Формальные языки и автоматы

Игнатьев Валерий Николаевич

vignatyev@cs.msu.ru

Организационные вопросы и отчетность

- 2 контрольных на лекциях
  (40 баллов)
- Экзамен письменный
  (30 баллов)
- Задачи в ejudge (ДЗ)
  (10,10,5, … баллов)
- Контест на Regexp
  (5 баллов)

Оценка	Перед экзаменом	Бонус	экзамен (22г)
отл.	~50	+1 к оценке на экзамене	20 / 25
хор.	~42		16 / 25
удв.	~35		12 / 25

Слайды, материалы, результаты проверочных работ

https://t.me/+AfdgEOSZCiAwMWEy

- Табличка с результатами

  - будет сообщение в группе

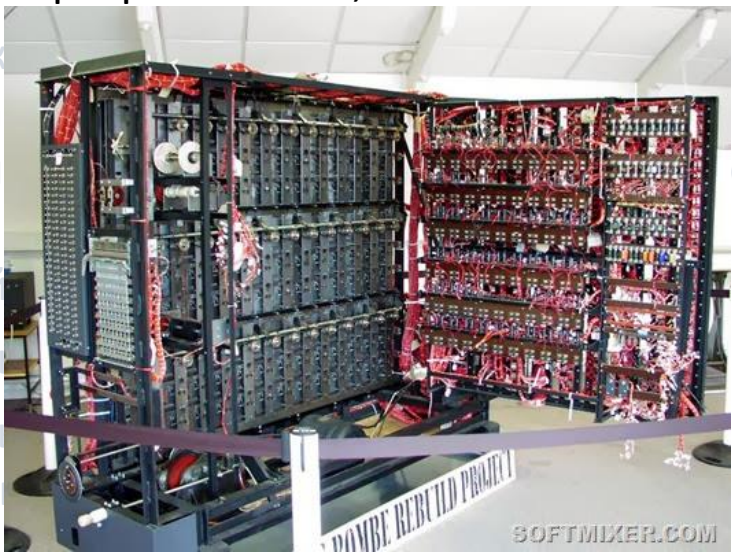
Актуальность

- **Регулярные выражения (PB, RE)**
  - текстовые редакторы
  - утилиты обработки текстов (sed, grep)
  - UNIX shell,
  - тривиальный разбор текста
- **Конечные автоматы – модель для многих HW&SW компонентов**
  - Лексический анализ компилятора
  - Model checking
  - Дизайн и верификация цифровых электронных схем
- **Контекстно-свободные грамматики используются для**
  - формализации синтаксиса практически всех ЯП
  - в распознавании естественного языка (natural language processing)
- **Конструирование неоптимизирующего компилятора**

История

1930-е

- Тьюринг – абстрактная машина
- Проблема разрешимости, останова



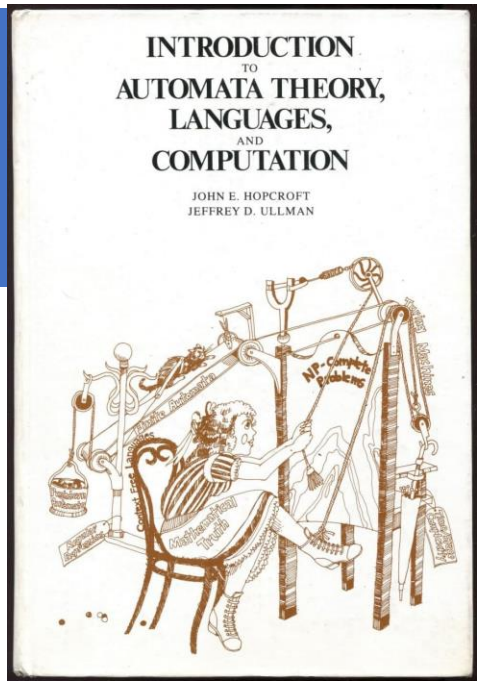
История

1930-е	• Тьюринг – абстрактная машина • Проблема разрешимости, останова
1940 – 1950-е	• теория конечных автоматов разработка и верификация цифровых схем, лек еры-краулеры, протоколы свя • ии ФЯ
1950-е	• ки
1950 –1960-е	• ки применяются при реализации первых ЯП

История

1930-е	• Тьюринг – абстрактная машина • Проблема разрешимости, останова
1940 – 1950-е	• теория конечных автоматов разработка и верификация цифровых схем, лексический анализ, парсеры-краулеры, протоколы связи • Хомский – основы теории ФЯ
1950-е	• Формальные грамматики
1950 –	• Формальные грамматики применяются при реализации первых ЯП

Литература

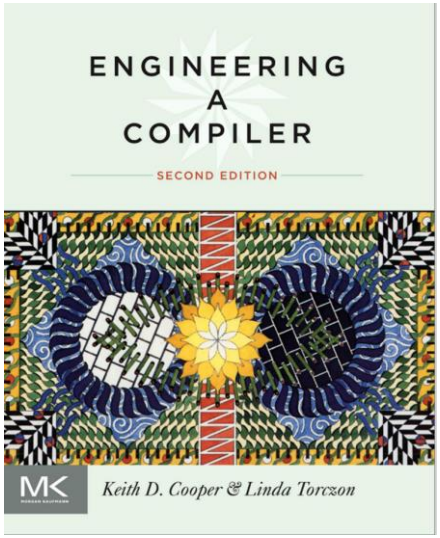


- *Джон Э. Хопкрофт, Раджив Мотвани, Джеффри Д. Ульман*

«Введение в теорию автоматов, языков и вычислений»

- *В.А. Серебряков*

«Теория и реализация языков программирования»

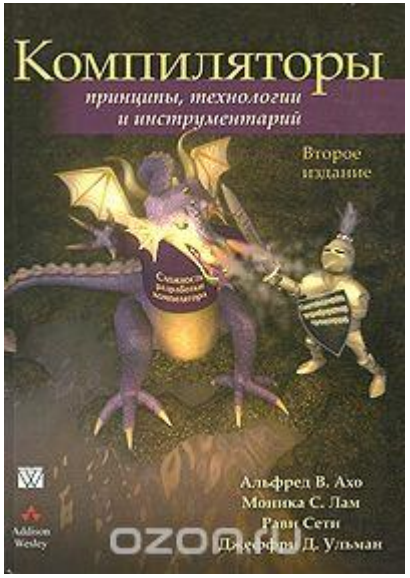


- *Альфред В. Ахо, Моника С. Лам, Рави Сети, Джеффри Д. Ульман*

«Компиляторы. Принципы, технологии и инструментарий»

- *Cooper, Keith D., Torczon, Linda*

“Engineering a compiler”



Основные понятия теории автоматов
Язык как формальная система

- **Алфавит (V)** – конечное непустое множество *символов*

Пример: $\{a, b, c, \dots, z\}$ – латинский, $\{0,1\}$ – двоичный

- **Слово (строка, предложение)** – любая цепочка конечной длины, составленная из символов алфавита.

