Practice Problems with Solution

Shift Reduce Parsing

Problem-01:

Consider the following grammar-

$$E \rightarrow E - E$$

$$E \rightarrow E \times E$$

$$E \rightarrow id$$

Parse the input string id – id x id using a shift-reduce parser.

Solution-

The priority order is: id > x > –

Stack	Input Buffer	Parsing Action
$	id – id x id $	Shift
$ id	– id x id $	Reduce E → id
$ E	– id x id $	Shift
$ E –	id x id $	Shift
$ E – id	x id $	Reduce E → id
$ E – E	x id $	Shift
$ E – E x	id $	Shift

$ E – E x id	$	Reduce E → id
$ E – E x E	$	Reduce E → E x E
$ E – E	$	Reduce E → E – E
$ E	$	Accept

Problem-02:

Consider the following grammar-

S → (L) | a

L → L , S | S

Parse the input string (a , (a , a)) using a shift-reduce parser.

Solution-

Stack	Input Buffer	Parsing Action
$	(a , (a , a)) $	Shift
$ (	a , (a , a)) $	Shift
$ (a	, (a , a)) $	Reduce S → a
$ (S	, (a , a)) $	Reduce L → S
$ (L	, (a , a)) $	Shift
$ (L ,	(a , a)) $	Shift

$ (L , (	a , a)) $	Shift
$ (L , (a	, a)) $	Reduce S → a
$ (L , (S	, a)) $	Reduce L → S
$ (L , (L	, a)) $	Shift
$ (L , (L ,	a)) $	Shift
$ (L , (L , a	)) $	Reduce S → a
$ (L , (L , S)	)) $	Reduce L → L , S
$ (L , (L	)) $	Shift
$ (L , (L)	) $	Reduce S → (L)
$ (L , S	) $	Reduce L → L , S
$ (L	) $	Shift
$ (L)	$	Reduce S → (L)
$ S	$	Accept

Problem-03:

Consider the following grammar-

$$S \rightarrow T\ L$$

$$T \rightarrow int \mid float$$

$$L \rightarrow L\ ,\ id \mid id$$

Parse the input string int id , id ; using a shift-reduce parser.

Solution-

Stack	Input Buffer	Parsing Action
$	int id , id ; $	Shift
$ int	id , id ; $	Reduce T → int
$ T	id , id ; $	Shift
$ T id	, id ; $	Reduce L → id
$ T L	, id ; $	Shift
$ T L ,	id ; $	Shift
$ T L , id	; $	Reduce L → L , id
$ T L	; $	Shift
$ T L ;	$	Reduce S → T L
$ S	$	Accept

Problem-04:

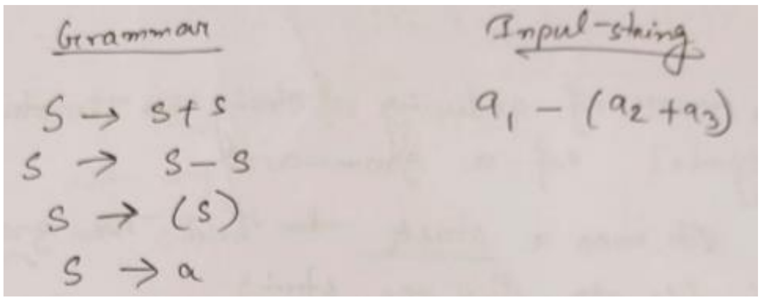
Considering the string "10201", design a shift-reduce parser for the following grammar-

S → 0S0 | 1S1 | 2

Solution-

Stack	Input Buffer	Parsing Action
$	1 0 2 0 1 $	Shift
$ 1	0 2 0 1 $	Shift
$ 1 0	2 0 1 $	Shift
$ 1 0 2	0 1 $	Reduce S → 2
$ 1 0 S	0 1 $	Shift
$ 1 0 S 0	1 $	Reduce S → 0 S 0
$ 1 S	1 $	Shift
$ 1 S 1	$	Reduce S → 1 S 1
$ S	$	Accept

Problem 5:



Grammar

$S \to S+S$

$S \to S-S$

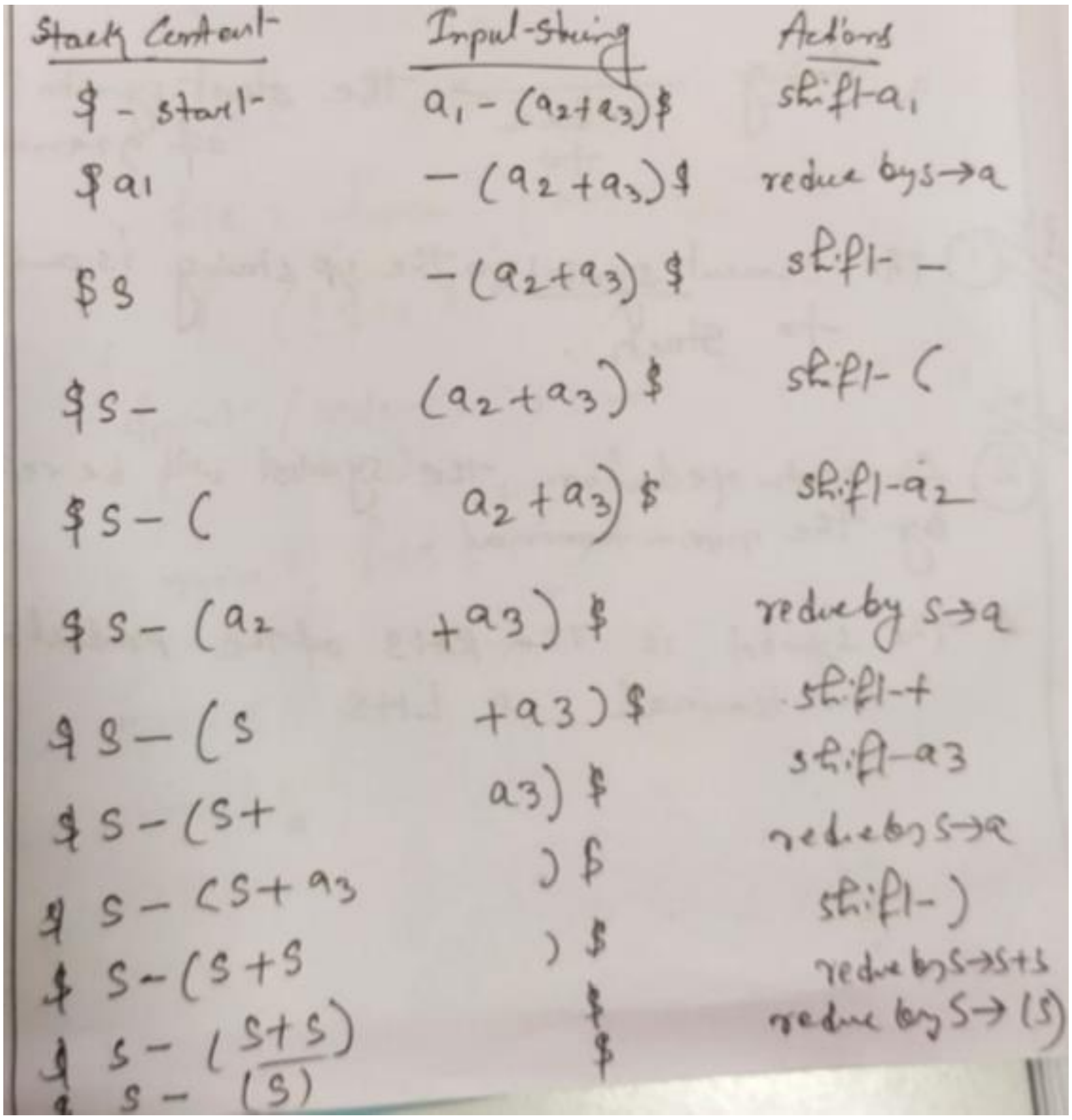
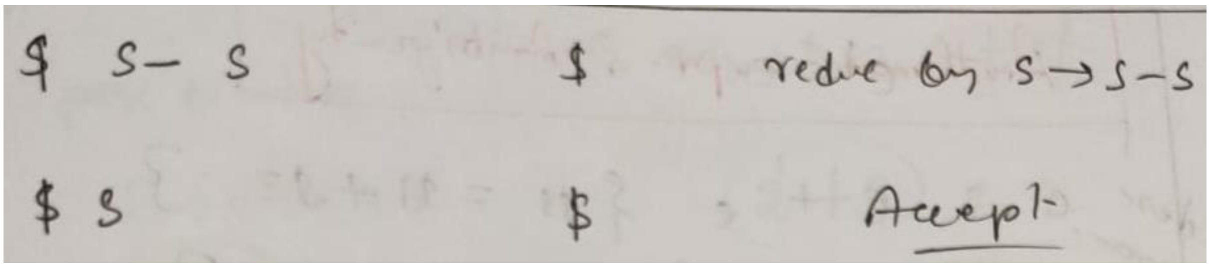
$S \to (S)$

