Midterm-2 (1st term 2018-2019)
CSC 340
KSU

21 / 36

Student Name: [redacted]
Section:

Student Number: [redacted]
Seat Number:

---

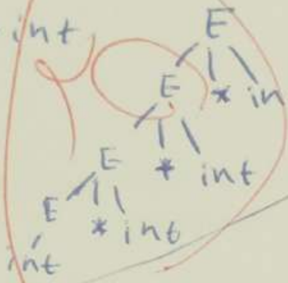
Q1) Consider the following CFG (12 marks)

S→ int * E| int +E
E→ int

A) Write two left most derivation for the string int*int+int*int (2 marks)

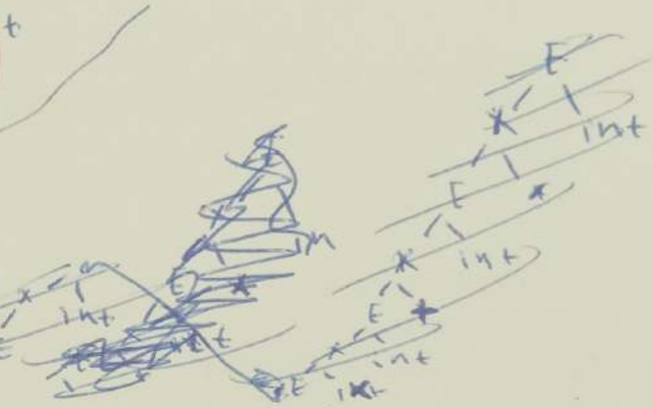
First Derivation:

E → E * int | E + int
E → int

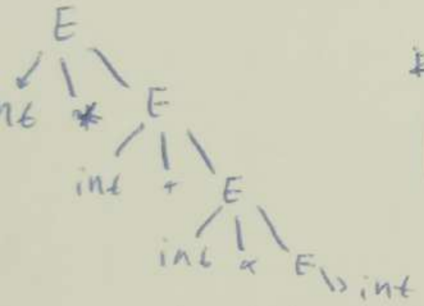


Second Derivation:

~~E → Xint~~
~~X → E* | E* | ε~~



B) Since the string int*int+int*int has more than one leftmost derivations, can we conclude that the grammar is ambiguous. (2 marks)



~~is the grammar~~
the string can do only in one pars tree

so is nuambiguous

C) Which operation has the highest precedence? Why? (2 marks)

is *, becues highest ~~pre~~ Priotify then +
it's

D) Is the associativity of the operation * right or left associativity? (2 marks)

right associativity

E) Rewrite the above grammar so that ~~you reverse~~ the * operation has higher precedence than the + operation and both operations have right associativity. (4 marks)

E → int E\`

E\` → * E | x | ϵ

x → + E | ϵ

Q2. Consider the following CFG (please notice that it is slightly different than the grammar in the previous question) (12 marks)

S→ S* E| E + S

E→ int | ε

a) Find the first and follow set for each terminal and non-terminal symbol. (4 marks)

| Symbol | First Set | Follow Set |
|---|---|---|
| S | {int, ϵ} | {*, $} |
| E | {int, ϵ} | {+, *, $} |
| int | int | {+, *, $} |
| * | * | {int, *, $} |
| + | + | {int, *, $} |

+

b) Draw the LL(1) parsing Table (3 marks)

|  | int | * | + | $ |
|---|---|---|---|---|
| S | E + S | ϵ |  | ϵ |
| E | int | ϵ | ϵ | ϵ |

c) Is the grammar LL(1) grammar? Why? or why not? (2 mark)

Yes, ti's LL(1) grammar since you not have

~~[illegible]~~ ~~cointans~~ two Path in non

d) If your answer to the above question was no, then rewrite the grammar so that the grammar is LL(1). (3 marks)