$\boldsymbol{\varepsilon}$ или $\boldsymbol{\epsilon}, \boldsymbol{e}$ – обозначение **пустой строки/слова** – не содержит ни одного символа

V^* – множество всех слов, составленных из символов V

$V^+ = V^* \setminus \{\varepsilon\}$

Пример: $V = \{0,1\}$, тогда $V^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}$

- $|x| -$ длина строки $x \in V$,
- $|\varepsilon| = 0$
- Пусть $x, y \in V^*, x = a_1 a_2 \dots a_i, y = b_1 b_2 \dots b_j$,

xy – конкатенация, $xy = a_1 a_2 \dots a_i b_1 b_2 \dots b_j$.

Язык как формальная система

- **Язык (L)** – любое множество строк в алфавите $L \subseteq V^*$

Пример:

- $L_1 = \emptyset$ – пустой язык;
- $L_2 = \{\varepsilon\}$ — язык, содержащий только пустую цепочку
  (L_1 и L_2 — различные языки)
- $L_3 = \{\varepsilon, a, b, aa, ab, ba, bb\}$ — язык, содержащий все цепочки из a и b, длина которых не превосходит 2;
- $L_4 - \{a^{n^2} : n > 0\}$ — язык цепочек из a, длины которых представляют собой квадраты натуральных чисел.

Проблемы

- Как задавать язык?
- Для каждого ли языка существует конечное представление?
- Для каких классов языков существуют конечные представления?
- Является ли данная цепочка элементом определенного языка?

Проблемы

Как задавать язык?	
Распознавание	**Порождение**
• алгоритм или процедура, которые определяют, принадлежит ли заданное слово языку • алгоритм: «да» ^ «нет» • процедура: «да» ^ «нет» ^ не завершается	• процедура, которая систематически порождает предложения языка последовательно в некотором порядке

- По алгоритму или процедуре распознавания языка можно построить процедуру, порождающую этот у язык:
  генерировать все предложения из V^* и проверять с помощью известной процедуры или алгоритма

А что если процедура распознавания не завершится на некотором слове?

Построение порождающей процедуры по процедуре распознавания

Пусть V – алфавит из p символов

- Занумеруем слова из V^* (как числа p-ичной системы счисления)

Пример: $V = \{a, b, c\}$

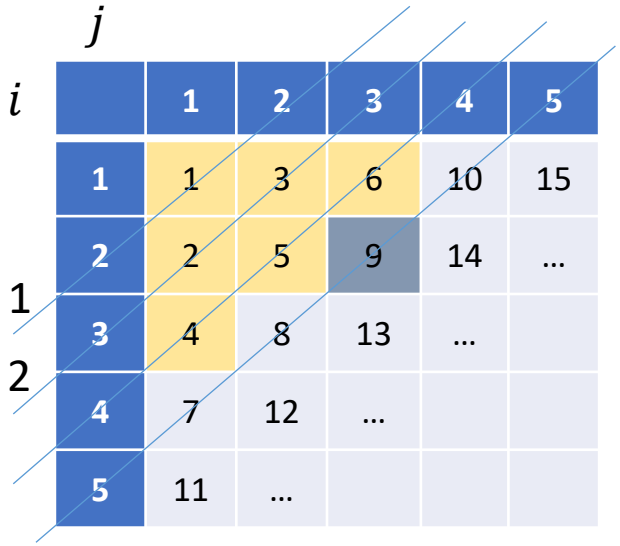
1.	ε	5.	aa	9.	bb
2.	a	6.	ab	10.	bc
3.	b	7.	ac	11.	…
4.	c	8.	ba		

- Пусть P – распознающая процедура для L. Она состоит из шагов, т.е. существует некоторый i-й шаг

Построение порождающей процедуры по процедуре распознавания

- Зададим способ нумерации (k) пар положительных целых чисел (i, j)

$$k = \mathrm{c}(i, j)$$



- Пусть k имеет координаты (i, j) (серый)

Заметим, что

Пусть

N – число эл-тов треугольника (желтый)

$k = N + i$

$$N = 1 + 2 + .. + n = \frac{n(n+1)}{2},$$

$$n: (1,1), (i + j - 1, 1), (1, \mathrm{i} + \mathrm{j} - 1); i + j = n + 2$$

$$N = 1 + 2 + .. + (i + (j - 1) - 1)$$

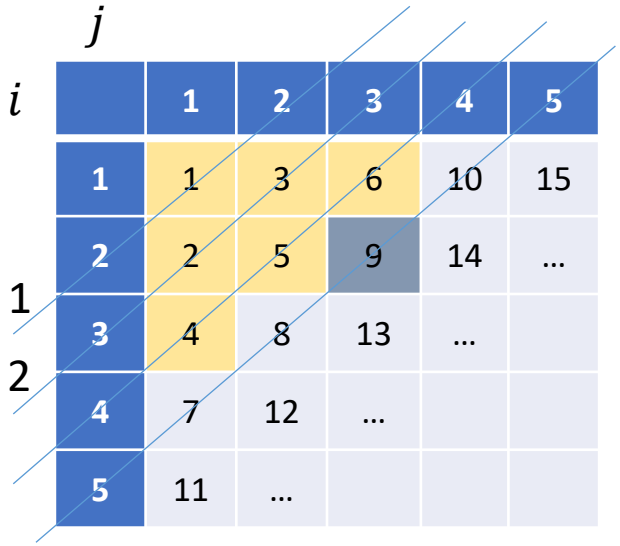
$$N = \frac{(i + j - 2)(i + j - 1)}{2}$$

$$\boldsymbol{k = \frac{(i + j - 2)(i + j - 1)}{2} + i}$$

Построение порождающей процедуры по процедуре распознавания

- Зададим способ нумерации (k) пар положительных целых чисел (i, j)

$$k = \mathrm{c}(i, j)$$



- Пусть k имеет координаты (i, j)

Заметим, что

$k = N + i$

N – число эл-тов треугольника

$N = 1 + 2 + .. + (i + (j - 1) - 1)$

$$N = \frac{(i + j - 2)(i + j - 1)}{2}$$

$$\boldsymbol{k = \frac{(i + j - 2)(i + j - 1)}{2} + i}$$

Для каждой пары (i, j):

- выбирается i-е слово из V^*, на котором
- выполняются первые j шагов процедуры распознавания P

Если P распознает слово за j шагов, то оно добавляется к списку слов языка L.

