

## Содержание

1 Билет 1	3
2 Билет 2	5
3 Билет 3	5
4 Билет 4	6
5 Билет 5	7
6 Билет 6	7
7 Билет 7	7
8 Билет 8	8
9 Билет 9	9
10 Билет 10	12
11 Билет 11	12
12 Билет 12	12
13 Билет 13	13
14 Билет 14	13
15 Билет 15	14
16 Билет 16	14
17 Билет 17	16
18 Билет 18	18
19 Билет 19	19
20 Билет 20	21
21 Билет 21	24
22 Билет 22	25
23 Билет 23	25
24 Билет 24	25

<b>25 Билет 25</b>	<b>26</b>
<b>26 Билет 26</b>	<b>26</b>
<b>27 Лемма о накачке для регулярных языков</b>	<b>28</b>
<b>28 Эквивалентность регулярных языков</b>	<b>29</b>
<b>29 Построение произведения автоматов L и M</b>	<b>29</b>
<b>30 Проблема вложения регулярных языков</b>	<b>30</b>

# 1 Билет 1

## Общая структура компиляторов



### Лексический анализ

Лексический анализ — деление текста на слова, выделение токенов.

Задача лексического анализа — выделить лексемы, классифицировать их и передать их на стадию синтаксического анализа.

```
\tif (i == j)\n\t\tz = 0;\n\telse\n\t\tz = 1;
```

- Оператор
- Пробел
- Идентификатор
- Ключевое слово
- Число
- Спец. символы

```
| if (i==j)
|     z=0;
| else
|     z=1;
```

### Синтаксический анализ

Синтаксический анализ — определение структуры предложения.

Задача: иерархическая группировка в соответствии с грамматикой языка программирования.

**position := initial + rate \* 60**



## Семантический анализ

Семантический анализ — устранение неоднозначностей.

- Пример:

- Петя оставил её задание дома

- Из-за «несоответствия типов» между «её» и «Петя» мы узнаём, что это разные люди.

## Оптимизация кода

Оптимизация кода:

1. На естественном языке оптимизация не имеет строгих правил и сводится к редактированию;
2. Для программ она предполагает следующее:
  - (a) Увеличение скорости работы программы;
  - (b) Уменьшение объёма используемой памяти;
  - (c) И так далее.

## Генерация кода

Генерация кода — трансляция исходного кода на другой язык программирования.

Обычно результатом является ассемблерный код.

**Дополнительную информацию смотри на слайдах 1-39 первой половины лекций.**

## 2 Билет 2

## 3 Билет 3

### Определение алфавита

Алфавит — это конечное множество символов.

### Определение слова в $\Sigma$

Словом в алфавите  $\Sigma$  называется любая конечная последовательность символов этого алфавита.

### Операции над цепочками символов

#### 1. Конкатенация

**Опр.** Если  $a$  и  $b$  — цепочки, то цепочка  $ab$  (результат приписывания цепочки  $b$  в конец цепочки  $a$ ), называется *конкатенацией* (или *сцеплением*) цепочек  $a$  и  $b$ . Конкатенацию можно считать двуместной операцией над цепочками:  $a \times b = ab$ .

Например, если  $w = ab$  и  $z = cd$ , то  $w \times z = abcd$ .

Для любой цепочки  $a$ :  $a\varepsilon = \varepsilon a = a$ .

Для любых цепочек  $a, b, g$  справедливо свойство ассоциативности операции конкатенации  $(ab)g = a(bg) = abg$ .

#### 2. Обращение

• **Опр.** *Обращением* (или *реверсом*) цепочки  $\alpha$  называется цепочка, символы которой записаны в обратном порядке.

Обращение цепочки  $\alpha$  будем обозначать  $\alpha^R$ .

Например, если  $\alpha = abcdef$ , то  $\alpha^R = fedcba$ .

Для пустой цепочки:  $\varepsilon^R = \varepsilon$ .

#### 3. Возведение в степень

• **Опр.**  $n$ -ой степенью цепочки  $\alpha$  (будем обозначать  $\alpha^n$ ) называется конкатенация  $n$  цепочек  $\alpha$ :

$$\alpha^n = \underbrace{\alpha \alpha \dots \alpha}_n$$

Свойства степени:  $\alpha^0 = \varepsilon$ ;  $\alpha^n = \alpha \alpha^{n-1} = \alpha^{n-1} \alpha$

Дополнительную информацию смотрите на слайдах 40-42 первой презентации.

## 4 Билет 4

• **Опр.** Обозначим через  $\Sigma^*$  множество, содержащее все цепочки в алфавите  $\Sigma$ , включая пустую цепочку  $\varepsilon$ .  
Например, если  $\Sigma = \{0, 1\}$ , то  $\Sigma^* = \{\varepsilon, 0, 1, 00, 11, 01, 10, 000, 001, 011, \dots\}$ .

### Формальное определение языка

• **Опр.** *Язык* в алфавите  $\Sigma$  — это подмножество множества всех цепочек в этом алфавите. Для любого языка  $L$  справедливо  $L \subseteq \Sigma^*$ .

### Операции над языками

#### 1. Конкатенация

• **Опр.** *Конкатенацией* двух языков  $L_1$  и  $L_2$  называется язык

$$L_3 = \{\alpha\beta \mid \alpha \in L_1, \beta \in L_2\}.$$

$$\text{Обозн.: } L_3 = L_1 L_2$$

**Пример:**  $\{01, 111, 10\} \{00, 01\} = \{0100, 0101, 11100, 11101, 1000, 1001\}$ .

#### 2. Объединение

• **Опр.** *Объединением* двух языков  $L_1$  и  $L_2$  называется язык

$$L = L_1 \cup L_2 = \{\alpha \mid \alpha \in L_1 \text{ или } \alpha \in L_2\}.$$

**Пример:**  $\{01, 111, 10\} \cup \{00, 01\} = \{01, 111, 10, 00\}$ .

#### 3. Степень

• **Опр.** *Степень* языка  $L$ :

$$1. L^0 = \{\varepsilon\}$$

$$2. L^1 = L$$

$$3. L^k = L^{k-1} L$$

#### 4. Итерация

- **Опр. Итерация** языка  $L$ :

$$L^* = \bigcup_{k=0}^{\infty} L^k \quad L^+ = \bigcup_{k=1}^{\infty} L^k \quad L^+ = L^* L$$

$$L^* = \{\varepsilon\} \cup L \cup LL \cup LLL \cup \dots$$

**Пример:**  $\{0,10\}^* = \{\varepsilon, 0, 10, 00, 010, 100, 1010, \dots\}$

## 5 Билет 5

### Способы описания языков

- Конечный язык можно описать простым перечислением его цепочек.
- Как представлять бесконечные языки?
  - спецификация (описание)
  - механизм распознавания
  - механизм порождения (генерации).
- Не каждый формальный язык можно задать с помощью конечного описания.

## 6 Билет 6

- **Спецификация** — Описание языка, как множества слов, удовлетворяющих некоторому условию. (Для регулярных языков — это регулярное выражение\_.

