Lena Kemmelmeier

CS 326 - Programming Languages, Concepts and Implementation

Homework 1

Due: February 8th, 2024

CS 326 – Programming Languages, Concepts and Implementation

Homework 1

(Due February 8)

1. (20 pts) Errors in a computer program can be classified according to when they are detected and, if they are detected at compile time, what part of the compiler detects them. Using your favorite programming language **(C++)**, give an example of:

   (a) (5 pts) A lexical error, detected by the scanner.

   int β = 75; // β is an invalid token

   (b) (5 pts) A syntax error, detected by the parser.

   int b = 75 // missing a semicolon at the end of this statement!

   (c) (5 pts) A static semantic error, detected (at compile-time) by semantic analysis.

   bool y = "test!"; // the data types do not match

   (d) (5 pts) A dynamic semantic error, detected (at run-time) by code generated by the compiler.

   int a;

   int c = a + 7; // a has not been initialized with a value!

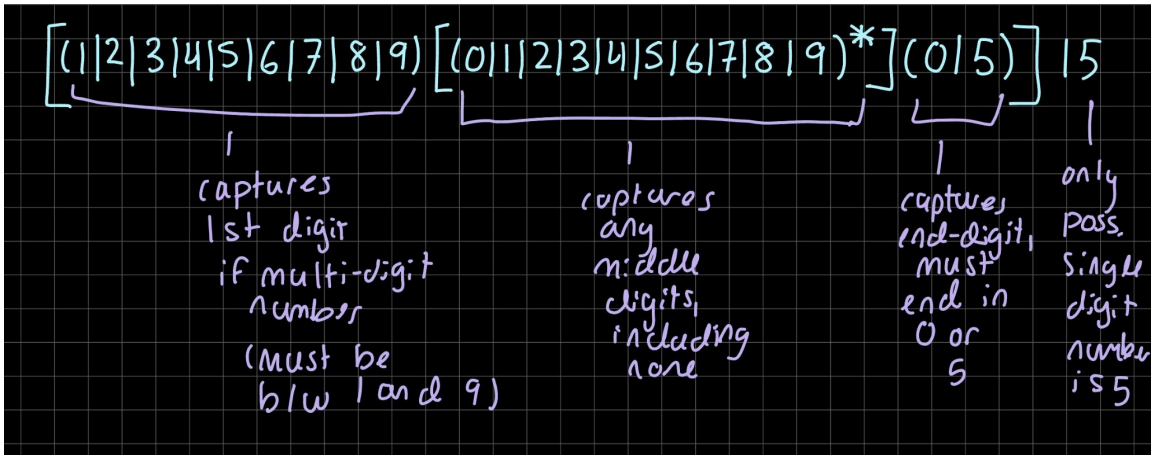
2. (18 pts) For each of the following languages, write a regular expression that describes the language.

   (a) (6 pts) The set of strings that begin with *ab* and end with *ba*, over alphabet {*a, b*}. Note: the string *aba* is in the language.

   R.E.: ab(a | b)*ba | aba

   (b) (6 pts) The set of natural numbers divisible by 5. [Does not include 0].

   R.E.: (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)[(0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)*](0 | 5) | 5



(c) (6 pts) The set of strings that consist of an even number of *a*'s (including none), over alphabet {*a*}.

R.E.: (aa)*

3. (18 pts) For each of the following languages, write a grammar that describes the language.

(a) (6 pts) The set of strings that begin with *ab* and end with *ba*, over alphabet {*a, b*}. Note: the string *aba* is in the language.

S → abTba | aba
T → aT | bT | ε

One check, generate abaaba: S → abTba →abaTba → abaaTba → abaaba

(b) (6 pts) Strings that consist of a sequence of *a*'s followed by a sequence of *b*'s, where the number of *a*'s is even, and equal to the number of *b*'s, over alphabet {*a,b*}. In other words, the language $\{a^n b^n | \ n \geq 0$ and n is even$\}$.

S → aaSbb | ε

One check, generate aaaabbbb: S → aaSbb → aaaaSbbbb → aaaabbbb

(c) (6 pts) The set of strings of parentheses ( ), brackets [ ], and braces { } that are properly nested. For instance, ( ) [{ } ( )] is properly nested, while ([ )] is not.