Построение распознающей процедуры по порождающей

Пусть P – порождающая процедура языка L

Обозначим как P' – распознающую процедуру

Тогда $\forall x \in V^* \; P'$ определяет истинность $x \in L$

```
procedure P'(x):
    do
        y ← P
    while (y ≠ x);
    return true;
```

Может не завершиться

Типы языков

- **Рекурсивно перечислимый** – существует *процедура* распознавания
  - Или существует порождающая процедура
- **Рекурсивный** – существует *алгоритм* распознавания

Теорема

Пусть $L \subseteq V^*$ – некоторый язык, а $\bar{L} = V^* \setminus L$ – его дополнение. Если языки L и $\bar{L}$ рекурсивно перечислимы, то L рекурсивен.

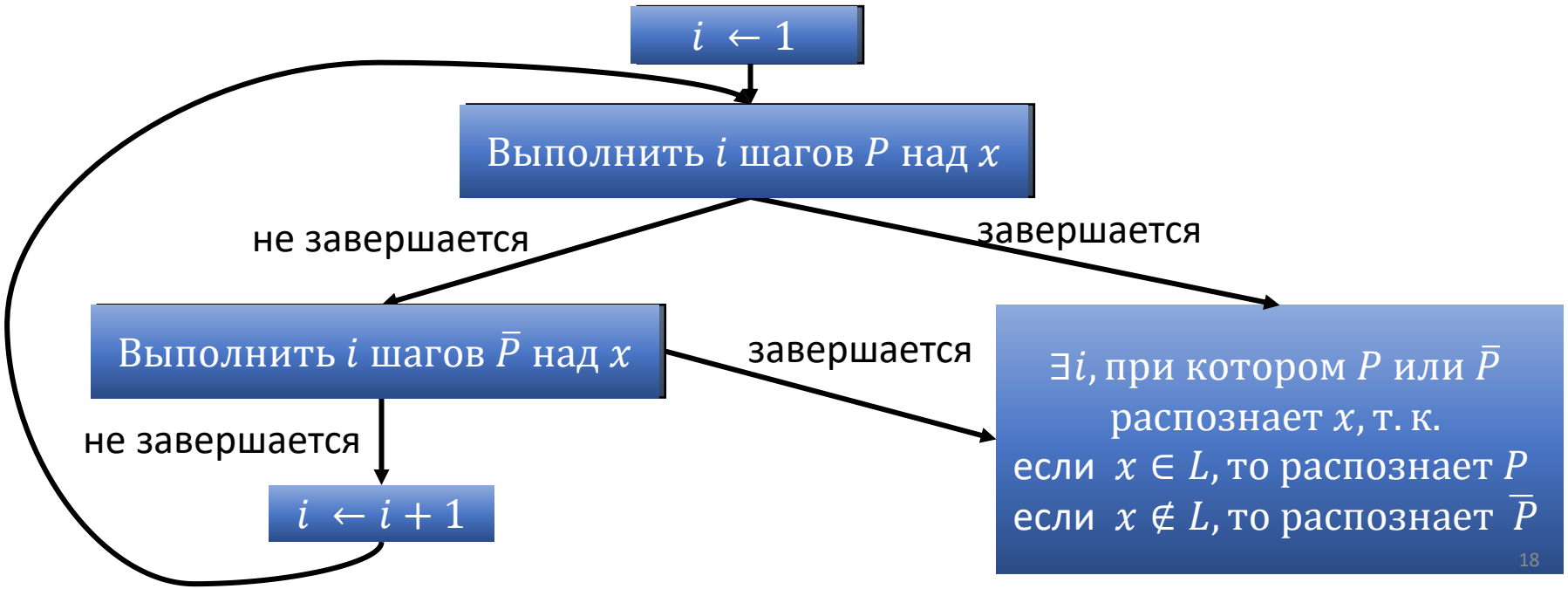
Доказательство теоремы

Теорема

Пусть $L \subseteq V^*$ – некоторый язык, а $\bar{L} = V^* \setminus L$ – его дополнение. Если языки L и $\bar{L}$ рекурсивно перечислимы, то L рекурсивен.