## 7 Билет 7

- Механизм распознавания (*распознаватель*), по сути, является процедурой специального вида, которая по заданной цепочке определяет, принадлежит ли она языку.
- Если принадлежит, то процедура останавливается с ответом «да», т. е. *допускает* цепочку; иначе — останавливается с ответом «нет» или закликивается.
- *Язык, определяемый распознавателем* — это множество всех цепочек, которые он допускает.

- Механизм, который является процедурой специального вида, которая по заданной цепочке определяет, принадлежит ли она языку.
- Если принадлежит, то процедура останавливается с ответом «да», т. е. *допускает* цепочку; иначе — останавливается с ответом «нет» или заикливается.
- *Язык, определяемый распознавателем* — это множество всех цепочек, которые он допускает.



## 8 Билет 8

**Опр. Порождающая грамматика**  $G$  — это четверка  $\langle T, N, P, S \rangle$ ,

где

$T$  — алфавит терминальных символов (терминалов);

$N$  — алфавит нетерминальных символов (нетерминалов),  $T \cap N = \emptyset$ ;

$P$  — конечное подмножество множества  $(T \cup N)^+ \times (T \cup N)^*$ ;

где элемент  $(\alpha, \beta)$  записывается в виде  $\alpha \rightarrow \beta$  и называется *правилом вывода*;

$\alpha$  называется *левой частью* правила,  $\beta$  — *правой частью* правила.

Левая часть любого правила из  $P$  обязана содержать хотя бы один нетерминал;

$S$  — начальный символ грамматики,  $S \in N$ .

Для записи правил вывода с одинаковыми левыми частями

$$\alpha \rightarrow \beta_1 \quad \alpha \rightarrow \beta_2 \quad \dots \quad \alpha \rightarrow \beta_n$$

используют сокращенную запись  $\alpha \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$ .

**Опр.** Цепочка  $\beta \in (T \cup N)^*$  *непосредственно выводима* из цепочки  $\alpha \in (T \cup N)^+$  в грамматике  $G = \langle T, N, P, S \rangle$  (обозначается  $\alpha \rightarrow_e \beta$ ), если

$$\alpha = \xi_1 \gamma \xi_2, \quad \beta = \xi_1 \delta \xi_2,$$

где

$$\xi_1, \xi_2, \delta \in (T \cup N)^*, \quad \gamma \in (T \cup N)^+$$

и правило вывода  $\gamma \rightarrow \delta$  содержится в  $P$ .



**Опр.** Цепочка  $\beta \in (T \cup N)^*$  *выводима* из цепочки  $\alpha \in (T \cup N)^+$  в грамматике  $G = \langle T, N, P, S \rangle$  (обозначается  $\alpha \Rightarrow \beta$ ), если существуют цепочки  $\gamma_0, \gamma_1, \dots, \gamma_n$  ( $n \geq 0$ ), такие, что

$$\alpha = \gamma_0 \rightarrow \gamma_1 \rightarrow \dots \rightarrow \gamma_n = \beta.$$

Последовательность  $\gamma_0, \gamma_1, \dots, \gamma_n$  называется *выводом длины  $n$* .  
Длину вывода  $n$  показывают обозначением:  
Вывод за некоторое число шагов (м.б. 0 шагов) обозначается:

Индекс  $G$  в обозначении  $\Rightarrow$  опускают, если понятно, какая грамматика подразумевается.

**Опр.** *Языком, порождаемым грамматикой*  $G = \langle T, N, P, S \rangle$ , называется множество

$$L(G) = \{ \alpha \in T^* \mid S \Rightarrow \alpha \}.$$

Другими словами,  $L(G)$  — это все цепочки в алфавите  $T$ , которые выводимы из  $S$  с помощью правил  $P$ .  
Например,  $L(G_{example}) = \{ 0^n 1^n \mid n > 0 \}$ .

**Опр.** Цепочка  $\alpha \in (T \cup N)^*$ , для которой  $S \Rightarrow \alpha$ , называется *сентенциальной формой* в грамматике  $G = \langle T, N, P, S \rangle$ .

Таким образом, *язык, порождаемый грамматикой*, можно определить как *множество терминальных сентенциальных форм*.

Дополнительную информацию смотри на слайдах 54-60 первой презентации.

## 9 Билет 9

### Классификация грамматик и языков по Хомскому

- Тип грамматики определяется типом ограничений на вид правил вывода.
- Всего определено четыре типа грамматик:  
тип 0, тип 1, тип 2, тип 3.
- Каждому типу грамматик соответствует свой класс языков.
- Если язык порождается грамматикой типа  $i$  (для  $i = 0, 1, 2, 3$ ), то он является *языком типа  $i$* .

#### 1. Тип 0

### Тип 0

Любая порождающая грамматика является грамматикой типа 0.

На вид правил грамматик этого типа не накладывается никаких дополнительных ограничений.

Класс языков типа 0 совпадает с классом рекурсивно перечислимых языков.

## 2. Тип 1

### Классификация грамматик и языков по Хомскому: Тип 1

**Опр.** Грамматика  $G = \langle T, N, P, S \rangle$  называется *неукорачивающей*, если правая часть каждого правила из  $P$  не короче левой части (т. е. для любого правила  $\alpha \rightarrow \beta \in P$  выполняется неравенство  $|\alpha| \leq |\beta|$ ).

В виде исключения в неукорачивающей грамматике допускается наличие правила  $S \rightarrow \varepsilon$ , при условии, что  $S$  (начальный символ) не встречается в правых частях правил.

Грамматикой *типа 1* называют неукорачивающую грамматику.

### Классификация грамматик и языков по Хомскому: Тип 1

Другое определение:

**Опр.** Грамматика  $G = \langle T, N, P, S \rangle$  называется *контекстно-зависимой (КЗ)*, если каждое правило из  $P$  имеет вид  $\alpha \rightarrow \beta$ ,

где  $\alpha = \xi_1 A \xi_2$ ,  $\beta = \xi_1 \gamma \xi_2$ ,  $A \in N$ ,  $\gamma \in (T \cup N)^+$ ,  $\xi_1, \xi_2 \in (T \cup N)^*$ .

В виде исключения в КЗ-грамматике допускается наличие правила  $S \rightarrow \varepsilon$ , при условии, что  $S$  (начальный символ) не встречается в правых частях правил.

Язык, порождаемый контекстно-зависимой грамматикой, называется *контекстно-зависимым языком*.

## 3. Тип 2

## Классификация грамматик и языков по Хомскому: Тип 2

**Опр.** Грамматика  $G = \langle T, N, P, S \rangle$  называется *контекстно-свободной* (КС), если каждое правило из  $P$  имеет вид  $A \rightarrow \beta$ , где  $A \in N$ ,  $\beta \in (T \cup N)^*$ . Заметим, что в КС-грамматиках допускаются правила с пустыми правыми частями. Язык, порождаемый контекстно-свободной грамматикой, называется *контекстно-свободным* языком. Грамматикой *типа 2* будем называть контекстно-свободную грамматику.

