Министерство образования и науки Российской Федерации

Национальный исследовательский
ядерный университет «МИФИ»

М.А. Короткова, Е.Е.Трифонова

Задачник по курсу

«МАТЕМАТИЧЕСКАЯ ЛИНГВИСТИКА И ТЕОРИЯ АВТОМАТОВ»

Рекомендовано УМО «Ядерные физика и технологии»
в качестве учебного пособия для студентов
высших учебных заведений

Москва 2012

УДК 519.713(075)+ 519.76(075)
ББК 22.18я7
К 68

Короткова М.А., Трифонова Е.Е. Задачник по курсу «Математическая лингвистика и теория автоматов». *Учебное пособие.* М.: НИЯУ МИФИ, 2012. 92 с.

В пособие включены задачи по лингвистике, теории автоматов и кодированию. Задачи разделены по темам, каждый раздел содержит краткое изложение базовой теории. Для ряда задач даны ответы или указания.

Данное пособие предназначено для студентов факультета кибернетики и информационной безопасности, изучающих математическую лингвистику и теорию автоматов, и может быть также рекомендовано всем интересующимся математической лингвистикой и теорией автоматов.

Пособие подготовлено в рамках Программы создания и развития НИЯУ МИФИ.

Рецензент: канд. физ.-мат. наук А.Д. Яшунский

ISBN 978-5-7262-1702-4

Содержание

Предисловие

Математическая лингвистика как самостоятельная дисциплина является достаточно молодой — она сформировалась лишь в 60-е года XX века. В связи с этим, если теоретическая часть курса в какой-то мере обеспечена фундаментальными методически продуманными трудами, то изданий с задачами по математической лингвистике на настоящий момент практически нет. Данное учебное пособие призвано устранить этот пробел.

Основное внимание в пособии уделено задачам, позволяющим студентам освоиться с понятиями и методами математической лингвистики как фундаментальной дисциплины, содержатся также и задачи, демонстрирующие связи с другими дисциплинами и возможные применения математической лингвистики.

Материал пособия разбит на разделы, к каждому из разделов приведены необходимые теоретические сведения, достаточные для решения большинства задач, а также алгоритмы решения типовых задач. Для части задач приведены ответы и указания к решению.

Задачи обеспечивают материал для семинарских занятий по всему курсу «Теория автоматов и математическая лингвистика», а наличие теоретического материала позволяет использовать пособие и для самостоятельной работы студентов при освоении дисциплины.

Задачи

Тема 1. Языки, способы задания, операции над языками

Теория. V (алфавит) – любое множество различных символов.

Цепочка (слово) в алфавите V – произвольная последовательность символов $v_1v_2...v_n$, $v_i \in V$, $i \in [0,n]$. При n= 0 получается пустое слово, оно обозначается λ.

V^* - множество всех слов над алфавитом V. Длина слова $v_1v_2...v_n$ (обозначается $|v_1v_2...v_n|$) равна n . Длина пустого слова равна 0.

Конкатенация слов x и y – слово, полученное приписыванием справа к слову x слова y (обозначается xy). Свойства конкатенации:

- *в общем случае $xy \neq yx$*
- *$(xy)z = x(yz)$*
- *$\lambda x = x\lambda = x$*
- $x^n = \underbrace{xx...x}_{n...раз}$

Если x = uv, то u – начало слова x, а v – хвост (или конец) x. λ – начало и хвост любого слова.

Если x = uyv, то говорят, что имеет место вхождение y в x. При этом u называют левым крылом вхождения, а v – правым крылом.

Если λ – левое крыло, то y называют начальным вхождением, если λ – правое крыло, то y – концевое вхождение.

Инверсия.

$\tilde{X} = \widetilde{\quad} = \widetilde{\quad} = x_n x_{n-1} \ldots x_2 x_1$

$x_1x_2\ldots x_{n-1}x_n$ $\qquad$ $x_1x_2\ldots x_{n-1}\,x_1$

При этом $\tilde{\lambda} = \lambda$.

Формальным языком L на V называется произвольное подмножество V^*. *Пустой язык (язык, который не содержит ни одного слова) -* $\varnothing$. $\{\lambda\}$- λ-*язык (язык, который содержит одно слово - пустое).*

Операции над языками:

1. $L = L_1 \cup L_2 = \{x \mid x \in L_1 \vee x \in L_2\}$ – *объединение;*
2. $L = L_1 \cap L_2 = \{x \mid x \in L_1 \,\&\, x \in L_2\}$ – *пересечение;*
3. $L = L_1 \setminus L_2 = \{x \mid x \in L_1 \,\&\, x \notin L_2\}$ – *вычитание (разность);*
4. $L = L_1L_2 = \{xy \mid x \in L_1 \,\&\, y \in L_2\}$ – *произведение (конкатенация);*
5. $L_1^n = \underbrace{L_1 L_1 \ldots L_1}_{n \text{ раз}}$ – *возведение языка в степень;*
6. $L^0 = \{\lambda\}$;
7. $L_1^* = L_1^0 \cup L_1^1 \cup L_1^2 \cup \ldots = \bigcup_{n=0}^{\infty} L_1^n$ — *итерация языка* L_1;
8. $L_1^+ = L_1^1 \cup L_1^2 \cup L_1^3 \cup \ldots = \bigcup_{n=1}^{\infty} L_1^n$ — *усечённая итерация языка* L_1.

Пусть $\boldsymbol{V_T}$ *– конечный алфавит. Регулярные множества в алфавите* $\boldsymbol{V_T}$ *определяются рекурсивно:*

1) $\varnothing$ *– регулярное множество;*

2) $\boldsymbol{\{\lambda\}}$ *– регулярное множество;*

3) $\boldsymbol{\{a\}}$ *– регулярное множество для любого* $\boldsymbol{a \in V_T}$;

4) если $\boldsymbol{P}$ *и* $\boldsymbol{Q}$ *– регулярные множества, то регулярными множествами являются и* $\boldsymbol{P \cup Q}$, $\boldsymbol{PQ}$, $\boldsymbol{P^*}$;

5) никаких других регулярных множеств нет.

Регулярные выражения в алфавите V_T *обозначают регулярные множества и определяются рекурсивно следующим образом:*

1) $\mathbf{0}$ *– регулярное выражение, обозначающее регулярное множество* $\varnothing$*;*

2) $\mathbf{1}$ *– регулярное выражение, обозначающее регулярное множество* $\{\lambda\}$*;*

3) если $a \in V_T$*, то* a *– регулярное выражение, обозначающее регулярное множество* $\{a\}$*;*

4) если p *и* q *– регулярные выражения, обозначающие регулярные множества* P *и* Q*, то:*

а) $(p+q)$*– регулярное выражение, обозначающее регулярное множество* $P \cup Q$ *,*

б) (pq) *– регулярное выражение, обозначающее регулярное множество* PQ*,*

в) (p^*) *– регулярное выражение, обозначающее регулярное множество* P^**;*

5) никаких других регулярных выражений нет.

Задачи:

1. Задан алфавит V = {0; 1; 2} и слова x = 01; y = 2; z = 001. Требуется записать выражения для xy, yx, xyz, x^4, x^2y^2, $(xy)^2$ через символы алфавита.
2. Построить все возможные начала для цепочки $(xy)^2$ и всевозможные концы для цепочки xyz из предыдущей задачи.
3. Определить, имеет ли место вхождение x в z (см. задачу 1). Каково левое крыло вхождения, правое? Является ли вхождение концевым или начальным?
4. Построить все возможные начала и концы слова abcdc.
5. Определить длины слов, построенных в предыдущей задаче.

6. Пусть x=bbc, y= ac, z=b. Построить слова xy, $(xy)^2$, x^2y^2, y^3, xyz, xzy, xz^2y.
7. Доказать, что для любых слов *x* и *y* справедливо $\widetilde{xy} = \tilde{x}\,\tilde{y}$.
8. Задан алфавит V = {a; b}. Определить, являются ли множества L_1 = {a; b; baaa}; L_2 = {ba; ab; ac} и L_3 = { λ; a; ba} языками над этим алфавитом.
9. Даны языки L_1 = {ab; c} и L_2 = {c; ca}. Найти результат выполнения следующих операций: $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 \setminus L_2$, L_1L_2, L_2L_1, $L_1^2 \setminus L_2^2$, $L_1^2 \cup L_2^2$.
10. Пусть A, B, C, B – слова, AB = CD и |A| ≤ |C|. Доказать, что ∃X: C = AX и B = XD.
11. Даны языки L_1 = {a; b} и L_2 = {aa; bb}. Чему будет равен результат выполнения следующих действий: L_1L_2, L_2L_1, $L_1L_2 \cap L_2L_1$?
12. Среди приведённых выражений для языков указать те языки, которые равны между собой для любого языка L: L∅; L{λ}; ∅L; {λ}L; ∅; L; {λ}.
13. Даны произвольные языки L_1, L_2, L_3. Требуется проверить, справедливы ли следующие соотношения (доказать, что они выполняются, или привести опровергающий пример):
    a) $L_1 (L_2 \cup L_3) = L_1L_2 \cup L_1L_3$;
    b) $L_1 (L_2 \cap L_3) = L_1L_2 \cap L_1L_3$.
14. Даны два языка L_1 = {ab; c} и L_2 = {c; ca}. Чему равны значения для выражений L_i^2, L_i^3, L_i^n, L_i^*, L_i^+, где i = 1,2.
15. Даны L_1={ab, ba}, L_2={ba, aa, bb}. Построить $L_1 \cap L_2$, $L_1 \cup L_2$, $L_1 \setminus L_2$, $L_2 \setminus L_1$.
16. Даны L_1={ab, ba}, L_2={bb, aa}. Построить L_1L_2, L_2L_1, L_1^2, L_2^2.
17. Дан L_1={ab, ba}. Построить L_1^3, L_1^n, L_1^+, L_1^*.

18. Язык L_3 состоит из различных двоичных чисел. Записать этот язык в виде множества, в виде регулярного выражения.
19. Даны $L_1=\{aba, ba\}$, $L_2=\{ab, aa\}$. Чему будут равны L_1L_2, L_2L_1, L_1^2, L_2^2?
20. Пусть алфавит V = {0; 1}. В множестве L содержатся всевозможные положительные двоичные целые числа, в L_1 – всевозможные положительные двоичные нечётные целые числа.
    a) Записать выражения для L и L_1 в виде регулярных выражений;
    b) определить, чему равны результаты операций LL_1 и L_1L;
    c) вычислить, чему будут равны L^+ и L^*.
21. Пусть дан произвольный алфавит $\boldsymbol{V}$. Верно ли, что $\boldsymbol{\lambda} \in \boldsymbol{V}^*$?
22. Для каких языков $\boldsymbol{L^*=L^+}$?
23. Существует ли язык $\boldsymbol{L}$ такой, что для любого языка $\boldsymbol{L_1}$ верно, что $\boldsymbol{L_1 = L\ L_1}$?
24. Существует ли язык $\boldsymbol{L}$ такой, что для любого языка $\boldsymbol{L_1}$ верно, что $\boldsymbol{L\ L_1 = L}$?
25. Пусть $\boldsymbol{|L_1|=n}$, $\boldsymbol{|L_2|=m}$. Оцените величину $\boldsymbol{|L_1L_2|}$.
26. Пусть $\boldsymbol{|L_1|=n,\ n>0,\ |L_2|=m,\ m>0}$. В каком случае $\boldsymbol{|L_1L_2|= nm}$?
27. Записать регулярное выражение для множества всех слов в алфавите {a; b}, содержащих
    a) не менее одной буквы a;
    b) не более двух букв b подряд;
    c) не менее одной буквы a и не менее одной буквы b.
28. Записать регулярное выражение для множества всех слов в алфавите{a; b}, не содержащих
    a) двух букв b подряд;
    b) нечётное число букв b подряд;
    c) ни одного вхождения слова aba.

29. Записать регулярное выражение для множества всех слов в алфавите{a, b, c}, содержащих
    a) не более одной буквы a;
    b) хотя бы одну букву a, хотя бы одну букву b.

Тема 2. Порождающие грамматики

Теория. Порождающей грамматикой называется четверка объектов $G=< V_N, V_T, S, R>$, где

- *V_T – алфавит терминальных или основных символов;*
- *V_N — алфавит нетерминальных или вспомогательных символов ($V_T \cap V_N = \varnothing$);*
- *S – начальный символ ($S \in V_N$);*
- *R – конечное множество правил или продукций вида $\varphi \to \psi$, где $\varphi \in V_T^* V_N (V_T \cup V_N)^*$, $\psi \in (V_T \cup V_N)^*$ — различные цепочки, "$\to$" – метасимвол.*

Цепочка ω_1 непосредственно выводима из цепочки ω_0 (обозначается $\omega_0 \Rightarrow \omega_1$), если существуют такие ξ_1, ξ_2, φ, ψ, что $\omega_0=\xi_1 \varphi \xi_2$, $\omega_1=\xi_1 \psi \xi_2$ и правило $\varphi \to \psi \in R$.

Цепочка ω_n выводится из цепочки ω_0 за один или более шагов (обозначается $\omega_0 \Rightarrow^+ \omega_n$), если существует последовательность цепочек ω_0, ω_1, $\omega_2, \ldots$, ω_n ($n>0$), такая, что $\omega_i \Rightarrow \omega_{i+1}$, $i \in \{0, \ldots n-1\}$. Последовательность цепочек ω_0, ω_1, $\omega_2, \ldots$, ω_n называется выводом. Если $\omega_0 \Rightarrow^+ \omega_n$ или $\omega_0=\omega_n$, то пишут $\omega_0 \Rightarrow^* \omega_n$.

Языком L(G), порождаемым грамматикой G , называется множество цепочек в основном алфавите, выводимых из начального символа:

$$L(G) = \{x \,/\, S \Rightarrow^* x \;\&\; x \in V_T^*\}.$$

Контекстно-зависимые (КЗ-грамматики) – грамматики, все правила которых имеют вид $\varphi \to \psi$, где $\varphi \in V_T^* V_N (V_T \cup V_N)^*$, $\psi \in (V_T \cup V_N)^+$ и $|\varphi| \le |\psi|$.

Контекстно-свободные (КС-грамматики) – грамматики, все правила которых имеют вид $A \to \psi$, *где* $A \in V_N$, $\psi \in (V_T \cup V_N)^*$.

В задачах грамматики часто задаются только последовательностью правил, первое правило в последовательности определяется для начального нетерминала.

Для сокращения записи грамматик и выводов будем изображать нетерминальные символы прописными буквами латинского алфавита **A, B, C , … , S** *с индексами или без них, терминальные символы – строчными буквами* **a, b, c,…** *и цифрами. Прописными буквами* **U, V, Z** *будем обозначать символы, которые могут быть как терминальными, так и нетерминальными; строчными буквами* **u, v, x, y, z** *с индексами или без них – цепочки, составленные из терминальных символов, а буквами* $\alpha, \beta, \gamma \ldots$ *– из любых символов. Кроме того, для обозначения правил с одинаковыми левыми частями, имеющими вид:* $\alpha \to \beta_1$, $\alpha \to \beta_2, \ldots,$ $\alpha \to \beta_n$ *будем пользоваться записью* $\alpha \to \beta_1 / \beta_2 / \ldots / \beta_n$.

Задачи:

1. Проверить, выводится ли слово abaaba в грамматиках G_1, G_2 и G_3. Если да, то построить вывод.
   $G_1 : \; S \to aS \mid bS \mid a \mid b$
   $G_2 : \; S \to SS \mid a \mid b$
   $G_3 : \; S \to aSa \mid bSb \mid \lambda$
2. Пусть V={0,1}. Записать порождающие грамматики G и G_1 для языков L = {1}V^* и L_1 = {1}∪{1}V^*{1}.
3. Даны грамматики G_1 и G_2:

$$G_1:\begin{cases} S \to aS \mid aB \\ B \to bB \mid bC \\ C \to cC \mid c \end{cases};\quad G_2:\begin{cases} S \to aS \mid B \\ B \to bC \mid C \\ C \to cC \mid c \end{cases}.$$

Какие языки порождаются этими грамматиками?

