A R S D I G I T A V N I V E R S I T Y

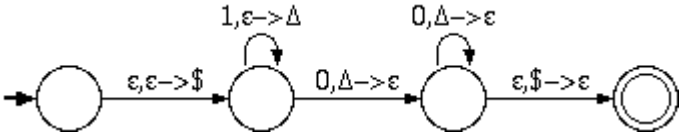
Month 8: Theory of Computation

Problem Set 3 Solutions - Mike Allen

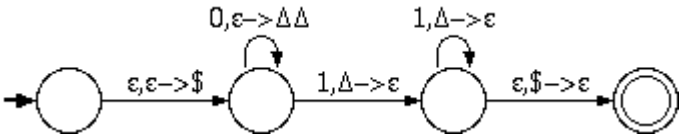
1. NPDAs

Construct non-deterministic pushdown automata to accept the following languages.

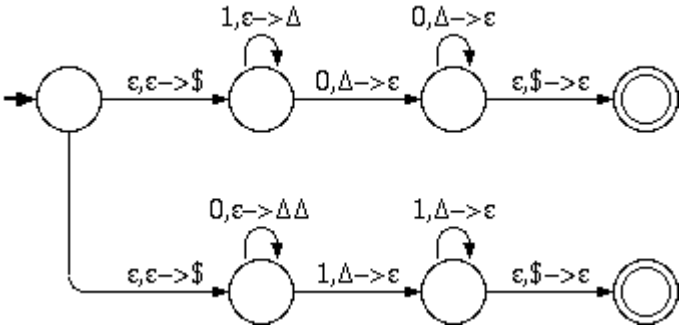
a. $\{1^n0^n \mid n>0\}$



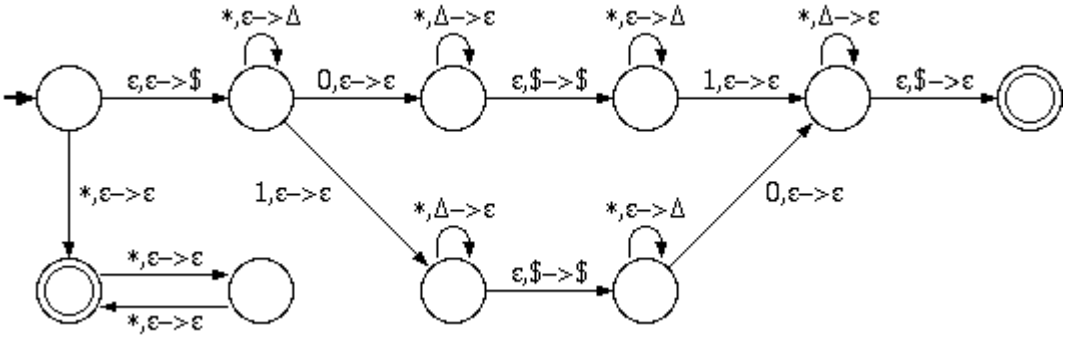
b. $\{0^n1^{2n} \mid n>=0\}$



c. $\{1^n0^n \mid n>0\} \cup \{0^n1^{2n} \mid n>=0\}$



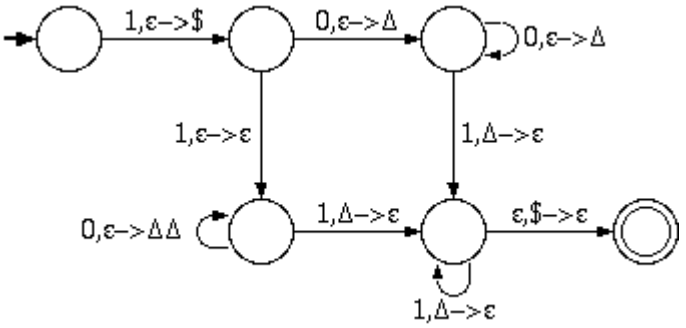
d. (0+1)* - {ww | w in {0,1}*} (complement of ww)



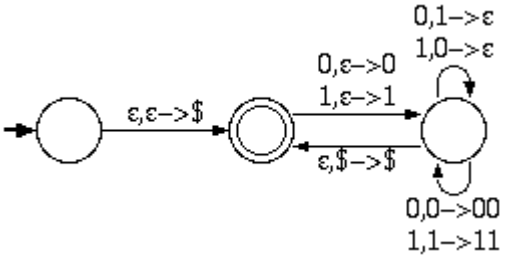
2. DPDAs

Construct deterministic pushdown automata to accept the following languages.