8

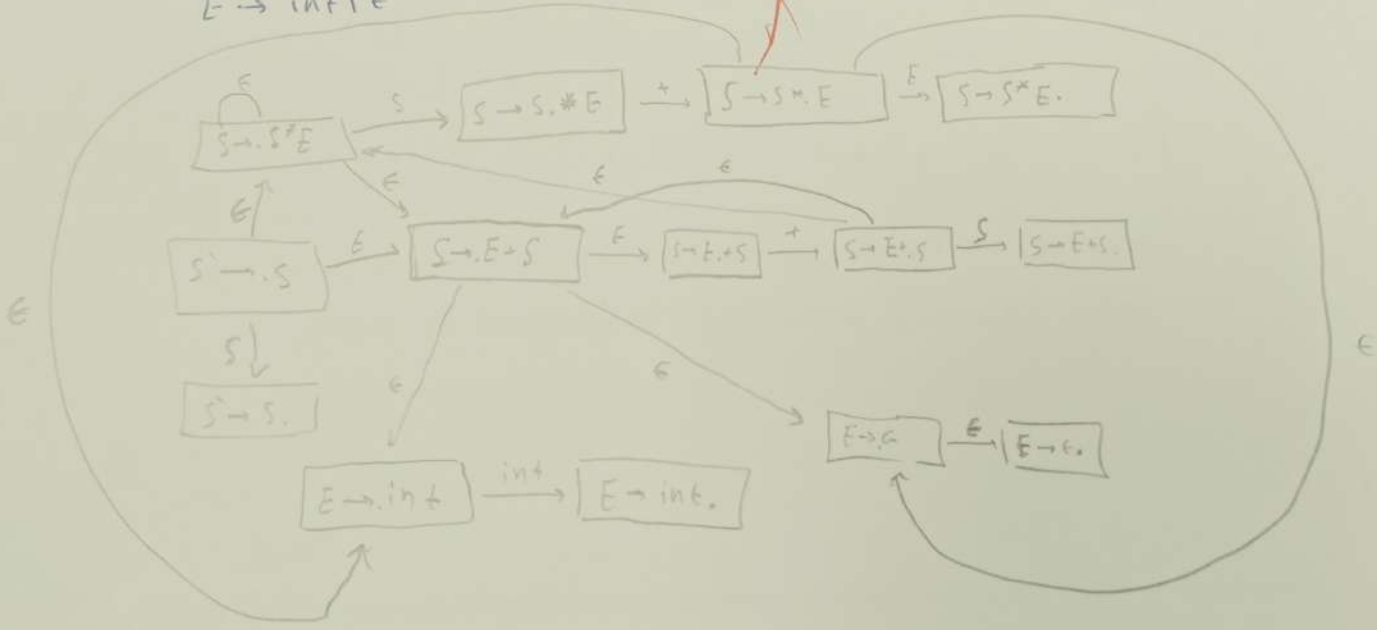
elimination left recursion.

Q3) Consider the CFG (16 marks)

**S→ S* E| E + S**
**E→ int | ε**

a) Design an NFA that recognizes the viable prefixes for the above grammar. (5 marks)

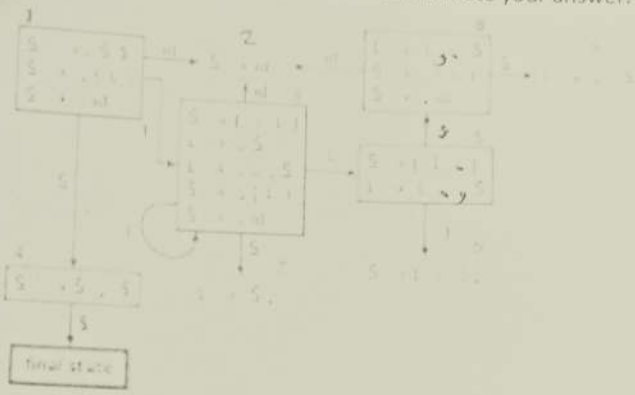
S' → S
S → S*E | E+S
E → int | ϵ



5

...ion def f(x,y) = e the activation
follows:

b) Use the following LR(0) DFA to parse the string **(id, id + id )$** using the SLR parsing algorithm. Please use a suitable table to illustrate your answer. (5 marks)



|  |  |  |
|---|---|---|
| .(id,id+id)$ | 1 | shaft |
| (.id,id+id)$ | 3 | shaft |
| (id.,id+id)$ | 2 | reduce S→id |
| (S.,id+id)$ | 7 | reduce L→S |
| (L.,id+id)$ | 5 | shaft |
| (L,.id+id)$ | 8 | chaft |
| (L,id.+id)$ | 2 | reduce S→id |
| (L,S.+id)$ | 9 | reduce L→L,S |
| (L.+id)$ | 9 | shaft |
| (L+.id)$ | ~~8~~ | ~~shaft~~ Reject |
| ~~(L+id.)$~~ | ~~9~~ | ~~shaft~~ |
| ~~(L+id).$~~ | ~~6~~ | ~~reduce S→(L+id)~~ |
| ~~S.$~~ | ~~4~~ | ~~shaft~~ |
| ~~S$.~~ | ~~final state~~ | ~~final state~~ |

5

c) Is the CFG represented using the above DFA an SLR grammar? Why? (2 marks)

easy to implement and ~~do~~ to ~~test the~~ inPut

d) Give two examples of problems that cannot be detected by both the lexical analyzer and the parser. (4 marks)

1.

2.