4. Построить грамматику, порождающую язык a*b*c*.
5. Описать язык, порождаемый грамматикой G:
   $G: S \to aSb \mid ab$
6. Построить грамматику, порождающую язык $\{a^n b^n\}$ для:
   a) $n \geq 0$;
   b) $n \geq 2$.
7. Как будет выглядеть грамматика, которая будет порождать язык $L = \{a^n b^n c^q;\ n, q \geq 1\}$? Доказать, что построенная грамматика порождает именно этот язык.
8. Построить грамматики G_1, G_2 и G_3 для языков $L_1 = \{a^n b^m c^m;\ n, m \geq 1\}$, $L_2 = \{a^n b^m c^m;\ n, m \in \{1; 2\}\}$, и $L_3 = \{a^m b^n c^m;\ n, m \geq 1\}$ соответственно.
9. Пусть алфавит $V = \{0; 1\}$. Построить грамматику, порождающую:
   a) все цепочки из 0 и 1, где непосредственно справа от любого 0 стоит 1;
   b) все цепочки, результаты чтения которых справа и слева совпадают (двоичные палиндромы);
   c) все непустые строки из 0 и 1, где 0 в два раза больше, чем 1.
10. Какой язык порождает грамматика G_1?

$$G_1 = \begin{cases} S \to aSBa \mid aba \\ aB \to Ba \\ bB \to bb \end{cases}$$

11. Определить язык, который порождает грамматика:

$$G:\begin{cases} S \to aSBC \mid aBC \\ CB \to BC \\ aB \to ab \\ bB \to bb \\ bC \to bc \\ cC \to cc \end{cases}$$

12. Записать грамматику для языка, полученного в предыдущей задаче, в более простой форме (содержащую меньшее число правил).
13. Построить грамматику, порождающую язык L = $\{(001)^n; (010)^n; n \geq 2\}$.
14. Записать грамматику, порождающую все двоичные числа, делящиеся на 8.
15. Построить грамматику, которая будет порождать все неотрицательные десятичные числа:
    a) делящиеся на 10;
    b) делящиеся на 3;
    c) делящиеся на 6.
16. Какие цепочки будет порождать следующая грамматика?

$$G: S \to SbS \mid SaS \mid c$$

17. Какой язык порождает грамматика G?

$$G:\begin{cases} S \to bA \mid aB \\ A \to a \mid aS \mid bAA \\ B \to b \mid bS \mid aBB \end{cases}$$

18. Пусть алфавит V = {0; 1}. Построить грамматику, порождающую все цепочки, в которых единицы идут парами.
19. Для алфавита {a, b} построить грамматику, порождающую все непустые цепочки, в которых число букв b в два раза больше, чем число букв a.

20. Для алфавита V = {a; b; c; d} построить грамматики, порождающие следующие языки:
    a) $\{a^n b^m c^m d^n, n \geq 1, m \geq 1\}$;
    b) $\{a^n b^m c^n d^m, n \geq 0, m \geq 0\}$;
    c) $\{a^n b^n c^n d^m, n \geq 1, m \geq 1\}$;
    d) $\{a^n b^n c^m d^m, n \geq 1, m \geq 0\}$;
    e) $\{a^n b^n c^n d^n, n \geq 1\}$.
21. Для грамматики с множеством правил S→a|aaS построить вывод слова a^7.
22. Для грамматики G_6, заданной своими правилами, построить вывод слов a^2b^2, a^3b^3, a^4b^4. Записать грамматику полностью.

$$G_6 : \begin{cases} S \to abA \mid ab \\ bA \to Ab \\ aA \to aabA \\ aA \to aab. \end{cases}$$

23. Для грамматики G_7 с правилами:

$$G_7 : \begin{cases} S \to aS \mid aB \\ B \to bB \mid bC \\ C \to cC \mid c \end{cases}$$

построить вывод слов abc, a^2b^3c, $a^3b^2c^3$. Записать грамматику полностью. Определить язык, порождаемый грамматикой.

24. Для грамматики G_8 с правилами:

$$G_8 : \begin{cases} S \to aS \mid B \\ B \to C \mid bC \\ C \to cC \mid c \end{cases}$$

построить вывод слов abc, a^2bc^3, b^2c^3. Записать грамматику полностью. Определить язык, порождаемый грамматикой.

25. Определить класс грамматики G_6 с правилами:

$$G_6: \begin{cases} S \to abA \mid ab \\ bA \to Ab \\ aA \to aabA \\ aA \to aab \end{cases}$$

26. Построить грамматику для порождения языка L. Определить класс построенной грамматики.
    a) L= $\{a^nb^nc^m, n\geq 1, m>1\}$;
    b) L= $\{a^nb^{2n}c^m, n\geq 1, m>1\}$;
    c) L= $\{a^nb^mc^n, n\geq 1, m\geq 0\}$;
    d) L= $\{c^ma^nb^n, n\geq 0, m\geq 1\}$;
    e) L= $\{a^nb^nc^md^m, n\geq 1, m\geq 1\}$.

Тема 3. А-грамматики, конечные автоматы

Теория. Автоматные грамматики (А-грамматики) – это грамматики, все правила которых имеют вид $A \to a$, $A \to a\,B$, $A \to \lambda$, $a \in V_T$, $A,B \in V_N$.

Пусть дана А-грамматика $G=< V_N, V_T, S, R>$. *Диаграмма А-грамматики – граф с помеченными вершинами и дугами. Множество вершин графа соответствует множеству нетерминалов А-грамматики, приведенной к каноническому виду, а множество дуг – множеству правил грамматики.*

Преобразование грамматики к каноническому виду:

1. *Вводим дополнительный нетерминальный символ* Z. $V_N' = V_N \cup \{Z\}$.
2. *Заменяем все правила вида* $A \to a$ *на правила* $A \to aZ$.

3. Вводим дополнительное правило $Z \to \lambda$

После такого преобразования все правила грамматики приобретают вид $A \to aB$ *или* $A \to \lambda$

Построение диаграммы может быть описано следующими правилами.

1. Каждому нетерминальному символу поставим в соответствие вершину и пометим ее этим символом.

2. Каждому правилу $A \to aB$ *сопоставим дугу из вершины* A *в вершину* B *и пометим ее терминальным символом* a.

3. Отметим в графе как начальную вершину - вершину, соответствующую начальному символу, и как заключительные - все такие вершины B, *что* $B \to \lambda$ *(на диаграмме используется символ #) .*

Лингвистический автомат *– это* $S_L = \langle Q, V_T, q_0, F, K \rangle$, *где*

- $Q = \{q_0, q_1, \ldots q_k\}$, $k \geq 0$ *– множество состояний автомата (внутренний алфавит),*
- $V_T = \{a_1, a_2, \ldots a_m\}$, $m \geq 1$ *– множество терминальных символов (внешний алфавит) автомата,*
- q_0 *– начальное состояние автомата,* $q_0 \in Q$,
- $F: Q \times V_T \to Q$ *функция переходов,*
- $K \subseteq Q$ *– множество конечных (заключительных) состояний.*

В начале работы автомат находится в состоянии q_0, *на входе – цепочка* $a_1 a_2 \ldots a_n$, *обозревается самый левый символ цепочки. Назовём конфигурацией автомата пару* $H=(q, x)$, *где* q *— текущее состояние автомата;* x *— остаток входной цепочки, самый левый символ которой обозревается входной головкой. Говорят, что конфигурация* (p, x_1) *получена из конфигурации* (q, x) *за один такт (обозначается* $(q, x) \vdash (p, x_1)$ *), если* $x = a x_1$ *и* $F(q, a) = p$.

Если $H_0, H_1, \ldots, H_n$ $(n \geq 1)$ *— последовательность конфигураций, таких, что* $H_i \vdash H_{i+1}$, $i \in \{0,1,\ldots, n-1\}$, *то, как и раньше, будем использовать обозначения* $H_0 \vdash^+ H_n$, *и* $H_0 \vdash^* H_n$, *если справедливо* $H_0 \vdash^+ H_n \vee H_0 = H_n$.

Пусть x *— анализируемая цепочка. Начальная конфигурация имеет вид* (q_0, x), *заключительная –* (q_s, λ), $q_s \in K$. *Говорят, что автомат* A *допустил цепочку* x, *если* $(q_0, x) \vdash^* (q, \lambda)$ *и* $q \in K$ *(Использование отношения* $\vdash^*$ *позволяет включить в множество допускаемых цепочек и пустую цепочку* λ, *если* $q_0 \in K$*).*

Языком $L(A)$, *допускаемым конечным автоматом A, называется множество допускаемых им цепочек:* $L(A) = \{ x / (q_0, x) \vdash^* (q, \lambda) \ \& \ q \in K\}$.

По диаграмме автомата можно записать грамматику, порождающую язык, распознаваемый автоматом.

Правила грамматики по диаграмме автомата строятся следующим образом:

1. *Каждой вершине диаграммы сопоставляем нетерминал грамматики.*
2. *Каждой дуге из вершины* P *в вершину* Q, *помеченной терминалом* a, *сопоставляется правило грамматики* $P \to aQ$.
3. *Каждой заключительной вершине* R *сопоставляется правило* $R \to \lambda$.
4. *Начальной вершине диаграммы сопоставляется начальный символ грамматики.*

Будем называть недетерминированным конечным автоматом S пятерку объектов $S = \langle Q, V_T, q_0, F, K \rangle$, *где интерпретация* Q, V_T, q_0, K *такая же, как и раньше, а* F *– отображение* $Q \times V_T$ *в* $P(Q)$. *Здесь* $P(Q)$ *обозначает*

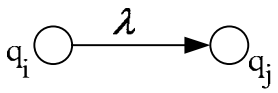
множество подмножеств множества ***Q****. Множество состояний* $\{q_{i_1}, q_{i_2}, ..., q_{i_l}\}$ *будем обозначать как обобщённое состояние* $[q_{i_1}, q_{i_2}, ..., q_{i_l}]$.

Алгоритм построения детерминированного автомата по недетерминированному:

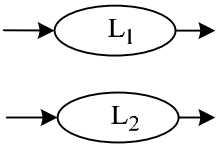
1. *Строим начальное состояние* $q_0' = [q_0]$*, помечаем его как начальное.*
2. *Для каждого состояния, построенного на предыдущем шаге, строим* $F(q_i', a)$ *для всех* $a \in V_T$*. Если для какого-нибудь из построенных состояний функция перехода ещё не построена, возвращаемся к шагу 2.*
3. *Помечаем как конечные все состояния* $q_i' = [q_{i_1}, q_{i_2}, ... q_{i_l}]$ *такие, что* $\{q_{i_1}, q_{i_2}, ... q_{i_l}\} \cap K \neq \varnothing$.

Конечность процесса обеспечивается конечностью множества $P(Q)$.

Автоматы с λ *- переходами естественно возникают в различных приложениях, и позволяют представить любой автомат в виде двухполюсников с одним входом и одним выходом, а также строить сети из таких автоматов, сохраняя в них единственный вход и единственный выход. От рассмотренных ранее автоматов они отличаются тем, что в них присутствуют переходы, осуществляемые без чтения входной цепочки (на диаграмме такие переходы обозначаются стрелками, помеченными символом* λ*).*

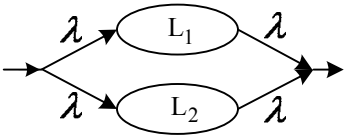


*Очевидно, что если **L** – А-язык, то ему можно сопоставить некоторый двухполюсник.*
*Пусть языкам **L_1** и **L_2** сопоставлены соответствующие двухполюсники (см рис.).*

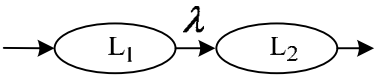


*Тогда их объединению, конкатенации и итерации языка **L_1** будут, соответственно, сопоставлены двухполюсники:*

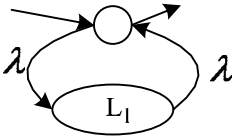
$L_1 \cup L_2$:



$L_1 L_2$:



L_1^*



Алгоритм минимизации лингвистического автомата (метод Хаффмана).

1. *Составляем таблицу переходов. По строкам указываются входные символы, по столбцам – состояния.*
2. *Составляем классы предположительно эквивалентных состояний (первоначально разбиванием на класс конечных и неконечных состояний).*
3. *Составляем таблицу переходов, в которой вместо состояний указываются классы, которым принадлежат состояния. Если в пределах одного класса состояния ведут себя по-разному (есть отличия в столбцах таблицы переходов), то класс разбивается на подклассы состояний, для которых столбцы одинаковы, и возвращаемся к шагу 3.*
4. *Если во всех классах состояния ведут себя одинаково, то это действительно классы эквивалентности. Строится автомат, состояниями которого являются классы эквивалентности исходного автомата.*

Систему уравнений с регулярными коэффициентами назовем стандартной над множеством неизвестных $\Delta=\{X_1,X_2,\ldots,X_n\}$, *если она имеет вид*

$$\begin{cases} X_1 = \alpha_{10} + \alpha_{11}X_1 + \alpha_{12}X_2 + \ldots + \alpha_{1n}X_n; \\ X_2 = \alpha_{20} + \alpha_{21}X_1 + \alpha_{22}X_2 + \ldots + \alpha_{2n}X_n; \\ \ldots \\ X_n = \alpha_{n0} + \alpha_{n1}X_1 + \alpha_{n2}X_2 + \ldots + \alpha_{nn}X_n; \end{cases}$$

где все $\alpha_{ij}, i\in[1,n], j\in[0,n]$ *– регулярные выражения. Если какое-либо* i*-е уравнение не содержит переменную* X_j, *то достаточно положить соответствующий коэффициент* $\alpha_{ij}=0$, *если* $\alpha_{ij}=1$, *то этот коэффициент можно не писать.*

Пусть есть А-грамматика $G = < V_T, V_N, S, R>$, *правила для* A_i:

$$A_i \to a_1 \mid a_2 \mid \ldots a_k \mid b_1A_{j1} \mid b_2A_{j2} \mid \ldots \mid b_mA_{jm}$$

где $a_s, b_q \in V_T, A_{js} \in V_N$. *Обозначим* X_k *– язык, порождаемый грамматикой* G_k, *в которой в качестве начального символа*

выбран символ A_k*.Тогда множество правил для нетерминала* A_i *соответствуют следующему уравнению:*

$$X_i = a_1 + a_2 + \ldots + a_k + b_1 X_{j1} + b_2 X_{j2} + \ldots + b_m X_{jm}.$$

Задачи.

1. Дана А-грамматика:

$$G: \begin{cases} S \to aA \mid bS \mid \lambda \\ A \to aB \mid bA \\ B \to aS \mid bB \end{cases}$$

   Для этой грамматики:
   a) показать порождение цепочки abaab;
   b) построить диаграмму грамматики, показать порождение цепочки abaab на ней;
   c) определить язык, порождаемый грамматикой;
   d) записать регулярное выражение для языка;
   e) построить конечный автомат, допускающий язык (записать как пятёрку объектов);
   f) показать, используя конфигурации, процесс допускания цепочки X= abaab.
2. Является ли конечный язык, например, L = {ab, abb}, А-языком? Доказать, что любой конечный язык – А-язык.
3. Определить, какой язык порождается А-грамматикой (записать регулярное выражение):

$$G: \begin{cases} S \to aA \mid bB \\ A \to b \mid aA \\ B \to \lambda \end{cases}$$

4. Построить А-грамматику для порождения следующих языков, а так же диаграмму грамматики:

a) $L = \{a^n b^m, n \geq 1, m \geq 1\}$;

b) $L = \{a^n b^m, n \geq 1, m \geq 0\}$;

c) $L = \{a^n b^m, n \geq 0, m \geq 1\}$;

d) $L = \{a^n b^m, n \geq 0, m \geq 0\}$.

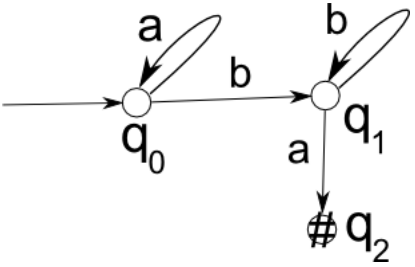
5. Можно ли построить А-грамматику для языков L_1, L_2? Ответ обосновать.

   a) $L_1 = \{a^n b^n, n \geq 1\}$;

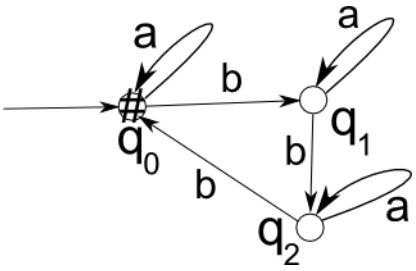
   b) $L_2 = \{a^n b^n, 1 \leq n \leq 3\}$

6. Построить А-грамматику для языка L = $\{ (001)^n; (010)^n; n \geq 2\}$. Изобразить диаграмму А-грамматики, рассмотреть её как диаграмму автомата, построить эквивалентный детерминированный автомат.
7. Задана диаграмма автомата. Записать функцию переходов автомата, построить соответствующую грамматику.

a.

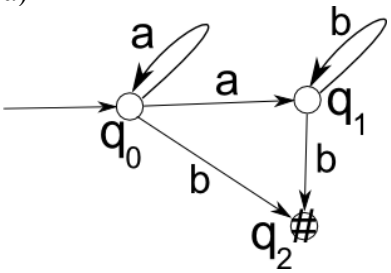


b.

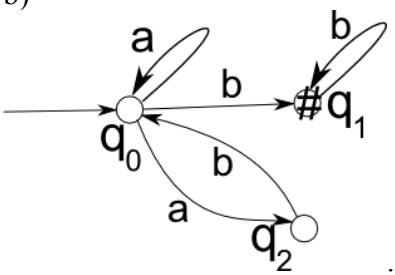


8. Недетерминированный автомат задан графом переходов. Необходимо записать эквивалентную грамматику и построить эквивалентный детерминированный автомат.

a)



b)



9. Грамматика G задана списком правил. Построить для неё эквивалентный недетерминированный и детерминированный автоматы.

$$G:\begin{cases} S \to bB \mid bS \\ B \to aB \mid aC \mid \lambda \\ C \to bC \mid b \end{cases}$$

10. Для каждой из заданных А-грамматик G_1 и G_2 построить двухполюсники, распознающие языки $L_1=L(G_1)$ и $L_2=L(G_2)$ соответственно.

$$G_1:\begin{cases} S \to aB \mid b \\ B \to bS \mid b \end{cases};\ G_2:\begin{cases} S \to bA \mid b \\ A \to aA \mid bS \end{cases}$$

11. Для языков L_1 и L_2, определённых в предыдущей задаче, построить двухполюсники, распознающие языки:
    a) $L_1 \cup L_2$;
    b) $L_1 L_2$;
    c) $L_1 *$.
12. Построить детерминированный автомат для каждого из двухполюсников, построенных в предыдущей задаче.
13. Пусть задан алфавит {a, b} и язык L, состоящий из всех непустых цепочек из a и b, где за каждой буквой b непосредственно справа следует a (то есть bb не встречается). Для данного языка:
    a) построить А-грамматику, порождающую данный язык;
    b) изобразить диаграмму автомата, распознающего этот язык;
    c) произвести детерминизацию автомата;
    d) построить грамматику, порождающую язык, допускаемый автоматом;
    e) построить по грамматике стандартную систему уравнений и решить её;
    f) построить А-грамматику по полученному регулярному выражению.

