

## Содержание

1	Общая структура компиляторов. Лексический анализ. Синтаксический анализ. Семантический анализ. Оптимизация кода. Генерация кода.	4
2	Формальные языки. Основные понятия и определения формальных языков.	6
3	Алфавит. Цепочка символов (или слово) в алфавите $\Sigma$ . Операции над цепочками символов.	6
4	Язык в алфавите $\Sigma$ . Операции над языками.	7
5	Методы представления бесконечных языков.	8
6	Спецификация языка.	8
7	Механизм распознавания (распознаватель). Язык, определяемый распознавателем.	8
8	Грамматики. Порождающая грамматика. Выводимость цепочки. Язык грамматики.	9
9	Классификация грамматик и языков по Хомскому.	10
10	Эквивалентность неукорачивающих и КЗ-грамматик.	13
11	КС-грамматика	13
12	Праволинейная и леволинейная грамматики. Эквивалентность этих грамматик.	14
13	Регулярные языки. Автоматные грамматики.	14
14	Иерархия языков.	15
15	Алгоритмическая неразрешимость проблемы отнесения языка к определенному типу по классификации Хомского.	15
16	Примеры грамматик и языков.	16
17	Регулярные множества и регулярные выражения. Алгебраические законы для регулярных выражений.	17
18	Уравнения с регулярными коэффициентами. Алгоритм решения стандартной системы уравнений с регулярными коэффициентами.	19

<b>19 Регулярные множества и праволинейные грамматики.</b>	<b>21</b>
<b>20 Конечные автоматы и регулярные множества. Язык автомата.</b>	<b>23</b>
<b>21 Эквивалентность автоматных и праволинейных языков</b>	<b>26</b>
<b>22 Свойства регулярных языков. Свойства замкнутости. Разрешимые свойства.</b>	<b>27</b>
<b>23 Проблема принадлежности слова языку.</b>	<b>28</b>
<b>24 Проблема пустоты регулярного языка.</b>	<b>28</b>
<b>25 Проблема бесконечности регулярного языка.</b>	<b>28</b>
<b>26 Бесконечность регулярного языка.</b>	<b>29</b>
<b>27 Лемма о накачке для регулярных языков</b>	<b>31</b>
<b>28 Эквивалентность регулярных языков</b>	<b>32</b>
<b>29 Построение произведения автоматов <math>L</math> и <math>M</math></b>	<b>32</b>
<b>30 Проблема вложения регулярных языков</b>	<b>33</b>
<b>31 Минимальный КДА для регулярных языков. Эффективная минимизация числа состояний.</b>	<b>34</b>
<b>32 Теорема о минимальном автомате.</b>	<b>39</b>
<b>33 Замкнутость класса регулярных языков относительно регулярных операций.</b>	<b>42</b>
<b>34 Замыкание класса регулярных языков по пересечению.</b>	<b>43</b>
<b>35 Использование свойства замыкания пересечения регулярных языков.</b>	<b>44</b>
<b>36 Замыкание класса регулярных языков по вычитанию.</b>	<b>44</b>
<b>37 Замыкание класса регулярных языков по дополнению.</b>	<b>45</b>
<b>38 Замыкание класса регулярных языков по обращению</b>	<b>45</b>
<b>39 Гомоморфизмы. Замыкание по гомоморфизму. Обратный гомоморфизм. Замыкание класса регулярных языков по обратному гомоморфизму.</b>	<b>46</b>

<b>40 Контекстно-свободные грамматики.</b>	<b>48</b>
<b>41 Деревья синтаксического разбора.</b>	<b>48</b>
<b>42 Левые и правые выводы. Деревья синтаксического разбора. Сечение, крона сечения.</b>	<b>48</b>
<b>43 Теорема о взаимном соответствии деревьев разбора, левых и правых выводов.</b>	<b>51</b>
<b>44 Неоднозначные грамматики. Существенно неоднозначный язык.</b>	<b>53</b>
<b>45 Нормальные формы КС-грамматик. Нормальная форма Хом- ского.</b>	<b>54</b>
<b>46 Алгоритм удаления бесполезных переменных.</b>	<b>54</b>
<b>47 Алгоритм удаления <math>\epsilon</math>-правил.</b>	<b>55</b>
<b>48 Алгоритм удаления единичных правил.</b>	<b>57</b>
<b>49 Нормальная форма Хомского.</b>	<b>58</b>
<b>50 Автоматы с магазинной памятью. Определение. Мгновен- ная конфигурация МПА. Функционирование МП-автомата. Детерминированные МП-автоматы.</b>	<b>59</b>
<b>51 Языки для МП-автоматов. Теорема об эквивалентности опре- деления языков МПА по конечному состоянию и опустоше- нием стека.</b>	<b>61</b>
<b>52 Эквивалентность МП-автоматов и КС-грамматик.</b>	<b>63</b>
<b>53 Лемма о накачке для КС-языков. Формулировка. Приложе- ния.</b>	<b>64</b>
<b>54 Свойства КС-языков. Разрешающие свойства. Свойства за- мыкания</b>	<b>65</b>
<b>55 Машины Тьюринга. Мгновенные конфигурации МТ.</b>	<b>70</b>
<b>56 Рекурсивно-перечислимые языки. Рекурсивные языки.</b>	<b>72</b>
<b>57 Свойства замыкания для рекурсивных и рекурсивно-перечислимых языков.</b>	<b>72</b>
<b>58 Универсальный язык. Неразрешимые проблемы.</b>	<b>73</b>

# 1 Общая структура компиляторов. Лексический анализ. Синтаксический анализ. Семантический анализ. Оптимизация кода. Генерация кода.

Структура компилятора



## 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:
  - (а) Увеличение скорости работы программы;
  - (б) Уменьшение объёма используемой памяти;
  - (с) 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 Формальные языки. Основные понятия и определения формальных языков.
- 3 Алфавит. Цепочка символов (или слово) в алфавите  $\Sigma$ . Операции над цепочками символов.

#### 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 *сплением*) 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  $a$ :  $a\epsilon = \epsilon a = a$ .

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:  $\epsilon^R = \epsilon$ .

##### 3. Возвведение в степень

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

$$\alpha^n = \underbrace{\alpha \dots \alpha}_{n}.$$

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

Дополнительную информацию смотрите на слайдах 40-42 первой презентации.

## 4 Язык в алфавите $\Sigma$ . Операции над языками.

• **Опр.** Обозначим через  $\Sigma^*$  множество, содержащее все цепочки в алфавите  $\Sigma$ , включая пустую цепочку  $\epsilon$ .

Например, если  $\Sigma = \{0, 1\}$ , то  $\Sigma^* = \{\epsilon, 0, 1, 00, 11, 01, 10, 000, 001, 011, \dots\}$ .

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. Контатенация

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

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

Обозн.:  $L_3 = L_1L_2$

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.  $L^0 = \{\epsilon\}$

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 Методы представления бесконечных языков.

### 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.

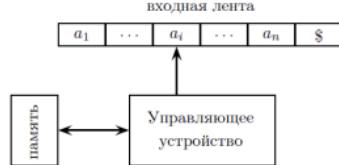
## 6 Спецификация языка.

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

## 7 Механизм распознавания (распознаватель). Язык, определяемый распознавателем.

- 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 зацикливается.
- *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 зацикливается.
- **Язык, определяемый распознавателем** — это множество всех цепочек, которые он допускает.



## 8 Грамматики. Порождающая грамматика. Выводимость цепочки. Язык грамматики.

**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  $\alpha \rightarrow \beta_1 | \beta_2 | \dots | \beta_n$ .

**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  $\alpha \rightarrow_\sigma \beta$ ), 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 ), 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  $\Rightarrow_n$  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 Классификация грамматик и языков по Хомскому.

### Классификация грамматик и языков по Хомскому

- Тип грамматики определяется типом ограничений на вид правил вывода.
- Всего определено четыре типа грамматик:  
типа 0, типа 1, типа 2, типа 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  $S \rightarrow \epsilon$ , 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 (К3)*, 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  $S \rightarrow \epsilon$ , 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** (*KC*), 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 К3-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:

$$\text{Регулярные неукорачивающие} \subset \text{КС неукорачивающие} \subset \text{К3} \subset \text{Тип 0}$$

## 10 Эквивалентность неукорачивающих и КЗ-грамматик.

### 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 КС-грамматика

**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 Праволинейная и леволинейная грамматики. Эквивалентность этих грамматик.

**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 леволинейная грамматика  $G_2$ , такая что  $L = L(G_2)$ .

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

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

## 13 Регулярные языки. Автоматные грамматики.

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

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

$A \rightarrow a$  либо  $A \rightarrow aB$  (для леволинейной, 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 леволинейной) 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 Иерархия языков.

### 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 К3-языком, но существуют К3-языки, которые не являются КС-языками, например:

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

3) Каждый К3-язык является языком типа 0 (т. е. рекурсивно перечислимым языком), но существуют языки типа 0, которые не являются К3-языками.

Из утверждения 6 следует иерархия классов языков:

$$Tun 3 (\text{Регулярные}) \subset Tun 2 (\text{КС}) \subset Tun 1 (\text{К3}) \subset Tun 0$$



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 Алгоритмическая неразрешимость проблемы отнесения языка к определенному типу по классификации Хомского.

- **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 Примеры грамматик и языков.

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

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

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

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  $\begin{aligned} S &\rightarrow A\perp \mid B\perp && \text{леволинейная;} \\ A &\rightarrow a \mid Ba \\ B &\rightarrow b \mid Bb \mid Ab \end{aligned}$

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:  $L = \{\omega\perp \mid \omega \in \{a, b\}^+, aa \notin \omega\}$ .