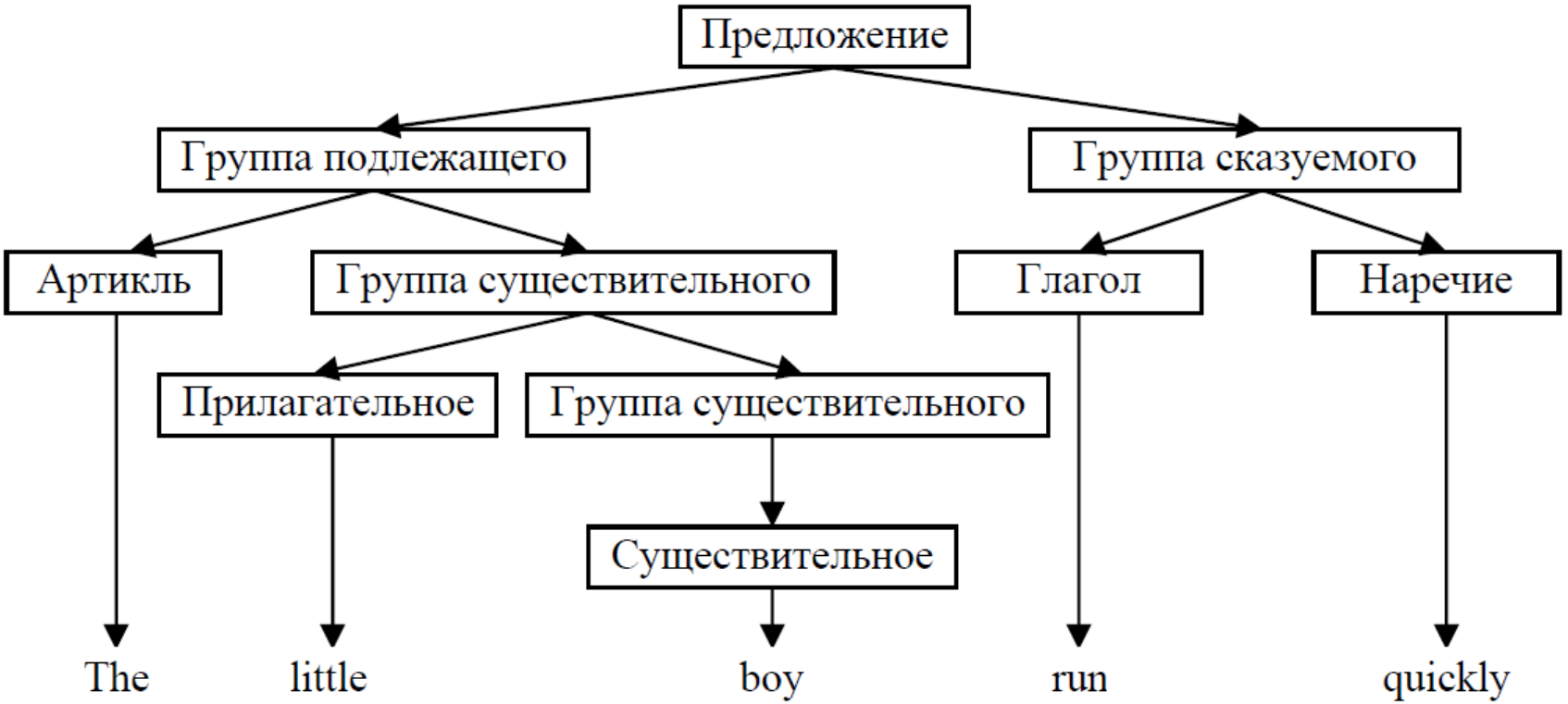
Пусть L распознается процедурой P, а $\bar{L}$ распознается $\bar{P}$.

Построим алгоритм распознавания L: $(x \in L)?$:

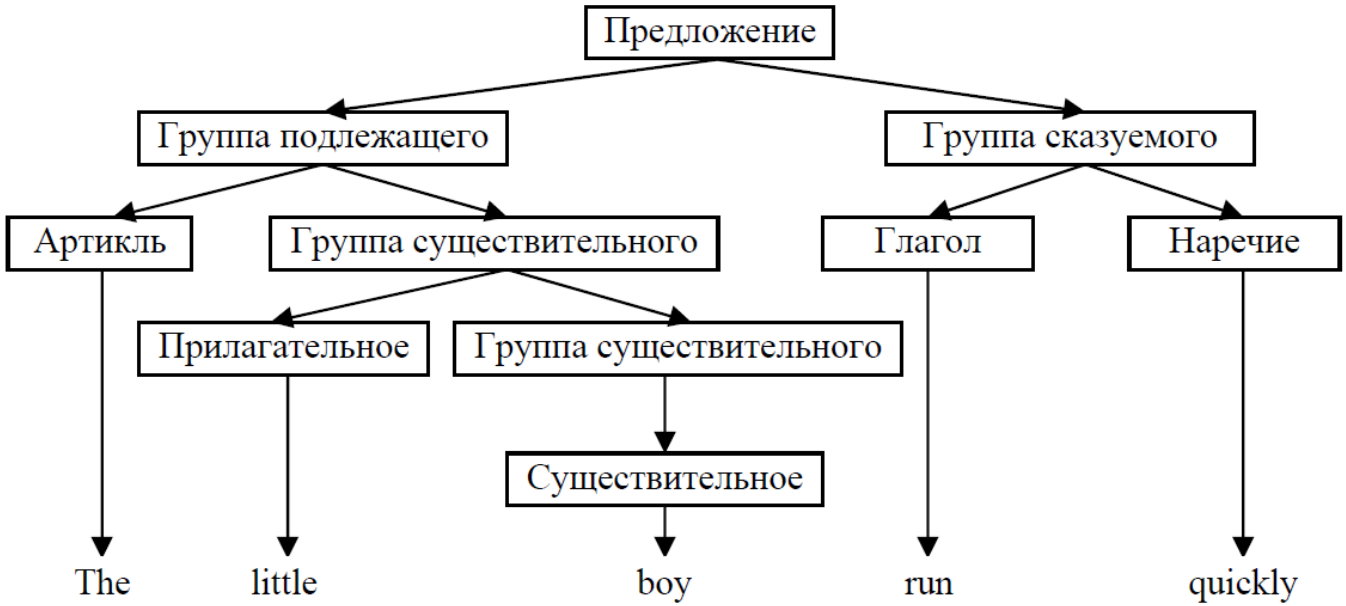


Грамматики

- Способ порождения языков



Грамматики



⟨ предложение⟩ → ⟨ группа подлежащего⟩ ⟨ группа сказуемого⟩

⟨ группа подлежащего⟩ → ⟨ артикль⟩ ⟨ группа существительного⟩

⟨ группа существительного⟩ → ⟨ прилагательное⟩ ⟨ группа существительного⟩

⟨ группа существительного⟩ → ⟨ существительное⟩

⟨ группа сказуемого⟩ → ⟨ глагол⟩ ⟨ наречие⟩

⟨ артикль⟩ → The

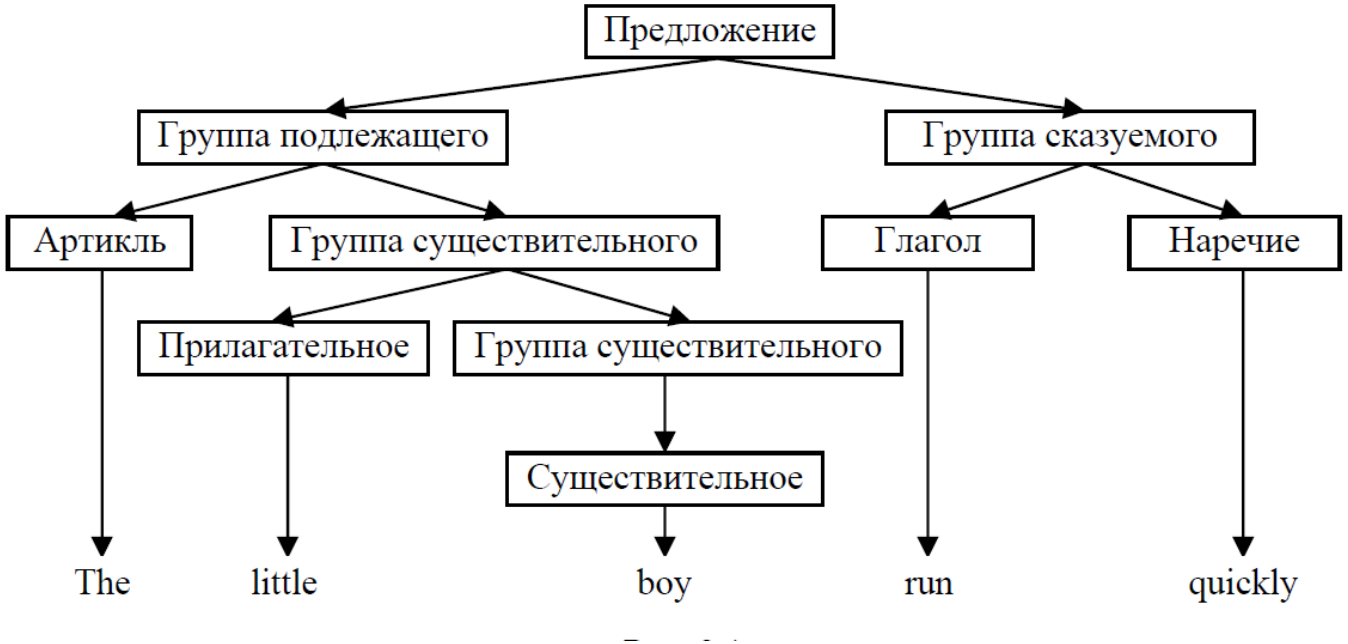
⟨ прилагательное⟩ → little

⟨ существительное⟩ → boy

⟨ глагол⟩ → run

⟨ наречие⟩ → quickly

Грамматики



⟨ предложение⟩ → ⟨ группа подлежащего⟩ ⟨ группа сказуемого⟩

⟨ группа подлежащего⟩ → ⟨ артикль⟩ ⟨ группа существительного⟩

⟨ группа существительного⟩ → ⟨ прилагательное⟩ ⟨ группа существительного⟩

⟨ группа существительного⟩ → ⟨ существительное⟩

⟨ группа сказуемого⟩ → ⟨ глагол⟩ ⟨ наречие⟩

⟨ артикль⟩ → The

⟨ прилагательное⟩ → little

⟨ существительное⟩ → boy

⟨ глагол⟩ → run

⟨ наречие⟩ → quickly

нетерминалы

терминалы

правила

Грамматики

Грамматикой называется четвёрка $G = (V_N, V_T, P, S)$, где

V_N – алфавит нетерминальных символов,

V_T – алфавит терминальных символов,

причём $V_N \cap V_T = \emptyset,\ V = V_N \cup V_T$

P – конечное множество правил вида $\alpha \to \beta,\ \alpha \in V^* V_N V^*, \beta \in V^*$

$S \in V_N$ – начальный нетерминал (аксиома грамматики, стартовый символ)

Обозначения:

$A, B, C, S \in V_N$ – нетерминалы

$a, b, c \in V_T$ – терминалы

$x, y, z \in V_T^*$ – строки терминалов

$\alpha, \beta, \gamma \in V^*$ – строки из объединенного алфавита V

- Пусть $\alpha \to \beta \in P$ – правило; $\gamma, \delta \in V^*$
  Тогда $\gamma\alpha\delta \underset{G}{\Rightarrow} \gamma\beta\delta$ читается как *«из $\gamma\alpha\delta$ непосредственно выводится $\gamma\beta\delta$»* в грамматике G.
- Пусть $\alpha_1, \alpha_2, \ldots \alpha_m \in V^*$, $\alpha_1 \underset{G}{\Rightarrow} \alpha_2$, $\alpha_2 \underset{G}{\Rightarrow} \alpha_3$, …, $\alpha_{m-1} \underset{G}{\Rightarrow} \alpha_m$
  Тогда $\alpha_1 \overset{*}{\underset{G}{\Rightarrow}} \alpha_m$ читается как *«из α_1 выводится α_m в грамматике G»*
- Рефлексивность: $\alpha \overset{*}{\underset{G}{\Rightarrow}} \alpha$

Языки и грамматики

- **Язык, порождаемый** грамматикой G:

  $L(G) = \{\omega | \omega \in V_T^*, S \overset{*}{\underset{G}{\Rightarrow}} \omega\}$ -- множество всех терминальных строк, выводимых из начального нетерминала грамматики

- **Сентенциальной формой** называется любая строка α такая что

  $$\alpha \in V^* \text{ и } S \overset{*}{\underset{G}{\Rightarrow}} \alpha$$

- Грамматики G_1 и G_2 **эквивалентны**, если $L(G_1) = L(G_2)$

Пример: $G = (\{S\}, \{0,1\}, \{S \to 0S1, S \to 01\}, S)$

$$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 0^3S1^3 \Rightarrow \cdots \Rightarrow 0^{n-1}S1^{n-1} \Rightarrow 0^n1^n$$

$$L(G) = \{0^n1^n | \mathrm{n} > 0\}$$

Пример грамматики

Пример 2: $G = (\{S, B, C\}, \{a, b, c\}, P, S)$

$$P = \begin{cases} S \to aSBC\ (1) \\ S \to aBC\ (2) \\ CB \to BC\ (3) \\ aB \to ab\ (4) \\ bB \to bb\ (5) \\ bC \to bc\ (6) \\ cC \to cc\ (7) \end{cases}$$

$$S \xRightarrow{(1)\times(n-1)} a^{n-1}S(BC)^{n-1} \xRightarrow{(2)} a^n(BC)^n \xRightarrow{(3)\times\frac{n(n-1)}{2}} a^nB^nC^n$$

$$\xRightarrow{(4)} a^nbB^{n-1}C^n \xRightarrow{(5)\times(n-1)} a^nb^nC^n \xRightarrow{(6)} a^nb^ncC^{n-1} \xRightarrow{(7)} a^nb^nc^n$$

$$L(G) = \{a^nb^nc^n | n \geq 1\}$$

Возможен только такой порядок применения правил, т.к. иначе порождается нетерминальная строка.