$S \to a$

Input-string

$a_1 - (a_2 + a_3)$

Solution

Stack Content	Input String	Actions
$\$ - \text{start}$	$a_1 - (a_2+a_3)\$$	shift a_1
$\$a_1$	$-(a_2+a_3)\$$	reduce by $S \to a$
$\$S$	$-(a_2+a_3)\$$	shift $-$
$\$S-$	$(a_2+a_3)\$$	shift (
$\$S-($	$a_2+a_3)\$$	shift a_2
$\$S-(a_2$	$+a_3)\$$	reduce by $S \to a$
$\$S-(S$	$+a3)\$$	shift +
$\$S-(S+$	$a3)\$$	shift a3
$\$S-(S+a_3$	$)\$$	reduce by $S \to a$
$\$S-(S+S$	$)\$$	shift)
$\$S-(S+S)$	$\$$	reduce by $S \to S+S$
$\$S-(S)$	$\$$	reduce by $S \to (S)$
$\$S-S$	$\$$	reduce by $S \to S-S$
$\$S$	$\$$	Accept

LEFT RECURSION ELIMINATION-

Problem-01:

Consider the following grammar and eliminate left recursion-

A → ABd / Aa / a

B → Be / b

Solution-

The grammar after eliminating left recursion is-

A → aA’

A’ → BdA’ / aA’ / ∈

B → bB’

B’ → eB’ / ∈

Problem-02:

Consider the following grammar and eliminate left recursion-

E → E + E / E x E / a

Solution-

The grammar after eliminating left recursion is-

E → aA

A → +EA / xEA / ∈

Problem-03:

Consider the following grammar and eliminate left recursion-

$$E \rightarrow E + T \,/\, T$$

$$T \rightarrow T \times F \,/\, F$$

$$F \rightarrow id$$

Solution-

The grammar after eliminating left recursion is-

$E \rightarrow TE'$

$E' \rightarrow +TE' \,/\, \in$

$T \rightarrow FT'$

$T' \rightarrow xFT' \,/\, \in$

$F \rightarrow id$

Problem-04:

Consider the following grammar and eliminate left recursion-

$$S \rightarrow (L) \,/\, a$$

$$L \rightarrow L\, ,\, S \,/\, S$$

Solution-

The grammar after eliminating left recursion is-

$S \rightarrow (L) \,/\, a$

$L \rightarrow SL'$

$L' \rightarrow ,SL' \,/\, \in$

Problem-05:

Consider the following grammar and eliminate left recursion-

$$S \rightarrow S0S1S \,/\, 01$$

Solution-

The grammar after eliminating left recursion is-

$S \rightarrow 01A$

$A \rightarrow 0S1SA \,/\, \in$

Problem-06:

Consider the following grammar and eliminate left recursion-

$$S \rightarrow A$$

$$A \rightarrow Ad \,/\, Ae \,/\, aB \,/\, ac$$

$$B \rightarrow bBc \,/\, f$$

Solution-

The grammar after eliminating left recursion is-

$S \rightarrow A$

$A \rightarrow aBA' \,/\, acA'$

$A' \rightarrow dA' \,/\, eA' \,/\, \in$

$B \rightarrow bBc \,/\, f$

Problem-07:

Consider the following grammar and eliminate left recursion-

$$A \rightarrow AA\alpha \,/\, \beta$$

Solution-

The grammar after eliminating left recursion is-

A → βA’

A’ → AαA’ / ∈

Problem-08:

Consider the following grammar and eliminate left recursion-

A → Ba / Aa / c

B → Bb / Ab / d

Solution-

This is a case of indirect left recursion.

Step-01:

First let us eliminate left recursion from A → Ba / Aa / c

Eliminating left recursion from here, we get-

A → BaA’ / cA’

A’ → aA’ / ∈

Now, given grammar becomes-

A → BaA’ / cA’

A’ → aA’ / ∈

B → Bb / Ab / d

Step-02:

Substituting the productions of A in B → Ab, we get the following grammar-

A → BaA’ / cA’

A’ → aA’ / ∈

B → Bb / BaA’b / cA’b / d

Step-03:

Now, eliminating left recursion from the productions of B, we get the following grammar-

A → BaA’ / cA’

A’ → aA’ / ∈

B → cA’bB’ / dB’

B’ → bB’ / aA’bB’ / ∈

This is the final grammar after eliminating left recursion.

