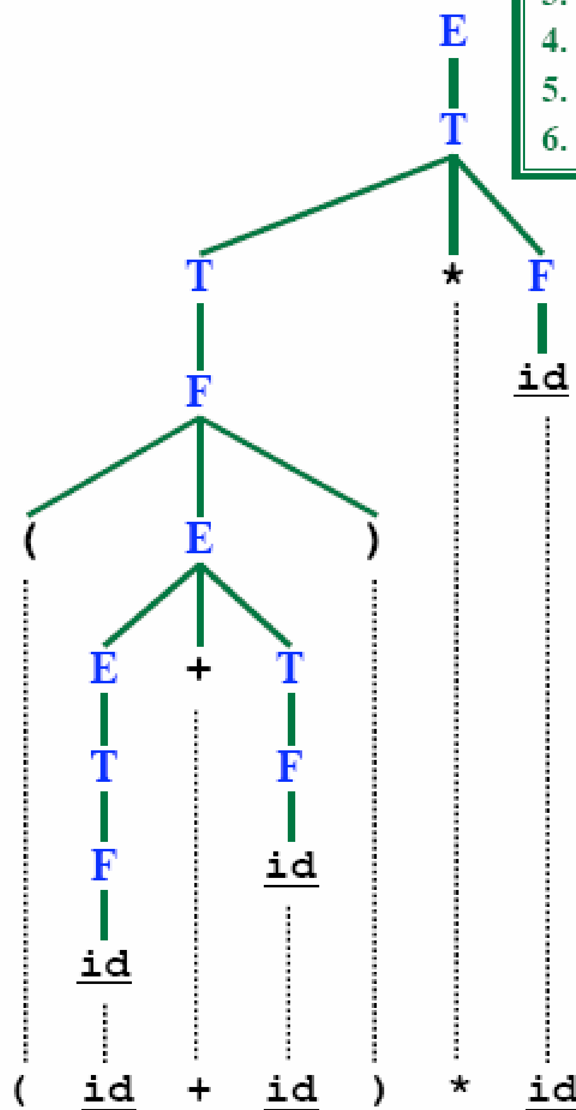
1. E → E + T
2. E → T
3. T → T * F
4. T → F
5. F → (E)
6. F → id

Rules Used:

- F → id
- T → F
- E → T
- F → id
- T → F
- E → E + T
- F → (E)
- T → F
- F → id
- T → T * F
- E → T

Right-Sentential Forms:

- (id + id) * id
- (F + id) * id
- (T + id) * id
- (E + id) * id
- (E + F) * id
- (E + T) * id
- (E) * id
- F * id
- T * id
- T * F
- T
- E



LR parsing corresponds to rightmost derivation in reverse

Reduction

- A reduction step replaces a specific substring (matching the body of a production)

(id + id) * id

(F + id) * id

(T + id) * id

(E + id) * id

(E + F) * id

(E + T) * id

(E) * id

F * id

T * id

T * F

T

E

1. E → E + T
2. E → T
3. T → T * F
4. T → F
5. F → (E)
6. F → id

- Reduction is the opposite of derivation
- Bottom up parsing is a process of reducing a string ω to the start symbol S of the grammar

Handle

- Informally, a **handle** is a substring (in the parsing string) that matches the right side of a production rule.
  - But not every substring matches the right side of a production rule is handle

- A **handle** of a right sentential form γ ($\equiv \alpha\beta\omega$) is

  a production rule $A \rightarrow \beta$ and a position of γ

  where the string β may be found and replaced by A to produce

  the previous right-sentential form in a rightmost derivation of γ.

$$S \overset{*}{\underset{rm}{\Rightarrow}} \alpha A\omega \underset{rm}{\Rightarrow} \alpha\beta\omega$$

Handle Pruning

- A right-most derivation in reverse can be obtained by **handle-pruning**.

n-th right-sentential form

$$S=\gamma_0 \overset{rm}{\Rightarrow} \gamma_1 \overset{rm}{\Rightarrow} \gamma_2 \overset{rm}{\Rightarrow} \ldots \overset{rm}{\Rightarrow} \gamma_{n-1} \Rightarrow \gamma_n = \omega$$

input string

- Start from γ_n, find a handle $A_n \rightarrow \beta_n$ in γ_n, and replace β_n in by A_n to get γ_{n-1}.
- Then find a handle $A_{n-1} \rightarrow \beta_{n-1}$ in γ_{n-1}, and replace β_{n-1} in by A_{n-1} to get γ_{n-2}.
- Repeat this, until we reach S.