Классификация грамматик по Хомскому

- Грамматика **типа 0** – без ограничений
- Грамматика **типа 1 (неукорачивающая, контекстно-зависимая (КЗ), context-sensitive grammar (CSG))**

  Каждое правило имеет вид $\boldsymbol{\alpha \to \beta \in P}$: $|\boldsymbol{\alpha}| \leq |\boldsymbol{\beta}|$

  Почему контекстно-зависимые?

  $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2, \quad A \in V_N, \beta \neq \varepsilon$

  Можно показать, что порождаемые языки одинаковы.
- Грамматика **типа 2 (контекстно-свободная (КС), context-free grammar)**

  Каждое правило имеет вид $\boldsymbol{A \to \beta \in P, A \in V_N,\ \beta \in V^+}$
- Грамматика **типа 3 (регулярная, regular grammar)**

  Каждое правило имеет вид $\boldsymbol{A \to aB}$ или $\boldsymbol{A \to a}$, $\boldsymbol{a \in V_T, A, B \in V_N}$

Можно показать, что правила вида $A \to xB$ или $A \to x$, $x \in V_T^+, A, B \in V_N$ также задают регулярную грамматику

Язык, порождаемый грамматикой типа i, и не порождаемый грамматикой типа $i+1$, называют языком типа i.

Пусть K_i -- класс языков типа i, тогда $K_3 \subset K_2 \subset K_1 \subset K_0$

Пустое слово

- Пустое слово: $!\exists L \in K_1\ (K_2, K_3)$: $\varepsilon \in L$

  Расширим 1-3 классы, добавим правило $S \to \varepsilon$, при условии, что S не появляется в правой части никакого правила.

Хотим доказать, что
если $L \in K_i$ $(i = 1,2,3)$, то $L \cup \{\varepsilon\} \in K_i$ и $L \setminus \{\varepsilon\} \in K_i$

Лемма

Если $G = (V_N, V_T, P, S)$ – КЗ-, КС- или регулярная грамматика, то существует G_1 *такого же типа*, которая порождает тот же самый язык, и в которой ни одно правило не содержит стартовый нетерминал в своей правой части.

Пустое слово

Если $G = (V_N, V_T, P, S)$ – КЗ-, КС- или регулярная грамматика, то существует G_1 такого же типа, которая порождает тот же самый язык и в которой ни одно правило не содержит начальный символ в своей правой части.

Доказательство

Пусть S_1: $S_1 \notin V_T \cup V_N$

Пусть $G_1 = (V_N \cup \{S_1\}, V_T, P_1, S_1)$,

$P_1 = P \cup \{S_1 \to \alpha\}$, для каждого правила $S \to \alpha \in P$

$S_1 \notin V \Rightarrow S_1$ – нет в правой части.

Покажем $L(G) = L(G_1)$.

1. $L(G) \subseteq L(G_1)$

Пусть $x \in L(G)$, тогда $S \Rightarrow_G^* x$. Пусть 1-ое правило $S \to \alpha \in P$, т. е.

$S \Rightarrow_G \alpha \Rightarrow_G^* x$, тогда существует правило $S_1 \to \alpha \in P_1$: $S_1 \Rightarrow_{G_1} \alpha$ и $S_1 \Rightarrow_{G_1}^* x$.

2. $L(G) \supseteq L(G_1)$

Пусть $x \in L_1(G)$, тогда $S_1 \Rightarrow_G^* x$. Пусть 1-ое правило $S_1 \to \alpha \in P_1$, тогда есть $S \to \alpha \in P$ и $S \Rightarrow_G \alpha$. $\alpha \Rightarrow_{G_1} x$ и $S_1 \notin \alpha$, поэтому используются правила из P и $\alpha \Rightarrow_G^* x$.

3. 1 & 2 => $L(G) = L(G_1)$

Пустое слово

Теорема

Если L - КЗ-, КС- или регулярный язык, то $L \cup \{\varepsilon\}$ и $L \setminus \{\varepsilon\}$ -- КЗ-, КС- или регулярный язык соответственно.

Доказательство

Согласно лемме существует G порождающая L и не содержащая S в правой части правил.

1. $\varepsilon \notin L(G)$.

G_1: добавим $S \to \varepsilon$. S нет в правой части, значит $S \to \varepsilon$ – первое и единственное

$L(G_1) = L(G) \cup \{\varepsilon\}$

2. $\varepsilon \in L(G)$.

Тогда среди правил есть $S \to \varepsilon$, отбросив которое получим G_1 , порождающую $L(G_1) = L(G) \setminus \{\varepsilon\}$

3. Согласно лемме типы грамматик G и G_1 совпадают

Примеры

- $G = (\{S, A, B\},\ \{0,1\},\ \{S \to AB, A \to 0A, A \to 0, B \to 1B, B \to 1\},\ S)$
  Контекстно-свободная

  $L(G) = \{0^n 1^m | n, m > 0\}$

- $G_1 = (\{S, A\},\ \{0,1\},\ S \to 0S, S \to 0A, A \to 1A, A \to 1\},\ S)$
  Регулярная

Если язык порождается некоторой грамматикой, то это не означает, что не существует грамматики с более сильными ограничениями, порождающей заданный язык

Лексический анализ

- Первый этап анализа исходного текста программы в компиляторе
- Входная строка разбивается на цепочку лексем (token)
  - *Например*: <константа,5>, <ключевое слово, for>, <идентификатор, int>
- Отдельная стадия анализа или «по запросу» синтаксического анализа
- Задается конечным автоматом
  - На практике регулярное выражение или регулярная грамматика

Итерация на множествах

- Пусть $\mathrm{P}, Q \subseteq V^*$ – некоторые языки.
- **Произведением (конкатенацией)** PQ *языков* P и Q называется множество, состоящее из всех конечных слов в алфавите A, которые составлены конкатенацией некоторого слова из P и некоторого слова из Q.
- Т.е. $PQ = \{pq \mid p \in P, q \in Q\}$.