Problem-09:

Consider the following grammar and eliminate left recursion-

X → XSb / Sa / b

S → Sb / Xa / a

Solution-

This is a case of indirect left recursion.

Step-01:

First let us eliminate left recursion from X → XSb / Sa / b

Eliminating left recursion from here, we get-

X → SaX’ / bX’

X’ → SbX’ / ∈

Now, given grammar becomes-

X → SaX’ / bX’

X’ → SbX’ / ∈

S → Sb / Xa / a

<u>Step-02:</u>

Substituting the productions of X in S → Xa, we get the following grammar-

X → SaX’ / bX’

X’ → SbX’ / ∈

S → Sb / SaX’a / bX’a / a

<u>Step-03:</u>

Now, eliminating left recursion from the productions of S, we get the following grammar-

X → SaX’ / bX’

X’ → SbX’ / ∈

S → bX’aS’ / aS’

S’ → bS’ / aX’aS’ / ∈

This is the final grammar after eliminating left recursion.

Problem-10:

Consider the following grammar and eliminate left recursion-

S → Aa / b

A → Ac / Sd / ∈

Solution-

This is a case of indirect left recursion.

Step-01:

First let us eliminate left recursion from S → Aa / b

This is already free from left recursion.

Step-02:

Substituting the productions of S in A → Sd, we get the following grammar-

S → Aa / b

A → Ac / Aad / bd / ∈

Step-03:

Now, eliminating left recursion from the productions of A, we get the following grammar-

S → Aa / b

A → bdA' / A'

A' → cA' / adA' / ∈

This is the final grammar after eliminating left recursion.

Leftmost Derivation/ Rightmost Derivation

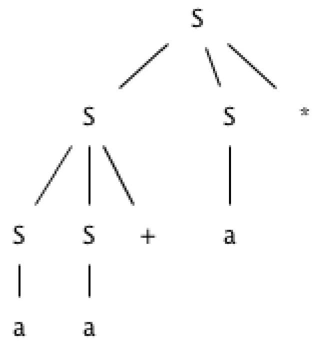
Consider the context-free grammar:

```
S -> S S + | S S * | a
```
and the string aa + a*.

1. Give a leftmost derivation for the string.
2. Give a rightmost derivation for the string.
3. Give a parse tree for the string.
4. ! Is the grammar ambiguous or unambiguous? Justify your answer.
5. ! Describe the language generated by this grammar.

Answer

1. S =lm=> SS* => SS+S* => aS+S* => aa+S* => aa+a*
2. S =rm=> SS* => Sa* => SS+a* => Sa+a* => aa+a*



3. Unambiguous
4. The set of all postfix expressions consist of addition and multiplication

4.2.2

Repeat Exercise 4 . 2 . 1 for each of the following grammars and strings:

1. S -> 0 S 1 | 0 1 with string 00011l.
2. S -> + S S | * S S | a with string + * aaa.
3. ! S -> S (S) S | ε with string (()())
4. ! S -> S + S | S S | (S) | S * | a with string (a+a)*a
5. ! S -> (L) | a 以及 L -> L, S | S with string ((a,a),a,(a))
6. !! S -> a S b S | b S a S | ε with string aabbab
7. The following grammar for boolean expressions:

```
8. bexpr -> bexpr or bterm | bterm
9. bterm -> bterm and bfactor | bfactor
   bfactor -> not bfactor | (bexpr) | true | false
```

Answer

1. S =lm=> 0S1 => 00S11 => 000111

2. S =rm=> 0S1 => 00S11 => 000111
3. Omit
4. Unambiguous
5. The set of all strings of 0s and followed by an equal number of 1s

2、

1. S =lm=> +SS => +*SSS => +*aSS => +*aaS => +*aaa
2. S =rm=> +SS => +Sa => +*SSa => +*Saa => +*aaa
3. Omit
4. Unambiguous
5. The set of all prefix expressions consist of addition and multiplication.

