[Question 1]

Consider the following CFG:

Prove that this generates the language defined by the regular expression *a*∗*ba*

Given

Let the language generated by the CFG be

Let the language defined by the regular expression be

To show , we must prove that

|  |  |
| --- | --- |
| 1. | *Every string generated by the CFG is also in the language* |
| 2. | *Every string in the language can be generated by the CFG is also* |

*1.*

|  |  |
| --- | --- |
| **: Terminal(s)** | , |
| : **Non-terminal(s)** |  |
| **: Production Rule(s)** | P1.        *will generate words with arbitrary number of a’s*  P2.    *will generate just the word ba* |

Therefore, any string generated by will be in the form

*2.*

can be generated with the production P1:

can be generated with the production P2:

Therefore, any string generated by can be generated by

Thus,

[Question 2]

Find CFGs for the following languages over the alphabet :

All words that do not have the substring .

|  |  |
| --- | --- |
| **Terminal(s)** | , |
| : **Non-terminal(s)** | , , , |
| **Production Rule(s)** | **P1.**  **P2.**  **P3.**  …will generate the empty string  or A  or |
|  | **P4.**  **P5.**  **P6.**  …will generate strings starting with a, followed by more a's.  or strings starting with a, followed by .  or the empty string |
|  | **P7.**  **P8.**  …will generate strings starting with b, followed by more b's  *or* strings starting with b, followed by .  *or just the word b* |
|  | **P9.**  **P10.**  …will generate strings starting with b, followed by more a's.  or the empty string |

can be generated with the production PROD 1:

can be generated with the production PROD 2:

Therefore, any string generated by can be generated by

Thus,

guarantees that no generated string contains the substring "ab."

**Question 3**

Investigate each of the CFGs provided and decide whether the word *abba* is generated by the given CFGs.

In the case where *abba* is not generated a brief discussion why a particular CFG does not generate *abba*.

If abba is indeed generated, then draw the corresponding syntax tree illustrating the generation of *abba*.

*1. CFG 1:*

|  |  |
| --- | --- |
| **Terminal(s)** | , |
| : **Non-terminal(s)** |  |
| **Production Rule(s)** | P1: |
|  | P2: |

b

a

S

S

b

a

|  |  |
| --- | --- |
| Production Rule | **Terminal(s) generated** |
| PROD 1: | a |
| PROD 1: | b |
| PROD 1: | b |
| PROD 2: | a |

*2. CFG 2:*

|  |  |
| --- | --- |
| **Terminal(s)** | , |
| : **Non-terminal(s)** |  |
| **Production Rule(s)** | P1: |
|  | P2: |
|  | P3: |

S

b

S

S

a

a

|  |  |
| --- | --- |
| Production Rule | **Terminal(s) generated** |
| PROD 3: | a |
| PROD 2: | bb |
| PROD 3: | a |

b

*3. CFG 3*

|  |  |
| --- | --- |
| **Terminal(s)** | , |
| : **Non-terminal(s)** | **,** |
| **Production Rule(s)** | P1: |
|  | P2: |
|  | P3: |
|  | P4: |
|  | P5: |

X

S

Sb

a

|  |  |
| --- | --- |
| Production Rule | **Terminal(s) generated** |
| PROD 1: | a |
| PROD 1: | b |
| PROD 1: | b |
| PROD 3: | a |

a

a

X

b

a

*4. CFG 4:*

|  |  |
| --- | --- |
| **Terminal(s)** | , |
| : **Non-terminal(s)** | **,** |
| **Production Rule(s)** | P1: |
|  | P2: |
|  | P3: |
|  | P4: |
|  | P5: |

|  |  |
| --- | --- |
| Production Rule | **Terminal(s) generated** |
| PROD 1: | a |
| PROD 4: | aa |
| PROD 1: | aab |
| PROD 1: | aabb |
| PROD 2: | aabba |
| PROD 3: | a**abba**b |
| PROD 2: | abba |

*5. CFG 5:*

|  |  |
| --- | --- |
| **Terminal(s)** | , |
| : **Non-terminal(s)** | **, ,** |
| **Production Rule(s)** | P1: |
|  | P2: |
|  | P3: |
|  | P4: |
|  | P5: |
|  | P6: |
|  | P7: |
|  | P8: |

|  |  |
| --- | --- |
| Production Rule | **Terminal(s) generated** |
| PROD 1: | a |
| PROD 7: | ab |
| PROD 3: | abb |
| PROD 3: | abba |
| PROD 5: |  |

S

B

a

a

A

a

B

a

S

b

**Question 4**

Convert the grammar below to CNF.

1. Eliminate ε-productions from the grammar

2. Eliminate any non-terminal that produces a single terminal

3. Ensure productions of the form or :

*Rewrite as:*

Chomsky Normal Form.

**Question 5**

Develop a DPDA accepting the language

Define DPDA

|  |  |
| --- | --- |
|  | input alphabet |
|  | stack alphabet |
|  | Tape of infinite length containing a string |
| The states as defined on page 307 of the textbook. |  |
|  | transition function  read b, push  read b, push  read a, push  read a, pop from stack |