### Контекстно-свободные

1. Грамматика  $\begin{aligned} S &\rightarrow aQb \mid accb \\ Q &\rightarrow cSc \end{aligned}$

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

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

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):

$$\begin{aligned} S &\rightarrow A \mid \epsilon \\ A &\rightarrow aAa \mid bAb \mid aa \mid bb \end{aligned}$$

### **Неукорачивающие и контекстно-зависимые**

1. Грамматика:

$$\begin{array}{lcl} S & \rightarrow & aSBC \mid abC \\ CB & \rightarrow & BC \\ BB & \rightarrow & bb \\ bC & \rightarrow & bc \\ cC & \rightarrow & cc \end{array}$$

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:

$$\begin{array}{lcl} S & \rightarrow & aSBC \mid abC \\ CB & \rightarrow & CD \\ CD & \rightarrow & BD \\ BD & \rightarrow & BC \\ bB & \rightarrow & bb \\ bC & \rightarrow & bc \\ cC & \rightarrow & cc \end{array}$$

**1. Грамматика типа 0**

$$\begin{array}{lcl} S & \rightarrow & SS \\ SS & \rightarrow & \epsilon \end{array}$$

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

2. Грамматика

$$\begin{array}{lcl} S & \rightarrow & SS \\ SS & \rightarrow & Sa \mid S \end{array}$$

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

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

## **17 Регулярные множества и регулярные выражения. Алгебраические законы для регулярных выражений.**

Определение регулярного выражения

- **Опр.** Пусть  $\Sigma$  — алфавит, не содержащий символов \*, +, ε, ∅, (, ).  
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)  $\epsilon$  есть регулярное выражение;  $L(\epsilon) = \{\epsilon\}$
- 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)  $\{\epsilon\}$  – 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 Уравнения с регулярными коэффициентами. Алгоритм решения стандартной системы уравнений с регулярными коэффициентами.

### 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  $\beta_0 + \beta_1 X_{i+1} + \dots + \beta_n X_n$ ;  $\beta_i$  - 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:

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

$$\dots$$

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

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 Регулярные множества и праволинейные грамматики.

- 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:  $S_1 \Rightarrow_{G_4}^+ xS_2 \Rightarrow_{G_4}^+ xy$ , где  $w = xy$  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  $S_2 \Rightarrow_{G_Q}^+ y$ .

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, оканчивающейся 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 РВ  $f(S)$ , которое определяет язык  $L(G)$ .

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

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

## 20 Конечные автоматы и регулярные множества. Язык автомата.

- Класс регулярных множеств – наименьший класс языков, содержащий  $\emptyset$ ,  $\{\epsilon\}$ ,  $\{a\}$  для всех  $a$  и замкнутый относительно операций объединения, конкатенации и итерации.
- Регулярное множества – множества, определяемые регулярными выражениями.
- Регулярное множества – языки, порождаемые праволинейными грамматиками.
- Регулярное 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” являются синонимами.

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

## Функция переходов

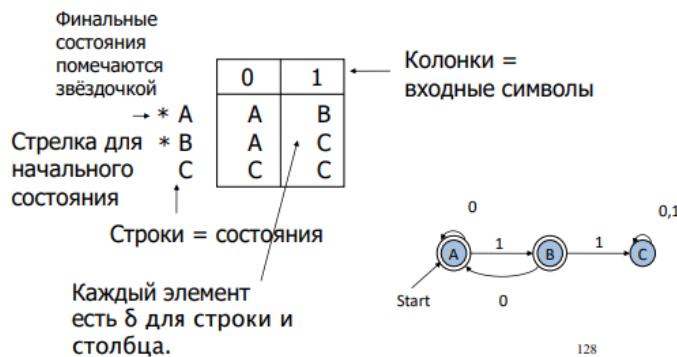
- Имеет два аргумента: состояние и входной символ.
- $\delta(q, a)$  = состояние, в которое КДА переходит, если он в состоянии  $q$  получает на вход символ  $a$ .
- **Замечание:** всегда есть следующее состояние – добавим **мёртвое состояние** если нет переходов (Пример далее).
- Форма задания: таблица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



128

## 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:**  $L(A) = \{w \mid \delta(q_0, w) \in F\}$ .

## 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

- Стока  $w$  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 Эквивалентность автоматных и праволинейных языков

- 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  $r \in F$ , то  $P$  содержит правило  $r \rightarrow \epsilon$ .  
Каждый шаг вывода в грамматике  $G$  имитирует торт работы автомата  $A$ .  
Индукция по  $i$  – длине вывода.

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

Индукция.  $s \Rightarrow^i x$  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 Свойства регулярных языков. Свойства замкнутости. Разрешимые свойства.

### Свойства замкнутости для регулярных языков

- Замкнутость класса регулярных языков относительно регулярных операций
- Замкнутость по дополнению
- Замкнутость по пересечению
- Замкнутость по вычитанию
- Замкнутость по обращению (слов)
- Замкнутость по гомоморфизму
- Замкнутость по обратному гомоморфизму

### Разрешимые свойства для регулярных языков - ИТОГИ

- **Следующие проблемы разрешимы для регулярный языков:**
  - Проблема принадлежности языку:  $w \in L?$
  - Проблема пустоты:  $L=\emptyset?$
  - Проблема эквивалентности:  $L = M?$
  - Проблема вложения языков:  $L \subseteq M?$
  - Проблема бесконечности языка:  $|L| = \infty?$

## 23 Проблема принадлежности слова языку.

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 Проблема пустоты регулярного языка.

### 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 Проблема бесконечности регулярного языка.

### Проблема бесконечности

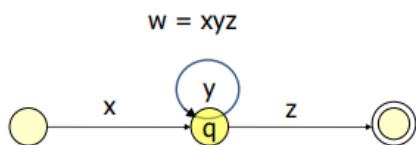
- Является данный регулярный язык бесконечным?
- Рассмотрим КДА для языка.
- **Ключевая идея:** если КДА имеет  $n$  состояний и язык содержит какую либо строку длины  $n$  или более, то язык бесконечный.
- В противном случае, язык, безусловно, конечный.
  - Можно ограничиться строками длины  $n$  или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

## 26 Бесконечность регулярного языка.

### 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)



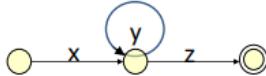
Тогда  $xy^iz$  является языком для всех  $i \geq 0$ .

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

Число состояний КДА для  $L$

для каждой строки  $w$  из  $L$  длины  $\geq n$

мы можем записать ее как  $w = xyz$  так, что:

Метки вдоль первого цикла пути, помеченного  $w$

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

## 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 $M$

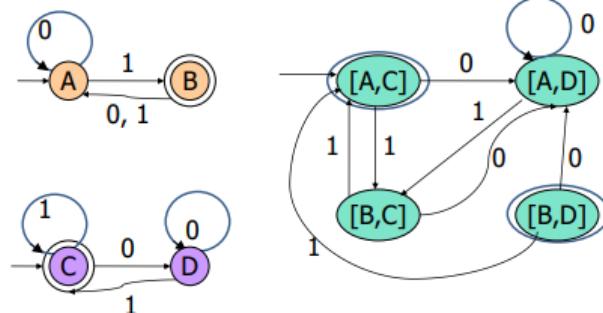
### Построение произведения автоматов $L$ и $M$

- 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

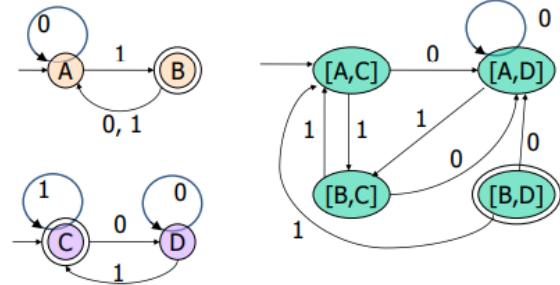
- По данным регулярным языкам  $L$  и  $M$  определить,  $L \subseteq M$ ?
- Алгоритм также использует конструкцию произведения автоматов.
- Как нужно определить финальное состояние  $[q, r]$  автомата-произведения, чтобы его язык был пуст т.и.т.т., когда  $L \subseteq M$ ?

**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



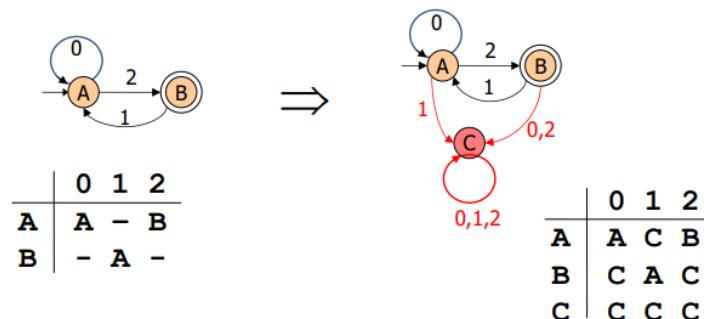
Замечание: только одно финальное состояние недостижимо, язык для автомата произведения пуст, т.о. вложение выполняется.

### 31 Минимальный КДА для регулярных языков. Эффективная минимизация числа состояний.

Минимальный КДА для регулярных языков

- В принципе, т.к. мы можем протестировать КДА на эквивалентность, то мы можем по данному КДА A найти КДА с наименьшим числом состояний принимающий  $L(A)$ .
- Можно проверить все КДА на эквивалентность с A.
- Но этот алгоритм, конечно, никуда не годится.

Доопределение КДА до полного



## Эффективная минимизация числа состояний

- Построим таблицу со всеми парами состояний.
- Если мы найдем строку, которая *различает* два состояния (переводит точно одно состояние из двух в заключительное), то пометим эту пару.
- В противном случае состояния в паре будут *неразличимыми*, и они могут быть объединены в одно состояние.
- Алгоритм является рекурсивным по длине кратчайших различающих строк.

## Минимизация числа состояний – (2)

- **Базис:** Пометим пары с точно одним финальным состоянием.
- **Индукция:** пометим  $[q, r]$ , если для некоторого входного символа  $a$ ,  $[\delta(q, a), \delta(r, a)]$  является помеченной.
- После того, как больше нельзя пометить ни одно состояние, непомеченные пары состояний объявляем эквивалентными и объединяем в одно состояние.