14. Регулярно ли множество цепочек в алфавите {a, b}, у которых число b равно числу a? Если множество регулярно, построить соответствующий автомат. Ответ обосновать.

15. Регулярно ли множество цепочек с числом букв a, равным числу букв b, у которых для любого начала число a превышает число b не более чем на 2, а число b никогда не превышает число a? Если множество регулярно, построить соответствующий автомат.

16. Регулярно ли множество цепочек в алфавите {a, b}, у которых число букв b чётно, а число букв a – нечётно? Если множество регулярно, построить соответствующий автомат.

17. Записать по A-грамматике G_5 систему стандартных уравнений для языка $L(G_5)$. Решить систему. Определить $L(G_5)$.

$$G_5 : \begin{cases} S \to aS \mid bA \\ A \to aA \mid a \end{cases}$$

18. Записать по A-грамматике G_6 систему стандартных уравнений для языка $L(G_6)$. Решить систему. Определить $L(G_6)$.

$$G_6 : \begin{cases} S \to aB \mid b \\ B \to bS \mid b \end{cases}$$

19. Регулярно ли множество цепочек с чётным числом нулей и нечётным числом единиц? Является ли это множество A-языком? Если да, то построить A-грамматику, порождающую данное множество, построить по грамматике систему стандартных уравнений и решить её.

20. Решить систему уравнений в классе регулярных выражений

$$\begin{cases} X_1 = (ab^*+b)X_1 + X_2 \\ X_2 = bb + bX_1 + aaX_3 \\ X_3 = X_1 + X_2 \end{cases}$$

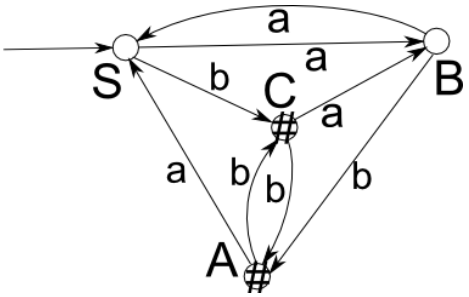
Упростить полученные выражения.

21. Построить детерминированный автомат, допускающий (распознающий) цепочки, заданные регулярными выражениями:

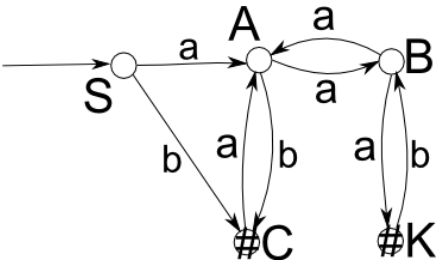
a) $a(ba+b)^*+b$;
b) $(ab)^*(bc)^*$;
c) $(a+bc)(a+bc)^*$;
d) $(ab+b^*)^*ba+b$;
e) $((b^*a)^*ab^*)^*$;
f) $c(acc+aa)^*a$;
g) $a(b^*+a)^*b+a^*bb^*$;

22. Пусть автомат задан своим графом переходов. Построить эквивалентный минимизированный автомат.

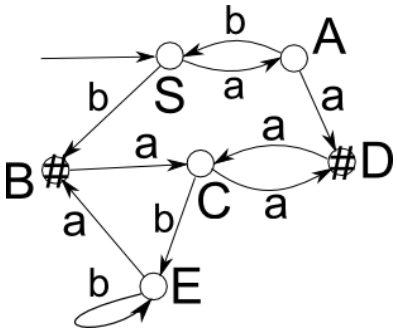
а)



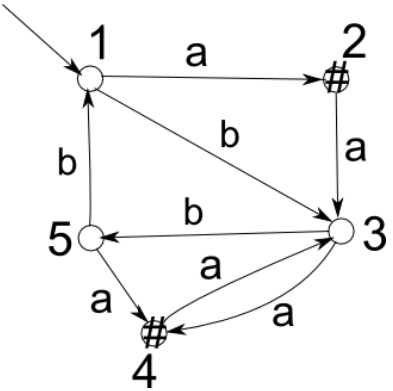
b)



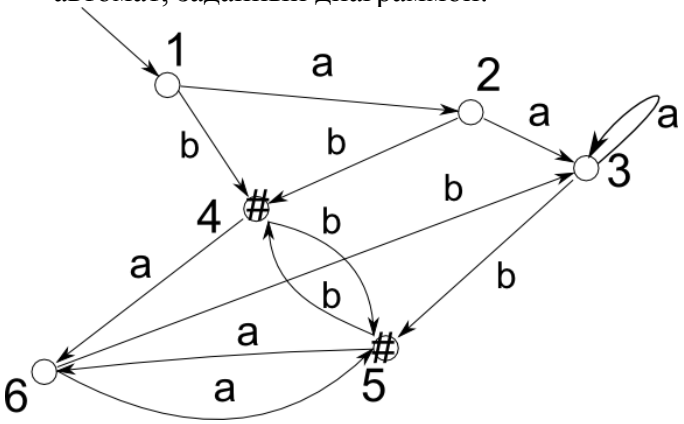
c)



23. Минимизировать число состояний автомата, заданного диаграммой:



24. Минимизировать детерминированный лингвистический автомат, заданный диаграммой:



25. Для заданного регулярного выражения построить автомат, детерминизировать его, минимизировать число состояний, для минимального автомата построить А-грамматику:
    a) $a(ac*+c)*c$;
    b) $a*ba*+ac(b+c*a)*$;
    c) $(ab*+a)a*+ab$.
26. Записать регулярное выражение для языка, чьи слова составлены из символов алфавита {a;b;c} и содержат не

менее 1 буквы a и одной буквы b. Построить А-грамматику, порождающую этот язык.

27. Записать регулярное выражение для языка, чьи слова составлены из символов алфавита {a;b;c}, а третий символ с конца слова – c. Построить А-грамматику, порождающую этот язык.
28. Записать регулярное выражение для языка, чьи цепочки составлены из a и b, а число букв b делится на 5. Построить А-грамматику, порождающую этот язык.
29. Записать регулярное выражение для языка, у которого цепочки состоят из a и b и не содержат вхождений слова bab. Построить А-грамматику, порождающую этот язык.

Тема 4. Бекусовы (бекусовские) нормальные формы

Теория. БНФ – нотация, применяемая для задания КС-грамматик.

КС- грамматика: правила имеют вид:
$A \to \psi, A \in V_N, \psi \in (V_T \cup V_N)^*$.

БНФ(бекусова нормальная форма):
1. Нетерминалы обозначаются произвольным словом и заключаются в угловые скобки (<слово>).
2. Терминалы записываются так, как они есть.
3. $\to$ *заменяется на ::=.*
4. Альтернативы разделяются вертикальной чертой |.

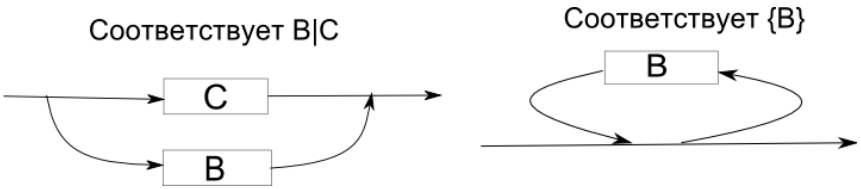
РБНФ(расширенная бекусова нормальная форма):
1. Нетерминалы обозначаются словами.
2. Терминалы заключаются в апострофы (‘a’).
3. $\to$ *заменяется на =.*

4. Вертикальная черта используется для разделения альтернатив.
5. Необязательная конструкция заключается в квадратные скобки([слово]).
6. Конструкция, которая может повторяться 0 и более раз, заключается в фигурные скобки ({конструкция }).
7. ( ) служат для факторизации, т.е. конструкция $\varphi\psi|\gamma\psi$ *может быть представлена в виде* $(\varphi|\gamma)\psi$*. Это позволяет использовать конструкцию выбора на более глубоком уровне.*
8. В конце каждого описания ставится точка.

Синтаксическая диаграмма. Синтаксическая диаграмма – графическая форма представления РБНФ. Для каждого нетерминала рисуется своя диаграмма. Приняты следующие обозначения:

1. *Нетерминалы заключаются в прямоугольники.*
2. *Терминалы заключаются в овалы.*
3. *Образы нетерминалов и терминалов соединяются линиями (со стрелками или без, мы будем использовать стрелки для указания направления движения) таким образом, чтобы множество путей соответствовало множеству цепочек из терминалов и нетерминалов, которое задаётся РБНФ – правилами.*

Диаграммы для конструкции выбора и итерации представлены на рисунке.

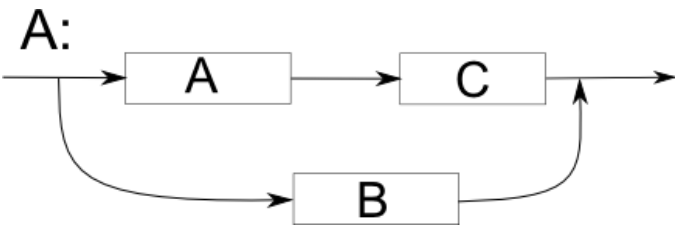


Задачи.

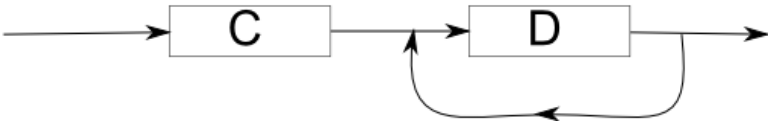
1. Пусть дано описание целого числа с помощью БНФ:
   <число без знака> ::= <целая часть>|<дробь>e<показатель степени>|<дробь>|e<показатель степени>
   <дробь> ::= <целая часть>.<дробная часть>|.<дробная часть>
   <целая часть> ::= <цифра>|<цифра без 0><непустая последовательность>
   <цифра> ::= <цифра без 0>|0
   <цифра без 0> ::= 1|2|3|4|5|6|7|8|9
   <дробная часть> ::= <непустая последовательность>
   <непустая последовательность> ::= <цифра>|<цифра><непустая последовательность>
   <показатель степени> ::= <непустая последовательность>|<знак><непустая последовательность>
   <знак>::=+|-

Описать данный язык с помощью А-грамматики, построить диаграмму автомата, распознающего этот язык. Описать язык с помощью РБНФ.

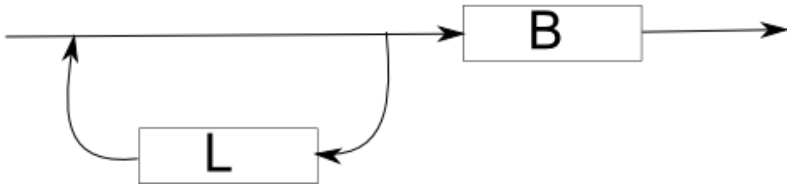
2. По приведённым ниже диаграммам записать БНФ и РБНФ:



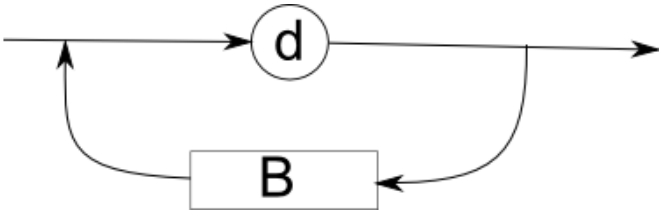
B:



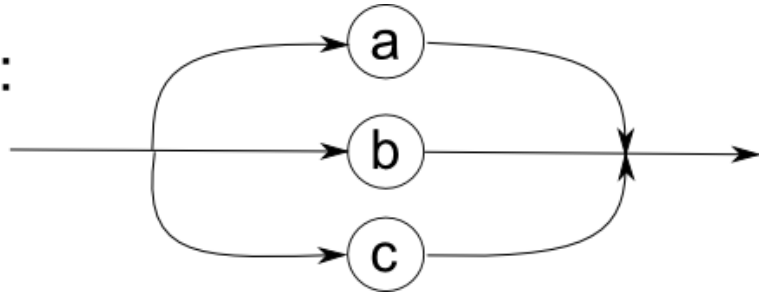
C:



D:



L:

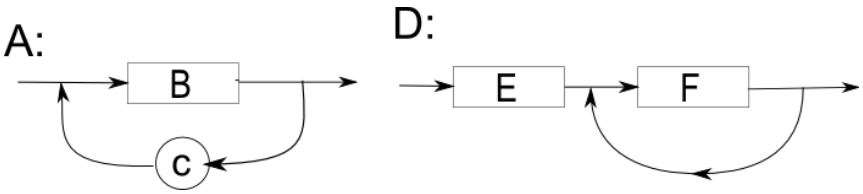


3. Дана БНФ. Построить РБНФ и диаграмму, избавившись от «самоссылочности».
   <последовательность операций>::=<операция>|<операция>;<последовательность операций>
4. Дана РБНФ. Записать в виде БНФ, нарисовать диаграмму.
   Число = [(‘+’|’-’)]цифра{цифра}[‘,’{цифра}].
   Цифра = ‘1’|’2’.
5. Построить грамматику, БНФ и РБНФ для построения логических выражений из символов 0 и 1 с использованием связок &,∨,¬ и скобок: (,) . Символом ¬ обозначается отрицание.
6. Построить синтаксическую диаграмму и БНФ по приведённой РБНФ: A=[‘a’{B’a’}]’b’{‘b’}.
7. Задана БНФ для условия:
   < условие >::= < терм > < оператор сравнения > < терм >
   < терм > ::= < переменная >|< число >|(< выражение >)
   < оператор сравнения > ::= <|>|=|≤|≥
   < выражение >::= < терм > < арифметическая операция > < выражение >
   |< терм >
   < арифметическая операция > ::= +|-|×|/
   Записать РБНФ по БНФ. Построить синтаксическую диаграмму.
8. Задана РБНФ для условного оператора.
   Условный_оператор = 'If' условие 'then' последовательность_операторов ['else' последовательность_операторов] 'endif'.
   последовательность_операторов = оператор {оператор}.

оператор = переменная '='
выражение | условный_оператор.

Построить синтаксическую диаграмму. Построить БНФ.

9. Построить БНФ, синтаксическую диаграмму и РБНФ для оператора CASE Visual Basic.
10. Построить БНФ, синтаксическую диаграмму и РБНФ для оператора FOR Visual Basic.
11. Построить БНФ и РБНФ для синтаксических диаграмм

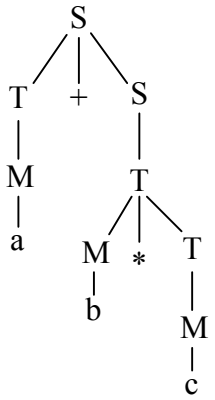


Тема 5. Структура цепочек. СУ-схемы

Теория. На синтаксическом уровне сталкиваемся с наличием структуры у анализируемых цепочек, принадлежащих языку, построенному с помощью КС-грамматики.

Существуют различные варианты представления структуры:

1. *Синтаксическое дерево*



2. *Скобочная запись*

*(a+(b*c))*

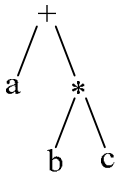
3. *Обратная польская запись*

abc+*

4. *Любые формы, удобные разработчику; но наше описание обязано приписывать каждой цепочке её структуру – структурный уровень описания языка. Например, КС-грамматика даёт возможность успешно решать эту задачу.*

Для синтаксических деревьев (называемых также полными синтаксическими деревьями) существует сокращённая форма. Сокращённое дерево получается из полного синтаксического дерева «подтягиванием» знаков операций в вышележащие узлы синтаксического дерева и исключением всех нетерминалов. Скобки в сокращённом дереве опускаются, так как при таком задании структуры вложенность операций определена однозначно.

Для приведённого выше полного синтаксического дерева сокращённое дерево имеет вид:



При использовании КС-грамматик для задания структуры цепочки повышаются требования к грамматикам: они должны приписывать правильную структуру, например, для арифметических выражений должен учитываться приоритет операций.

Синтаксическое дерево – графическая форма представления вывода в заданной грамматике. Всегда по выводу можем построить графическое представление.

Левосторонний обход (ЛКП) терминальных вершин дерева даёт нам терминальную цепочку. Внешним вершинам дерева приписываются терминальные символы, внутренним вершинам – нетерминальные символы. Одному и тому же дереву могут соответствовать разные выводы, в зависимости от порядка применения правил.