### 4. Тип 3

## Классификация грамматик и языков по Хомскому: Тип 3

**Опр.** Грамматика  $G = \langle T, N, P, S \rangle$  называется *праволинейной*, если каждое правило из  $P$  имеет вид  $A \rightarrow wB$  либо  $A \rightarrow w$ , где  $A, B \in N$ ,  $w \in T^*$ .  
**Опр.** Грамматика  $G = \langle T, N, P, S \rangle$  называется *леволинейной*, если каждое правило из  $P$  имеет вид  $A \rightarrow Bw$  либо  $A \rightarrow w$ , где  $A, B \in N$ ,  $w \in T^*$ .

### Дополнительная информация

## Иерархия грамматик Хомского

**Утверждение 5.** Справедливы следующие утверждения:

- 1) любая регулярная грамматика является КС-грамматикой;
- 2) любая неукорачивающая КС-грамматика является КЗ-грамматикой;
- 3) любая неукорачивающая грамматика является грамматикой типа 0.

Утверждение 5 следует непосредственно из определений.

Рассматривая только неукорачивающие регулярные и неукорачивающие КС-грамматики, получаем следующую иерархию классов грамматик:

*Регулярные неукорачивающие*  $\subset$  *КС неукорачивающие*  $\subset$  *КЗ*  $\subset$  *Тип 0*

## 10 Билет 10

### Эквивалентность неукорачивающих и КЗ-грамматик

**Утверждение 1.** Пусть  $L$  — формальный язык. Следующие утверждения эквивалентны.

- 1) существует контекстно-зависимая грамматика  $G_1$ , такая что  $L = L(G_1)$ ;
- 2) существует неукорачивающая грамматика  $G_2$ , такая что  $L = L(G_2)$ .

**Док-во.** Очевидно, что  $(1) \Rightarrow (2)$ : любая контекстно-зависимая грамматика удовлетворяет ограничениям неукорачивающей грамматики (см. определения).

Т.к. каждое неукорачивающее правило можно заменить эквивалентной серией контекстно-зависимых правил, следовательно  $(2) \Rightarrow (1)$ .

Т.о., язык, порождаемый неукорачивающей грамматикой, — контекстно-зависимый. ■

Т.е., неукорачивающие и КЗ-грамматики определяют один и тот же класс языков.

## 11 Билет 11

**Опр.** Грамматика  $G = \langle T, N, P, S \rangle$  называется **контекстно-свободной** (КС), если каждое правило из  $P$  имеет вид  $A \rightarrow \beta$ , где  $A \in N$ ,  $\beta \in (T \cup N)^*$ .

Заметим, что в КС-грамматиках допускаются правила с пустыми правыми частями. Язык, порождаемый контекстно-свободной грамматикой, называется **контекстно-свободным** языком.

Грамматикой **типа 2** будем называть контекстно-свободную грамматику.

КС-грамматика может являться неукорачивающей, т.е. удовлетворять ограничениям неукорачивающей грамматики.

**Утверждение 2.** Для любой КС-грамматики  $G$  существует неукорачивающая КС-грамматика  $G'$ , такая что  $L(G) = L(G')$ .

## 12 Билет 12

**Опр.** Грамматика  $G = \langle T, N, P, S \rangle$  называется **праволинейной**, если каждое правило из  $P$  имеет вид  $A \rightarrow wB$  либо  $A \rightarrow w$ , где  $A, B \in N$ ,  $w \in T^*$ .

**Опр.** Грамматика  $G = \langle T, N, P, S \rangle$  называется **леволинейной**, если каждое правило из  $P$  имеет вид  $A \rightarrow Bw$  либо  $A \rightarrow w$ , где  $A, B \in N$ ,  $w \in T^*$ .

**Утверждение 3.** Пусть  $L$  — формальный язык. Следующие два утверждения эквивалентны:

- 1) существует праволинейная грамматика  $G_1$ , такая что  $L = L(G_1)$ ;
- 2) существует левوليнейная грамматика  $G_2$ , такая что  $L = L(G_2)$ .

Т.е., праволинейные и левوليнейные грамматики определяют один и тот же класс языков. Такие языки называются **регулярными**.

- Право- и левوليнейные грамматики также называют регулярными.
- Регулярная грамматика является грамматикой **типа 3**.

## 13 Билет 13

Т.е., праволинейные и левوليнейные грамматики определяют один и тот же класс языков. Такие языки называются **регулярными**.

**Опр.** **Автоматной** грамматикой называется праволинейная (левوليнейная) грамматика, такая, что каждое правило с непустой правой частью имеет вид:

$A \rightarrow a$  либо  $A \rightarrow aB$  (для левوليнейной, соответственно,  $A \rightarrow a$  либо  $A \rightarrow Ba$ ),

где  $A, B \in N, a \in T$ .

Автоматная грамматика является более простой формой регулярной грамматики.

Существует алгоритм, позволяющий по регулярной (право- или левوليнейной) грамматике построить соответствующую автоматную грамматику.

Таким образом, **любой регулярный язык может быть порожден автоматной грамматикой**.

Существует алгоритм, позволяющий устранить из регулярной (автоматной) грамматики все  $\epsilon$ -правила (кроме  $S \rightarrow \epsilon$  в случае, если пустая цепочка принадлежит языку; при этом  $S$  не будет встречаться в правых частях правил).

**Утверждение 4.** Для любой регулярной (автоматной) грамматики  $G$  существует неукорачивающая регулярная (автоматная) грамматика  $G'$ , такая что  $L(G) = L(G')$ .

## 14 Билет 14

### Иерархия языков

**Утверждение 6.** Справедливы следующие утверждения:

- 1) Каждый регулярный язык является КС-языком, **но существуют КС-языки, которые не являются регулярными**, например:

$$L = \{a^n b^n \mid n > 0\};$$

- 2) Каждый КС-язык является КЗ-языком, **но существуют КЗ-языки, которые не являются КС-языками**, например:

$$L = \{a^n b^n c^n \mid n > 0\};$$

- 3) Каждый КЗ-язык является языком типа 0 (т. е. рекурсивно перечислимым языком), **но существуют языки типа 0, которые не являются КЗ-языками**.

Из утверждения 6 следует иерархия классов языков:

$$\text{Тип 3 (Регулярные)} \subset \text{Тип 2 (КС)} \subset \text{Тип 1 (КЗ)} \subset \text{Тип 0}$$



Для  $k = 1, 2, 3$  язык типа  $k$  является также и языком типа  $k - 1$  (класс языков типа  $k$  является подклассом класса языков типа  $k - 1$ ).

## 15 Билет 15

- **Утверждение 7.** Проблема «Можно ли язык, описанный грамматикой типа  $k$  ( $k = 0, 1, 2$ ), описать грамматикой типа  $k + 1$ ?» является *алгоритмически неразрешимой*.
- Т.е., нет алгоритма, позволяющего по заданному описанию языка  $L$  (например, по грамматике), определить максимальное  $k$ , такое что  $L$  является языком типа  $k$

## 16 Билет 16

### Регулярные

1. Грамматика  $S \rightarrow aS \mid a$  является праволинейной (неукорачивающей) грамматикой и порождает регулярный язык  $\{a^n \mid n > 0\}$ .