## Транзитивность “Неразличимости”

- Если состояние  $p$  неразличимо от  $q$ , и  $q$  неразличимо от  $r$ , то  $p$  неразличимо от  $r$ .
- **Доказательство:** Результат (принятие или нет) входа  $w$  для  $p$  и  $q$  будет один и тот же, и результат для  $q$  и  $r$  на  $w$  тот же, тогда аналогичный результат будет для  $p$  и  $r$ .

## Построение минимального КДА

- Предположим, что  $q_1, \dots, q_k$  - неразличимые состояния.
- Заменим их одним их *представляющим* состоянием  $q$ .
- Тогда  $\delta(q_1, a), \dots, \delta(q_k, a)$  есть все неразличимые состояния.
- **NB!**: в противном случае, нам следовало бы пометить по крайней мере на одну пару состояний больше.
- Пусть  $\delta(q, a) = \text{состояние - представитель для этой группы.}$

**Пример:** минимизация числа состояний



**Пример – продолжение**

	r	b				
→	A   B   C					
	B   D   E					
	C   D   F					
	D   D   G					
	E   D   G					
	*F   D   C					
	*G   D   G					

	G	F	E	D	C	B
A	x	x				
B	x	x				
C	x	x				
D	x	x				
E	x	x				
F						

Вход r бесполезен,  
т.к. пара [B, D]  
непомечена.

## Пример – продолжение

	r	b
A	B	C
B	D	E
C	D	F
D	D	G
E	D	G
*	F	D
*	G	D

	G	F	E	D	C	B
A	x	x	x	x	x	
B	x	x	x	x	x	
C	x	x				
D	x	x				
E	x	x				
F	x					

Но вход b различает состояния из {A,B,F}  
от состояний из {C,D,E,G}. Например, [A, C]  
помечается, т.к. [C, F] помечено.

## Пример – продолжение

	r	b
A	B	C
B	D	E
C	D	F
D	D	G
E	D	G
*	F	D
*	G	D

	G	F	E	D	C	B
A	x	x	x	x	x	
B	x	x	x	x	x	
C	x	x	x	x	x	
D	x	x				
E	x	x				
F	x					

[C, D] и [C, E] помечаются,  
т.к. есть переход по b в  
Помеченную пару [F, G].

## Пример – продолжение

	r	b		G	F	E	D	C	B	
→	A	B	C	A	x	x	x	x	x	x
	B	D	E	B	x	x	x	x	x	x
	C	D	F	C	x	x	x	x		
	D	D	G	D	x	x				
	E	D	G	E	x	x				
	*	F	D	C	F	x				
	*	G	D	G						

[A, B] помечается, т.к.  
имеются переходы по г  
в помеченную пару [B, D].

[D, E] никогда не будут  
помечены, т.к. по обоим  
входам они переходят в  
одно и то же состояние.

## Пример – заключение

	r	b		r	b		G	F	E	D	C	B	
→	A	B	C	A	B	C	A	x	x	x	x	x	x
	B	D	E	B	H	H	B	x	x	x	x	x	x
	C	D	F	C	H	F	C	x	x	x	x		
	D	D	G	H	H	G	D	x	x				
	E	D	G				E	x	x				
	*	F	D	C	*	F	H	C					
	*	G	D	G	*	G	H	G					

Заменим D и E на H.  
Результат – КДА с минимальным  
числом состояний.

## Удаление недостижимых состояний

- К сожалению, комбинирование неразличимых состояний может оставить нас с недостижимыми состояниями в минимальном КДА.
- Таким образом, до или после, нужно удалить состояния, которые недостижимы из начального состояния.

## Удаление недостижимых состояний

- **Опр.** Состояние  $q$  конечного автомата  $M = (Q, \Sigma, \delta, q_0, F)$  называется недостижимым, если не найдется цепочки  $\alpha \in \Sigma^*$ , приводящей автомат из начального состояния  $q_0$  в состояние  $q$ .  
Т.е.,  $\nexists \alpha \in \Sigma^*: \delta(q_0, \alpha) = q$

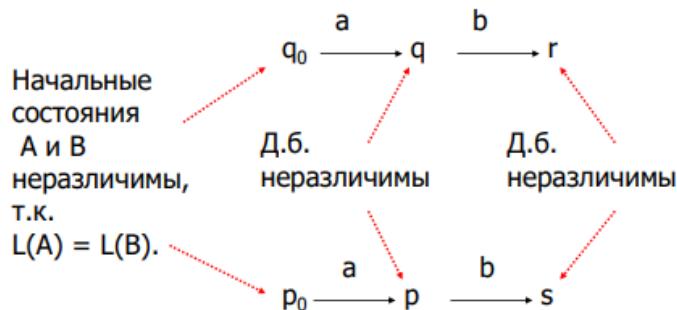
## 32 Теорема о минимальном автомате.

- **Теорема.** (О минимальном автомате)
  - Для любого КДА автомата  $M = (Q, \Sigma, \delta, q_0, F)$  найдется эквивалентный КДА  $A = (Q, \Sigma, \delta, F)$  без неразличимых состояний, допускающий тот же язык. Причём, если  $M$  –автомат без недостижимых состояний, то не существует КДА, допускающего  $L(M)$  и имеющего состояний меньше, чем у автомата  $A$ .

## Доказательство: Нет другого «меньшего» автомата

- Пусть А – наш минимизированный КДА; пусть В – его «меньший» эквивалент.
- Предположим, что А эквивалентен В.
- Рассмотрим объединённый комбинированный автомат с состояниями из А и В (неважно какое состояние начальное, но заключительные состояния – это те, что были в А и В).
- Используем “различимость” в ее контрапозитивной форме:
  - То есть, если  $w$  различает  $\delta(q, a)$  от  $\delta(p, a)$ , то, конечно,  $aw$  различает  $q$  от  $p$ .
  - Таким образом, если состояния  $q$  и  $p$  **неразличимы**, то их последователи по некоторому  $a$ :  $\delta(q, a)$  и  $\delta(p, a)$  – также неразличимы.

## Наследование неразличимости



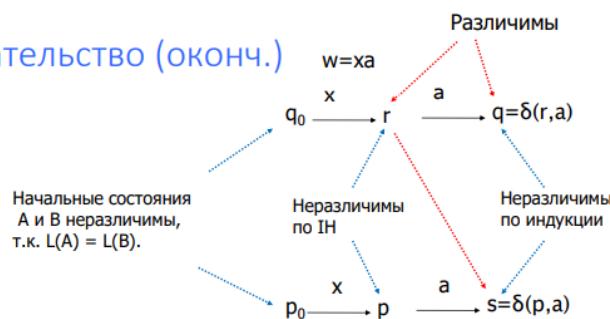
## Индуктивная гипотеза

- Каждое состояние  $q$  автомата А неразличимо с некоторым состоянием автомата В.
- Индукция проводится по длине кратчайшей строки, переводящей начальное состояние автомата А в  $q$ .

## Доказательство

- **Базис:** Начальные состояния А и В неразличимы, т.к.  $L(A) = L(B)$ .
- **IH:** Пусть строка длины  $n < |w|$  переводит А в некоторое состояние  $r$ , которое неразличимо с некоторым состоянием  $p$  автомата В.
- **Индукция:** Предположим, что  $w = xa$  есть кратчайшая строка, переводящая автомат А в состояние  $q$ .
- Согласно IH, строка  $x$  переводит А в некоторое состояние  $r$ , которое неразличимо с некоторым состоянием  $p$  автомата В.
- Тогда  $\delta_A(r, a) = q$  неразличимо с  $\delta_B(p, a)$ .

## Доказательство (оконч.)



- Однако, два состояния автомата А не могут быть неразличимы с тем же состоянием в В, или они были бы неразличимы одно от другого.
  - Это нарушает транзитивность "неразличимости."
- Таким образом, В имеет, по крайней мере, столько же состояний, что и А.

### 33 Замкнутость класса регулярных языков относительно регулярных операций.

Замкнутость класса регулярных языков относительно регулярных операций

- **Теорема.** Класс леволинейных языков замкнут относительно регулярных операций.

#### • Док-во.

Пусть  $L$  и  $L'$  - леволинейные языки, определенные с помощью грамматик  $G = (N, \Sigma, P, S)$  и  $G' = (N', \Sigma', P', S')$  соответственно.

Пусть  $N \cap N' = \emptyset$

1. Язык  $L \cup L'$  определяется грамматикой

$$G_1 = (N \cup N' \cup \{S_0\}, \Sigma \cup \Sigma', P \cup P' \cup \{S_0 \rightarrow S, S_0 \rightarrow S'\}, S_0)$$

#### Док-во, прод.

2. Заменим в  $P'$  правила вида  $A \rightarrow \alpha$ , где  $A \in N'$ ,  $\alpha \in \Sigma'^*$  на правила  $A \rightarrow S\alpha$ . Полученное множество правил обозн. через  $P_1$ .

Грамматика  $G_2 = (N \cup N', \Sigma \cup \Sigma', P \cup P_1, S')$  порождает язык  $LL'$ .

Если  $\beta \in L'$ , то  $S' \Rightarrow_{G_1} \beta$ . Но тогда  $S' \Rightarrow_{G_2} S\beta$ .

Таким образом,  $\forall \alpha \in L \quad S' \Rightarrow_{G_2} S\beta \Rightarrow_{G_2} \alpha\beta$ . Все конкатенации из  $LL'$  можно вывести в  $G_2$  из  $S'$ .

Никакие другие слова вывести из  $S'$  нельзя.

#### Док-во, финал

3. Заменим в  $P'$  правила вида  $A \rightarrow \alpha$ , на  $A \rightarrow S'\alpha$ .