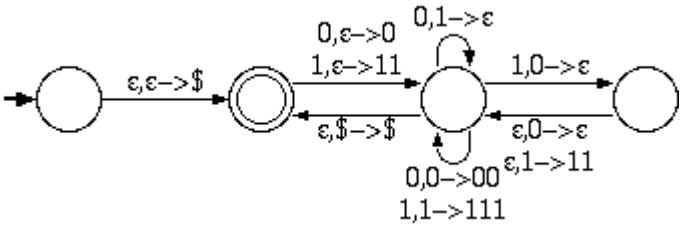
a. $\{10^n1^n \mid n>0\} \cup \{110^n1^{2n} \mid n>0\}$



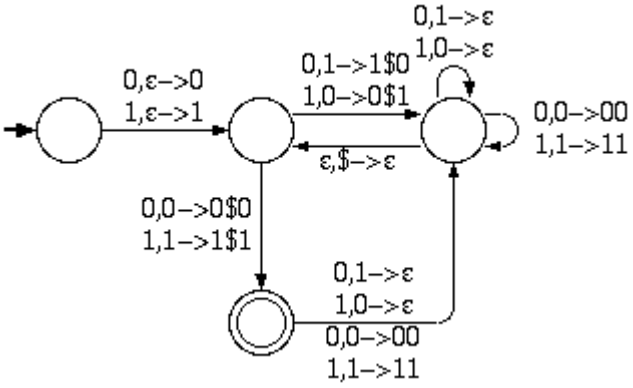
b. Binary strings tha contain an equal number of 1s and 0s.



c. Binary strings with twice as many 1s as 0s.



d. Binary strings that start and end with the same symbol and have the same number of 0s as 1s.



3. CFGs

Construct context free grammars to accept the following languages.

a. (2.4b) {w | w starts and ends with the same symbol}

```
S -> 0A0 | 1A1
A -> 0A | 1A | e
```

b. (2.4c) {w | |w| is odd}

```
S -> 0A | 1
```

```
A -> 0S | 1S | e
```

c. (2.4d) {w | |w| is odd and its middle symbol is 0}

```
S -> 0 | 0S0 | 0S1 | 1S0 | 1S1
```

d. $\{0^n1^n \mid n>0\} \cup \{0^n1^{2n} \mid n>0\}$

```
S -> 0A1 | 0B11
A -> 0A1 | e
B -> 0B11 | e
```

e. $\{0^i1^j2^k \mid i!=j \text{ or } j!=k\}$

```
S -> AC | BC | DE | DF
A -> 0 | 0A | 0A1
B -> 1 | B1 | 0B1
C -> 2 | 2C
D -> 0 | 0D
E -> 1 | 1E | 1E2
F -> 2 | F2 | 1F2
```

f. Binary strings with twice as many 1s as 0s.

```
S -> e | 0S1S1S | 1S0S1S | 1S1S0S
```

4. Ambiguity

a. Explain why the grammar below is ambiguous.

```
S -> 0A | 1B
A -> 0AA | 1S | 1
B -> 1BB | 0S | 0
```

The grammar is ambiguous because we can find strings which have multiple derivations:

S	**S**
0**A**	0**A**
0 0**AA**	0 0**AA**
00 1**S** 1	00 1 1**S**
001 1**B** 1	0011 0**A**
0011 0 1	00110 1

b. (extra credit)

c. Demonstrate that G (see problem set) is ambiguous.