Вывод называется левым (правым), если на каждом шаге правило применяется к самому левому (правому) вхождению нетерминала.

Утверждение: Между правым (левым) выводом и синтаксическим деревом существует взаимно однозначное соответствие

Цепочка называется неоднозначной в грамматике G, если для неё существует более одного дерева. Грамматика называется неоднозначной, если она порождает неоднозначные цепочки. Существуют существенно-неоднозначные языки, которые не могут быть порождены ни одной однозначной грамматикой.

Для задания структуры цепочек часто используются СУ-схемы, т.е. схемы синтаксически управляемого перевода. СУ- схема $T=< V_{вх}, V_{вых}, V_N, I, R>$, *где*

- $V_{вх}$ *– входной алфавит ,*
- $V_{вых}$ *– выходной алфавит ,*
- V_N *– множество нетерминалов,*
- I *– начальный нетерминал,*
- $R=\{ A \to \varphi_1, \varphi_2\}$, *где* $A \in V_N$, $\varphi_1 \in (V_{вх} \cup V_N)^*$, $\varphi_2 \in (V_{вых} \cup V_N)^*$, *а «,» - метасимвол, разделяющий* φ_1 *и* φ_2.

При этом количество и состав нетерминалов в φ_1 *и* φ_2 *совпадают. Если в* φ_1 *более одного вхождения некоторого нетерминала A, то устанавливается соответствие между всеми вхождениями A в* φ_1 *и* φ_2: $A^{(1)}, A^{(2)} \ldots A^{(n)}$.

СУ-схема называется простой, если порядки вхождений нетерминалов в φ_1 *и* φ_2 *совпадают.*

Вывод в СУ-схеме начинает строиться из пары начальных символов <I, I>. Определение выводимости в СУ-схеме подобно определению выводимости (выводимость за один шаг обозначаем $\Rightarrow$ *) для обычных грамматик:* $<\omega_1,\delta_1> \Rightarrow <\omega_2,\delta_2>$ *(из* $<\omega_1,\delta_1>$ *непосредственно выводима* $<\omega_2,\delta_2>$*), если* $\omega_1=\theta_1 A \theta_2$, $\omega_2=\theta_1 \varphi_1 \theta_2$, $\delta_1=\alpha_1 A \alpha_2$, $\delta_2 =\alpha_1 \varphi_2 \alpha_2$, *и* $A \to \varphi_1, \varphi_2 \in R$, *при этом вхождения A в* ω_1 *и* δ_1 *– соответствующие.*

Выводимость является рефлексивным и транзитивным замыканием непосредственной выводимости.

СУ-схема применяется к цепочке следующим образом: правила применяются одновременно к двум входным символам таким образом, чтобы до запятой получилась исходная цепочка, в этом случае после запятой получается перевод этой цепочки.

Задачи.

1. Даны СУ-схемы, построить перевод для цепочек i+i*i, i+i*(i+i), i*i+(i+i)*i с помощью каждой из них, сравнить результаты:

$$\text{T1:}\begin{cases} S \to T + S, TS + \\ S \to T, T \\ T \to M * T, MT * \\ T \to M, M \\ M \to (S), S \\ M \to i, i \end{cases}$$

$$\text{T2:}\begin{cases} S \to T + S, [T + S] \\ S \to T, T \\ T \to M * T, [M * T] \\ T \to M, M \\ M \to (S), [S] \\ M \to i, i \end{cases}$$

2. Имеется КС-грамматика G_1. Построить синтаксические деревья для цепочек i, (i), i+(i*i), (i+i)*i, (i+i)*i+i*i.

$$G_1\begin{cases} S \to S + T \,|\, T \\ T \to T * M \,|\, M \\ M \to (S) \,|\, i \end{cases}$$

3. Построить синтаксические деревья для цепочек a↑b×2+c, a+b×2↑c, 7×1↑b↑(a+7), (1+a)+2×c в грамматике G_1.

$$G_1: \begin{cases} S \to T + S \mid T \\ T \to M \times T \mid M \\ M \to M \uparrow K \mid K \\ K \to (S) \mid a \mid b \mid c \mid D \\ D \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{cases}$$

4. Дана грамматика G_6 со следующим множеством правил:

$$G_6: \begin{cases} S \to aSc \mid B \\ B \to bB \mid b \end{cases}$$

Построить в ней синтаксическое дерево для цепочек:

- abc;
- $a^2b^3c^2$;
- ab^2c;
- a^2bc^2;
- $a^3b^3c^3$.

5. Дана грамматика G_7 со следующим множеством правил:

$$G_7: \begin{cases} S \to +TS \mid T \\ T \to *MT \mid M \\ M \to S \mid a \mid c \mid b \mid d \end{cases}$$

Построить в ней синтаксическое дерево для цепочек:

- +a*b+cd;
- *+a*bcd;
- +*+abcd;
- +a**bcd.

6. Для грамматик G_1, G_2 построить синтаксические деревья для цепочек i+i+i, i*i*i, сокращённые деревья. Сравнить полученные сокращённые деревья.

$$G_1\begin{cases} S \to S+T \,|\, T \\ T \to T*M \,|\, M \\ M \to (S) \,|\, i \end{cases} \qquad G_2\begin{cases} S \to T+S \,|\, T \\ T \to M*T \,|\, M \\ M \to (S) \,|\, i \end{cases}$$

7. Является ли грамматика G однозначной? Что означает неоднозначность грамматики при интерпретации цепочек? Построить всевозможные сокращённые деревья для цепочек i+i*i, i+i+i.

$$G:\begin{cases} S \to SAS \,|\, (S) \,|\, i \\ A \to + \,|\, \backslash \,|\, * \,|\, - \end{cases}$$

8. Однозначна ли грамматика G? Если нет, то привести пример неоднозначных цепочек.

$$G:\ \{S \to SS \,|\, (S) \,|\, a \,|\, b$$

9. Определить, однозначна ли грамматика. Если нет, то привести пример неоднозначных цепочек.

$$G:\begin{cases} S \to bA \,|\, aB \\ A \to a \,|\, aS \,|\, bAA \\ B \to b \,|\, bS \,|\, aBB \end{cases}$$

10. Построить однозначную грамматику, порождающую семантически правильные деревья для правильных булевских выражений. Используются следующие связки: $\supset, \vee, \wedge, \neg, (,)$ и константы i, t, f.
11. Доказать, что язык $\{a^n b^n c^n;\ n \geq 1\}$ не может быть построен КС-грамматикой.

Тема 6.Эквивалентные преобразования КС-грамматик

Теория. Будем рассматривать такие преобразования грамматик, которые не выводят нас из класса эквивалентности, т.е. грамматика G_1 таким образом преобразуется в грамматику G_2, чтобы $L(G_1)=L(G_2)$.

Устранение непроизводящих правил

Непроизводящими правилами грамматики называются правила, применение которых никогда не приводит к построению терминальных цепочек.

Алгоритм устранения непроизводящих нетерминалов:

Пусть дана грамматика $\boldsymbol{G=< V_N, V_T, S, R>}$, *строится эквивалентная грамматика без непроизводящих символов* $\boldsymbol{G' =< V_N', V_T, S, R'>}$.

1. *Построение множества производящих нетерминалов N_R:*
   a. *Строится множество $N_R^1 = \{A/ A \to \beta_j, \beta_j \in V_T^*\}$*
   b. *Последовательно строятся множества $N_R^{i+1} = N_R^i \cup \{ A/ A \to \beta_j, \beta_j \in (N_R^i \cup V_T)^*\}$*
   c. *Построение продолжается до тех пор, пока не получим $N_R^{i+1} = N_R^i$ или же $N_R^i = V_N$. Тогда $N_R = N_R^i$.*
2. *Исключаем из грамматики все правила, в которых присутствуют нетерминалы из $V_N \setminus N_R$.*

Устранение недостижимых нетерминалов

Недостижимыми называются нетерминалы, которые никогда не могут встретиться в выводе, начальным символом которого является начальный символ грамматики.

Алгоритм построения грамматики, в которой используются только достижимые нетерминалы (множество достижимых нетерминалов –D) , выглядит следующим образом:

Пусть дана грамматика $G=<V_N, V_T, S, R>$*, строится эквивалентная грамматика без недостижимых нетерминалов* $G'=<V_N', V_T, S, R'>$.

1. *Построение множества достижимых нетерминалов:*

   a. *Начальное значение* $D_0=S$.

   b. *Итерационное построение множества* D:
   $D_{i+1}= D_i \cup \{A/(B \to \alpha A \beta \in R) \& (B \in D_i)\}$

   c. *Построение продолжается, до тех пор, пока не будет выполнено* $D_{i+1}= D_i$ *или* $D_{i+1}= V_N$, *в обоих случаях* $V_N'= D_{i+1}$,

2. *Построение множества правил преобразованной грамматики:*

$$R'=\{ R_i / R_i=(A \to \varphi), A \in V_N', \varphi \in (V_N' \cup V_T)^*\}$$

Построенная грамматика не содержит недостижимых нетерминалов.

Устранение λ-правил.

Пусть дана грамматика $G=< V_N, V_T, S, R>$*, строится эквивалентная грамматика* $G' =< V_N', V_T, S, R'>$.

1. *Построение множества* $N_\lambda = \{ A / A \Rightarrow^+ \lambda \}$ *- множество нетерминальных символов, из которых возможен вывод пустой цепочки. Множество* N_λ *строится итерационно:*
   a) *на первом шаге строится* N_λ^0*:*
   $N_\lambda^0 = \{ B / B \to \lambda \in R\}$;
   b) *последовательно строятся множества* $N_\lambda^{i+1} = N_\lambda^i \cup \{ B / (B \to \varphi \in R) \& (\varphi \in (N_\lambda^i)^*)\}$;
   c) *построение продолжается до тех пор, пока не получим* $N_\lambda^{i+1} = N_\lambda^i$*, тогда* $N_\lambda = N_\lambda^i$.
2. *Построение* R' *— множества правил эквивалентной грамматики:*
   a) *Если* $A \to \alpha_0 B_1 \alpha_1 B_2 \dots B_k \alpha_k \in R$, $k \geq 0$ *и* $B_i \in N_\lambda$ *для* $0 \leq i \leq k$*, цепочки* α_j, $1 \leq j \leq k$ *не содержат символов из* N_λ*, то включить в* R' *все правила вида*
   $A \to \alpha_0 X_1 \alpha_1 X_2 \dots X_k \alpha_k$*, где* X_i *либо* B_i*, либо* λ *(при этом правило* $A \to \lambda$ *не включается, даже если все* $\alpha_i = \lambda$*).*
   b) *если* $S \in N_\lambda$*, включить в* R' *также правило* $S' \to S / \lambda$*, где* S' *- новый начальный символ.*

Таким образом, любая КС-грамматика может быть приведена к виду, когда R *не содержит* λ*-правил, либо есть точно одно правило* $S' \to \lambda$ *и* S' *не встречается в правых частях остальных правил из* R.

Устранение цепных правил (правил вида $A \to B$***)***

Применение цепных правил приводит к увеличению длины ветвей синтаксического дерева, исключение цепных правил часто приводит к большей «прозрачности» грамматики и уменьшению длины выводов, которые можно построить.

Пусть дана грамматика $G=<V_N, V_T, S, R>$, *строится эквивалентная грамматика* $G'=<V_N', V_T, S, R'>$ *без цепных правил.*

1. *Построение для каждого* $A \in V_N$ *множества* $N_A = \{ B / A \to B \}$, *т.е. множества нетерминальных символов, выводимых из данного символа. Итерационная процедура построения* N_A*:*
   a) *начальное значение* $N_A^0 = \{A\}$;
   b) $N_A^{i+1} = N_A^i \cup \{ C / (B \to C \in R) \ \& \ (B \in N_A^i) \}$;
   c) *построение продолжается до тех пор, пока не получим* $N_A^{i+1} = N_A^i$, *тогда* $N_A = N_A^i$.
2. *Построение множества* R' *(множества правил эквивалентной грамматики): если* $B \to \alpha \in R$ *и не является цепным правилом, то включить в* R' *все правила вида* $A \to \alpha$, *для всех таких* A, *что* $B \in N_A$.

Задачи.

1. Преобразовать грамматику G в эквивалентную, устранив λ-правила, а затем – цепные правила. Определить язык, порождаемый грамматикой. Построить более простую грамматику G_1, которая порождает этот язык.

$$G: \begin{cases} S \to ABC \\ A \to BB \mid \lambda \\ B \to CC \mid a \\ C \to AA \mid b \end{cases}$$

2. Устранить цепные правила в грамматике G:

$$G:\begin{cases} S \to A \mid B \\ A \to C \mid D \\ B \to D \mid E \\ C \to S \mid a \mid e \\ D \to S \mid b \\ E \to S \mid c \mid e \end{cases}$$

3. Устранить цепные правила в грамматике G:

$$G:\begin{cases} S \to S + T \mid T \\ T \to T * F \mid F \\ F \to (S) \mid a \end{cases}$$

4. Преобразовать грамматику G в следующем порядке:
   1) устранить непроизводящие нетерминалы;
   2) устранить недостижимые нетерминалы;
   3) устранить λ-правила;
   4) устранить цепные правила.

   Упростить получившуюся грамматику, сохраняя эквивалентность.

$$G:\begin{cases} S \to AB \\ A \to SA \mid BB \mid bB \\ B \to b \mid aA \mid aE \mid \lambda \\ E \to EE \mid aEF \\ F \to a \mid b \end{cases}$$

5. Преобразовать грамматики G_1, G_2, G_3 в том же порядке, что и для предыдущей задачи. Определить языки, порождаемые грамматиками.

$$G_1:\begin{cases} S \to aS \mid B \mid aE \\ A \to a \mid b \\ B \to aBA \mid bB \\ C \to cC \mid c \mid \lambda \\ E \to bC \mid C \end{cases}, \quad G_2:\begin{cases} S \to aAB \mid bE \mid c \mid bB \mid BB \\ A \to AA \mid bAE \mid cA \mid AD \\ B \to bB \mid \lambda \\ D \to a \mid bD \\ E \to AE \mid aE \end{cases}$$

$$, \; G_3:\begin{cases} S \to aS \mid AB \mid AE \\ A \to aA \mid AA \mid \lambda \\ B \to bB \mid b \mid BC \\ C \to cC \mid CC \mid \lambda \\ D \to aD \mid aA \mid bD \\ E \to ED \mid aE \end{cases}$$

Тема 7. LL(k), строго LL(k)-грамматики

Теория. Определим следующие понятия:

$$\overline{first_k}(x) = \begin{cases} u, & uy = x, |u| = k \\ x, & 0 < |x| < k \\ \varnothing, & x = \lambda \end{cases}$$

для $x \in V_T^*$.

Для $\alpha \in (V_T \cup V_N)^*$ $First_k(\alpha) = \{ x \mid \alpha \Rightarrow^* Z, Z \in V_T^*, x = \overline{first_k}(Z) \}$.

Множество $\mathbf{Follow_1(X)=\{a \mid S \Rightarrow^+ \omega X a \beta \,\&\, a \in V_T\}}$, *для* $\mathbf{X \in (V_N \cup V_T)}$.

LL(k) грамматика – это грамматика, для которой для любых двух левых выводов, таких что:

$$S \Rightarrow^* \omega\, A\alpha \Rightarrow \omega\, \beta_1\alpha \Rightarrow^* \omega\, x_1$$
$$S \Rightarrow^* \omega\, A\alpha \Rightarrow \omega\, \beta_2\alpha \Rightarrow^* \omega\, x_2\ ,$$

где $A \in V_N$, $\omega, x_1, x_2 \in V_T^*$, $\alpha, \beta_1, \beta_2 \in (V_N \cup V_T)^*$, *из условия* $First_k(x_1) = First_k(x_2)$ *следует, что* $\beta_1 = \beta_2$.

Будем называть грамматику строго LL(k)-грамматикой, если для любых двух левых выводов:

$$S \Rightarrow^* \omega_1 A\alpha_1 \Rightarrow \omega_1\ \beta_1\alpha_1 \Rightarrow^* \omega_1\ x_1$$
$$S \Rightarrow^* \omega_2 A\alpha_2 \Rightarrow \omega_2\ \beta_2\alpha_2 \Rightarrow^* \omega_2\ x_2\ ,$$

где $A \in V_N$, $\omega_1, \omega_2, x_1, x_2 \in V_T^*$, $\alpha_1, \alpha_2, \beta_1, \beta_2 \in (V_N \cup V_T)^*$, *из условия* $First_k(x_1) = First_k(x_2)$, *следует, что* $\beta_1 = \beta_2$.

Для грамматики $G=< V_N, V_T, S, R>$ *соответствующая пополненная грамматика есть* $G'=< V_N \cup \{S'\}, V_T \cup \{\$\}, S', R'>$, *где множество правил* $R'=R \cup \{S' \rightarrow S\ \$\ \}$, *здесь каждая цепочка имеет справа граничный маркер* **($)**.

Обозначим $M_A^{\beta_i} = \bigcup_{\substack{\alpha \\ S \Rightarrow^* \omega A \alpha}} First_k(\beta_i\alpha)$ *и потребуем, чтобы*

$M_A^{\beta_i} \cap M_A^{\beta_j} = \varnothing$ *при* $i \neq j$.
Тогда G является строго LL(k) грамматикой тогда и только тогда, когда для любого $A \in V_N$ *из того, что* $A \rightarrow \beta \in R$, $A \rightarrow \gamma \in R$, $\beta \neq \gamma$, *следует, что* $M_A^{\beta} \cap M_A^{\gamma} = \varnothing$.

