Exercises on Grammars

1. Consider the following grammar:

S -> (L) | a

L -> L , S | S

- Is this grammar ambiguous ?
- Is this grammar LL(1) ?
- Compute the First and Follow sets for the new grammar.
- Construct the parsing table for the LL(1) parser

Finding an LL(1) grammar

- No procedural way ! Practice …
- But there are some recommended practices that generally help in finding one.
- Eg. try to eliminate left recursion.
  - There is a procedure for this but you don’t have to faithfully follow the entire approach.
  - Just think of what left recursion brings and what can be done to eliminate them

Removing Left Recursion

S -> (L) | a

L -> L , S | S

- How does a derivation starting from ‘L’ look ?
- L => L , S

  => L , S , S

  =>* L , S , … , S

  => S , … , S
- L -> L , S | S is equivalent to L -> S , L | S

  S -> (L) | a

  L -> S , L | S

Removing Left Recursion

- In general, L -> L α | β_1 | … | β_n
- L -> β_1 Z | … | β_n Z
- Z -> α Z | ϵ

- This will remove immediate recursion
- But, what if

  S -> L a

  L -> S a | b

Removing Left Recursion

- Order nonterminals Eg. (1) S , (2) L
- Enforce that if A -> B then A should precede B in the ordering
- S -> L a and L -> b satisfy the constraint but L -> S a doesn't
- Inline the production of S in L -> S a
- L -> L a a
  - Remove left recursion. Result ??
  - If this wasn't left recursive or doesn't satisfy the constraints, inline again.

Example 1 [Cont.]

S -> (L) | a

L -> L , S | S

- After eliminating left recursion

S -> (L) | a

L -> S , L | S

- Is this LL(1) now ?

Example 1 [Cont.]

S -> (L) | a

L -> L , S | S

- After eliminating left recursion

S -> (L) | a

L -> S , L | S

- Is this LL(1) now ?

Left factorization

S -> (L) | a

L -> S , L | S

- Identify a common prefix and push the suffixes to a new nonterminal.

S -> (L) | a

L -> S Z

Z -> , L | ϵ

- Is this LL(1) now ?

Exercise 2

Consider a grammar for expressions where the multiplication sign is optional.

ex ::= ex + ex | ex * ex | ex ex |ID

- Find a LL(1) grammar recognizing the same language
- Using your grammar to derive a string
- Create the LL(1) parsing table.

Exercise 3

Balanced Parentheses over { (, [}

S ::= (S)| [S] | S S | ϵ

- Find a LL(1) grammar recognizing the language

Exercise 4

Prove that every LL(1) grammar is unambiguous.

Exercise 5

Say that a grammar has a cycle if there is a *reachable, productive* non-terminal A such that $A \Rightarrow^* A$, i.e. it is possible to derive the nonterminal A from A by a nonempty sequence of production rules.

Show that if a grammar has a cycle, then it is not LL(1).

Exercise 6

Show that the regular languages can be recognized with LL(1) parsers. Describe a process that, given a regular expression, constructs an LL(1) parser for it.

- a
- Concatenation: $r_1\ r_2$
- Union: $r_1 \mid r_2$
- Closure: r_1^*