Этот язык может быть порожден и леволинейной грамматикой:  $S \rightarrow Sa \mid a$ .

Обе эти грамматики являются автоматными.

2. Грамматика  $S \rightarrow aS \mid \varepsilon$  является праволинейной и порождает регулярный язык  $\{a^n \mid n \geq 0\}$ .

Для любого регулярного языка существует неукорачивающая регулярная грамматика (см. утверждение 4):

$$\begin{aligned} S &\rightarrow aA \mid a \mid \varepsilon \\ A &\rightarrow a \mid aA \end{aligned}$$

Правило с пустой правой частью может применяться только один раз и только на первом шаге вывода; остальные правила таковы, что их правая часть не короче левой, т. е. грамматика неукорачивающая.

3. Грамматика  $S \rightarrow A\perp \mid B\perp$  левوليнейная;  
 $A \rightarrow a \mid Ba$   
 $B \rightarrow b \mid Bb \mid Ab$

она порождает регулярный язык, состоящий из всех непустых цепочек в алфавите  $\{a, b\}$ , заканчивающихся символом  $\perp$  (маркер конца) и не содержащих подцепочку  $aa$ .

То есть в цепочках этого языка символ  $a$  не может встречаться два раза подряд, хотя бы один символ  $b$  обязательно присутствует между любыми двумя  $a$ .

Данный язык описывается как:  $L = \{\omega\perp \mid \omega \in \{a, b\}^+, aa \not\subset \omega\}$ .

### Контекстно-свободные

1. Грамматика  $S \rightarrow aQb \mid accb$   
 $Q \rightarrow cSc$

является контекстно-свободной (неукорачивающей) и порождает КС-язык  $\{(ac)^n (cb)^n \mid n > 0\}$ , который, не является регулярным.

2. Грамматика  $S \rightarrow aSa \mid bSb \mid \varepsilon$

порождает КС-язык  $\{xx^R, x \in \{a, b\}^*\}$ . Данный язык не является регулярным.

Грамматика не удовлетворяет определению неукорачивающей, но для нее существует эквивалентная неукорачивающая грамматика (см. утверждение 2):

$S \rightarrow A \mid \varepsilon$   
 $A \rightarrow aAa \mid bAb \mid aa \mid bb$

### Неукорачивающие и контекстно-зависимые

1. Грамматика:  $S \rightarrow aSBC \mid abC$   
 $CB \rightarrow BC$   
 $bB \rightarrow bb$   
 $bC \rightarrow bc$   
 $cC \rightarrow cc$

неукорачивающая и порождает язык  $\{a^n b^n c^n \mid n > 0\}$ , который является языком типа 1, но не является языком типа 2.

Правило  $CB \rightarrow BC$  не удовлетворяет определению КЗ-грамматики. Заменим его тремя новыми

$CB \rightarrow CD, CD \rightarrow BD, BD \rightarrow BC$ . Получим эквивалентную серию контекстно-зависимых правил, которые меняют местами символы  $C$  и  $B$ . Таким образом, получаем эквивалентную КЗ-грамматику:

$S \rightarrow aSBC \mid abC$   
 $CB \rightarrow CD$   
 $CD \rightarrow BD$   
 $BD \rightarrow BC$   
 $bB \rightarrow bb$   
 $bC \rightarrow bc$   
 $cC \rightarrow cc$

$$\begin{array}{lcl} 1. \text{ Грамматика типа 0} & S & \rightarrow SS \\ & SS & \rightarrow \varepsilon \end{array}$$

Второе правило не удовлетворяет ограничениям неукорачивающей грамматики. В данной грамматике существует бесконечно много выводов, однако порождаемый язык конечен и состоит из единственной цепочки:  $\{\varepsilon\}$ .

$$\begin{array}{lcl} 2. \text{ Грамматика} & S & \rightarrow SS \\ & SS & \rightarrow Sa \mid S \end{array}$$

также не является неукорачивающей и порождает пустой язык, так как ни одна терминальная цепочка не выводится из  $S$ .

Языки  $L_\varepsilon = \{\varepsilon\}$  и  $L_\emptyset = \emptyset$  (пустой язык) могут быть описаны грамматиками типа 3.

## 17 Билет 17

### Определение регулярного выражения

- **Опр.** Пусть  $\Sigma$  — алфавит, не содержащий символов  $*$ ,  $+$ ,  $\varepsilon$ ,  $\emptyset$ ,  $($ ,  $)$ .  
Определим рекурсивно *регулярное выражение*  $\gamma$  над алфавитом  $\Sigma$  и регулярный язык  $L(\gamma)$ , задаваемый этим выражением:  
*Базис индукции:*  
1)  $a \in \Sigma$  есть регулярное выражение;  $L(a) = \{a\}$   
2)  $\varepsilon$  есть регулярное выражение;  $L(\varepsilon) = \{\varepsilon\}$   
3)  $\emptyset$  есть регулярное выражение;  $L(\emptyset) = \emptyset$

*Индукция:*

Если  $\alpha$  и  $\beta$  — регулярные выражения, то:

- 1)  $(\alpha) + (\beta)$  — регулярное выражение;  $L((\alpha) + (\beta)) = L(\alpha) \cup L(\beta)$ ;
- 2)  $(\alpha)(\beta)$  — регулярное выражение;  $L((\alpha)(\beta)) = L(\alpha)L(\beta)$ ;
- 3)  $(\beta)^*$  — регулярное выражение;  $L((\beta)^*) = (L(\beta))^*$ ;

Никаких других регулярных выражений, кроме тех, что построены в соответствии с описанным определением, нет.

### Определение регулярного множества

- **Опр.** Пусть  $\Sigma$  — конечный алфавит. Регулярное множество в алфавите  $\Sigma$  определяется следующим образом:  
1)  $\emptyset$  — регулярное множество в алфавите  $\Sigma$ .  
2)  $\{\varepsilon\}$  — регулярное множество в алфавите  $\Sigma$ .  
3)  $\{a\}$  — регулярное множество в алфавите  $\Sigma$  для каждого  $a \in \Sigma$ .  
4) Если  $Q$  и  $P$  — регулярные множества в алфавите  $\Sigma$ , то множества  $Q \cup P$ ,  $QP$  и  $P^*$  регулярные.  
5) Ничто другое не является регулярным множеством в алфавите  $\Sigma$ .



## Алгебраические законы для регулярных выражений

- Объединение и конкатенация ведут себя как сложение и умножение.
- $+$  является коммутативным и ассоциативным;
- Конкатенация является ассоциативной операцией.
- Конкатенация распределяется по  $+$ .
- Исключение: Конкатенация не коммутативна.