Тогда грамматика  $G_3 = (N' \cup \{S_0\}, \Sigma', P' \cup \{S_0 \rightarrow \varepsilon, S_0 \rightarrow S'\}, S_0)$  порождает язык  $L'^*$ .

В  $G_3$  возможны выводы:

$$\begin{aligned} S_0 &\Rightarrow_{G_3} \varepsilon \\ S_0 &\Rightarrow_{G_3} S' \\ S_0 &\Rightarrow_{G_3}^* \alpha \quad \forall \alpha \in L' \\ S_0 &\Rightarrow_{G_3}^* S'\alpha \quad \forall \alpha \in L' \end{aligned}$$

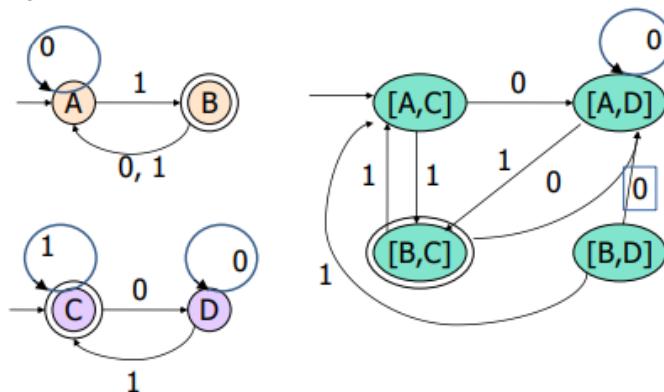
Следовательно, из  $S_0$  можно вывести любое слово составленное из слов языка  $L'$ , т.е.  $L(G_3) = L'^*$

### 34 Замыкание класса регулярных языков по пересечению.

#### Замыкание по пересечению

- **Теорема.** Если  $L$  и  $M$  – регулярные языки, то таковым будет язык  $L \cap M$ .
- **Доказательство:** Пусть  $A$  и  $B$  – КДА, определяющие языки  $L$  и  $M$ , соответственно.
- Построим автомат  $C$  – произведение автоматов  $A$  и  $B$ .
- Сделаем финальными состояниями  $C$  пары, состоящие из финальных состояний  $A$  и  $B$ .

**Пример:** КДА-произведение для пересечения



### 35 Использование свойства замыкания пересечения регулярных языков.

**Пример:** Использование свойства замыкания пересечения

- Мы доказали с использованием леммы о накачке, что  $L_1 = \{0^n 1^n \mid n \geq 0\}$  - не регулярный язык.
- $L_2$  = множество строк с равным числом 0 и 1 тоже не РЯ, но этот факт трудно доказать напрямую.
- Предположим, что  $L_2$  – регулярный язык.
- Регулярные языки замкнуты относительно  $\cap$ .
- Если  $L_2$  регулярный, то  $L_2 \cap L(0^* 1^*) = L_1$  тоже должен быть регулярным, но мы уже установили, что это не так.

### 36 Замыкание класса регулярных языков по вычитанию.

**Замыкание по вычитанию**

- **Теорема.** Если  $L$  и  $M$  – регулярные языки, то таким является язык  $L - M$  (строки из  $L$ , но не из  $M$ ).
- **Доказательство:** Пусть  $A$  и  $B$  - КДА, языками которых являются  $L$  и  $M$  соответственно.
- Построим автомат  $C$  – произведение автоматов  $A$  и  $B$ .
- Сделаем финальными состояниями автомата  $C$  пары, которые состоят из финальных состояний  $A$  и нефинальных  $B$ .
- Автомат  $C$  допускает точно строки из  $L - M$ .
- Следовательно,  $L - M$  – регулярный язык.

### 37 Замыкание класса регулярных языков по дополнению.

#### Замыкание по дополнению

- **Дополнением** языка  $L$  (по отношению к алфавиту  $\Sigma$ , так что  $\Sigma^*$  содержит  $L$ ) является язык  $\Sigma^* - L$ .
- Так как  $\Sigma^*$  очевидно регулярный и регулярные языки замкнуты по операции вычитания, дополнение регулярного языка всегда будет регулярным языком.

### 38 Замыкание класса регулярных языков по обращению

#### Замыкание по обращению

- Пусть дан язык  $L$ , и  $L^R$  - есть множество строк, обращение которых есть в  $L$ .
- **Пример:**  $L = \{0, 01, 100\}$ ;  $L^R = \{0, 10, 001\}$ .
- Если  $L$  – регулярный язык, то  $L^R$  – тоже регулярный язык.
- **Доказательство:** Пусть  $E$  есть РВ для  $L$ . Мы покажем как обратить  $E$ , чтобы произвести РВ  $E^R$  для  $L^R$ .

#### Обращение регулярного выражения

- **Базис:** Если  $E$  символ  $a$ ,  $\epsilon$ , или  $\emptyset$ , то  $E^R = E$ .
- **Индукция:** если  $E$  есть
  - $F+G$ , то  $E^R = F^R + G^R$ .
  - $FG$ , то  $E^R = G^R F^R$
  - $F^*$ , то  $E^R = (F^R)^*$ .

39 Гомоморфизмы. Замыкание по гомоморфизму. Обратный гомоморфизм. Замыкание класса регулярных языков по обратному гомоморфизму.

### Гомоморфизмы

- **Гомоморфизмом** на алфавите является функция, которая каждый символ этого алфавита заменяет некоторой строкой.
- **Пример:**  $h(0) = ab; h(1) = \epsilon$ .
- Расширим функцию на строки  $h(a_1\dots a_n) = h(a_1)\dots h(a_n)$ .
- **Пример:**  $h(01010) = ababab$ .

### Замыкание по гомоморфизму

- Если  $L$  — регулярный язык, и  $h$  есть гомоморфизм на его алфавите, то  $h(L) = \{h(w) \mid w \in L\}$  также является регулярным языком.
- **Доказательство:** Пусть  $E$  — РВ для  $L$ .
- Применим  $h$  к каждому символу в  $E$ .
- Язык для результирующего РВ есть  $h(L)$ .

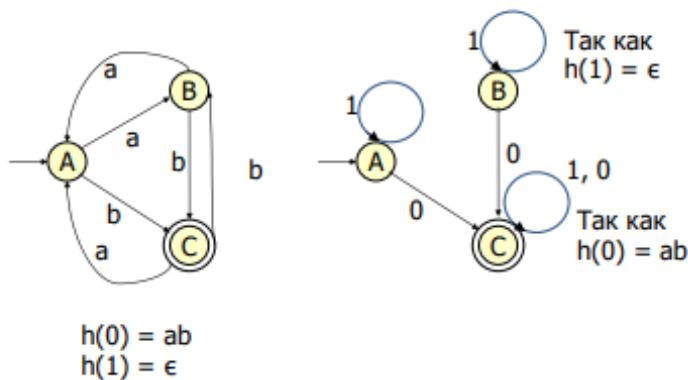
### Обратный гомоморфизм

- Пусть  $h$  — гомоморфизм и  $L$  есть язык, алфавитом которого является результирующий язык функции  $h$ .
- Определим:
- $h^{-1}(L) = \{w \mid h(w) \text{ есть в } L\}$ .

## Доказательство замыкания по обратному гомоморфизму

- Начнём с КДА А для L.
- Построим КДА В для  $h^{-1}(L)$ :
  - Тем же самым множеством состояний, что и у А,
  - Тем же начальным состоянием,
  - Тими же финальными состояниями.
  - Входной алфавит = символы, к которым применяется гомоморфизм  $h$ .
- Переходы для В вычисляются путём применения  $h$  к входному символу  $a$  и просмотра того, куда А мог бы пройти по последовательности символов  $h(a)$ .
- Формально,  $\delta_B(q, a) = \delta_A(q, h(a))$ .

**Пример:** построение обратного гомоморфизма



**Доказательство – обратный гомоморфизм**

- Индукция по  $|w|$  показывает, что
$$\delta_B(q_0, w) = \delta_A(q_0, h(w)).$$
- Таким образом, В принимает  $w$  т.и т.т., когда А принимает  $h(w)$ .

## 40 Контекстно-свободные грамматики.

### 41 Деревья синтаксического разбора.

#### Деревья синтаксического разбора

- *Деревья синтаксического разбора* – это деревья, помеченные символами соответствующей КС-грамматики.
- *Листья*: помечены терминалами или  $\epsilon$ .
- *Внутренние узлы*: помечены переменными.
  - Дочерние помечены телом продукции для родителя.
- *Корень*: должен быть помечен начальным символом.

## 42 Левые и правые выводы. Деревья синтаксического разбора. Сечение, крона сечения.

#### Левые выводы

- Будем говорить, что  $wA\alpha \Rightarrow_{lm} w\beta\alpha$  если  $w$  – строка только терминалов и  $A \rightarrow \beta$  - продукция.
- Также,  $\alpha \Rightarrow_{lm}^* \beta$  если  $\alpha$  становится  $\beta$  в результате последовательности 0 или более  $\Rightarrow_{lm}$  шагов.
- $lm$  = leftmost

#### Пример: Левые выводы

- Грамматика сбалансированных скобок:
- $S \rightarrow SS \mid (S) \mid ()$
- $S \Rightarrow_{lm} SS \Rightarrow_{lm} (S)S \Rightarrow_{lm} ((())S \Rightarrow_{lm} ((())()$
- Таким образом,  $S \Rightarrow_{lm}^* ((())()$
- $S \Rightarrow SS \Rightarrow S() \Rightarrow (S()) \Rightarrow ((())()$  есть вывод, но не левый вывод.