E → (E) | [E] | {E} | EE | ε

One check, generate ( ) [{ } ( )] : EE → (E)E → (E)[E] → (E)[E] → (E)[EE] → (E)[{E}E] → (E)[{E}(E)] → ( )[{E}(E)] → ( )[{ }(E)] → ( )[{ }( )]

4. (25 pts) Consider the following grammar, where P is the start symbol:

P → [B , P] | B
B → D | (P)
D → x | y | z

For each of the following strings, specify whether the string belongs to the language generated by the grammar, and if so, indicate a derivation:

(a) (5 pts) `z` **Yes; P → B → D → z**

(b) (5 pts) `(x)` **Yes; P → B → (P) → (B) → (D) → (x)**

(c) (5 pts) `[y]` **No; P → [B, P] This grammar requires two items in [].**

(d) (5 pts) `([x,y])` **Yes; P → B → (P) → ([B, P]) → ([D, P]) → ([x, P]) → ([x, B]) → ([x, D]) → ([x, y])**

(e) (5 pts) `[(x),y]` **Yes; P → [B, P] → [(P), P] → [(B), P] → [(D), P] → [(x), P] → [(x), B] → [(x), D] → [(x), y]**

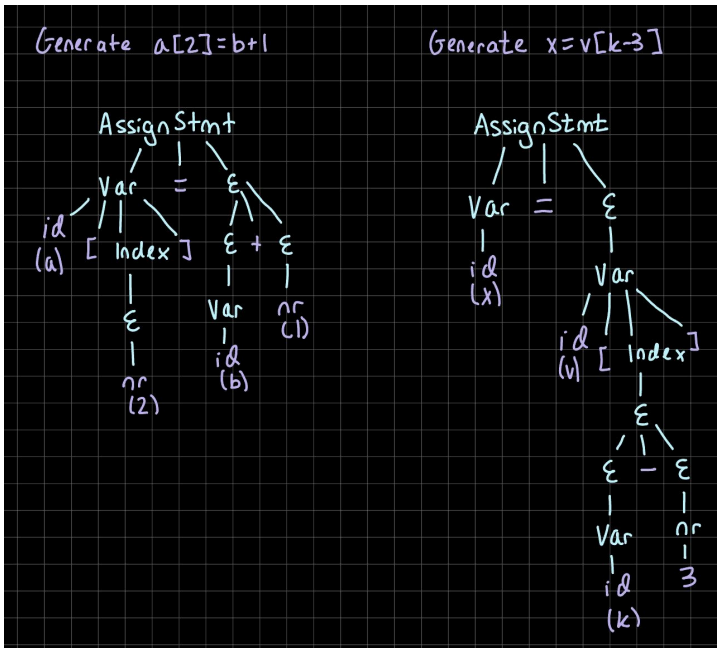
5. (19 pts) Consider the following grammar for simple assignment statements:

AssignStmt → Var = E
Var → id | id [Index]
Index → E
E → Var | nr | E + E | E – E | E * E | E / E | (E)

(a) (12 pts) Build a parse tree <u>for each</u> of the following assignment statements:

```
a[2] = b + 1
```

```
x = v[k-3]
```



(b) (7 pts) Rewrite the grammar so that it allows multi-dimensional arrays (instead of just one index, allow a list of indices separated by commas). For instance, each of the following should be a valid string under the new grammar:

```
m[5,3] = x
y = p[i+1,j,k-1]
```

AssignStmt → Var = E
Var → id | id [Indices]
Indices -> Indices, E | E
E → Var | nr | E + E | E – E | E * E | E / E | (E)

Check, generate m[5,3] = x: AssignStmt → Var = E → id[Indices] = E → id[Indices, E] = E → id[E, E] = E → id[nr, E] = E → id[nr, nr] = E → id[nr, nr] = id;

Check, generate y = p[i + 1, j, k - 1]: AssignStmt → Var = E → id = E → id = Var → id = id[Indices] → id = id[Indices, E] → id = id[Indices, E, E] → id = id[E, E, E] → id = id[E + E, E, E] → id = id[Var + E, E, E] → id = id[id + E, E, E] → id = id[id + nr, E, E] → id = id[id + nr, Var, E] → id = id[id + nr, id, E] → id = id[id + nr, id, E - E] → id = id[id + nr, id, Var - E] → id = id[id + nr, id, id - E] → id = id[id + nr, id, id - nr]

6. (Extra Credit - 10 pts) Write a regular expression that describes the following language: the set

of strings that contain an even number of *a*'s (including none), not necessarily adjacent, over alphabet $\{a, b\}$.

R.E.: b* | (b*ab*ab*)*

We can have a string with just b's or an empty string. Also, we don't want to restrict the a's in their position (they can be adjacent, non-adjacent, at the end beginning, or in the middle).