### Дополнительная информация

Лемма 1. Если  $\alpha, \beta, \gamma$  - регулярные выражения, то справедливы следующие соотношения:

- |  |  |
|--|--|
| 1) $\alpha + \beta = \beta + \alpha$                       | 7) $\emptyset^* = \varepsilon$                           |
| 2) $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$ | 8) $\alpha(\beta\gamma) = (\alpha\beta)\gamma$           |
| 3) $\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$   | 9) $(\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma$ |
| 4) $\alpha\varepsilon = \varepsilon\alpha = \alpha$        | 10) $\emptyset\alpha = \alpha\emptyset = \emptyset$      |
| 5) $\alpha^* = \alpha + \alpha^*$                          | 11) $(\alpha^*)^* = \alpha^*$                            |
| 6) $\alpha + \alpha = \alpha$                              | 12) $\alpha + \emptyset = \alpha$                        |

- $\emptyset$  является нейтральным элементом для  $+$ .  
 $R + \emptyset = R$ .
- $\varepsilon$  является «единицей» для конкатенации.  
 $\varepsilon R = R\varepsilon = R$ .
- $\emptyset$  является «нулем» для конкатенации.  
 $\emptyset R = R\emptyset = \emptyset$ .

## 18 Билет 18

### Уравнения с регулярными коэффициентами

- Рассм. уравнение  $X=aX+b$ , где  $a$  и  $b$  – РВ.

$X=a*b$  – решение уравнения:

$$a*b=aa*b+b$$

$$a*b=(aa^*+\varepsilon)b$$

$$a*b=a*b$$

Если множество, определяемое рег. выражением  $a$ , содержит  $\varepsilon$ , то ур-е имеет бесконечно много решений:  $X=a*(b+c)$  для любого РВ  $c$ .

В этом случае берут «наименьшее решение» – *наименьшую неподвижную точку*.

### Алгоритм решения стандартной системы уравнений с регулярными коэффициентами

- Вход. Стандартная система  $Q$  уравнений с рег. коэффициентами в алфавите  $\Sigma$  и множеством неизвестных  $\Delta = \{X_1, X_2, \dots, X_n\}$ .
- Выход. Решение системы  $Q$  в виде  $X_1 = a_1, X_2 = a_2, \dots, X_n = a_n$ , где  $a_i$  – РВ в алфавите  $\Sigma$ .

- Метод. (см. решение СЛУ методом Гаусса)

Шаг 1.  $i=1$ .

Шаг 2. Если  $i=n$ , перейти к шагу 4.

Иначе, с помощью тождеств РВ записать ур-е для  $X_i$  в виде  $X_i = \alpha X_i + \beta$ , где  $\alpha$  – РВ в алфавите  $\Sigma$ , а  $\beta$  – РВ вида  $\beta_0 + \beta_i X_{i+1} + \dots + \beta_n X_n$ ;  $\beta_i$  – РВ в алфавите  $\Sigma$ .

Затем, в правых частях уравнений для  $X_{i+1}, \dots, X_n$  заменить  $X_i$  выражением  $\alpha^* \beta$ .

Шаг 3.  $i=i+1$ . Перейти к шагу 2.

Шаг 4. Записать ур-е для  $X_n$  в виде  $X_n = \alpha X_n + \beta$ , где  $\alpha$  и  $\beta$  – РВ в алфавите  $\Sigma$ .

Перейти к шагу 5 ( $i=n$ ).

Шаг 5. Уравнение для  $X_i$  имеет вид  $X_i = \alpha X_i + \beta$ , где  $\alpha$  и  $\beta$  – РВ в алфавите  $\Sigma$ .

Записать на выходе  $X_i = \alpha^* \beta$  и в ур-я для  $X_{i-1}, \dots, X_1$  подставить  $\alpha^* \beta$  вместо  $X_i$ .

Шаг 6. Если  $i=1$ , остановиться. В противном случае  $i=i-1$  и вернуться к шагу 5.

### Дополнительная информация

- Опр. Система уравнений с рег. коэффициентами наз. *стандартной системой* с мн-вом неизвестных  $\Delta = \{X_1, X_2, \dots, X_n\}$ , если она имеет вид:

$$X_1 = a_{10}X_1 + a_{12}X_2 + \dots + a_{1n}X_n$$

$$\dots\dots\dots$$

$$X_n = a_{n0}X_1 + a_{n2}X_2 + \dots + a_{nn}X_n$$

где все  $a_{ij}$  - регулярные выражения в алфавите, не пересекающимся с  $\Delta$ .

Если  $a_{ij} = \emptyset$ , то соотв. слагаемое отсутствует, если  $a_{ij} = \varepsilon$ , то слагаемое равно  $X_j$ .

## 19 Билет 19

- Язык определяется праволинейной грамматикой т.и.т.т., когда он является регулярным множеством.
- Лемма 3. Множества  $\emptyset$ ,  $\{\varepsilon\}$  и  $a$  для всех  $a \in \Sigma$  являются праволинейными языками.
- Док-во.
  - 1)  $G = (\{S\}, \Sigma, \{P\}, S)$  – праволинейная грамматика, для которой  $L(G) = \emptyset$ .
  - 2)  $G = (\{S\}, \Sigma, \{S \rightarrow \varepsilon\}, S)$  – праволинейная грамматика, для которой  $L(G) = \{\varepsilon\}$ .
  - 3)  $G_a = (\{S\}, \Sigma, \{S \rightarrow a\}, S)$  – праволинейная грамматика, для которой  $L(G_a) = \{a\}$ .
- Лемма 4. Если  $P$  и  $Q$  – праволинейные языки, то языки
  - 1)  $P \cup Q$ , 2)  $PQ$ , 3)  $P^*$  тоже праволинейные.
- Док-во.
 

Т.к.  $P$  и  $Q$  – праволинейные, то  $\exists$  праволинейные грамматики  $G_P = (N_1, \Sigma, P_1, S_1)$  и  $G_Q = (N_2, \Sigma, P_2, S_2)$ , для которых  $L(G_P) = P$  и  $L(G_Q) = Q$ . Считаем, что  $N_1 \cap N_2 = \emptyset$ .

  - 1)  $G_3 = (N_1 \cup N_2, \Sigma, P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 | S_2\}, S_3)$  – праволинейная.  $L(G_3) = L(G_P) \cup L(G_Q)$  т.к. для каждого вывода  $S_3 \Rightarrow_{G_3}^+ w$  существует либо вывод  $S_1 \Rightarrow_{G_P}^+ w$ , либо вывод  $S_2 \Rightarrow_{G_Q}^+ w$  и обратно.

Т.к.  $G_3$  - праволинейная грамматика, то  $L(G_3)$  – праволинейный язык.

2)  $G_4 = (N_1 \cup N_2, \Sigma, P_4, S_1)$  – праволинейная.

$P_4$ : (1) Если  $A \rightarrow xB$  есть в  $P_1$ , то  $A \rightarrow xB$  принадлежит  $P_4$ .

(2) Если  $A \rightarrow x$  есть в  $P_1$ , то  $A \rightarrow xS_2$  принадлежит  $P_4$ .