Справедлива следующая теорема: . LL(1) грамматика всегда строго LL(1) грамматика.

Задачи.
1. Дана грамматика G,

$$G:\begin{cases} S \to V_1 \\ V_1 \to V_2 \mid V_1 i V_2 \\ V_2 \to V_3 \mid V_2 + V_3 \mid i V_3 \\ V_3 \to)V_1 * \mid (\end{cases}$$

a) построить в данной грамматике выводы цепочек (; (+(;)(*; (+(i(, i)(*; (+)(*i(; (ii(;

b) найти $First_1(N_i)$, $First_2(N_i)$, $Follow_1(T_i)$, для всех $N_i \in V_N$, $T_i \in V_N \cup V_T$.

2. Найти множества $first_1$ и $follow_1$ для нетерминалов грамматики G_1 и множества $First_1$, $Follow_1$ для нетерминалов грамматики G_1 и множества $First_1$, $First_2$, $follow_1$ для нетерминалов грамматики G_2.

$$G_1:\begin{cases} S \to TS+ \mid T \\ T \to MT* \mid M \\ M \to S \mid i \end{cases}, \; G_2:\begin{cases} S \to T + S \mid T \\ T \to M * T \mid M \\ M \to i \mid (S) \end{cases}$$

3. Проверить, являются ли LL(1)-грамматиками грамматики G_1, G_2:

$$G_1:\begin{cases} S \to aAB \mid bS \\ A \to aA \mid bB \\ B \to AB \mid c \end{cases}, \; G_2:\begin{cases} S \to AB \mid BC \mid CA \\ A \to aB \mid bA \\ B \to cB \mid bb \\ C \to aC \mid c \end{cases}$$

4. Построить грамматику для порождения префиксной польской записи для арифметических выражений, построенных с использованием знаков + и * из a, b, c. Проверить, является ли построенная грамматика LL(1), LL(2), LL(k) для какого-либо k.
5. Проверить, является ли грамматика G LL(k)-грамматикой, строго LL(k)-грамматикой для какого-либо k.

$$G:\begin{cases} S \to aAaa \mid bAba \\ A \to b \mid \lambda \end{cases}$$

6. Проверить, является ли грамматика G LL(k)-грамматикой для какого-либо k.

$$G:\begin{cases} S \to A \mid B \\ A \to 0A0 \mid 0 \\ B \to 0B1 \mid 1 \end{cases}$$

7. Для грамматики G_3 построить пополненную грамматику и проверить, является ли полученная грамматика LL(1)-грамматикой.

$$G_3:\begin{cases} S \to AS \mid \lambda \\ A \to aA \mid b \end{cases}$$

8. Устранить λ-правила в грамматике предыдущего упражнения. Проверить, является ли полученная грамматика LL(1)-грамматикой, LL(k) для какого-либо k.
9. Проверить, является ли LL(1)-грамматикой следующая грамматика:

$$\begin{cases} S \to AB \mid BC \mid CA \\ A \to aB \mid bA \\ B \to cB \mid bb \\ C \to aC \mid c \end{cases}$$

Определить язык, порождаемый грамматикой.

10. Дана грамматика:

$$\begin{cases} S \to aAB \,|\, bAC \\ A \to d \,|\, \lambda \\ B \to dbB \,|\, a \\ C \to bdC \,|\, c \end{cases}$$

Проверить, является ли данная грамматика LL(k)-грамматикой для какого-либо k, строго LL(k) для какого-либо k. Определить язык, порождаемый грамматикой.

11. Дана грамматика:

$$\begin{cases} S \to aB \,|\, bC \,|\, \lambda \\ B \to bB \,|\, aB \,|\, \lambda \\ C \to cB \,|\, aD \\ D \to aB \,|\, Bc \end{cases}$$

Построить пополненную грамматику. Проверить, является ли грамматика LL(1)-грамматикой. Устранить λ-правила. Является ли преобразованная грамматика LL(k) для какого-либо k? Строго LL(k) для какого-либо k?

12. Проверить, является ли грамматика G LL(k)-грамматикой для какого-либо k.

$$G: \begin{cases} S \to aB \mid bAS \mid bA \\ A \to bAA \mid a \\ B \to aBB \mid b \end{cases}$$

13. Дана грамматика:

$$\begin{cases} S \to aAB \,|\, bAC \\ A \to d \,|\, \lambda \\ B \to dbB \,|\, a \\ C \to bdC \,|\, c \end{cases}$$

Является ли данная грамматика LL(k) для какого-либо k? Строго LL(k) для какого-либо k? Определить язык, порождаемый грамматикой.

Тема 8. Грамматики простого предшествования (ГПП), восходящий анализ

Теория. Грамматика G называется ***грамматикой простого предшествования****, если:*

1. *между любой парой символов определено не более одного отношения предшествования;*
2. *нет λ-правил;*
3. *нет правил с одинаковыми правыми частями.*

Отношения простого предшествования с указанными свойствами могут быть определены на $V_N \cup V_T$ *следующим образом:*

- $X \lessdot Y$*, если в* R *есть правило* $A \to \alpha X B \beta$*, и при этом* $B \Rightarrow^{+} Y\omega$*;*
- $X \doteq Y$*, если в* R *есть правило* $A \to \alpha X Y \beta$*;*
- $X \gtrdot a$*, если в* R *есть правило* $A \to \alpha B Y \beta$*, и* $B \Rightarrow^{+} \gamma X$*,* $Y \Rightarrow^{*} a\delta$*.*

Отношение $\gtrdot$ *определяется на* $(V_N \cup V_T) \times V_T$ *, так как непосредственно справа от основы может быть только терминальный символ.*

Обозначим
$Head(A)=\{X/A \Rightarrow^{+} X\alpha\}$ $(First_1(A)=Head(A) \cap V_T)$,
$Tail(A)=\{X/A \Rightarrow^{+} \alpha X\}$,
тогда
$X <\circ Y \Leftrightarrow (A \to \alpha X B \beta)$ & $(Y \in Head(B))$,
$X \circ > a \Leftrightarrow (A \to \alpha B C \beta)$ & $(X \in Tail(B))$ & $(a \in First_1(C)$.

Удобно заключить анализируемую цепочку в концевые маркеры $ и $, положив для $X \in V_N \cup V_T$, $X \circ > \$$ *для всех* X, *для которых* $S \Rightarrow^{+} \alpha X$ *и* $\$ <\circ X$ *для всех* X, *для которых* $S \Rightarrow^{+} X \alpha$. *Для заключения цепочки в маркеры вводим новый начальный символ* S' *и правило* $S' \to \$S\$$.

***Алгоритм разбора для ГПП**:*

1. Анализируемая цепочка заключается в маркеры.

2. Берём очередной символ из входного буфера (слева направо). Если между верхним символом стека и первым символом входной цепочки отношение $<\circ$ *или* $\doteq$, *то заносим этот символ из входной цепочки в стек и возвращаемся к шагу 2, если же между верхним символом стека и первым символом входной цепочки отношение* $\circ>$, *то тогда переходим к шагу 3. Если между символами нет никакого отношения предшествования, то цепочка не принадлежит языку, порождаемому грамматикой.*

3. Обратное движение: из стека вынимаются символы до первого отношения $<\circ$ *между первым символом стека и символом цепочки во входном буфере. Если такой символ появился, то переходим к шагу 4, иначе цепочка не принадлежит языку, порождаемому грамматикой.*

4. Применяем свёртку (заменяем выделенный фрагмент на левую часть правила грамматики, правая часть которого совпадает с основой) и возвращаемся к шагу 2. Если свёртка неприменима (нет такой правой части правила), то цепочка не принадлежит языку, порождаемому грамматикой.

Если в результате применения свёртки мы приходим к цепочке $ S $, то исходная цепочка принадлежит языку, порождаемому грамматикой, в противном случае цепочка не принадлежит языку, порождаемому грамматикой.

Задачи.

1. Являются ли грамматики G_1, G_2 грамматиками простого предшествования?

$$G_1: \begin{cases} S \to AS \mid \lambda \\ A \to aA \mid b \end{cases}, \quad G_2: \begin{cases} S \to AS \mid c \\ A \to aA \mid b \end{cases}$$

2. Является ли грамматика G ГПП? Какие цепочки она порождает? Построить разбор цепочек (aa), (((aa)a)a).

$$G: \begin{cases} S \to (R \mid a \\ R \to Sa) \end{cases}$$

3. Является ли данная грамматика G грамматикой простого предшествования?

$$G: \begin{cases} S \to V_1 \\ V_1 \to V_2 \mid V_1 i V_2 \\ V_2 \to V_3 \mid V_2 + V_3 \mid i V_3 \\ V_3 \to)V_1 * \mid (\end{cases}$$

4. Проверить, является ли грамматика G ГПП, LL(1)-грамматикой?

$$G: \begin{cases} S \to aBc \mid cDa \\ D \to dS \mid f \\ B \to bS \end{cases}$$

Построить восходящий разбор для цепочек abcfac, cdcfaa.

5. Проверить, является ли ГПП следующая грамматика:

$$\begin{cases} S \to S + T \,|\, T \\ T \to T * M \,|\, M \\ M \to (S) \,|\, i \end{cases}$$

Обосновать ответ.

6. Проверить, является ли ГПП следующая грамматика:

$$\begin{cases} A \to aABC \,|\, CB \\ B \to aB \,|\, c \\ C \to b \end{cases}$$

Обосновать ответ.

Тема 9. Конечные автоматы Мили и Мура

Теория.

Конечный автомат (автомат Мили): $S=< V_a, Q, V_b, q_0, F, G>$, *где*

- $V_a=\{a_1,a_2,\ldots a_m\}$, $m\geq 1$ – *входной алфавит автомата,*
- $V_b= \{b_1, b_2, \ldots, b_n\}$, $n\geq 1$ – *выходной алфавит автомата,*
- $Q= \{q_0,q_1,\ldots q_k\}$, $k\geq 0$ – *внутренний алфавит (алфавит состояний),*
- $q_0 \in Q$ – *начальное состояние автомата,*
- F – *функция переходов;* $F: Q\times V_a \to Q$,
- G – *функция выходов,* $G: Q\times V_a \to V_b$.

Автомат однозначно задает отображение $V_a^* \to V_b^*$ *(входной цепочки в выходную), называемое автоматным отображением. Мы будем рассматривать только инициальные автоматы, где входное состояние задано*

статически, в ряде случаев начальное состояние рассматривается как $q_0 = q(0)$.

Автоматы Мура отличаются от автоматов Мили тем, что одному состоянию (а не переходу) соответствует один выход.

Конечный автомат Мура: $S = < V_a, Q, V_b, q_0, F, G >$, где

- *$V_a = \{a_1, a_2, \ldots a_m\}$, $m \geq 1$ – входной алфавит автомата,*
- *$V_b = \{b_1, b_2, \ldots, b_n\}$, $n \geq 1$ – выходной алфавит автомата,*
- *$Q = \{q_0, q_1, \ldots q_k\}$, $k \geq 0$ – внутренний алфавит (алфавит состояний),*
- *$q_0 \in Q$ – начальное состояние автомата (инициальный автомат, иногда рассматривают $q_0 = q(0)$),*
- *F - функция переходов; $F: Q \times V_a \rightarrow Q$,*
- *G - функция выходов, $G: Q \rightarrow V_b$.*

Приняты две схемы задания автоматов Мура:

Первая схема	*Вторая схема*
$\begin{cases} q(t+1) = f(q(t), a(t)) \\ b(t) = g(q(t)) \end{cases}$	$\begin{cases} q(t+1) = f(q(t), a(t)) \\ b(t) = g(q(t+1)) \end{cases}$

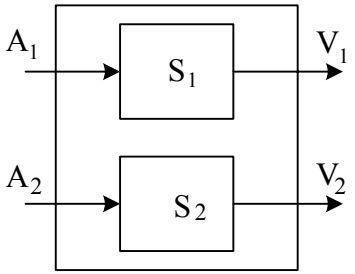
При работе по первой схеме выход автомата однозначно соответствует состоянию, из которого совершается переход, по второй – состоянию, в которое автомат переходит.

Алгоритм построения по автомату Мили эквивалентного автомата Мура (работающего по второй схеме) выглядит следующим образом.

1. *Расщепляем состояния (Общий случай: в некоторые вершины q_k ведут дуги, помеченные разными символами. В этом случае все такие вершины q_k расщепляются на множество вершин (расщепление состояния), помечаемых символами $<q_k,b_j>$). Если все дуги, ведущие в некоторое состояние q_k, имеют одинаковые выходные пометки b_s, то эта пометка просто переносится на это состояние.*
2. *Вводим дополнительное входное состояние (если входное состояние расщепилось), в которое не ведет ни одна дуга.*
3. *Строим дуги, соответствующие дугам исходного автомата.*
4. *Переносим выходные символы на соответствующие состояния.*
5. *Переобозначаем состояния.*

Виды соединения автоматов:

1. *Параллельное соединение*
   а) С разделительными входами и алфавитами A_1 и A_2:

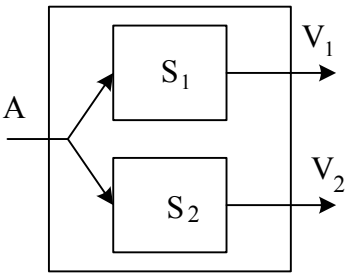


$S = <A, Q, V, q_0, F, G>$

В этом случае входной алфавит $A = A_1 \times A_2$, внутренний алфавит $Q = Q_1 \times Q_2$, выходной алфавит $V = V_1 \times V_2$, $q_0 = (q_0^1, q_0^2)$. S называется прямым произведением автоматов S_1 и S_2. В этом случае $a = (a^1, a^2)$ (Здесь верхний индекс

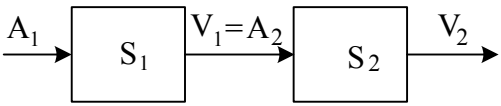
означает отнесение к соответствующему алфавиту). Функция переходов $f((q^1,q^2), (a^1,a^2))=(f_1(q^1,a^1),f_2(q^2,a^2))$. Мы рассмотрели случай, соединения двух автоматов, в результирующем автомате два входа и два выхода. Аналогично может быть построено соединение произвольного числа автоматов.

b) С общим входом и алфавитом A



В этом случае $f((q^1,q^2), a)=(f_1(q^1,a),f_2(q^2,a))$. Определение выходов в обоих случаях очевидно.

2. Последовательное соединение автоматов:



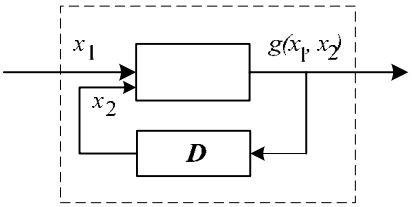
$S =<A, Q,V, q_0, F,G>$, $A=A_1$, $V=V_2$, $V_1=A_2$, $Q= Q_1 \times Q_2$.

Для F и G существенна задержка g_1.

Если задержка g_1 равна 0, т.е. $g_1(q^1(t), a(t))=v^1(t)$, то $q(t+1)=(q^1(t+1),q^2(t+1))= (f_1(q^1(t),a(t)), f_2(q^2(t), g_1 (q^1(t),a(t)))$, т.е. зависимость существует только от q(t), a(t), при этом состояние $q(t+1)=f(q(t),a(t))$, выход $g((q^1,q^2),a)=g_2(q^2,g_1(q^1,a))$.

Если же задержка g_1 равна 1, т.е. $g_1(q^1(t), a(t))=v^1(t+1)$, то $q(t+1)= (f_1(q^1(t),a(t)), f_2(q^2(t), g_1 (q^1(t-1),a(t-1))))$, и такой простой зависимости, как для прошлого случая, нет.

3. Соединение автоматов с обратной связью. Общая схема:



Алгоритм минимизации автомата Мили (метод Хаффмана).

1. *Составляем таблицы переходов и выходов. По строкам указываются входы, по*

столбцам – состояния, в первом подстолбце каждого столбца указаны переходы, во втором – соответствующие выходы.

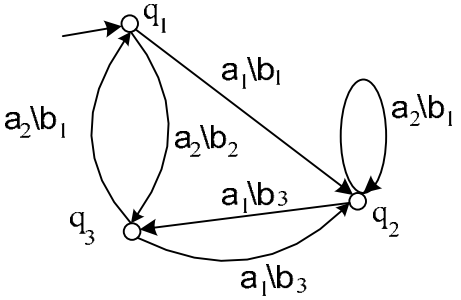
2. Составляем классы предположительно эквивалентных состояний (первоначально объединяем в классы эквивалентности состояния, у которых для одинаковых входов одинаковые выходы).

3. Составляем таблицу переходов, в которой вместо состояний указываются классы, которым принадлежат состояния. Если в пределах одного класса состояния ведут себя по-разному (им соответствуют разные столбцы в таблице), то класс разбивается на соответствующие подклассы, и возвращаемся к шагу 3.