Итерация (замыкание Клини) языка L обозначается L^* и представляет собой множество всех слов, которые можно образовать путем конкатенации любого количества конечных слов из L.
При этом допускаются повторения, т.е. одна и та же цепочка из L может быть выбрана для конкатенации более одного раза

$$\boldsymbol{P^* = \bigcup_{n=0}^{\infty} P^n,\ P^0 = \{\varepsilon\}, P^1 = P}$$

- Пусть $P = \{0, 11\}$
  $P^0 = \{\varepsilon\},$
  $P^1 = P = \{0, 11\}$
  $P^2 = \{\, 0 \cdot 0 = \mathbf{00}, 0 \cdot 11 = \mathbf{011}, 11 \cdot 0 = \mathbf{110}, 11 \cdot 11 = \mathbf{1111}\}$
  $P^3 = \{000, 0011, 0110, 1100, 01111, 11011, 11110, 111111\}$
- $\emptyset^* = \{\boldsymbol{\varepsilon}\}$

Регулярные множества/языки

- **Регулярное множество** в алфавите V_T определяется рекурсивно:
  1. $\emptyset$ – регулярное множество в алфавите V_T;
  2. $\{\varepsilon\}$ – регулярное множество в алфавите V_T;
  3. $\{a\}$ – регулярное множество в алфавите $V_T, a \in V_T$;
  4. Если P и Q – регулярные множества в алфавите V_T, то регулярными множествами являются и множества:
     a) $P \cup Q$,
     b) $P \cdot Q = PQ$ (конкатенация: $\{pq | p \in P, q \in Q\}$),
     c) P^* (итерация: $P^* = \bigcup_{n=0}^{\infty} P^n$)., $\boldsymbol{P^0 = \{\varepsilon\}, P^1 = P}$
  5. Ничто другое регулярным множеством в алфавите V_T не является.

Регулярные выражения

- Форма записи регулярных множеств
- **Регулярное выражение (РВ)** в алфавите V_T и обозначаемое им регулярное множество определяется следующим образом:
  1. $\emptyset$ – регулярное выражение, обозначающее регулярное множество $\emptyset$;
  2. ε – регулярное выражение, обозначающее регулярное множество $\{\varepsilon\}$;
  3. a – регулярное выражение, обозначающее регулярное множество $\{a\}$;
  4. Если p и q – регулярные выражения, обозначающие регулярные множества P и Q соответственно, то:
     a) $(p|q)$ – РВ, обозначающее $P \cup Q$,
     b) (pq) – РВ, обозначающее PQ,
     c) (p^*) – РВ, обозначающее P^*
  5. Ничто другое регулярным выражением в алфавите V_T не является.

- Приоритет операций:
  1. * Итерация
  2. · Конкатенация
  3. |

$$01^*|1 = \big(0(1^*)\big)|1$$

- $p^+ = pp^*$
- запись $L(r)$ для регулярного множества, обозначаемого регулярным выражением r.

Регулярные множества/языки

- Примеры
- Какое множество обозначает РВ?
  - $a(\varepsilon|a)|b$

Регулярные множества/языки

- Примеры
- Какое множество обозначает РВ?
  - $a(\varepsilon|a)|b$ обозначает $\{a, b, aa\}$

Регулярные множества/языки

- Примеры
  - $a(\varepsilon|a)|b$ обозначает $\{a, b, aa\}$
- Какое множество обозначает РВ?
  - $a(a|b)^*$?

Регулярные множества/языки

- Примеры
  - $a(\varepsilon|a)|b$ обозначает $\{a, b, aa\}$
- Какое множество обозначает РВ?
  - $a(a|b)^*$?

Регулярные множества/языки

- Примеры

  - $a(\varepsilon|a)|b$ обозначает $\{a, b, aa\}$
  - $a(a|b)^*$ обозначает множество всевозможных цепочек, состоящих из a и b, начинающихся с a;

- Какое множество обозначает РВ?

  - $(a|b)^*(a|b)(a|b)^*$?

Регулярные множества/языки

- Примеры

  - $a(\varepsilon|a)|b$ обозначает $\{a, b, aa\}$
  - $a(a|b)^*$ обозначает множество всевозможных цепочек, состоящих из a и b, начинающихся с a;
  - $(a|b)^*(a|b)(a|b)^*$ обозначает множество всех непустых цепочек, состоящих из a и b, т. е. множество $\{a, b\}^+$;

- Какое множество обозначает РВ?

  - $\big((0|1)(0|1)(0|1)\big)^*$?

Регулярные множества/языки

➢ Примеры

- $a(\varepsilon|a)|b$ обозначает $\{a, b, aa\}$
- $a(a|b)^*$ обозначает множество всевозможных цепочек, состоящих из a и b, начинающихся с a;
- $(a|b)^*(a|b)(a|b)^*$ обозначает множество всех непустых цепочек, состоящих из a и b, т. е. множество $\{a, b\}^+$;
- $\big((0|1)(0|1)(0|1)\big)^*$ обозначает множество всех цепочек, состоящих из нулей и единиц, длины которых делятся на 3.

➢ Будем говорить, что регулярные выражения **равны**, или **эквивалентны** $(=)$, если они обозначают одно и то же регулярное множество.

Регулярные множества/языки

- Примеры

  - $a(\varepsilon|a)|b$ обозначает $\{a, b, aa\}$
  - $a(a|b)^*$ обозначает множество всевозможных цепочек, состоящих из a и b, начинающихся с a;
  - $(a|b)^*(a|b)(a|b)^*$ обозначает множество всех непустых цепочек, состоящих из a и b, т. е. множество $\{a, b\}^+$;
  - $\big((0|1)(0|1)(0|1)\big)^*$ обозначает множество всех цепочек, состоящих из нулей и единиц, длины которых делятся на 3.

- Напишите регулярное выражение, задающее множество всех цепочек в алфавите {0, 1}, в которых символ 1 является четвертым с конца.

Регулярные множества/языки

- Примеры

  - $a(\varepsilon|a)|b$ обозначает $\{a, b, aa\}$
  - $a(a|b)^*$ обозначает множество всевозможных цепочек, состоящих из a и b, начинающихся с a;
  - $(a|b)^*(a|b)(a|b)^*$ обозначает множество всех непустых цепочек, состоящих из a и b, т. е. множество $\{a, b\}^+$;
  - $\big((0|1)(0|1)(0|1)\big)^*$ обозначает множество всех цепочек, состоящих из нулей и единиц, длины которых делятся на 3.