## Правые выводы

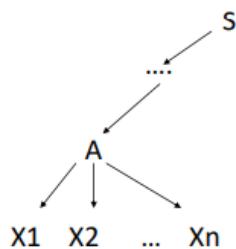
- Будем говорить, что  $\alpha Aw \Rightarrow_{rm} \alpha\beta w$  если  $w$  – строка только терминалов и  $A \rightarrow \beta$  - продукция.
- Также,  $\alpha \Rightarrow_{rm}^* \beta$  если  $\alpha$  становится  $\beta$  в результате последовательности 0 или более  $\Rightarrow_{rm}$  шагов.
- $rm = rightmost$

Пример: Правые выводы

- Грамматика сбалансированных скобок :
  - $S \rightarrow SS \mid (S) \mid ()$
  - $S \Rightarrow_{rm} SS \Rightarrow_{rm} S() \Rightarrow_{rm} (S)() \Rightarrow_{rm} ((())()$
  - Таким образом,  $S \Rightarrow_{rm}^* ((())()$
  - $S \Rightarrow SS \Rightarrow SSS \Rightarrow S()S \Rightarrow ()()S \Rightarrow ()()()$  не левый и не правый вывод.

## Деревья выводов

- $G = (N, \Sigma, P, S)$
- $A \rightarrow X_1, X_2, \dots, X_n$
- .....
- 



## Дерево вывода

- **Опр.** Помеченное упорядоченное дерево  $D$  называется **деревом вывода** (или **деревом разбора**) в КС-грамматике  $G = (N, \Sigma, P, S)$ , если выполнены следующие условия:
  1. Корень дерева  $D$  помечен  $S$ .
  2. Каждый узел имеет метку из  $N \cup \Sigma \cup \{\varepsilon\}$
  3. Если узел имеет хотя бы одного потомка, метка этого узла – нетерминальный символ
  4. Если  $D_1, D_2, \dots, D_k$  – поддеревья с корнями  $X_1, X_2, \dots, X_k$ , которые являются прямыми потомками, то в множестве правил  $P$  присутствует правило  $X_2 \dots X_k$ . При этом поддерево  $D_i$  может быть узлом с меткой корня  $X_i = \varepsilon$  только в случае, если  $X_i$  – единственный потомок узла  $A$  и в  $P$  присутствует правило  $A \rightarrow \varepsilon$ .

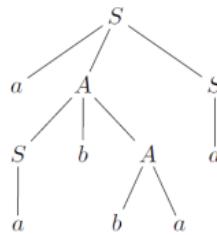
## Пример дерева разбора

Рассм.  $G = (\{S, A\}, \{a, b\}, P, S)$ ,

где

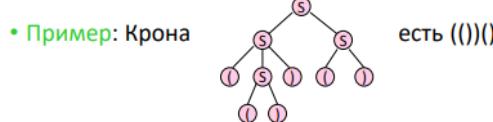
$P: S \rightarrow aAS \mid a$

$A \rightarrow SbA \mid ba \mid SS$



$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aSbAa \Rightarrow aabAa \Rightarrow aabbaa$

- **Опр.** Стока, являющаяся конкатенацией меток листьев в порядке слева направо, называется **кроной** дерева вывода (синтаксического разбора).
  - То есть, в порядке определенного обхода.

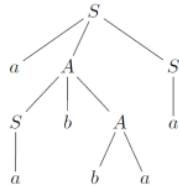


- **Оп.** Сечением дерева D называется такое множество C вершин дерева D, что
  - (1) Никакие две вершины из C не лежат на одном пути в D,
  - (2) Каждая вершина из C принадлежит хотя бы одному пути к корню дерева D.
  - (3) Ни одну вершину дерева D нельзя добавить в C, не нарушив свойства (1).

**Кроной сечения** называют строку – конкатенацию вершин сечения

- Примеры:

- Корень дерева есть сечение.
- Множество листьев есть сечение.
- Крона дерева есть сечение.
- Для дерева на рис.:  $(a, A, a)$ -сечение



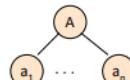
### 43 Теорема. о взаимном соответствии деревьев разбора, левых и правых выводов.

#### Деревья разбора, левый и правый выводы

- **Теорема.** Деревья разбора, левый и правый выводы находятся во взаимном соответствии.
- Мы докажем:
  1. Если есть дерево разбора с корнем A и кроной w, то  $A \Rightarrow^*_{\text{lm}} w$ .
  2. Если  $A \Rightarrow^*_{\text{lm}} w$ , то существует дерево разбора с корнем A и кроной w.

#### Доказательство – Часть 1

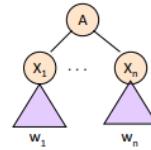
- Индукция по **высоте** (длине наиболее длинного пути от корня) дерева.
- **Базис:** высота 1. Дерево выглядит так:
  - $A \rightarrow a_1 \dots a_n$  должно быть продукцией.
  - Таким образом,  $A \Rightarrow^*_{\text{lm}} a_1 \dots a_n$ .



## Часть 1 – Индукция

- Предположим (1) выполняется для деревьев высоты  $< h$ , и пусть дерево имеет высоту  $h$ :

- По ИН,  $X_i \Rightarrow^*_{\text{lm}} w_i$ .
  - Заметим: Если  $X_i$  – терминал, то  $X_i = w_i$ .
- Таким образом,  $A \Rightarrow_{\text{lm}} X_1 \dots X_n \Rightarrow^*_{\text{lm}} w_1 X_2 \dots X_n \Rightarrow^*_{\text{lm}} w_1 w_2 X_3 \dots X_n \Rightarrow^*_{\text{lm}} \dots \Rightarrow^*_{\text{lm}} w_1 \dots w_n$ .

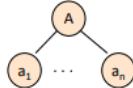


## Доказательство: Часть 2

- По данному левому выводу терминальной строки нам нужно доказать существование дерева разбора.
- Доказательство проводится индукцией по длине вывода.

## Часть 2 – Базис

- Если  $A \Rightarrow_{\text{lm}} a_1 \dots a_n$  вывод за один шаг, то должно существовать дерево разбора:



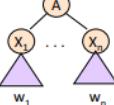
- Если  $A \Rightarrow_{\text{lm}} \varepsilon$ , то



## Часть 2 – Индукция

- ИН:** Предположим, что (2) выполняется для выводов за число шагов меньшее, чем  $k$ ,
- Пусть  $A \Rightarrow^*_{\text{lm}} w$  будет выводом за  $k$  шагов.
- Первый шаг есть  $A \Rightarrow_{\text{lm}} X_1 \dots X_n$ .
- Ключевой момент:**  $w$  может быть раздelenо так, что первая часть выводится из  $X_1$ , следующая – из  $X_2$ , и.т.д.
  - Если  $X_i$  – терминал, то  $w_i = X_i$ .

## Индукция – (2)

- То есть,  $X_i \Rightarrow^*_{\text{им}} w_i$  для всех  $i$  таких, что  $X_i$  есть переменная и вывод имеет меньше, чём  $k$  шагов.
- По **IH**, Если  $X_i$  – переменная, то существует дерево разбора с корнем  $X_i$  и кроной  $w_i$ .
- Таким образом, существует дерево разбора:  


```
graph TD; A((A)) --- X1((X1)); A --- dots[...]; A --- Xn((Xn)); X1 --- w1[w1]; dots --- wn[wn];
```
- Корень даёт первую продукцию вывода,  $X_i$  каждый  $X_i$  есть либо терминал, либо корень дерева с выводом  $w_i$
- Этим доказывается индуктивный шаг и, следовательно, если есть левый вывод  $w$  из  $A$ , то существует дерево разбора с корнем  $A$  и кроной  $w$ .

## Деревья разбора и правые выводы

- Приведенные утверждения имеют зеркальное отражение для правых выводов.
- Если есть правый вывод  $w$  из  $A$ , то существует дерево разбора с корнем  $A$  и кроной  $w$ , и наоборот.
- Доказательство – аналогично.

## 44 Неоднозначные грамматики. Существенно неоднозначный язык.

- Таким образом, эквивалентным определением для “неоднозначной грамматики” служит:
  1. Существует строка языка, которая имеет два различных левых вывода.
  2. Существует строка языка, которая имеет два различных правых вывода.
- **Оп.** КС-язык  $L$  называется *существенно неоднозначным*, если все его грамматики неоднозначны.
- Если хотя бы одна грамматика языка  $L$  является однозначной, то  $L$  является *однозначным* языком.

## 45 Нормальные формы КС-грамматик. Нормальная форма Хомского.

### Нормальная форма Хомского

- **Опр.** Говорят, что КС-грамматика находится в *нормальной форме Хомского*, если каждое ее правило имеет один из следующих видов:
  1.  $A \rightarrow BC$  (тело имеет две переменные).
  2.  $A \rightarrow a$  (тело – единичный терминал).
- **Теорема:** Если  $L$  – КС-язык, то  $L - \{\epsilon\}$  имеет КС-грамматику в НФХ.

## 46 Алгоритм удаления бесполезных переменных.

Алгоритм удаления переменных, из которых ничего не выводится

1. Найдём все переменные, из которых выводятся терминальные строки. Назовем такие переменные *полезными*.
2. Для всех других переменных (*бесполезных*), удалим из грамматики все правила, в которых они появляются либо в голове, либо в теле.

### Недостижимые символы

- Другой случай кандидата на удаление терминала или переменной - если она **не может** появиться в каком-либо выводе из начального символа.
- **Базис:** Мы всегда можем достичь  $S$  (начальный символ).
- **Индукция:** Если мы можем достичь  $A$ , и есть правило  $A \Rightarrow \alpha$ , то мы можем достичь  $\alpha$ .

## Удаление бесполезных символов

- **Опр.** Символ называется *полезным*, если он появляется в некотором выводе какой-либо терминальной строки из начального символа.
- В противном случае, он является *бесполезным*.
- Удалим все бесполезные символы:
- **Идея алгоритма:**
  - 1) Удалим все символы, из которых не выводятся терминальные строки.
  - 2) Удалим недостижимые символы.

## Устранение бесполезных символов

**Алгоритм УБС.** Устранение бесполезных символов

Вход: КС-грамматика  $G = (T, N, P, S)$ , у которой  $L(G) \neq \emptyset$ .