(3) Все правила из  $P_2$  принадлежат  $P_4$ .

Отметим, что если  $S_1 \Rightarrow_{G_P}^+ w$ , то  $S_1 \Rightarrow_{G_4}^+ wS_2$ , а если  $S_2 \Rightarrow_{G_Q}^+ x$ , то  $S_2 \Rightarrow_{G_4}^+ x$ .

Таким образом,  $L(G_P)L(G_Q) \subseteq L(G_4)$ .

Пусть  $S_1 \Rightarrow_{G_4}^+ w$ . Т.к. в  $G_4$  нет правил  $A \rightarrow x$ , попавших из  $P_1$ , то этот вывод можно сделать так:  $S_1 \Rightarrow_{G_4}^+ xS_2 \Rightarrow_{G_4}^+ xu$ , где  $w = xu$  и все правила в выводе  $S_1 \Rightarrow_{G_4}^+ wS_2$  попали в  $P_4$  по (1) и (2).

Следовательно, должны быть выводы  $S_1 \Rightarrow_{G_P}^+ x$  и  $S_2 \Rightarrow_{G_Q}^+ u$ .

Отсюда  $L(G_4) \subseteq L(G_P)L(G_Q)$ .

3) Пусть  $G_5 = (N_1 \cup \{S_5\}, \Sigma, P_5, S_5)$  такая, что  $S_5 \notin N_1$  и

$P_5$ : (1) Если  $A \rightarrow xB$  есть в  $P_1$ , то  $A \rightarrow xB$  принадлежит  $P_5$ .

(2) Если  $A \rightarrow x$  есть в  $P_1$ , то  $A \rightarrow xS_5$  и  $A \rightarrow x$  принадлежат  $P_5$ .

(3)  $S_5 \rightarrow S_1 | \varepsilon$  принадлежат  $P_5$ .

Очевидно, что

$$S_5 \Rightarrow_{G_5}^+ x_1 S_5 \Rightarrow_{G_5}^+ x_1 x_2 S_5 \Rightarrow_{G_5}^+ \dots \Rightarrow_{G_5}^+ x_1 x_2 \dots x_{n-1} x_n$$

т.и.т.т., когда

$$S_1 \Rightarrow_{G_P}^+ x_1, S_1 \Rightarrow_{G_P}^+ x_2, \dots, S_1 \Rightarrow_{G_P}^+ x_n.$$

Отсюда следует, что  $L(G_4) = (L(G_P))^*$ . ■

• Теорема. Язык является регулярным множеством т.и.т.т., когда он праволинейный.

• Док-во.

Необходимость.

Следует из лемм 3 и 4, индукцией по числу шагов построения регулярного множества, где один шаг – это применение одного из правил, определяющих регулярные множества.

Достаточность.

Пусть  $G = (N, \Sigma, P, S)$  – праволинейная грамматика и  $N = \{A_1, A_2, \dots, A_n\}$ .

Можно построить стандартную систему уравнений с регулярными коэффициентами, неизвестными которой являются нетерминалы из  $N$ .

Уравнение для  $A_i$  будет иметь вид:  $A_i = \alpha_{i0} + \alpha_{i1}A_1 + \dots + \alpha_{in}A_n$ , где

(1)  $\alpha_{i0} = w_1 + \dots + w_k$ , если  $A_i \rightarrow w_1 | \dots | w_k$  - все правила с левой частью  $A_i$  и правой частью, состоящей только из терминалов (если  $k=0$ , то  $\alpha_{i0} = \emptyset$ ).

(2) Для  $j>0$   $\alpha_{ij} = x_1 + \dots + x_m$ , если  $A_i \rightarrow x_1 A_j | \dots | x_m A_j$  - все правила с левой частью  $A_i$  и правой частью, оканчивающуюся на  $A_j$  (если  $m=0$ , то  $\alpha_{ij} = \emptyset$ ).

Решая эту систему уравнений получаем решение  $f$  для  $N = \{A_1, A_2, \dots, A_n\}$ .

Для  $S$  получаем  $PB f(S)$ , которое определяет язык  $L(G)$ .

Но алгоритм строит  $f(S)$  как язык, обозначаемый некоторым  $PB$ .

Таким образом,  $L(G)$  – регулярное множество.

## 20 Билет 20

- Класс регулярных множеств – наименьший класс языков, содержащий  $\emptyset$ ,  $\{\varepsilon\}$ ,  $\{a\}$  для всех  $a$  и замкнутый относительно операций объединения, конкатенации и итерации.
- Регулярные множества – множества, определяемые регулярными выражениями.
- Регулярные множества – языки, порождаемые праволинейными грамматиками.
- Регулярные множества – языки, распознаваемые конечными автоматами.

### Допустимые входы

- Дана последовательность входов (*входная строка*).
- Начать в начальном состоянии и следовать по переходу по каждому очередному символу входной строки.
- Вход *принимается*, если вы перенеслись в финальное (принимающее) состояние после чтения всех входных символов.

### Детерминированный конечный автомат

- Формализм для определения языков, состоящий из:

Конечного множества *состояний* (обозн. обычно  $Q$ ).

*Входной алфавит* ( $\Sigma$ , обычно).

*Функция переходов* ( $\delta$ , обычно).

*Начальное состояние* ( $q_0$ , в  $Q$ , обычно).

*Финальные состояния* ( $F \subseteq Q$ , обычно).

- "Финальный" и "принимающий" является синонимами.

$$A = (Q, \Sigma, \delta, q_0, F)$$

$$\delta: Q \times \Sigma \rightarrow Q$$

## Функция переходов

- Имеет два аргумента: состояние и входной символ.
- $\delta(q, a)$  = состояние, в которое КДА переходит, если он в состоянии  $q$  получает на вход символ  $a$ .
- **Замечание:** всегда есть следующее состояние – добавим **мёртвое состояние** если нет переходов (Пример далее).
- Форма задания: таблица переходов, граф.
- Функцию переходов можно доопределить для слов:
 
$$\delta^*(q, a) = \delta(q, a) \text{ если } |a|=1.$$

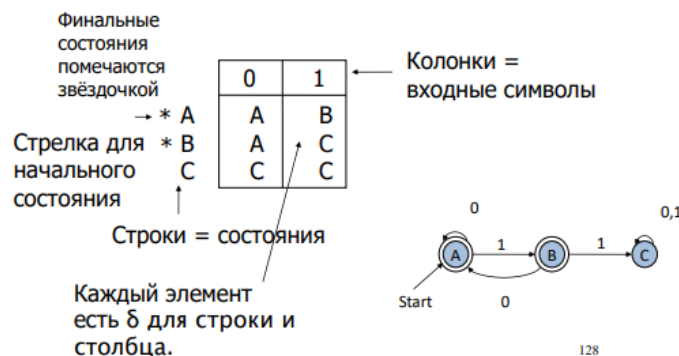
$$\delta^*(q, aw) = \delta(\delta^*(q, w), a)$$

## Представление КДА графом

- Вершины = состояния.
- Дуги представляют функцию переходов.
  - Дуга из состояния  $p$  в состояние  $q$  помечается всеми теми входными символами, по которым происходит переход из  $p$  в  $q$ .
- Стрелка помеченная как "Start" указывает на начальное состояние.
- Финальные состояния обозначаются двойной окружностью.