- Напишите регулярное выражение, задающее множество всех цепочек в алфавите $\{0, 1\}$, в которых символ 1 является третьим с конца.

  - $(0|1)^*1(0|1)(0|1)$

- Напишите регулярное выражение, задающее множество всех цепочек в алфавите $\{x, y, z\}$, которые **не содержат** подцепочку xy.

Регулярные множества/языки

- Примеры
  - $a(\varepsilon|a)|b$ обозначает $\{a, b, aa\}$
  - $a(a|b)^*$ обозначает множество всевозможных цепочек, состоящих из a и b, начинающихся с a;
  - $(a|b)^*(a|b)(a|b)^*$ обозначает множество всех непустых цепочек, состоящих из a и b, т. е. множество $\{a, b\}^+$;
  - $\big((0|1)(0|1)(0|1)\big)^*$ обозначает множество всех цепочек, состоящих из нулей и единиц, длины которых делятся на 3.
- Напишите регулярное выражение, задающее множество всех цепочек в алфавите $\{0, 1\}$, в которых символ 1 является третьим с конца.

  - $(0|1)^*1(0|1)(0|1)$

- Напишите регулярное выражение, задающее множество всех цепочек в алфавите $\{x, y, z\}$, которые **не содержат** подцепочку xy.

1. $(x^*z \mid y)^*x^*$
2. $y^*(zz^*\,y^* \mid x^*)^*$
3. $y^*\,(x^*\,z\,y^*\,)^*x^*$
4. $(y \mid z)^*(x^*\,zz^*\,y^* \mid z)^*x^*$

Свойства регулярных выражений

Пусть p, q, r – регулярные выражения, тогда

$$p|q = q|p$$

$$\emptyset^* = \varepsilon$$

$$\varepsilon^* = \varepsilon$$

$$p|(q|r) = (p|q)|r$$

$$p(qr) = (pq)r$$

$$p(q|r) = pq|pr$$

$$(p|q)r = pr|qr$$

$$p\varepsilon = \varepsilon p = p$$

$$\emptyset p = p\emptyset = \emptyset$$

$$p^* = p \mid p^*$$

$$(p^*)^* = p^*$$

$$p|p = p$$

$$p|\emptyset = p$$

Именование регулярных выражений

Пример

- 1.

$Letter = a|b|c|\dots|x|y|z,$
$Digit = 0|1|\dots|9,$
$Identifier = Letter(Letter|Digit)^*$

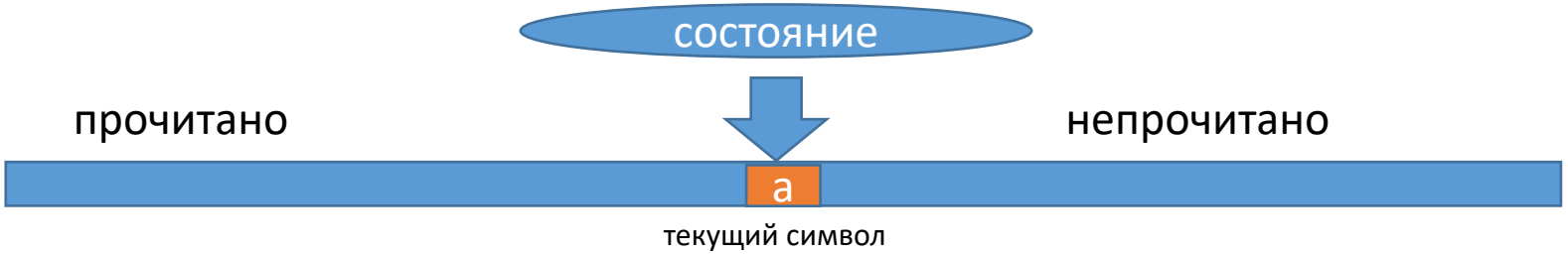
- 2.

$Digit = 0|1|\dots|9,$
$Integer = Digit^+$
$Fraction = .Integer|\varepsilon$
$Exponent = (E(+|-|\varepsilon)Integer)|\varepsilon$

Конечные автоматы

- Регулярные выражения -- для задания регулярных множеств
- Конечные автоматы – для распознавания регулярных множеств

- *Недетерминированный конечный автомат (НКА)* -- $M = (Q, \Sigma, \delta, q_0, F)$:
  - Q – конечное непустое множество состояний;
  - Σ – конечный входной алфавит;
  - $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \to Q$ – функция переходов
  - $q_0 \in Q$ – начальное состояние;
  - $F \subseteq Q$ – множество заключительных состояний.



- *Конфигурация автомата* $(q, w) \in Q \times \Sigma^*$. q – текущее состояние, w – цепочка символов под головкой и справа на ленте
  - (q_0, w) – *начальная* конфигурация
  - $(q, \varepsilon), q \in F$ – *заключительная* (*допускающая*) конфигурация

Конечные автоматы

- *Такт* автомата M ($\vdash$) – бинарное отношение, заданное на конфигурациях: если $p \in \delta(q, a), a \in \Sigma \cup \{\varepsilon\}$, то $(q, aw) \vdash (p, w)$ для всех $w \in \Sigma^*$

- $\vdash^*, \vdash^+$ -- рефлексивно-транзитивное (транзитивное) замыкание $\vdash$

- Автомат M ***допускает*** цепочку w, если $(\boldsymbol{q_0}, \boldsymbol{w}) \vdash^* (\boldsymbol{q}, \boldsymbol{\varepsilon}), \boldsymbol{q} \in \boldsymbol{F}$

- Язык L *распознаваемый (допускаемый, определяемый)* M*:*
$$L = L(M) = \{w | w \in \Sigma^* \wedge (q_0, w) \vdash^* (q, \varepsilon), q \in F\}$$

- ***Детерминированный*** **конечный автомат (ДКА)** – НКА $M = (Q, \Sigma, \delta, q_0, F)$, если
  - $\forall q \in Q : \delta(q, \varepsilon) = \emptyset$;
  - $\forall (q, a) \in Q \times \Sigma : |\delta(q, a)| \leq 1$