Выход: КС-грамматика  $G' = (T', N', P', S)$ , у которой  $L(G') = L(G)$  и в  $N' \cup T'$  нет бесполезных символов.

Метод.

1. Применив к  $G$  алгоритм определения непустоты языка, получить  $N_e$ .  
Положить  $G_1 = \langle T, N \cap N_e, P_1, S \rangle$ , где  $P_1$  состоит из правил мн-ва  $P$ , содержащих только символы из  $N_e \cap T$ .
2. Применив к  $G_1$  алгоритм устранения недостижимых символов, получить  $G' = (T', N', P', S)$

## 47 Алгоритм удаления $\epsilon$ -правил.

### $\epsilon$ -правила

- **Опр.** КС-грамматика  $G = \langle T, N, P, S \rangle$  называется грамматикой без  $\epsilon$ -правил, если в ней нет  $\epsilon$ -правил, за исключением, может быть, правила  $S \rightarrow \epsilon$  и если  $S$  не встречается в правых частях правил.

## $\epsilon$ -правила

- Мы можем почти избежать правил вида  $A \rightarrow \epsilon$  (называемых  *$\epsilon$ -правилами* ).
  - Проблема в том, что  $\epsilon$  не может быть в языке какой-либо грамматики, если в ней нет  $\epsilon$ -правил.
- **Теорема:** Если  $L$  – КС-язык, то язык  $L\{-\epsilon\}$  имеет КС-грамматику без  $\epsilon$ -правил.

## Пустые символы

- Чтобы удалить  $\epsilon$ -правила, нужно сначала найти *пустые символы* = переменные  $A$  такие, что  $A \Rightarrow^* \epsilon$ .
- **Базис:** Если есть правило  $A \rightarrow \epsilon$ , то  $A$  пустой.
- **Индукция:** Если есть правило  $A \rightarrow \alpha$ , и все символы в  $\alpha$  пустые, то  $A$  пустой.

## Удаление $\epsilon$ -правил

- **Идея:** преобразовать каждое правило  $A \rightarrow X_1\dots X_n$  в семейство правил.
- Для каждого подмножества пустых  $X$ -ов в теле правила, формируем одно новое правило с удалением этих  $X$ -ов в правой части исходного правила».
- **Исключение!** Если все  $X$ -ы пустые (или тело правила изначально было пустым), то не образуем правило с правой частью  $\epsilon$ .

## Удаление $\epsilon$ -правил

Алгоритм УЕП. Преобразование в грамматику без  $\epsilon$ -правил.

Вход: КС-грамматика  $G = (T, N, P, S)$

Выход: КС-грамматика  $G' = (T', N', P', S')$  без  $\epsilon$ -правил.

Метод.

1. Построить  $N_e = \{A | A \in N \text{ и } A \Rightarrow_G^+ \epsilon\}$
2. Построить  $P'$ :
  - Если  $A \rightarrow \alpha_0 B_1 \alpha_1 B_2 \alpha_2 \dots B_k \alpha_k$  принадлежит  $P$ ,  $k \geq 0$  и  $B_i \in N_e$  для  $1 \leq i \leq k$ , но ни один символ в цепочках  $\alpha_i$  ( $0 \leq i \leq k$ ) не принадлежит  $N_e$ , то включить в  $P'$  все правила вида  $A \rightarrow \alpha_0 X_1 \alpha_1 X_2 \alpha_2 \dots \alpha_{k-1} X_k \alpha_k$ , где  $X_l$  либо  $B_i$ , либо  $\epsilon$  (но не включать правило  $A \rightarrow \epsilon$  в случае, когда все  $\alpha_i = \epsilon$ ).
  - Если  $S \in N_e$ , включить в  $P'$  правила  $S' \rightarrow \epsilon | S$ , где  $S'$  – новый нетерминал и положить  $N' = N \cup \{S'\}$ . В противном случае положить  $N' = N$  и  $S' = S$ .
3. Определить  $G' = (T', N', P', S')$ .

## 48 Алгоритм удаления единичных правил.

### Единичные (цепные) правила

- Опр. *Единичные правила* – это те правила, в теле которых есть только одна переменная.
- Эти правила могут быть удалены.
- Ключевая идея: Если  $A \Rightarrow^* B$  в результате серии единичных выводов, и  $B \rightarrow \alpha$  неединичное правило, то добавим правило  $A \rightarrow \alpha$ .
- Затем, удалим все единичные правила.

### Единичные правила – (2)

#### ■ Алгоритм.

- Найдем все пары  $(A, B)$  такие, что  $A \Rightarrow^* B$  только последовательностью единичных правил.
- Базис: Очевидно  $(A, A)$  – переменная выводится сама из себя за 0 шагов.
- Индукция: Если мы уже нашли  $(A, B)$ , и  $B \rightarrow C$  – единичное правило, то добавим  $(A, C)$ .

## Алгоритм устранения единичных правил

- Алгоритм УЕП.
- Вход. КС-грамматика без  $\varepsilon$ -правил.
- Выход. Эквивалентная КС-грамматика  $G'$  без  $\varepsilon$ -правил и без единичных правил.
- Метод. 1. Для каждого  $A \in N$  построить  $N_A = \{B | A \Rightarrow^* B\}$  следующим образом:
  - (а) Положить  $N_0 = \{A\}$  и  $i=1$ .
  - (б) Положить  $N_i = \{C | B \rightarrow C \in P \text{ и } B \in N_{i-1}\} \cup N_{i-1}$ .
  - (в) Если  $N_i \neq N_{i-1}$ , то положить  $i=i+1$  и повторить шаг (б), иначе  $N_A = N_i$ .2. Построить  $P'$ : если  $B \rightarrow \alpha \in P$  и не является единичным правилом, включить в  $P'$  правило  $A \rightarrow \alpha$  для всех таких  $A$ , что  $B \in N_A$ .  
3. Положить  $G' = (N, \Sigma, P', S)$ .

## 49 Нормальная форма Хомского.

### Очистка грамматики— (2)

■ Док-во: Начнём с КС-грамматики для  $L$ .

■ Выполним следующие шаги:

1. Удалим  $\varepsilon$ -правила.
2. Удалим единичные правила.
3. Удалим переменные, из которых не выводятся терминальные строки.
4. Удалим переменные, которые недостижимы из начального символа.

Д.б. первым.  
Может породить  
единичные правила  
или бесполезные  
переменные.

### Нормальная форма Хомского

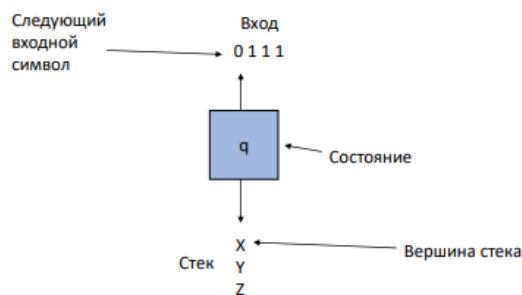
■ Опр. Говорят, что КС-грамматика находится в **нормальной форме Хомского**, если каждое ее правило имеет один из следующих видов:

1.  $A \rightarrow BC$  (тело имеет две переменные).
2.  $A \rightarrow a$  (тело – единичный терминал).

■ Теорема: Если  $L$  – КС-язык, то  $L - \{\epsilon\}$  имеет КС-грамматику в НФХ.

## 50 Автоматы с магазинной памятью. Определение. Мгновенная конфигурация МПА. Функционирование МП-автомата. Детерминированные МП-автоматы.

### Схема МПА



### МПА формально

- МПА определяется:  $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ 
  1. Конечным множеством *состояний* ( $Q$ , обычно).
  2. *Входным алфавитом* ( $\Sigma$ , обычно).
  3. *Алфавитом стека* ( $\Gamma$ , обычно).
  4. *Функцией переходов* ( $\delta$ , обычно).
  5. *Начальным состоянием* ( $q_0$ , в  $Q$ , обычно).
  6. *Начальным символом* ( $Z_0$ , в  $\Gamma$ , обычно).
  7. Множеством *конечных состояний* ( $F \subseteq Q$ , обычно).

### ФУНКЦИЯ ПЕРЕХОДОВ

- Имеет три аргумента:
  1. Состояние, в  $Q$ .
  2. Вход, который является либо символом в  $\Sigma$  или  $\epsilon$ .
  3. Верхний стековый символ в  $\Gamma$ .
- $\delta(q, a, Z)$  – есть множество нуль или более действий вида  $(p, \alpha)$ .
  - $p$  – новое состояние;  $\alpha$  - строка стековых символов, которая заменяет символ вершины стека.

## Действия МПА

- Если  $\delta(q, a, Z)$  содержит  $(p, \alpha)$  среди его действий, то единственным, что МПА может сделать в состоянии  $q$ , с  $a$  в начале входа, и  $Z$  в вершине стека, это:
  1. Изменить состояние на  $p$ .
  2. Удалить символ  $a$  из начальной части входа (но  $a$  может быть  $\epsilon$ ).
  3. Заменить  $Z$  в вершине стека на  $\alpha$ .
- МПА может иметь несколько альтернативных пар вида  $(p, \alpha)$  для  $\delta(q, a, Z)$ .
- Мы можем формализовать описание работы МПА на основе **мгновенных конфигураций** (ID- instantaneous description).
- ID – это тройка  $(q, w, \alpha)$ , где:
  1.  $q$  – текущее состояние.
  2.  $w$  – оставшийся непрочтенный вход.
  3.  $\alpha$  - содержимое стека, с вершиной слева.