Shift-Reduce Parsing

- Bottom-up parsing is also known as **shift-reduce parsing** because its two main actions are shift and reduce.
- data structures: input-string and stack
- Operations
  - At each shift action, the current symbol in the input string is pushed to a stack.
  - At each reduction step, the symbols at the top of the stack (this symbol sequence is the right side of a production) will replaced by the non-terminal at the left side of that production.
  - Accept: Announce successful completion of parsing
  - Error: Discover a syntax error and call error recovery

Shift Reduce Parsing

$S \rightarrow a\ T\ R\ e$
$T \rightarrow T\ b\ c \mid b$
$R \rightarrow d$

Remaining input: abbcde

Rightmost derivation:

$S \rightarrow a\ T\ R\ e$
$\rightarrow a\ T\ \mathbf{d}\ e$
$\rightarrow a\ \mathbf{T\ b\ c}\ d\ e$
$\rightarrow a\ b\ b\ c\ d\ e$

Shift Reduce Parsing

S → a T R e
T → T b c | b
R → d

Remaining input: bcde

→ Shift a, Shift b

a b

Rightmost derivation:

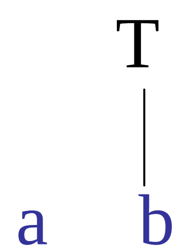
S → a T R e
→ a T **d** e
→ a **T b c** d e
→ a b b c d e

Shift Reduce Parsing

S → a T R e
T → T b c | b
R → d

Remaining input: bcde

- ➔ Shift a, Shift b
- ➔ Reduce T → b



Rightmost derivation:

S → a T R e
→ a T **d** e
→ <u>a **T**</u> **b c** d e
→ **a b** b c d e

Shift Reduce Parsing

S → a T R e
T → T b c | b
R → d

Remaining input: de

- → Shift a, Shift b
- → Reduce T → b
- → Shift b, Shift c

T

a b b c

Rightmost derivation:

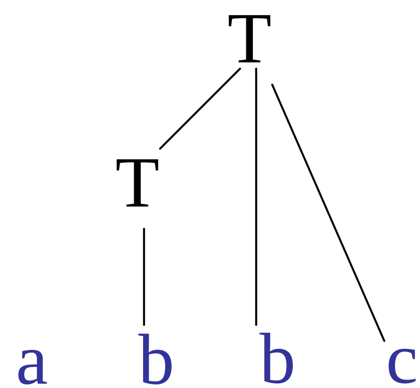
S → a T R e
→ a T **d** e
→ a **T b c** d e
→ **a b** b c d e

Shift Reduce Parsing

S → a T R e
T → T b c | b
R → d

Remaining input: de

➔ Shift a, Shift b
➔ Reduce T → b
➔ Shift b, Shift c
➔ Reduce T → T b c



Rightmost derivation:

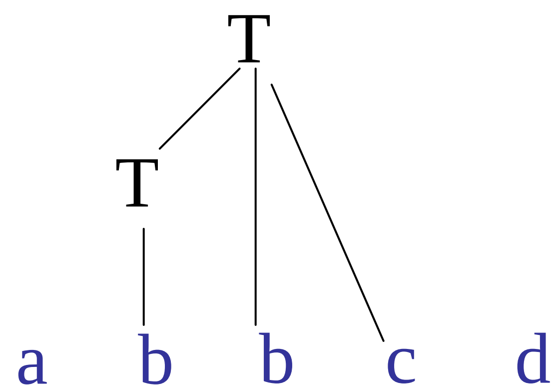
S ➔ a T R e
➔ <u>a T</u> **d** e
➔ a **T b c** d e
➔ **a b** b c d e

Shift Reduce Parsing

S → a T R e
T → T b c | b
R → d

Remaining input: e

- → Shift a, Shift b
- → Reduce T → b
- → Shift b, Shift c
- → Reduce T → T b c
- → Shift d



Rightmost derivation:

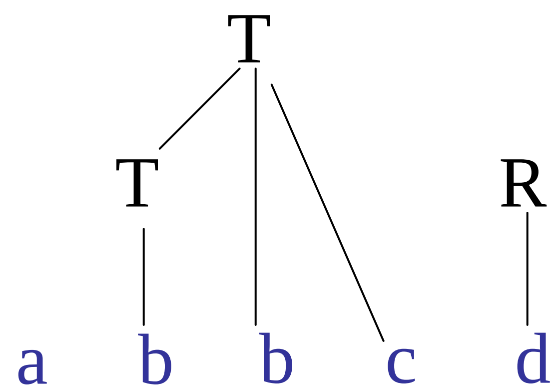
S → a T R e
→ a T **d** e
→ a **T b c** d e
→ **a b** b c d e

Shift Reduce Parsing

S → a T R e
T → T b c | b
R → d

Remaining input: e

➔ Shift a, Shift b
➔ Reduce T → b
➔ Shift b, Shift c
➔ Reduce T → T b c
➔ Shift d
➔ Reduce R → d



Rightmost derivation:

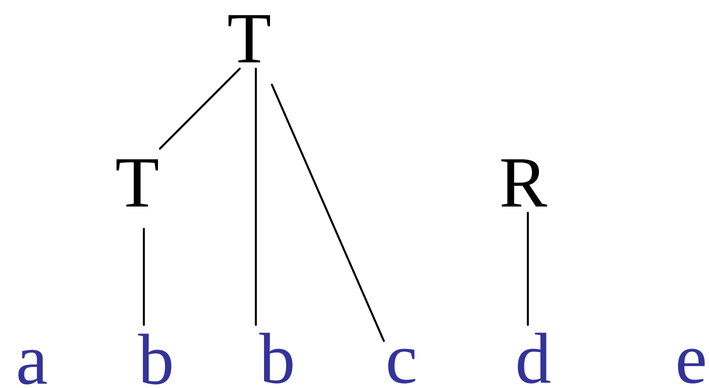
S → <u>a T R</u> e
→ a T **d** e
→ a **T b c** d e
→ **a b** b c d e

Shift Reduce Parsing

S → a T R e
T → T b c | b
R → d

Remaining input:

- → Shift a, Shift b
- → Reduce T → b
- → Shift b, Shift c
- → Reduce T → T b c
- → Shift d
- → Reduce R → d
- → Shift e



Rightmost derivation:

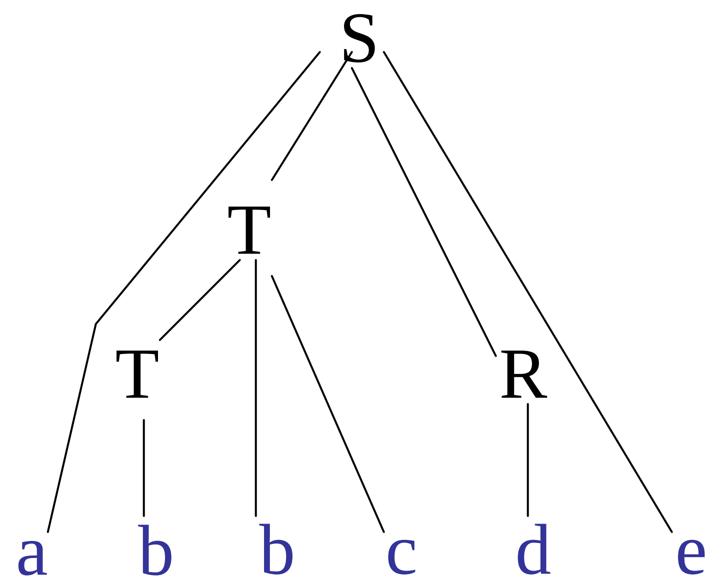
S → a T R e
→ a T **d** e
→ a **T b c** d e
→ **a b** b c d e

Shift Reduce Parsing

S → a T R e
T → T b c | b
R → d

- → Shift a, Shift b
- → Reduce T → b
- → Shift b, Shift c
- → Reduce T → T b c
- → Shift d
- → Reduce R → d
- → Shift e
- → Reduce S → a T R e

Remaining input:



Rightmost derivation:

S → a T R e
→ a T **d** e
→ a **T b c** d e
→ **a b** b c d e

Example Shift-Reduce Parsing

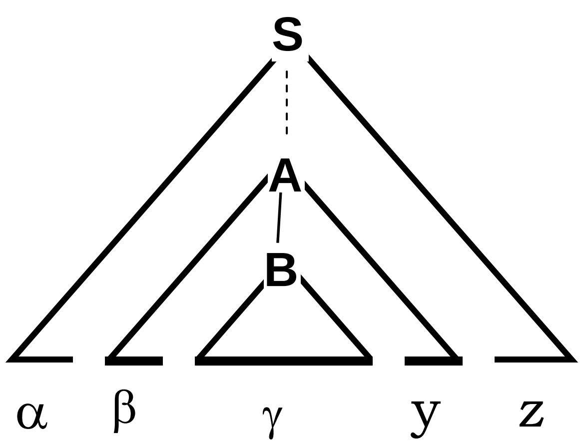
Consider the grammar:

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow \underline{\texttt{id}}$

Stack	Input	Action
\$	$id_1 + id_2\$$	shift
$\$id_1$	$+ id_2\$$	reduce 6
\$F	$+ id_2\$$	reduce 4
\$T	$+ id_2\$$	reduce 2
\$E	$+ id_2\$$	shift
\$E +	$id_2\$$	shift
$\$E + id_2$		reduce 6
\$E + F		reduce 4
\$E + T		reduce 1
\$E		accept

Shift-Reduce Parsing

- Handle will always appear on Top of stack, never inside
- Possible forms of two successive steps in any rightmost derivation
- CASE 1:

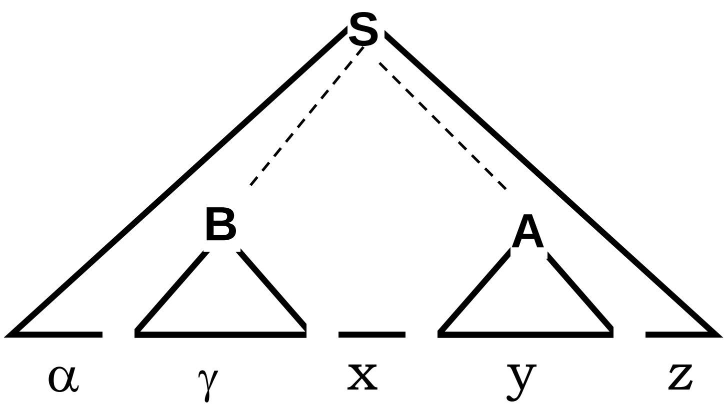


$$S \overset{*}{\underset{rm}{\Rightarrow}} \alpha Az \underset{rm}{\Rightarrow} \alpha\beta Byz \underset{rm}{\Rightarrow} \alpha\beta\gamma yz$$

STACK	**INPUT**
$\$\alpha\beta\gamma$	yz$
After Reducing the handle	
$\$\alpha\beta B$	yz$
Shifting from Input	
$\$\alpha\beta By$	z$
Reduce the handle	
$\$\alpha A$	z$

Shift-Reduce Parsing

- Case 2:



$$S \underset{rm}{\overset{*}{\Rightarrow}} \alpha B x A z \underset{rm}{\Rightarrow} \alpha B x y z \underset{rm}{\Rightarrow} \alpha\gamma x y z$$

STACK	**INPUT**
$\$\alpha\gamma$	xyz$
After Reducing the handle	
$\$\alpha B$	xyz$
Shifting from Input	
$\$\alpha Bxy$	z$
Reducing the handle	
$\$\alpha BxA$	z$

Conflicts During Shift-Reduce Parsing

- There are context-free grammars for which shift-reduce parsers cannot be used.
- Stack contents and the next input symbol may not decide action:
  - **shift/reduce conflict**: Whether make a shift operation or a reduction.
  - **reduce/reduce conflict**: The parser cannot decide which of several reductions to make.
- If a shift-reduce parser cannot be used for a grammar, that grammar is called as non-LR(k) grammar.

left to right scanning

right-mostk derivation

lookhead

- An ambiguous grammar can never be a LR grammar.

Shift-Reduce Conflict in Ambiguous Grammar

stmt → **if** *expr* **then** *stmt*
| **if** *expr* **then** *stmt* **else** *stmt*
| **other**

STACK	INPUT
….**if** *expr* **then** *stmt*	**else**….$

- We cant decide whether to shift or reduce?

- But we can adapt to parse certain ambiguous grammar to using shift-reducing parsers
  - We resolve in favor of SHIFT then we have a solution

Reduce Reduce conflict Example : Page 239-240