122

## Альтернативное представление: Таблица переходов



## Язык КДА

- Автоматы всех видов определяют языки.
- Если  $A$  - автомат,  $L(A)$  – его язык.
- Для КДА  $A$ ,  $L(A)$  есть множество строк, помечающих пути из начального состояния в финальное.
- **Формально:**  $L(A)$  = множество строк  $w$  таких, что  $\delta(q_0, w)$  есть в  $F$ .

## Формальное определение НКА

- Конечное множество состояний, обычно  $Q$ .
- Входной алфавит, обычно  $\Sigma$ .
- Функция переходов, обычно  $\delta$ .
- Начальное состояние в  $Q$ , обычно  $q_0$ .
- Множество конечных состояний  $F \subseteq Q$ .

$$A = (Q, \Sigma, \delta, q_0, F)$$

$$\delta: Q \times \Sigma \rightarrow 2^Q$$

## Язык НКА

- Строка  $w$  принимается НКА если  $\delta(q_0, w)$  содержит, по крайней мере, одно финальное состояние.
- Язык НКА – это множество всех принимаемых строк.

## НКА с $\epsilon$ -переходами

- Мы можем допустить переходы между состояниями по входу  $\epsilon$ .
- Эти переходы происходят спонтанно, без оглядки на входную строку.
- Иногда это удобно, но принимаются те же регулярные языки, что принимаются КДА и НКА.

## Заключение

- Теорема. КДА, НКА, и  $\epsilon$ -НКА все принимают в точности одно и то же множество языков: регулярные языки.
- По этой причине эти языки ещё называют **автоматными**.
- Типы НКА проще строить и они могут иметь экспоненциально меньше состояний, чем КДА.
- Но только КДА может быть реализован!

## 21 Билет 21

- Лемма. Если  $L=L(A)$  для некоторого конечного автомата  $A$ , то  $L=L(G)$  для некоторой праволинейной грамматики  $G$ .
- Док-во. Пусть  $A = (Q, \Sigma, \delta, q_0, F)$  – КДА. Определим грамматику  $G = (Q, \Sigma, P, q_0)$ , где  $P$  имеет вид:

Если  $\delta(q, a) = r$ , то  $P$  содержит правило  $q \rightarrow ar$ .

Если  $p \in F$ , то  $P$  содержит правило  $p \rightarrow \epsilon$ .

Каждый шаг вывода в грамматике  $G$  имитирует такт работы автомата  $A$ .

Индукция по  $i$  – длине вывода.

**Базис.**  $i=0$ .  $q \Rightarrow \epsilon$  т.и.т.т., когда  $(q, \epsilon) \vdash^0 (q, \epsilon)$ .

**ИН:**  $s \Rightarrow^i x$  т.и.т.т., когда  $(s, x) \vdash^{i-1} (r, \epsilon)$  для некоторого  $r \in F$ .

**Шаг индукции.** Пусть  $w = ax$ , где  $|x| = i$ .

Тогда  $q \Rightarrow^{i+1} w$  равносильно тому, что  $q \Rightarrow as \Rightarrow^i ax$  для некоторого  $s$ .

Но  $q \Rightarrow as$  равносильно  $\delta(q, a) = s$  или  $(q, a) \vdash^1 (s, \epsilon)$ .

Это означает, что  $(q, ax) \vdash^1 (s, x)$ .

По индукции  $s \Rightarrow^i x$  т.и.т.т., когда  $(s, x) \vdash^{i-1} (r, \epsilon)$  для некоторого  $r \in F$ .

Следовательно,  $q \Rightarrow^{i+1} w$  равносильно  $(q, ax) \vdash^1 (s, x) \vdash^{i-1} (r, \epsilon)$  или

$(q, w) \vdash^i (r, \epsilon)$  для некоторого  $r \in F$ .

ч. т. д.



- Лемма. Если  $L=L(G)$  для некоторой праволинейной грамматики  $G$ , то  $L=L(A)$  для некоторого конечного автомата  $A$ .

- Док-во. Пусть  $G = (Q, \Sigma, P, S)$  – праволинейная грамматика. Построим автомат  $A = (N \cup \{q_f\}, \Sigma, \delta, S, F)$ , где  $\delta$  определено как:

Если  $A \rightarrow aB \in P$ , то  $\delta(A, a) = B$  для  $A, B \in N$  и  $a \in \Sigma$ .

Если  $A \rightarrow a \in P$ , то  $\delta(A, a) = q_f$  для  $A \in N$  и  $a \in \Sigma$ .

$F = \{S, q_f\}$ , если в  $P$  есть  $S \rightarrow \varepsilon$  и  $F = \{q_f\}$  – в противном случае.

Очевидно, что построенный автомат определяет тот же язык, что и исходная праволинейная грамматика.

## 22 Билет 22

## 23 Билет 23

Пусть  $L$  — регулярный язык, заданный автоматом  $A = (Q, \Sigma, \delta, q_0, F)$  — КДА.

$$w \in L \Leftrightarrow \delta(q_0, w) \in F$$

- Таким образом, *проблема принадлежности произвольной строки (над определенным алфавитом) регулярному языку разрешима*.

## 24 Билет 24

### Проблема пустоты

- Для данного регулярного языка определить, содержит ли он хотя бы одну строку?
- Пусть представлением языка является КДА.
- Вычислим множество состояний, достижимых из начального.
- Если хотя бы одно из конечных состояний достижимо, то ответ “да”, иначе - “нет”.
- *Проблема пустоты для регулярных языков разрешима*.

## 25 Билет 25

### Проблема бесконечности

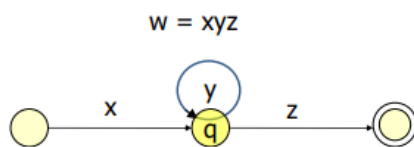
- Является данный регулярный язык бесконечным?
- Рассмотрим КДА для языка.
- **Ключевая идея:** если КДА имеет  $n$  состояний и язык содержит какую-либо строку длины  $n$  или более, то язык бесконечный.
- В противном случае, язык, безусловно, конечный.
  - Можно ограничиться строками длины  $n$  или меньше.

## 26 Билет 26

### Доказательство **ключевой идеи**

- Если КДА с  $n$  состояниями принимает строку  $w$  длины  $n$  или более, то должно быть состояние, в котором он окажется дважды на пути, помеченном  $w$  от начального состояния к конечному.
- Потому что существует, по крайней мере,  $n+1$  состояний вдоль всего пути.

### Доказательство – (2)



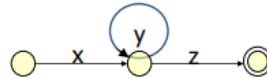
Тогда  $xy^iz$  является языком для всех  $i \geq 0$ .