## Детерминированные МПА

- Чтобы быть детерминированным, у МПА должен быть, по крайней мере, один выбор при переходе из любого состояния  $q$ , для входного символа  $a$ , и стекового символа  $X$ .
- Дополнительно, не должно быть выбора между использованием входа  $\epsilon$  или реального входа.
  - Формально:  $\delta(q, a, X)$  и  $\delta(q, \epsilon, X)$  не могут быть оба пусты.
- Обычно принятие входа Д-МПА определяется переходом в заключительное состояние, т.к. при пустом стеке мы не можем больше обрабатывать вход
- Хотя мы не будем углубляться далее в теорию МПА, отметим, что класс языков, принятых детерминированными МПА, содержит все регулярные языки (очевидно, поскольку он может моделировать детерминированный конечный автомат, просто игнорируя его стек), но не включает все контекстно-свободные языки.

## 51 Языки для МП-автоматов. Теорема об эквивалентности определения языков МПА по конечному состоянию и опустошением стека.

### Язык МПА

- Общим способом определения языка, определяемого МПА, является его *конечное состояние*.
- Если  $P$  есть МПА, то  $L(P)$  есть множество строк  $w$  таких, что  $(q_0, w, Z_0) \vdash^* (f, \epsilon, \alpha)$  для конечного состояния  $f$  и любого  $\alpha$ .

### Язык МПА – (2)

- Другим способом определения языка для того же МПА PDA является признак *пустоты стека*.
- Если  $P$  есть МПА, то  $N(P)$  есть множество строк  $w$  таких, что  $(q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon)$  для любого состояния  $q$ .

### Эквивалентность определения языков

#### Теорема

- Если  $L = L(P)$ , то существует другой МПА  $P'$  такой, что  $L = N(P')$ .
- Если  $L = N(P)$ , то существует другой МПА  $P''$  такой, что  $L = L(P'')$ .

*Доказательство:*  $L(P) \rightarrow N(P')$  интуитивно

- $P'$  моделирует  $P$ .
- Если  $P$  принимает строку,  $P'$  опустошит свой стек.
- $P'$  должен избежать случайного опустошения стека, для этого он использует специальный маркер дна стека на случай, когда  $P$  опустошает свой стек без принятия входа.

## Доказательство: $L(P) \rightarrow N(P')$

- $P'$  имеет все состояния, символы, и переходы, что и у  $P$ , плюс:
  1. Стековый символ  $X_0$  (начальный стековый символ  $P'$ ), используемый для отслеживания дна стека.
  2. Новое начальное состояние  $s$  и “удаляющее” состояние  $e$ .
  3.  $\delta(s, \epsilon, X_0) = \{(q_0, Z_0 X_0)\}$ . Начало работы с  $P$ .
  4. Добавим  $\{(e, \epsilon)\}$  к  $\delta(f, \epsilon, X)$  для любого финального состояния  $f$  автомата  $P$  и любого стекового символа  $X$ , включая  $X_0$ .
  5.  $\delta(e, \epsilon, X) = \{(e, \epsilon)\}$  для любого  $X$ .

## Доказательство: $N(P) \rightarrow L(P'')$ Интуитивно

- $P''$  моделирует  $P$ .
- $P''$  имеет специальный маркер дна стека, чтобы отследить ситуацию, когда  $P$  опустошает свой стек.
- Если это так, то,  $P''$  переходит в заключительное состояние и принимает входную строку.

## Доказательство : $N(P) \rightarrow L(P'')$

- $P''$  имеет все состояния, символы и переходы, что и у  $P$ , плюс:
  1. Стековый символ  $X_0$  (начальный символ), используемый для отслеживания дна стека.
  2. Новое начальное состояние  $s$  и финальное состояние  $f$ .
  3.  $\delta(s, \epsilon, X_0) = \{(q_0, Z_0 X_0)\}$ . Начало работы с  $P$ .
  4.  $\delta(q, \epsilon, X_0) = \{(f, \epsilon)\}$  для любого состояния  $q$  автомата  $P$ .

## 52 Эквивалентность МП-автоматов и КС-грамматик.

### Преобразование КСГ в МПА

- **Теорема.** Пусть  $G$  – КС-грамматика и  $L = L(G)$ .
- Тогда можно построить МПА  $P$  такой, что  $N(P) = L$ .
- $P$  имеет:
  - Одно состояние  $q$ .
  - Входные символы = терминалы грамматики  $G$ .
  - Стековые символы = все символы  $G$ .
  - Начальный символ = начальный символ  $G$ .

### Интуитивно о $P$

- На каждом шаге,  $P$  представляет некоторую **лево-сентенциальную форму** (шаг левого вывода) из начального символа  $S$ .
- Если стек  $P$  есть  $\alpha$ , и  $P$  только что обработал  $x$  из своего входа, то  $P$  представляет лево-сентенциальную форму  $\alpha x$ .
- При пустом стеке, обработанной строкой является строка в  $L(G)$ .
- Если никакая последовательность вариантов работы недетерминированного МП-автомата  $P$  не приводит к пустому стеку после обработки  $w$  из входа, то  $w$  не является терминальной строкой, порождаемой грамматикой, и  $P$ , соответственно, не принимает  $w$ .

### От МПА к КСГ

- Теперь предположим, что  $L = N(P)$ .
- **Теорема.** Пусть  $P$  – МПА и  $L = N(P)$ . Тогда можно построить КС-грамматику  $G$  такую, что  $L = L(G)$ .
- **Интуитивно:**  $G$  будет иметь переменные  $[pXq]$  генерирующие в точности строки  $w$ , которые являются причиной для  $P$  иметь эффект выталкивания символа  $X$  из стека при переходе из состояния  $p$  в состояние  $q$ .
- При этом  $P$  может наращивать стек значительно выше того, где был  $X$ .
  - При этом  $P$  никогда не опускается ниже этого  $X$ .

## 53 Лемма о накачке для КС-языков. Формулировка. Приложения.

### Формулировка леммы о накачке для КС-языков

**Лемма.** Для каждого КС-языка  $L$  существует целое число  $n$ , такое, что для каждой строки  $z$  из  $L$  длины  $\geq n$  существует ее представление  $z = uvwxy$  такое что:

1.  $|vwx| \leq n$ .
2.  $|vx| > 0$ .
3. Для всех  $i \geq 0$ ,  $uv^iwx^i y$  принадлежит  $L$ .

### Использование леммы о накачке

- $\{0^i 10^i \mid i \geq 1\}$  есть КС-язык.
  - Мы можем отметить одну пару счетчиков.
- Но  $L = \{0^i 10^i 10^i \mid i \geq 1\}$  - нет.
  - Мы не можем выделить две пары, или три счетчика как группу.
- [Доказывается](#) с использованием леммы о накачке.
- Предположим, что  $L$  – КС-язык.
- Пусть  $n$  будет константой леммы о накачке для языка  $L$ .

### Использование леммы о накачке – (2)

- Рассмотрим  $z = 0^n 10^n 10^n$ .
- Мы можем записать  $z = uvwxy$ , где  $|vwx| \leq n$ , и  $|vx| \geq 1$ .
- **Случай 1:**  $vx$  не имеет 0-ей.
  - Тогда, по крайней мере, одна из них имеет 1, и  $uw$  имеет, по крайней мере, одну 1, что при накачке не даёт строку из  $L$ .

### Использование леммы о накачке – (3)

- Мы все еще рассматриваем  $z = 0^n 1^n 10^n$ .
- **Случай 2:**  $vx$  имеет, по крайней мере, один 0.
  - $vwx$  слишком коротка (длина  $\leq n$ ), чтобы расширить все три блока 0-ей в  $0^n 1^n 10^n$ .
  - Рассмотрим строку  $uwv$ , которая, если L-КС-язык, должна быть в L.
  - Но,  $uwv$  имеет, по крайней мере, один блок из n 0-ей, и, по крайней мере, один блок с меньшим, чем n 0-ями.
  - Таким образом,  $uwv$  не входит в L.
  - Значит, наше предположение о контекстно-свободности языка L неверно!

## 54 Свойства КС-языков. Разрешающие свойства. Свойства замыкания

### Свойства замыкания для КС-языков

- КС-языки замкнуты относительно операций объединения, конкатенации и замыкания Клини (т.е. относительно регулярных операций).
- Также, КС-языки замкнуты относительно обратимости строк, гомоморфизма и обратного гомоморфизма.
- Но не по пересечению или разности.

### Замыкание КС-языков по объединению

- Пусть L и M – КС-языки с грамматиками G и H, соответственно.
- Предположим G и H не имеют общих переменных.
  - Имена переменных не влияют на язык.
- Пусть  $S_1$  и  $S_2$  – начальные символы G и H.

### Замыкание КС-языков по объединению – (2)

- Образуем новую грамматику для  $L \cup M$  путём комбинации всех символов и продукции G и H.
- Затем, добавим новый начальный символ S.
- Добавим продукцию  $S \rightarrow S_1 | S_2$ .

### Замыкание КС-языков по объединению – (3)

- В построенной новой грамматике, все выводы начинаются с  $S$ .
- На первом шаге происходит замена  $S$  на  $S_1$  или  $S_2$ .
- В первом случае, результатом должна стать строка в  $L(G) = L$ , а во втором – строка в  $L(H) = M$ .

### Замыкание КС-языков по конкатенации

- Пусть  $L$  и  $M$  – КС-языки с грамматиками  $G$  и  $H$ , соответственно.
- Предположим  $G$  и  $H$  не имеют общих переменных .
- Пусть  $S_1$  и  $S_2$  – начальные символы  $G$  и  $H$ .

### Замыкание КС-языков по конкатенации– (2)

- Образуем новую грамматику для  $LM$ , начав с того, что включим в неё все символы и продукции  $G$  и  $H$ .
- Добавим новый начальный символ  $S$ .
- Добавим продукцию  $S \rightarrow S_1S_2$ .
- Каждый вывод из  $S$  даст в результате строку в  $L$  с последующей за ней строкой из  $M$ .

## Замыкание КС-языков по операции Клини

- Пусть  $L$  имеет грамматику  $G$ , с начальным символом  $S_1$ .
- Образуем новую грамматику для  $L^*$  путём введения в  $G$  нового начального символа  $S$  и продукции  $S \rightarrow S_1S \mid \epsilon$ .
- Правый вывод из  $S$  генерирует последовательность из нуль или более переменных  $S_1$ , каждая из которых генерирует некоторую строку из  $L$ .