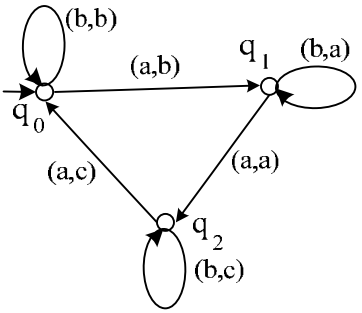
4. Если во всех классах состояния ведут себя одинаково, то это действительно классы эквивалентности. Строится автомат, состояниями которого являются классы эквивалентности исходного автомата.

Задачи.

1. Построить конечный автомат Мили, сдвигающий последовательность из 0 и 1 на 1 позицию, добавляя в начале последовательности 0. Например, для входа: 001110…, выход: -00111…
2. Построить автомат Мура для предыдущей задачи (рассмотреть 2 варианта печати)
3. Построить автомат Мура (2-я схема работы) по заданному автомату Мили

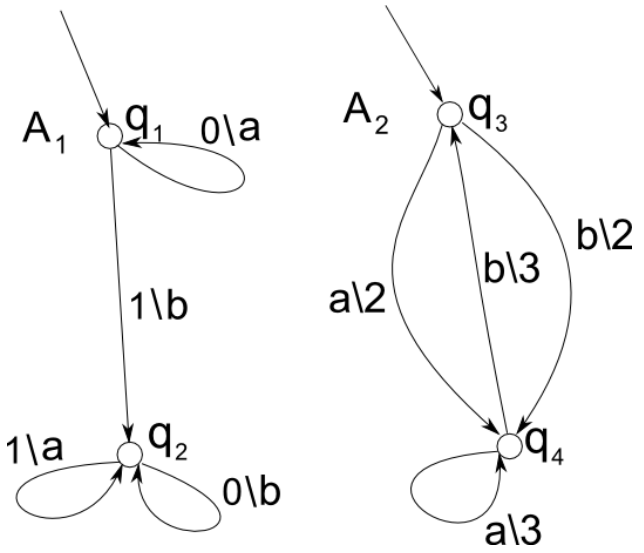


4. Автомат Мили задан диаграммой.

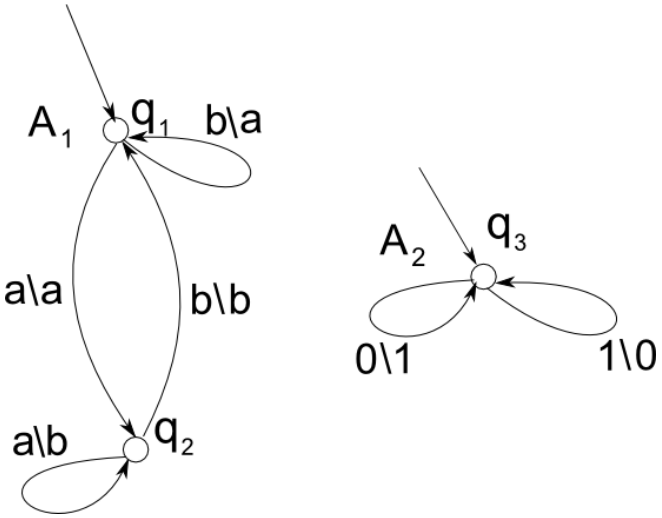


Построить автоматное отображение для цепочки *abbbaaab*. Определить автоматное отображение для автомата.

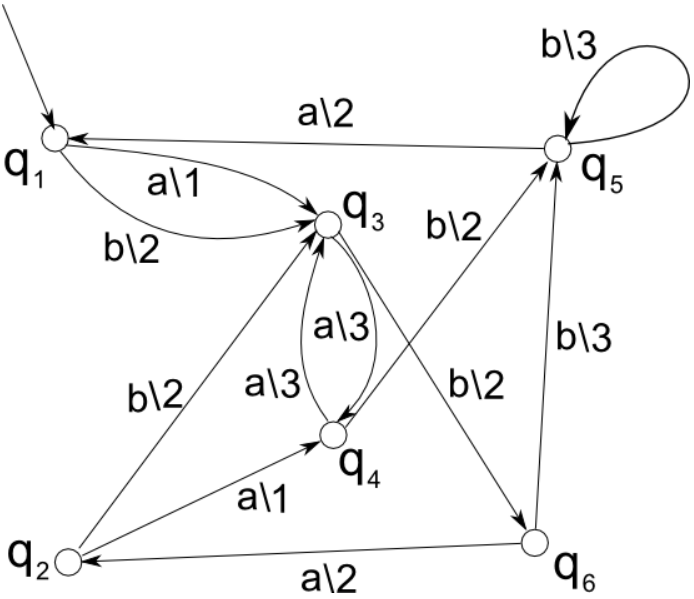
5. Построить автомат Мили, с входным алфавитом {0, 1, q}, который сохраняет в выходной последовательности 0 и 1, а при появлении q печатает ***ч*** или ***н*** в зависимости от чётности числа считанных к этому моменту единиц (если число единиц чётно, печатает ***ч***, и если нечётно – ***н***).
6. Построить автомат Мили, который каждую единицу, стоящую на чётном месте, заменяет на 0, если перед ней стояла 1, и сохраняет значения во всех остальных случаях.
7. Построить автомат Мура для предыдущей задачи.
8. Построить автомат Мили с входным алфавитом {a, b, c}, который заменяет все вхождения слова ababc на вхождения слова ababf, сохраняя все остальные символы.
9. Построить автомат Мили с входным алфавитом {a, b, c}, который заменяет все вхождения слова abbabc на вхождения слова abbabf, сохраняя все остальные символы.
10. Построить автомат A, последовательно соединив автоматы A_1 и A_2.



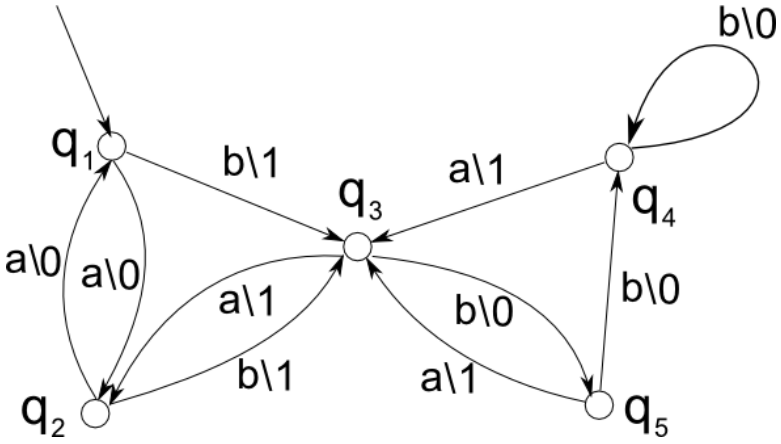
11. Построить автомат A как параллельное соединение автоматов A_1 и A_2.



12. Минимизировать число состояний полностью определённого автомата по методу Хаффмана.



13. Минимизировать число состояний автомата Мили.



14. Автомат Мили задан совмещённой таблицей переходов и выходов.

	A	B	C	D	E	F
a	D, b	E, c	F, b	B, b	E, c	E, b
b	E, c	C, b	B, c	F, c	A, b	D, c

Построить эквивалентный минимальный автомат.

15. Автомат Мили задан совмещённой таблицей переходов и выходов.

	A	B	C	D	E	F	G	H	I
a	B, a	A, b	A, b	H, a	F, b	H, a	F, b	D, b	G, a
b	D, b	A, a	F, a	A, b	D, b	I, b	A, b	D, a	I, b
c	D, b	E, a	E, a	A, b	C, a	F, b	C, a	G, a	G, b

Построить эквивалентный минимальный автомат.

16. Построить двоичный сумматор: автомат, которому на вход подаются, начиная с младшего разряда, два двоичных числа, а на выходе получается сумма этих чисел (считаем, что разрядность чисел одинакова, и в конце выдаются нули, чтобы сумма была правильной). Обосновать невозможность суммирования со старших разрядов.
17. Построить автомат Мили, прибавляющий к двоичному числу 3 (предполагается, что после введения числа будет введено достаточное число нулей для обеспечения правильности суммирования). Минимизировать число состояний автомата.
18. Построить автомат Мили, осуществляющий умножение двоичного числа на 3 (предполагается, что в после введения числа будет введено не менее двух нулей, что обеспечит правильность отработки единиц,

переносимых в старшие разряды). Произвольным образом закодировать состояния автомата. Построить логические функции переходов и выхода автомата.

Тема 10. Элементы теории кодирования

Теория. Пусть задан алфавит $V_b=\{b_1, b_2, \ldots b_n\}$. Пусть B – слово в алфавите V_b, $S'(V_b) \subseteq V_b^+$. Пусть задано отображение $F: S'(V_a) \rightarrow S'(V_b)$. Такое отображение ставит в соответствие произвольному слову A в алфавите V_a слово B в алфавите V_b. Слово B называется кодом сообщения A, а переход от слова A к слову B – кодированием. Отображение F задаётся некоторым алгоритмом. Алгоритм кодирования должен обеспечивать однозначное восстановление исходного сообщения по сообщению на выходе канала связи.

Основными типами кодирования являются следующие.

1. *Алфавитное. Это кодирование устанавливает взаимнооднозначное соответствие между буквами алфавита V_a и словами алфавита V_b. Это соответствие называют схемой кодирования. Слова, соответствующие буквам алфавита V_a, называются элементарными кодами.*
2. *Равномерное кодирование. Источник сообщений выдаёт слова одинаковой длины, при этом каждое слово однозначно раскладывается на элементарные коды. Каждому слову ставится в соответствие слово, называемое кодом слова.*

Рассмотрим алфавитное кодирование для $V_a=\{a_1,a_2,\ldots,a_k\}$.

Общий вид схемы кодирования Σ:

$a_1 – B_1$,

$a_2 – B_2$,

…

$a_k - B_k$

Здесь B_i $(i \in \{1,2,...k\})$ *- элементарные коды.*

Префиксом называется непустое начало слова. Схема кодирования Σ *обладает свойством префикса, если для любых* i , j $(1 \leq i, i \leq r, i \neq j)$ *элементарный код* B_i *не является префиксом элементарного кода* B_j. *Для схем кодирования, обладающих свойством префикса, может быть построено кодовое дерево. Кодовое дерево – дерево с корнем, ребрам которого приписаны буквы кодового алфавита таким образом, что путь от корня до висячей вершины соответствует элементарному коду. Висячей вершине приписывается соответствующая буква исходного алфавита.*

Теорема 1. *Если схема кодирования обладает свойством префикса, то алфавитное кодирование будет взаимнооднозначным.*

Теорема 2. *Если сама схема или её инвертирование обладает свойством префиксности, то алфавитное кодирование будет взаимнооднозначным.*

Критерий однозначности кодирования выглядит следующим образом:

Пусть алфавитное кодирование задано схемой Σ*:*

$a_1 - B_1$,

...

$a_n - B_n$

Для каждого элементарного кода B_i *рассмотрим все разложения вида* $B_i = \beta' Bi_1...Bi_m\beta''(*)$, *в которых* β', β'' *отличны от элементарных кодов. Тривиальные разложения* $B_i = \lambda\text{-}B_i\text{-}\lambda$ *не рассматриваются. Строим множество* ***К****, содержащее пустое слово и все* β, *присутствующие в разложениях вида* * *как в виде префиксов, так и в виде окончаний. Строится граф* $\Gamma(\Sigma)$. *Для каждого*

*нетривиального разложения вида * строится дуга из $\beta' \beta''$, помеченная $Bi_1 ... Bi_m$.*

***Теорема 3**. Алфавитное кодирование со схемой Σ не обладает свойством взаимной однозначности тогда и только тогда, когда $\Gamma(\Sigma)$ содержит ориентированный цикл, проходящий через вершину λ.*

***Теорема 4.** Если алфавитное кодирование обладает свойством взаимной однозначности,*

то $\sum_{i=1}^{n} \frac{1}{q^{l_i}} \le 1$ *(Неравенство Макмиллана).*

***Теорема 5**. Если числа l_i удовлетворяют неравенству Макмиллана, то существует алфавитное кодирование, Σ*

$a_1 - B_1$,

...

$a_n - B_n$,

обладающее свойством префикса, такое, что $l_i = l(B_i)$.

Определим среднюю длину элементарного кода $l_{ср} = \sum_{i=1}^{n} p_i l_i$,

где $l = l(B_i)$.

Для данного источника сообщений можно ввести величину l^*, где $l^* = \inf_{\Sigma} l_{ср}^{\Sigma}$.

Коды, определяемые схемой Σ с $l_{ср} = l^*$, называются кодами с минимальной избыточностью, или кодами Хаффмана.

Алгоритм построения схемы с минимальной избыточностью для двоичного кодирования

Все вероятности упорядочиваются в порядке невозрастания. Затем суммируются две минимальные вероятности. Получившиеся вероятности опять

упорядочиваются в порядке невозрастания. Процедура повторяется до получения двух величин. В результате будет построено дерево кодирования, которое и разворачивается в код с минимальной избыточностью.

Пусть кодовый алфавит состоит из двух букв $V_b=\{0, 1\}$. $A=\{A_1, A_2, \ldots A_n\}$ *- множество различных слов фиксированной длины* m ($n \le 2^m$), $A_i \in V_b^+$.
Рассматривается единичный m*-мерный куб как метрическое пространство. Расстояние* $\rho(b', b'')$ *между двумя векторами* $b'=(b'_1 b'_2 \ldots b'_m)$ *и* $b''=(b''_1 b''_2 \ldots b''_m)$ *определяется следующим образом :* $\rho(b', b'') = \sum_{i=1}^{m} |b'_i - b''_i|$, *т.е. число координат, в которых различаются вектора. Для коррекции* p *ошибок минимальное расстояние между двумя кодами должно быть* $\geq 2p+1$.

Коды Хемминга разработаны для обнаружения и исправления ошибок. Структура кода такова: все разряды кода, номера которых – степени двойки, являются контрольными, остальные разряды – смысловыми. Сначала заполняются смысловые разряды кода, затем строятся контрольные разряды. Контрольные разряды строятся следующим образом.
$b_1=b_3 \oplus b_5 \oplus b_7 \oplus \ldots \oplus b_{2n+1} \oplus \ldots$, *т.е. суммируются все нечётные разряды, кроме первого.*
$b_2=b_3 \oplus b_6 \oplus b_7 \oplus \ldots \oplus b_k \oplus \ldots$, *где* k *– числа, имеющие единицу во втором разряде двоичного кода (кроме самой двойки).*
$b_4=b_5 \oplus b_6 \oplus b_7 \oplus \ldots \oplus b_k \oplus \ldots$, *где* k *– числа, имеющие единицу в третьем разряде двоичного кода(кроме самой 4)...*
$b_n=\ldots \oplus b_k \oplus \ldots$ *где* $n= 2^j$, k *– числа, имеющие единицу в* j*-ом разряде двоичного кода (кроме самого n) .*
Для кода Хемминга при одной ошибке адрес ошибочного бита определяется следующим образом:

пусть $s_1=b_1 \oplus b_3 \oplus b_5 \oplus b_7 \oplus \ldots \oplus b_{2n+1} \oplus \ldots$

$s_2=b_2 \oplus b_3 \oplus b_6 \oplus b_7 \oplus \ldots \oplus b_k \oplus \ldots$, *где k – числа, имеющие единицу во втором разряде двоичного кода,*

$s_4=b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus \ldots \oplus b_k \oplus \ldots$, *где k – числа, имеющие единицу в третьем разряде двоичного кода,...*

$s_n=\ldots \oplus b_k \oplus \ldots$ *где* $n=2^j$, *k – числа, имеющие единицу в j-ом разряде двоичного кода.*

$S'=s_n \ldots s_4 s_2 s_1$ *– число, которое определяет адрес ошибочного бита.*

Если $s_1=0$, *то S' не принадлежит первой последовательности, если* $s_1=1$, *то S' принадлежит первой последовательности. Аналогично можно рассмотреть все остальные разряды.*

Если S'=0, то ошибки нет, иначе бит с адресом S' заменяется на своё дополнение.

Задачи.