Т.к.  $y$  не есть  $\epsilon$ , мы имеем бесконечное число строк в  $L$ .

## Бесконечность – продолжение

- Пока мы не имеем алгоритма.
- Существует бесконечное число строк длины  $> n$ , и мы не можем проверить их все.
- **Вторая ключевая идея:** если существует строка длины  $\geq n$  ( $n$  = числу состояний) в  $L$ , то в  $L$  существует строка длины между  $n$  и  $2n-1$ .

### Доказательство 2-ой ключевой идеи



- Вспомним:
- $y$  - это первый цикл в пути.
- Тогда:  $|xy| \leq n$ ; в частности,  $1 \leq |y| \leq n$ .
- То есть, одно состояние среди первых  $n+1$  повторяется.
- Т.о., если  $w$  имеет длину  $2n$  или более, то существует более короткая строка в  $L$ , которая имеет длину, по крайней мере,  $n$ .
- Продолжим сокращение длины путём удаления циклов, пока не достигнем длины в интервале  $[n, 2n-1]$ .

### Завершение доказательства алгоритма

- Для доказательства бесконечности регулярного языка построим для него КДА. Пусть он имеет  $n$  состояний
- Протестируем на принадлежность языку всех строк длины между  $n$  и  $2n-1$ .
  - Если какая-либо из них принимается, то язык бесконечен, иначе – конечен.
- Ужасный алгоритм. Если мы имеем  $k$  входных символов и  $n$  состояний, то число строк для проверки составит  $k^{2n}$ .
- **Лучше:** найти циклы между начальным и конечным состояниями. Для этого есть алгоритм сложности порядка  $kn$ .

## Поиск циклов

1. Исключим состояния, недостижимые из начального состояния.
2. Устраним состояния, из которых не достигаются конечные состояния.
3. Проверим наличие циклов в оставшемся графе переходов.

## Поиск циклов— (2)

- Простым, но менее эффективным способом поиска циклов является поиск вперед от заданного узла  $N$ .
- Если из  $N$  можно достичь  $N$ , то цикл есть.
- Сделайте это, начиная с каждого узла.
- Если есть цикл, то язык для данного КДА бесконечный. Если циклов нет, то конечный.
- *Проблема бесконечности регулярного языка разрешима.*

## 27 Лемма о накачке для регулярных языков

**Лемма.** Для каждого регулярного языка  $L$  существует целое  $n$ , такое, что для каждой строки  $w$  из  $L$  длины  $\geq n$  мы можем записать ее как  $w = xyz$  так, что:

Число состояний КДА для  $L$

1.  $|xy| \leq n$ .
2.  $|y| > 0$ .
3. Для всех  $i \geq 0$ ,  $xy^iz$  также принадлежит  $L$ .

Метки вдоль первого цикла пути, помеченного  $w$

## 28 Эквивалентность регулярных языков

### Проблема: Эквивалентность

- По данным регулярным языкам  $L$  и  $M$  определить  $L = M$ ?
- Алгоритм доказательства существования разрешающей процедуры базируется на построении КДА - *произведения* автоматов для  $L$  и  $M$ .

### Алгоритм установления эквивалентности

- Заключительные состояния автомата-произведения - те состояний  $[q, r]$ , в которых точно одно из  $q$  и  $r$  есть финальное состояние соответствующего автомата.
- Т.о., автомат-произведение принимает  $w$  т.и.т.т., когда  $w$  в точности принадлежит языку для одного из  $L$  или  $M$ .
- $L = M$  тогда и только тогда, когда язык для автомата произведения пуст.

Таким образом, эта проблема для регулярных языков разрешима.

## 29 Построение произведения автоматов $L$ и $M$

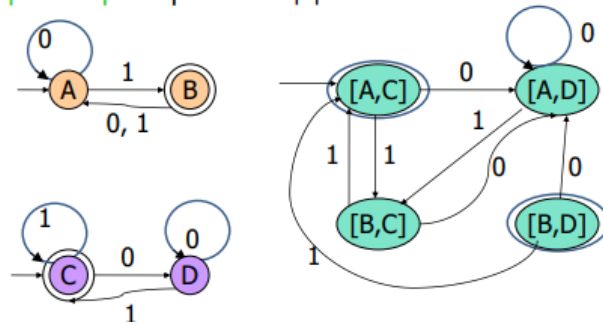
### Построение произведения автоматов $L$ и $M$

- Пусть КДА  $L$  и  $M$  имеют множества состояний  $Q$  и  $R$ , соответственно.
- $L = (Q, \Sigma, \delta_L, q_0, F_L)$ ,  $M = (R, \Sigma, \delta_M, r_0, F_R)$
- Автомат-произведение имеет множество состояний  $Q \times R$ .
  - т.е., пары  $[q, r]$ , где  $q \in Q, r \in R$ .
- Начальное состояние =  $[q_0, r_0]$  (начальные состояния КДА для  $L, M$ ).

## Произведение автоматов – прод.

- **Переходы:**  $\delta([q,r], a) = [\delta_L(q,a), \delta_M(r,a)]$ 
  - $\delta_L, \delta_M$  функции переходов для КДА L и M.
  - То есть, мы моделируем исходные КДА в компонентах двух состояний автомата-произведения.
- Определим заключительные состояния автомата - произведения из тех состояний  $[q, r]$ , в которых точно одно из q и r есть финальное состояние соответствующего автомата.

### Пример: Произведение автоматов



## 30 Проблема вложения регулярных языков

### Проблема вложения языков

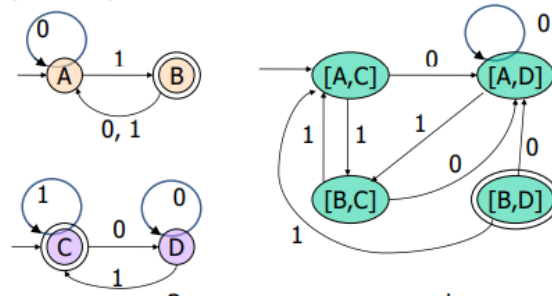
- По данным регулярным языкам L и M определить,  $L \subseteq M$ ?
- Алгоритм также использует конструкцию произведения автоматов.
- Как нужно определить финальное состояние  $[q, r]$  автомата-произведения, чтобы его язык был пуст т.и.т.т., когда  $L \subseteq M$ ?

**Ответ:** q - финальное; r - нет .

- То есть, L не содержится в M т.и.т.т., когда существует некоторая строка w такая, что  $w \in L$  но  $w \notin M$ . Такая строка могла бы перевести КДА для L в финальное состояние, но не смогла бы перевести КДА для M в финальное состояние.
- Так что вопрос вложения – это тот же самый вопрос о существовании строки, переводящей автомат-произведение в состояние  $[q,r]$  где q –заключительное, r - нет.

Таким образом, эта проблема для регулярных языков разрешима.

### Пример: вложение языков



Замечание: только одно финальное состояние недостижимо, язык для автомата произведения пуст, т.о. вложение выполняется.