## Замыкание КС-языков по обратимости

- Если  $L$  – КС-язык с грамматикой  $G$ , образуем грамматику для  $L^R$  путём обращения тела каждой продукции.
- **Пример:**
  - Пусть  $G$  имеет  $S \rightarrow 0S1 \mid 01$ .
  - Обращение языка  $L(G)$  имеет грамматику  $S \rightarrow 1S0 \mid 10$ .

## Замыкание КС-языков по гомоморфизму

- Пусть  $L$  – КС-язык с грамматикой  $G$ .
- Пусть  $h$  – гомоморфизм на терминальных символах  $G$ .
- Построим грамматику для  $h(L)$ , заменяя каждый терминальный символ  $a$  на  $h(a)$ .

## Пример: Замыкание КС-языков по гомоморфизму

- $G$  имеет продукцию  $S \rightarrow 0S1 \mid 01$ .
- $h$  определяется как  $h(0) = ab$ ,  $h(1) = \epsilon$ .
- $h(L(G))$  имеет грамматику с правилами  $S \rightarrow abS \mid ab$ .

## Замыкание КС-языков по обратному гомоморфизму

- Здесь нам не помогут грамматики, но вполне подойдет конструкция МПА.
- Пусть  $L = L(P)$  для некоторого МПА  $P$ .
- Построим МПА  $P'$  принимающий  $h^{-1}(L)$ .
- $P'$  моделирует  $P$ , но сохраняет, в качестве одной из частей двухкомпонентного состояния, буфер, который хранит результат применения  $h$  к одному входному символу.

## Отсутствие замыкания по пересечению

- В отличие от регулярных языков, класс КС-языков не замкнут относительно  $\cap$ .
- Мы знаем, что  $L_1 = \{0^n1^n2^n \mid n \geq 1\}$  не есть КС-язык (использовали лемму о накачке).
- Однако,  $L_2 = \{0^n1^n2^i \mid n \geq 1, i \geq 1\}$  КС-языком является.
  - КСГ:  $S \rightarrow AB$ ,  $A \rightarrow 0A1 \mid 01$ ,  $B \rightarrow 2B \mid 2$ .
- Это так и для  $L_3 = \{0^i1^n2^n \mid n \geq 1, i \geq 1\}$ .
- Но  $L_1 = L_2 \cap L_3$  !

## Отсутствие замыкания по разности

- Мы можем доказать нечто более общее :
  - Любой класс языков, замкнутый относительно разности, замкнут по пересечению.
- **Доказательство:**  $L \cap M = L - (L - M)$ .
- Таким образом, если бы КС-языки были замкнуты по разности, то они были бы замкнуты и по пересечению, но это не так.

### Разрешимые свойства для регулярных языков

- Проблема принадлежности языку:  $w \in L?$
- Проблема пустоты:  $L = \emptyset?$
- Проблема эквивалентности:  $L = M?$
- Проблема вложения языков:  $L \subseteq M?$
- Проблема бесконечности языка:  $|L| = \infty?$

### Разрешимые свойства для КС-языков языков

- Принадлежит ли строка  $w$  КС-языку  $L$ .
- Является ли КС-язык пустым.
- Является ли КС-язык бесконечным.

### Неразрешимые свойства

- Проблема эквивалентности:  $L = M?$
- Проблема связности:  $L \cap M = \emptyset?$

### Свойства замкнутости для регулярных языков

- Регулярные языки замкнуты относительно операций объединения, конкатенации и замыкания Клини (регулярных операций)
- Замкнутость по обращению (слов)
- Замкнутость по дополнению
- Замкнутость по пересечению
- Замкнутость по вычитанию (разности)
- Замкнутость по гомоморфизму и обратному гомоморфизму

### Свойства замыкания для КС-языков

- КС-языки замкнуты относительно операций объединения, конкатенации и замыкания Клини (т.е. относительно регулярных операций).
- КС-языки замкнуты относительно обратности строк,
- **КС-языки не замкнуты по пересечению**
- **КС-языки не замкнуты по разности.**
- КС-языки замкнуты относительно гомоморфизма и обратного гомоморфизма.
- Пересечение КС-языка с регулярным языком всегда будет КС-языком.

## 55 Машины Тьюринга. Мгновенные конфигурации МТ.

### Формальное определение МТ

- ТМ определяется:
  1. Конечным множеством *состояний* ( $Q$ , обычно).
  2. *Входной алфавит* ( $\Sigma$ , обычно).
  3. *Алфавит ленты* ( $\Gamma$ , обычно; contains  $\Sigma$ ).
  4. *Функция переходов* ( $\delta$ , обычно).
  5. *Начальный символ* ( $q_0$ , in  $Q$ , обычно).
  6. *Пустой символ* ( $B$ , in  $\Gamma - \Sigma$ , обычно).
    - Вся лента, за исключением входа, первоначально пуста.
  7. Множество *конечных состояний* ( $F \subseteq Q$ , обычно).

### Функция переходов

- Имеет два аргумента:
  1. Состояние, в  $Q$ .
  2. Символ ленты в  $\Gamma$ .
- $\delta(q, Z)$  - либо не определена, либо является тройкой вида  $(p, Y, D)$ .
  - $p$  - состояние.
  - $Y$  – новый символ ленты.
  - $D$  - *направление*, L или R.

### Мгновенные конфигурации МТ– (2)

- МК – это строка  $\alpha q \beta$ , где  $\alpha \beta$  содержимое ленты между самым левым и самым правым непробелами.
- Состояние  $q$  находится непосредственно слева от сканируемого головкой символа ленты.
- Если  $q$  находится с правого конца, то сканируется  $B$ .
  - Если  $q$  сканирует  $B$  слева, то следующие пробелы до конца справа от  $q$  являются частью  $\alpha$ .

## Формальное определение шага работы МТ

1. Если  $\delta(q, Z) = (p, Y, R)$ , то
  - $\alpha q Z \beta \vdash \alpha Y p \beta$
  - Если  $Z$  есть пробел  $B$ , то также  $\alpha q \vdash \alpha Y p$
2. Если  $\delta(q, Z) = (p, Y, L)$ , то
  - Для любого  $X$ ,  $\alpha X q Z \beta \vdash \alpha p X Y \beta$
  - Дополнительно,  $q Z \beta \vdash p B Y \beta$

## Языки МТ

- МТ обычно определяет язык по конечному состоянию.
- $L(M) = \{w \mid q_0 w \vdash^* l\}$ , где  $l$  есть конфигурация с конечным состоянием}.
- Или, МТ может определять язык по остановке.
- $H(M) = \{w \mid q_0 w \vdash^* l\text{, нет возможных переходов из данной конфигурации } l\}$ .

## Эквивалентность принятия и останова

1. Если  $L = L(M)$ , то существует МТ  $M'$  такая, что  $L = H(M')$ .
2. Если  $L = H(M)$ , то существует МТ  $M''$  такая, что  $L = L(M'')$ .

## 56 Рекурсивно-перечислимые языки. Рекурсивные языки.

### Рекурсивно-перечислимые языки

- Классы языков, определяемые машинами Тьюринга с использованием конечных состояний и останова, являются эквивалентными.
- Этот класс языков называется *рекурсивно-перечислимыми языками*.
- Для данного класса языком МТ является полуразрешимым алгоритмом.

### Рекурсивные языки

- Разрешимый алгоритм* - это МТ, принимающая вход индикацией по переходу в конечное состояние, и останавливающаяся на любом входе безотносительно того, принимается или нет вход.
- Если  $L = L(M)$  для некоторой МТ  $M$ , которая является алгоритмом, то говорят, что  $L$  – *рекурсивный язык*.
- Рекурсивные языки – это подкласс рекурсивно-перечислимых языков

## 57 Свойства замыкания для рекурсивных и рекурсивно-перечислимых языков.

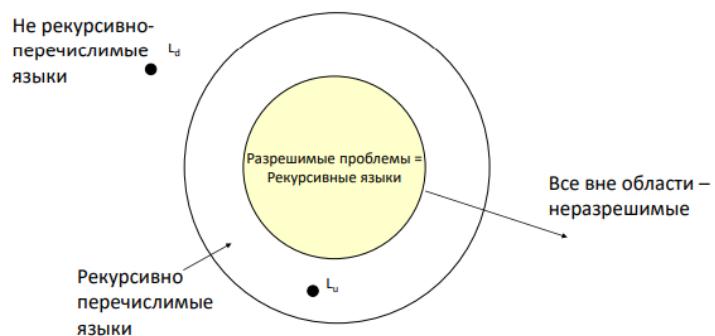
### Свойства замыкания для рекурсивных и рекурсивно-перечислимых языков

- Оба класса языков замкнуты относительно объединения, конкатенации, операции Клини, обращения, пересечения, обратного гомоморфизма.
- Класс рекурсивных языков замкнут по операциям вычитания и дополнения.
- Рекурсивно-перечислимые языки замкнуты по гомоморфизму.

## 58 Универсальный язык. Неразрешимые проблемы.

### Универсальный язык

- Примером рекурсивно-перечислимого, но не рекурсивного языка является язык  $L_u$  *универсальной машины Тьюринга (УМТ)*.
- То есть, на вход УМТ подается код некоторой МТ  $M$  и некоторая бинарная строка  $w$ , и она принимает их, т.и.т., когда  $M$  принимает  $w$ .



### Пример: Неразрешимые проблемы

- Можно ли выполнить определенную строку кода в программе?
- Неоднозначна ли данная контекстно-свободная грамматика?
- Генерируют ли две данных КС-грамматики один и тот же язык?
- Эквивалентен ли данный КС-язык языку  $\Sigma^*$ ?
- Является ли КС-язык регулярным?