1. Выяснить, обладает ли код С свойством префикса.
   a) C={a, ba, bb, bbba}
   b) C={ab, ba, bb, aab}
   c) C={ac, c, bb, abc, bac, abb, abc}
   d) C={a, ba,bba,…b^na,….}
   e) C={a, ab, abb,… ab^n,….}

   Код, инверсный коду С?
2. Выяснить, является ли в алфавите {0,1,2} при заданной схеме кодирования:

$$\sum : \begin{cases} 1-10 \\ 2-12 \\ 3-012 \\ 4-101 \\ 5-2100 \end{cases}$$

кодом сообщения слово Р.

a) 10120121012100
b) 1012101201210012
c) 0121001210201
d) 120120121001210
e) 1010122100
f) 12101210012
g) 101212101012
h) 1010012100101

3. Является ли однозначно декодируемым код:
   a) {01,201,112,122,0112}
   b) {001,021,102, 001121,01012101}
   c) {0,01,0010001001}
   d) {20,01202,22,2002,2012010,10201121,1112}
   e) {01,011,100,2100,10110,00112}
   f) {01,011,100,2100,101210,001210}
4. Построить двоичный префиксный код с последовательностью длин
   a) 1,2,3,3
   b) 1,2,4,4,4
   c) 2,2,3,3,4,4,4,4
   d) 2,2,2,4,4,4
   e) 2,2,3,4,4
   f) 2,3,3,3,4,4
5. Можно ли построить однозначный код в q-значном алфавите с длинами элементарных кодов:
   a) 1,2,2,3 q=2
   b) 1,2,2,3 q=3
   c) 2,2,2,4,4,4 q=2
   d) 1,2,2,2,3,3,3,3 q=3
   e) 1,1,2,2,3,3,3 q=3
   f) 1,1,1,2,2,2,2,3 q=4
6. Определить кодовое расстояние для системы кодов. Определить число ошибок, обнаруживаемых при

данной схеме кодирования, и число исправляемых ошибок:

a) C={11000, 10101, 01110}
b) C={111100, 110011,001111}
c) C={00001, 11111, 10100, 01010}
d) C={101010, 010110, 000001}
e) C={01101010, 11000110, 00011001, 1010110}

7. Определить оптимальный двоичный код для набора вероятностей:
   a) {0,4; 0,2; 0,2; 0,2}
   b) {0,7; 0,1; 0,1; 0,1}
   c) {0,2; 0,2; 0,2; 0,2; 0,2}
   d) {0,5; 02; 0,1; 0,09; 0,08; 0,03}
8. Определить набор вероятностей для оптимального двоичного кода с длинами:
   a) {1, 2, 3, 4, 5, 5}
   b) {2, 2, 2, 3, 3, 3}
   c) {2, 2, 3, 3, 4, 4}
   d) {1, 2, 4, 4, 4, 4}
9. Построить коды Хемминга для сообщений **0110, 1001, 1100, 1011 и 1111.**
10. Проверить корректность полученного сообщения и исправить при необходимости единичную ошибку в сообщении **0100100.** Выделить смысловую часть сообщения.
11. Проверить корректность полученного сообщения и исправить при необходимости единичную ошибку в сообщении **0110110.** Выделить смысловую часть сообщения.
12. Проверить корректность полученного сообщения и исправить при необходимости единичную ошибку в сообщении **1100101.** Выделить смысловую часть сообщения.

Ответы и указания

Тема 1. Языки, способы задания, операции над языками

4. Начала слова abcdc: abcdc, abcd, abc, ab, a, λ. Концы слова abcdc: abcdc, bcdc, cdc, dc, c, λ.

5. Длины начал $|abcdc|=5$, $|abcd|=4$, $|abc|=3$, $|ab|=2$, $|a|=1$, $|\lambda|=0$. Концы слова abcdc: $|abcdc|=5$, $|bcdc|=4$, $|cdc|=3$, $|dc|=2$, $|c|=1$, $|\lambda|=0$.

6. $xy=bbcac$, $(xy)^2=bbcacbbcac$, $x^2y^2=bbcbbcacac$, $y^3=acacac$, $xyz=bbcacb$, $xzy=bbcbac$, $xz^2y=bbcbbac$.

15. $L_1 \cap L_2 =\{ba\}$, $L_1 \cup L_2 =\{ ab, ba, aa, bb\}$, $L_1 \setminus L_2=\{ab\}$, $L_2 \setminus L_1=\{aa, bb\}$.

16. $L_1 L_2=\{abaa, abbb, baaa, babb\}$, $L_2L_1=\{aaab, aaba, bbab, bbba\}$, $L_1^2=\{abab, abba, baab, baba\}$, $L_2^2=\{aaaa, aabb, bbaa, bbbb\}=\{a^4,a^2b^2, b^2a^2,b^4\}$.

17. Для $L_1=\{ab, ba\}$ $L_1^3=\{ababab, ababba, abbaba, abbaab, baabab, baabab, bababa, bababa\}$. $L_1^n=\{x_1x_2\ldots x_n, x_i\in\{ab, ba\}, i\in[1,n]\}$. $L_1^+=\{x_1x_2\ldots x_n, x_i\in\{ab, ba\}, i\in[1,n], n\geq 1\}$, $L_1^+=\{x_1x_2\ldots x_n, x_i\in\{ab, ba\}, i\in[1,n], n\geq 0\}$.

24. Да, такой язык $\boldsymbol{L}$ существует, это $\boldsymbol{L}=\varnothing$.

25. Пусть $|\boldsymbol{L_1}|=\boldsymbol{n}$, $|\boldsymbol{L_2}|=\boldsymbol{m}$. Тогда $|\boldsymbol{L_1L_2}|\leq \boldsymbol{n\ m}$.

26. Пусть $|\boldsymbol{L_1}|=\boldsymbol{n}$, $\boldsymbol{n>0}$, $|\boldsymbol{L_2}|=\boldsymbol{m}$, $\boldsymbol{m>0}$. Тогда $|\boldsymbol{L_1L_2}|=\boldsymbol{n\ m}$ при условии, что $\forall i\ \forall j\ \forall k\ \forall p\ ((i\neq k)\&(j\neq p)\&(x_i\in \boldsymbol{L_1})\&(y_j\in\boldsymbol{L_2})\&(x_k\in \boldsymbol{L_1})\&(y_p\in\boldsymbol{L_2})\Rightarrow x_iy_j\neq x_ky_p)$, что означает, что никакое слово в конкатенации не может быть получено двумя различными способами.

29. a) (b+c)*(1+a)(b+c)*

b) c*a(a+c)*b(a+b+c)*+c*b (b+c)*a(a+b+c)*.

Тема 2. Порождающие грамматики

3. $L(G_1)=aa^*bb^*cc^*$, $L(G_2)= a^*(b+1)cc^*$.

8. Указание: Язык $L_1 = \{a^n b^m c^m;\ n, m \geq 1\}$ можно представить как конкатенацию двух языков: $L_A = \{a^n;\ n \geq 1\}$ и $L_B = \{b^m c^m;\ m \geq 1\}$. Пусть язык L_A порождается грамматикой с начальным символом A и множеством правил R_A, а язык L_B – грамматикой с начальным символом B и множеством правил R_B (предполагаем, что правила грамматик R_A и R_B не имеют общих нетерминалов). Тогда грамматика с начальным символом S и множеством правил $\{ S \to AB\} \cup R_A \cup R_B$ будет порождать язык $L_1 = \{a^n b^m c^m;\ n, m \geq 1\}$.

10. Грамматика порождает язык $\{a^n b^n a^n,\ n>0\}$.

11. Грамматика порождает язык $\{a^n b^n c^n,\ n>0\}$.

12. Указание: произвести замену «правого» терминала a на терминал c в правилах грамматики задачи 10.

13. Указание: Язык $L = \{(001)^n; (010)^n;\ n \geq 2\}$ можно представить как объединение двух языков: $L_1 = \{(001)^n;\ n \geq 2\}$ и $L_B = \{(010)^n;\ n \geq 2\}$. Пусть язык L_1 порождается грамматикой с начальным символом A и множеством правил R_1, а язык L_2 – грамматикой с начальным символом B и множеством правил R_2 (предполагаем, правила грамматик используют различные нетерминалы). Тогда грамматика с начальным символом S и множеством правил $\{ S \to A \mid B\} \cup R_A \cup R_B$ будет порождать язык $\{(001)^n; (010)^n;\ n \geq 2\}$.

15.a) Грамматика должна порождать последовательности десятичных цифр, не начинающиеся с нуля и оканчивающиеся нулём. Например, можно использовать грамматику с правилами:

$$\begin{cases} S \to 0 \mid 1A0 \mid 2A0 \mid 3A0 \mid 4A0 \mid 5A0 \mid 6A0 \mid 7A0 \mid 8A0 \mid 9A0 \\ A \to \lambda \mid 0A \mid 1A \mid 2A \mid 3A \mid 4A \mid 5A \mid 6A \mid 7A \mid 8A \mid 9A \end{cases}$$

b) Указание: Для того, чтобы полученное число делилось на 3, достаточно, чтобы сумма чисел этого числа делилась на 3. Грамматика, порождающая такие числа, подобна

грамматике пункта а), но надо учитывать остаток от деления на 3 суммы цифр построенного числа. Например, правила для начального нетерминала и нетерминала А, соответствующего сумме цифр, дающей при делении на 3 остаток 2. Правила для нетерминалов В и С строятся аналогично.

$$\begin{cases} S \to 0|3|6|9|1A|2B|3C|4A|5B|6C|7A|8B|9C \\ A \to 2|0A|1B|2C|3A|4B|5C|6A|7B|8C|9A|5|8 \end{cases}$$

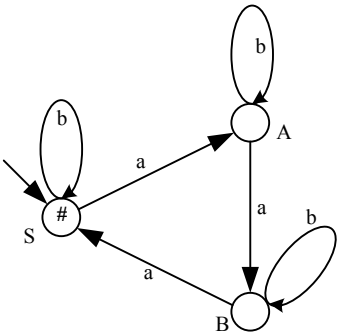
20. с) Указание: Язык $\{a^n b^n c^n d^m, n \geq 1, m \geq 1\}$ является конкатенацией языков $\{a^n b^n c^n, n \geq 1\}$ и $\{d^m, m \geq 1\}$, грамматики для которых уже известны.
е) Например, можно использовать грамматику с правилами:

$$G_1 = \begin{cases} S \to aBSCd|abcd \\ aB \to Ba \\ Bb \to bb \\ dC \to Cd \\ cC \to cc \end{cases}$$

25. Грамматика принадлежит классу КЗ-грамматик.
26. е) Язык L= $\{a^n b^n c^m d^m, n \geq 1, m \geq 1\}$ можно представить как конкатенацию языков $\{a^n b^n, m \geq 1\}$ и $\{c^m d^m, m \geq 1\}$. Построение грамматики очевидно.

Тема 3. А-грамматики, конечные автоматы

1. а) $S \Rightarrow aA \Rightarrow abA \Rightarrow abaB \Rightarrow abaaS \Rightarrow abaabS \Rightarrow abaab$
b) Диаграмма грамматики:



d) L=(b*ab*ab*a)*b*

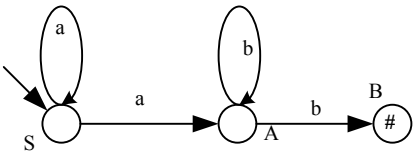
e) S=<{q_0, q_1, q_2}, {a, b}, q_0, F, {q_0} >,

F:

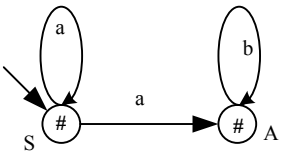
	q_0	q_1	q_2
a	q_1	q_2	q_0
b	q_0	q_1	q_2

f) (q_0, abaab) ⊢ (q_1,baab) ⊢ (q_1,aab) ⊢ (q_2,ab) ⊢ (q_0,b) ⊢ (q_0, λ)

4. a) Диаграмма грамматики:



d) Диаграмма грамматики:



5. а) Указание: определить число нетерминалов грамматики, порождающей язык.

7. а) F:

	q_0	q_1	q_2
a	q_0	q_1	-
b	q_1	q_2	-

Правила грамматики:

$$\begin{cases} S \to aS \mid bA \\ A \to bA \mid aB \\ B \to \lambda \end{cases}$$

8. а)Правила грамматики:

$$\begin{cases} S \to aS \mid aA \mid bB \\ A \to bA \mid bB \\ B \to \lambda \end{cases}$$

Детерминированный автомат:
S=<{q_0, q_1, q_2, q_3}, {a, b}, q_0, F, {q_2, q_3} >

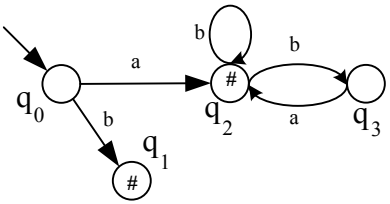
F:

	q_0	q_1	q_2	q_3
a	q_1	q_1	-	-
b	q_3	q_2	q_2	-

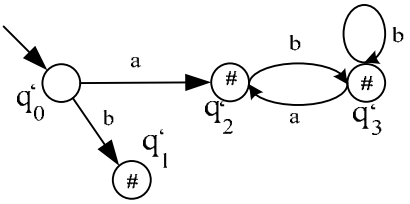
16. Множество цепочек регулярно, т.к. достаточно учитывать чётность-нечётность каждого типа букв.
17.

$$\begin{cases} X_S = aX_S + bX_A \\ X_A = aX_A + a \end{cases}$$

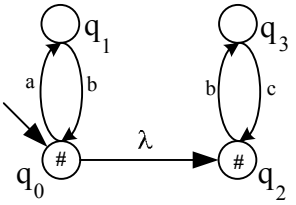
Откуда X_A=a*ba*a, X_B= a*a, следовательно, L(G)= a*ba*a.
21.a) Исходный недетерминированный автомат:



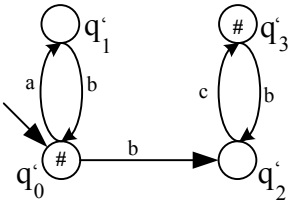
Соответствующий детерминированный автомат:



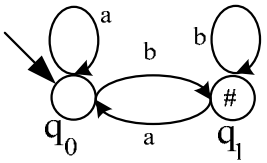
21. b) Исходный недетерминированный автомат:



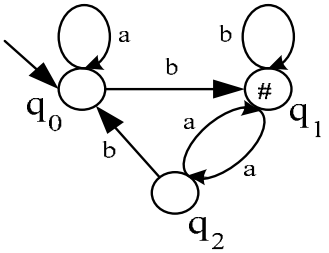
Соответствующий детерминированный автомат:



22. а) Диаграмма минимизированного автомата:



24. Диаграмма минимизированного автомата:



27. $(a+b+c)^*c(a+b+c)^2$
Правила А-грамматики, порождающей этот язык:

$$\begin{cases} S \to aS \,|\, bS \,|\, cS \,|\, cA \\ A \to aB \,|\, bB \,|\, cB \\ B \to a \,|\, b \,|\, c \end{cases}$$

Тема 4. Бекусовы (бекусовские) нормальные формы

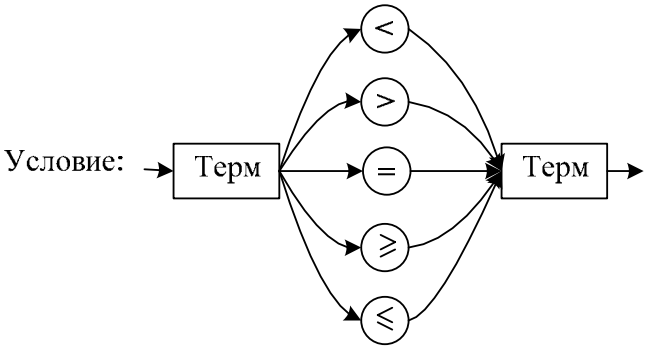
2. БНФ:
<A>::=<B>|<A><C>
<B>::=<C><D>|<B><D>
<C>::=<L><C>|<B>
<D>::= d |d <B><D>
<L>::=a|b|c

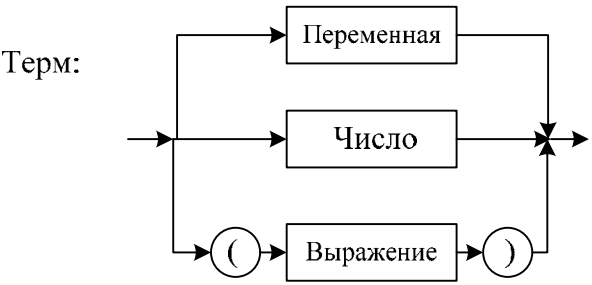
РБНФ:
A=B {C}.
B=C D{D}.
C={L} B.
D='d'{B'd'}.
L='a' |'b'|'c'.

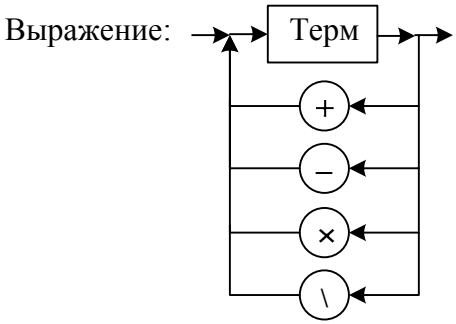
5. Указание. При построении надо учитывать силу связок: наиболее сильно связывает отрицание, слабее – конъюнкция, и самая слабая из этой группы – дизъюнкция.

7. РБНФ:
Условие = Терм {('<' |'>' |'=' |'≤' |'≥')}Терм.
Терм = Переменная |Число |(Выражение).
Выражение = {Терм ('+'|'-'|'×'|'\')}Терм

Синтаксическая диаграмма







11. БНФ:

<A>::=<B> | <B> c<A>
<D>::= <E> <F> | <D><F>

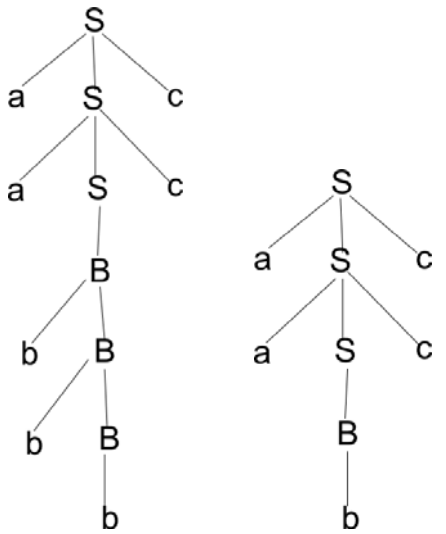
РБНФ: A= B{‘c’B}.