The ambiguity is primarly due to the following rules:

- `IF-THEN -> if condition then STMT`
- `IF-THEN-ELSE -> if condition then STMT else STMT`

If we start with an IF-THEN statement and substitute an IF-THEN-ELSE for the consequent, we get:

```
if condition then if condition then STMT else STMT
```

However, if we start with an IF-THEN-ELSE clause and substitute an IF-THEN for the consequent, we get the same thing. So, it is ambiguous. Note that many real programming languages (such as C and Java) exhibit this exact ambiguity in their syntax.

d. (extra credit)

5. Converting to Normal Form

a. Put the following grammar into Chomsky Normal Form.

```
S -> A | AB0 | A1A
A -> A0 | e
B -> B1 | BC
C -> CB | CA | 1B
```

Remove all e rules

```
S -> e | A | AB0 | A1A | B0 | A1 | 1A
A -> A0 | 0
B -> B1 | BC
C -> CB | CA | 1B
```

Remove unit rules

```
S -> e | A0 | 0 | AB0 | A1A | B0 | A1 | 1A
A -> A0 | 0
B -> B1 | BC
C -> CB | CA | 1B
```

Convert remaining rules into proper form

```
S -> e | A0 | 0 | AS1 | B0 | A1 | 1A
A -> A0 | 0
B -> B1 | BC
C -> CB | CA | 1B
S1 -> B0 | 1A

S -> e | AN0 | AS1 | BN0 | AN1 | N1A
A -> AN0 | 0
B -> BN1 | BC
C -> CB | CA | N1B
S1 -> BN0 | N1A
N0 -> 0
N1 -> 1
```

NOTE: We did not need to create a new start state because the given one did not appear in the right side of any rule.

b. Convert the following grammar into an equivalent one with no unit productions and no useless symbols.

```
S -> A | CB
A -> C | D
B -> 1B | 1
C -> 0C | 0
D -> 2D | 2
```

Converts to

```
S -> 0C | 0 | 2D | 2 | CB
A -> C | D
B -> 1B | 1
C -> 0C | 0
D -> 2D | 2
```

A is now useless and can be removed.

6. Derivations in Chomsky Normal Form

a. (2.19) Show that if G is a CFG in Chomsky normal form, then for any string w in L(G) of length n>=1, exactly 2n-1 steps are required for any derivation of w.

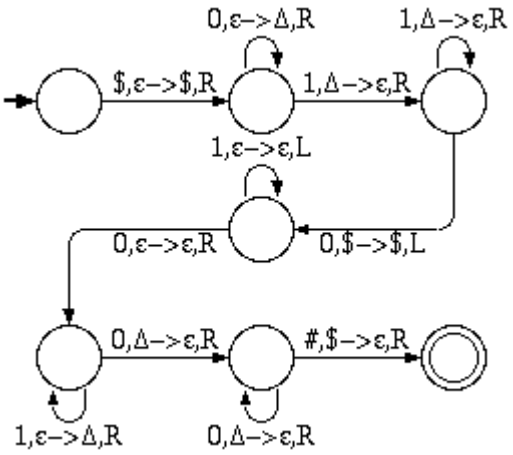
We begin with a single nonterminal ("S") and form the string by making substitutions according to the rules. Each rule of the form A->BC adds a new nonterminal, and each rule of the form A->a converts one nonterminal into a terminal. Since it takes n-1 steps to grow our string from the original nonterminal to n nonterminals, and then n steps to convert each of those nonterminals into terminals, it takes 2n-1 steps to derive a string of length n.

b. (2.20) Let G be a CFG in Chomsky normal from that contains b variables. Show that if G generates some string using a derivation with at least 2^b steps, then L(G) is infinite.

L(G) is infinite if any nonterminal shows up in a reduction of itself because this indicates a loop in the grammar. In the largest possible derivation, starting with the start variable ("S") we replace it with two other variables and each of those are replaced with two other variables and so on. Since no variable can be repeated along any branch of the tree without creating a cycle, the tree will be depth b and will contain 2^b-1 nodes. So, if a derivation requires 2^b or more steps, the grammar must contain a cycle and L(G) must be infinite.

7. Two Way PDAs

Show that the set $\{0^n1^n0^n \mid n>0\}$ is accepted by a 2-way PDA.



This 2-way PDA works by moving right across the string to make sure it begins with 0^n1^n. Then it moves left to the beginning of the 1s and continues to the right to check for 1^n0^n.