3、

1. S =lm=> S(S)S => (S)S => (S(S)S)S => ((S)S)S => (()S)S => (()S(S)S)S => (()(S)S)S => (()()S)S => (()())S => (()())
2. S =rm=> S(S)S => S(S) => S(S(S)S) => S(S(S)) => S(S()) => S(S(S)S()) => S(S(S)()) => S(S()()) => S(()()) => (()())
3. Omit
4. Ambiguous
5. The set of all strings of symmetrical parentheses

4、

1. S =lm=> SS => S*S => (S)*S => (S+S)*S => (a+S)*S => (a+a)*S => (a+a)*a
2. S =rm=> SS => Sa => S*a => (S)*a => (S+S)*a => (S+a)*a => (a+a)*a
3. Omit
4. Ambiguous
5. The set of all string of plus, mupplication, 'a' and symmetrical parentheses, and plus is not the beginning and end of the position, multiplication is not the beginning of the position

5、

1. S =lm=> (L) => (L, S) => (L, S, S) => ((S), S, S) => ((L), S, S) => ((L, S), S, S) => ((S, S), S, S) => ((a, S), S, S) => ((a, a), S, S) => ((a, a), a, S) => ((a, a), a, (L)) => ((a, a), a, (S)) => ((a, a), a, (a))
2. S =rm=> (L) => (L, S) => (L, (L)) => (L, (a)) => (L, S, (a)) => (L, a, (a)) => (S, a, (a)) => ((L), a, (a)) => ((L, S), a, (a)) => ((S, S), a, (a)) => ((S, a), a, (a)) => ((a, a), a, (a))
3. Omit

4. Unambiguous
5. Something like tuple in Python

6、

1. S =lm=> aSbS => aaSbSbS => aabSbS => aabbS => aabbaSbS => aabbabS => aabbab
2. S =rm=> aSbS => aSbaSbS => aSbaSb => aSbab => aaSbSbab => aaSbbab => aabbab
3. Omit
4. Ambiguous
5. The set of all strings of 'a's and 'b's of the equal number of 'a's and 'b's

7、 Unambiguous, boolean expression

Left Factoring

Problem-01:

Do left factoring in the following grammar-

S → iEtS / iEtSeS / a

E → b

Solution-

The left factored grammar is-

S → iEtSS’ / a

S’ → eS / ∈

E → b

Problem-02:

Do left factoring in the following grammar-

A → aAB / aBc / aAc

Solution-

Step-01:

A → aA’

A’ → AB / Bc / Ac

Again, this is a grammar with common prefixes.

Step-02:

A → aA’

A’ → AD / Bc

D → B / c

This is a left factored grammar.

Problem-03:

Do left factoring in the following grammar-

S → bSSaaS / bSSaSb / bSb / a

Solution-

Step-01:

S → bSS’ / a

S’ → SaaS / SaSb / b

Again, this is a grammar with common prefixes.

Step-02:

S → bSS’ / a

S’ → SaA / b

A → aS / Sb

This is a left factored grammar.

Problem-04:

Do left factoring in the following grammar-

S → aSSbS / aSaSb / abb / b

Solution-

Step-01:

S → aS’ / b

S’ → SSbS / SaSb / bb

Again, this is a grammar with common prefixes.

Step-02:

S → aS’ / b

S’ → SA / bb

A → SbS / aSb

This is a left factored grammar.

Problem-05:

Do left factoring in the following grammar-

S → a / ab / abc / abcd

Solution-

Step-01:

$$S \rightarrow aS'$$
$$S' \rightarrow b \,/\, bc \,/\, bcd \,/\, \in$$

Again, this is a grammar with common prefixes.

Step-02:

$$S \rightarrow aS'$$
$$S' \rightarrow bA \,/\, \in$$
$$A \rightarrow c \,/\, cd \,/\, \in$$

Again, this is a grammar with common prefixes.

Step-03:

$$S \rightarrow aS'$$
$$S' \rightarrow bA \,/\, \in$$
$$A \rightarrow cB \,/\, \in$$
$$B \rightarrow d \,/\, \in$$

This is a left factored grammar.

Problem-06:

Do left factoring in the following grammar-

$$S \rightarrow aAd \,/\, aB$$
$$A \rightarrow a \,/\, ab$$
$$B \rightarrow ccd \,/\, ddc$$

Solution-

The left factored grammar is-

$$S \rightarrow aS'$$

$S' \rightarrow Ad \,/\, B$

$A \rightarrow aA'$

$A' \rightarrow b \,/\, \in$

$B \rightarrow ccd \,/\, ddc$

First and Follow

Check Module 3 PPT problems