D = E F{ F }.

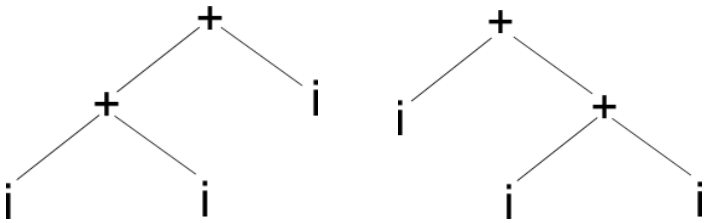
Тема 5. Структура цепочек. СУ-схемы

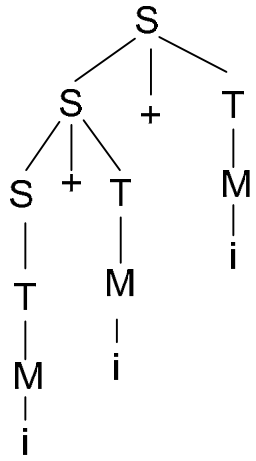
1. Переводами для цепочки i*i+(i+i)*i с помощью СУ-схем T1, T2 будут ii*ii+i*+ и [[i*i]+[[[i+i]]*i]] соответственно.

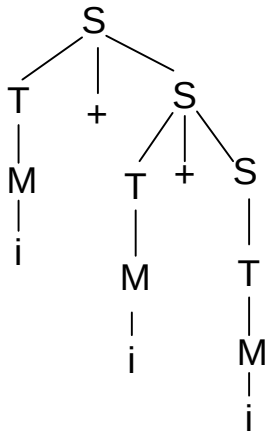
4. Деревья для цепочек $a^2b^3c^2$ и ab^2c выглядят так:



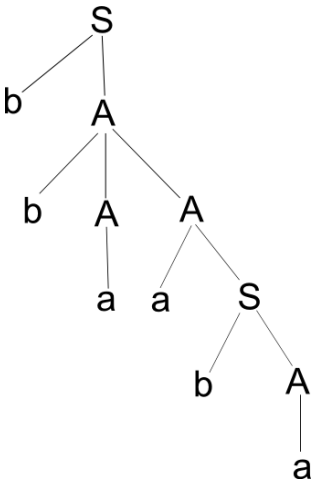
6. Сокращённые и полные деревья для цепочки i+i+i в грамматиках G_1, G_2 выглядят как:

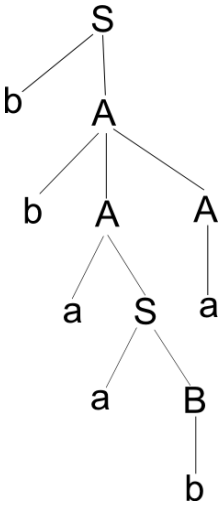






9. Грамматика неоднозначна. Например, для цепочки bbaaba могут быть построены два синтаксических дерева:





10. Примером грамматики, удовлетворяющей условиям, может служить:

$$G:\begin{cases} S \to T \supset S \mid T \\ T \to T \vee K \mid K \\ K \to K \wedge M \mid M \\ M \to \neg M \mid N \\ N \to (S) \mid i \mid t \mid f \end{cases}$$

Тема 6. Эквивалентные преобразования КС-грамматик

4. После преобразования получается грамматика G_1, кроме того, можно упростить грамматику, заметив, что у нетерминалов S и A одинаковые правые части. Получится грамматика G_2.

$$G_1:\begin{cases} S_1 \to S \mid \lambda \\ S \to AB \mid SA \mid BB \mid bB \mid b \mid aA \mid a \\ A \to AB \mid SA \mid BB \mid bB \mid b \mid aA \mid a \\ B \to b \mid aA \mid a \end{cases};$$

$$G_2:\begin{cases} S_1 \to S \mid \lambda \\ S \to SB \mid SS \mid BB \mid bB \mid b \mid aS \mid a \\ B \to b \mid aA \mid a \end{cases}$$

5. После преобразования получим следующие эквивалентные грамматики:

$$G_1:\begin{cases} S \to aS \mid aE \mid a \\ C \to cC \mid c \\ E \to bC \mid cC \mid b \mid c \end{cases}; \quad G_2:\begin{cases} S_1 \to S \mid \lambda \\ S \to bB \mid BB \mid b \mid c \\ B \to bB \mid b \end{cases}$$

Тема 7. LL(k), строго LL(k)-грамматики

1. а). Цепочки (+(и (ii(выводятся следующим образом:

$$S \Rightarrow V_1 \Rightarrow V_2 \Rightarrow V_2 + V_3 \Rightarrow V_3 + V_3 \Rightarrow (+V_3 \Rightarrow (+($$

$$S \Rightarrow V_1 \Rightarrow V_1 i V_2 \Rightarrow V_1 i i V_3 \Rightarrow V_2 i i V_3 \Rightarrow V_3 i i V_3 \Rightarrow (i i V_3 \Rightarrow (i i($$

b)

	S	V_1	V_2	V_3
$First_1$	i; (;)	i; (;)	i; (;)	(;)
$First_2$	(;)i;)(;)); i); i(; (+; (i	(;)i;)(;)); i); i(; (+; (i	(;)i;)(;)); i); i(; (+	(;)i;)(;))

	S	V_1	V_2	V_3	(	)	*	+	i
$Follow_1$	$	$; i; *	$; i; +; *	$; i; +; *	$; i; +; *	i; (;)	$; i; +; *	); (	); (; i

4. Подобная грамматика может выглядеть следующим образом:

$$G: \begin{cases} S \to +TS \mid T \\ T \to *MT \mid M \\ M \to a \mid b \mid c \mid S \end{cases}$$ и она не будет являться LL(k) ни для какого k.

5. Будет являться LL(2) –грамматикой и строго LL(3)-грамматикой.

7. Является LL(1)-грамматикой.

9. Не является LL(1)-грамматикой.

Тема 8. Грамматики простого предшествования (ГПП), восходящий анализ

1. G_1 не будет являться ГПП, так как в грамматике присутствует λ-правило. G_2 не будет являться ГПП, так как в

грамматике, например, между A и c может быть определено больше одного отношения предшествования (A <∘ c, A ∘> c).
2. Данная грамматика является ГПП со следующей таблицей отношений предшествования:

	S	R	(	a	)	$
S				≗		≗
R				∘>		∘>
(	<∘	≗	<∘	<∘		
a				∘>	≗	∘>
)				∘>		∘>
$	≗		<∘	<∘		

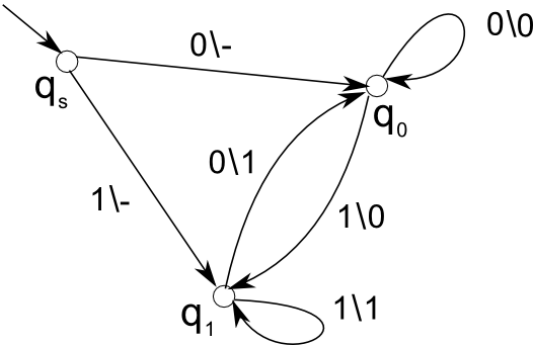
Свёртка цепочки (aa) осуществляется следующей последовательностью действий:

	стек	отношение	Входная строка	действие
1	$	<∘	(aa)$	перенос
2	$(	<∘	aa)$	перенос
3	$(a	∘>	a)$	свёртка
4	$(S	≗	a)$	перенос
5	$(Sa	≗	)$	перенос
6	$(Sa)	∘>	$	свертка
7	$(R	∘>	$	свертка
8	$S	≗	$	перенос
9	S			конец

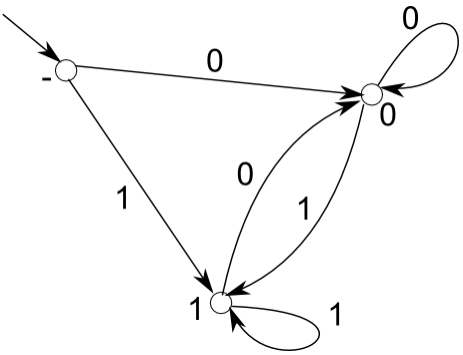
5. Не является, так как между некоторыми парами символов можно определить более одного отношения предшествования.

Тема 9. Конечные автоматы Мили и Мура

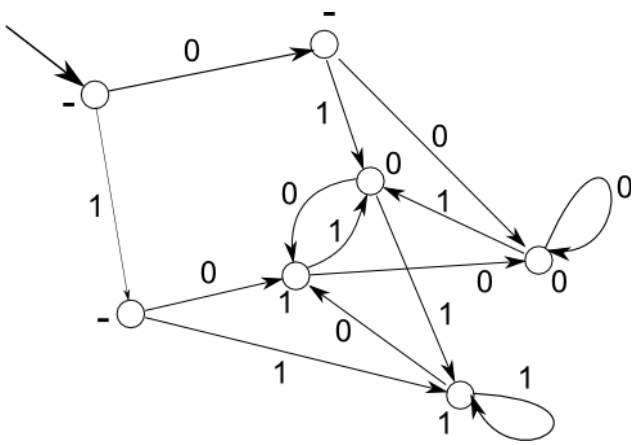
1. Диаграмма автомата Мили:



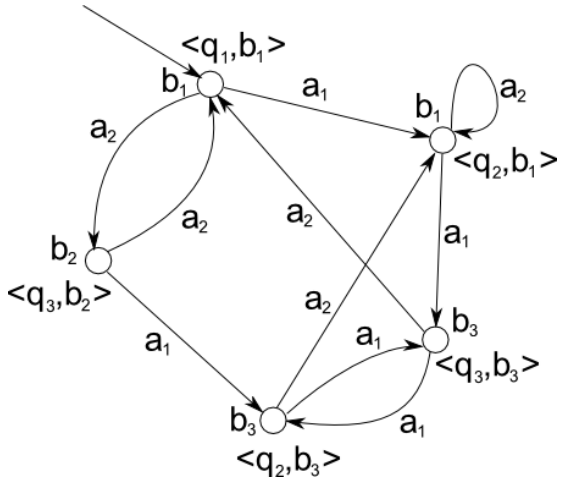
2. а). Диаграмма автомата Мура, работающего по первой схеме:



b). Диаграмма автомата Мура, работающего по второй схеме:

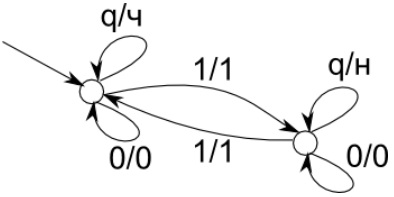


3. Диаграмма результирующего автомата Мура:

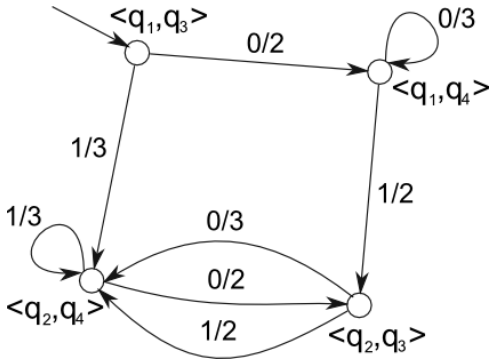


4. Автоматным отображением для цепочки abbbaaab будет цепочка abbbacab. Данный автомат каждую третью а заменяет на с, сохраняя все остальные символы.

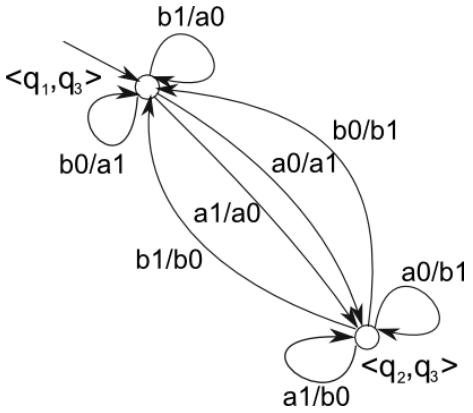
5. Диаграмма автомата:



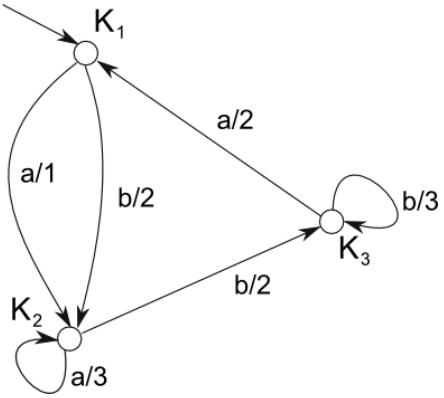
10. Диаграмма результирующего автомата:



11. Диаграмма результирующего автомата:



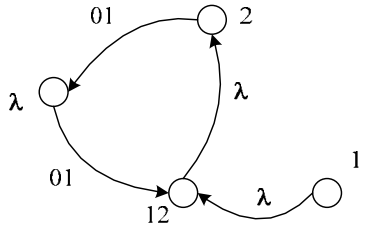
12. Диаграмма минимального автомата:



Тема 10. Элементы теории кодирования

1. a) Код не является префиксным, т.к. элементарный код bb является префиксом элементарного кода bbba. Инверсный код так же не является префиксным.
b) Код обладает свойством префикса, инверсный код свойством префикса не обладает.
c) Код обладает свойством префикса, а его инверсия не обладает.

d) Код обладает свойством префикса, а его инверсия не обладает.
e) Код не обладает свойством префикса, а его инверсия обладает.
3. a) Разложения вида * для элементарных кодов:
B_1: 0-λ-**1**; B_2:**2**-01-λ=20-λ-1; B_3: **1**-λ-**12**=11-λ-**2**; B_4: **12**-λ-**2**= **1**-λ-22; B_5: 0-112-λ=λ-01-**12**.
Множество K={λ, 1, 12, 2}. Граф Γ(Σ):



В графе существует цикл, проходящий через λ, поэтому кодирование не является однозначным. Неоднозначно декодируемое слово: 0112201 =01-122-01=0112-201.
3. b) Является однозначно декодируемым.
3. c) Не является однозначно декодируемым.
4. a) Например, схема с элементарными кодами: {0, 10,110,111}
4. b) Например, схема с элементарными кодами: {0, 10,1101,1100,1111}
4. c) Например, схема с элементарными кодами: {01, 00,101,100,1100,1101,1110,1111}
5. a)1/2+1/4+1/4+1/8=1,125>1, следовательно, такую схему кодирования построить нельзя.
5. b)1/3+1/9+1/9+1/27<1, следовательно, схему кодирования построить можно.
6. a) Кодовое расстояние равно 3, код позволяет обнаружить и исправить одну ошибку.

6. b) Кодовое расстояние равно 4, код позволяет обнаружить две и исправить одну ошибку.
7. a) {0,10,110,111} или {00,01,10,11}.
7. b) {0,10,110,111}.
8. a) Например, {1/2,1/4,1/8, 1/16, 1/32, 1/32}
10. Для сообщения **0100100**
$s_1=b_1 \oplus b_3 \oplus b_5 \oplus b_7=0 \oplus 0 \oplus 1 \oplus 0=1$
$s_2=b_2 \oplus b_3 \oplus b_6 \oplus b_7=1 \oplus 0 \oplus 0 \oplus 0 =1$
$s_4=b_4 \oplus b_5 \oplus b_6 \oplus b_7 =0 \oplus 0 \oplus 1 \oplus 0=1$, S'=111, следовательно, ошибка в 7-ом разряде, исправленное сообщение **0100101,** смысловая часть сообщения **0101.**

Список литературы

1. Короткова М.А. Математическая теория автоматов. М.: МИФИ, 2008.
2. Короткова М.А. Лингвистические методы анализа и синтеза систем. М.: МИФИ, 2006
3. Сергиевский Г.М., Короткова М.А. Введение в математическую лингвистику и теорию автоматов. Конспект лекций. – М.: МИФИ, 2004.
4. КузнецовО.П. Дискретная математика для инженера. М.: Лань, 2007
5. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. М : Вильямс, 2002.
6. Яблонский С.В. Введение в дискретную математику. М.: Высшая школа, 2001.
7. Гаврилов Г.П., Сапоженко А.А. Задачи и упражнения по курсу дискретной математики. М.: ФМЛ, 2005.
8. Ахо А., Ульман Дж. Теория синтаксического анализа перевода и компиляции.. Пер. с англ. Т. 1,2 М.: Мир 1978.

Мария Александровна Короткова
Екатерина Евгеньевна Трифонова

Задачник по курсу
«МАТЕМАТИЧЕСКАЯ ЛИНГВИСТИКА
И ТЕОРИЯ АВТОМАТОВ»

Редактор Т.В. Волвенкова
Оригинал-макет изготовлен М.А. Коротковой

Подписано в печать 15.11.2011. Формат 60×84 1/16
Печ. л. 5,5. Уч.-изд. л. 5,5. Тираж 180 экз.
Изд. № 1/49. Заказ № 23.

Национальный исследовательский ядерный университет «МИФИ»,
115409, Москва, Каширское ш., 31.

ООО «Полиграфический комплекс «Курчатовский».
144000, Московская область, г. Электросталь, ул. Красная, 42