Жуковский Павел, 12 группа

Вариант 4

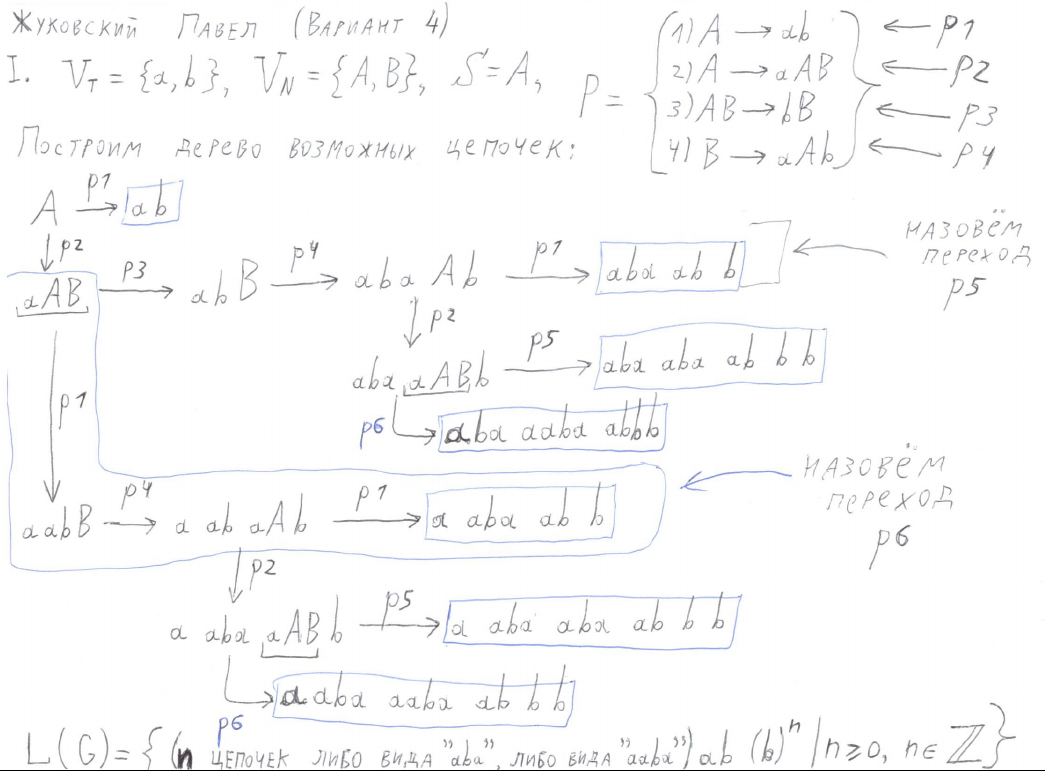
**Задания**

**I.** По заданной грамматике G = <VT, VN, P, S>, где VT, VN - конечные множества терминальных и нетерминальных символов, P - конечное множество правил грамматики, S - аксиома, определить язык, порождаемый данной грамматикой. Здесь и далее Lambda - пустая цепочка.

Вариант 4)

VT ={a,b}; VN={A,B}; S=A; P={ 1) A🡪 ab; 2) A🡪 aAB; 3) AB🡪 bB; 4) B🡪 aAb}

Для начала я предоставлю рисунок, на котором я изобразил всевозможные модификации исходной цепочки:



Для удобства, я присвоил правилам номера соответственно: p1, p2, p3 и p4. Над каждым переходом я пометил, по какому правилу я его совершал. Некоторые свои серии переводов я объединил ещё в два перевода и назвал их p5 и p6, которые также потом использовал. На рисунке я изобразил основные ответвления действий, которые мы можем делать с исходной цепочкой (дальнейшие являются продолжениями рекурсии).

На рисунке видно, что исходную цепочку «ab» можно модифицировать одним из следующих двух способов:

1. Слева добавить «aba» и справа добавить «b»
2. Слева добавить «aaba» и справа добавить «b»

Модифицировать цепочку одним из двух вышеперечисленных способов можно бесконечное количество раз. Если записать язык строго, то я бы описал его примерно вот так:

L(G) = {(**n** цепочек либо вида «aba», либо вида «aaba»)ab(b)n | n ≥ 0, n ∈ ℤ}

При чём цепочки «aba» и «aaba» могут следовать друг за другом в любом порядке в любых количествах (например, возможны цепочки (aba)(aaba)ab(b)(b) или же (aaba)(aba)ab(b)(b)).

**II.** По заданному языку построить грамматику, порождающую данный язык.

Вариант 4)

L(G) = {цепочки в алфавите {0,1}, причем каждый третий символ - "1"}

Можно по-разному записывать грамматику для данного языка. Самым наглядным (по моему мнению) способом будет вот такой:

VT = {0, 1};

VN = {A, B, C};

S = A;

P = { 1) A 🡪 0B | 1B | Lambda,

2) B 🡪 0C | 1C | Lambda,

3) C 🡪 1A | Lambda }

С помощью этого несложного набора правил мы можем строить любые цепочки (включая нулевую) из нулей и единиц, при чём каждый третий символ будет всегда равен единице (т.к. он попадает на терминальный символ C, который может перейти только в 1A, либо в Lambda). Здесь Lambda означает, что символ ни во что не переходит.