

## Регулярные выражения

Регулярные выражения – мощное, гибкое и эффективное средство обработки текстов. Универсальные шаблоны регулярных выражений сами по себе напоминают миниатюрный язык программирования, предназначенный для описания и разбора текста. При дополнительной поддержке со стороны конкретной утилиты или языка программирования регулярные выражения способны вставлять, удалять, выделять и выполнять самые невероятные операции с текстовыми данными любого вида. Они бывают очень простыми, вроде команды поиска в текстовом редакторе, или очень сложными, как специализированные языки обработки текстов.

Регулярные выражения тесно связаны с теорией автоматов и формальных языков, поэтому они часто распознаются с использованием инструментов данных разделов математики.

Поэтому для начала разберемся с теорией автоматов.

## Конечные автоматы и регулярные языки

Языком будем называть множество слов над некоторым алфавитом.

### Определение

Конечным автоматом будем называть пятерку  $(Q, \Sigma, \delta, q_0, F)$ , где:

- $Q$  – конечное множество состояний
- $\Sigma$  – конечное множество, называемое алфавитом
- $\delta : Q \times \Sigma \rightarrow Q$  – функция переходов
- $q_0 \in Q$  – начальное состояние
- $F \subseteq Q$  – множество конечных состояний

### Определение

Автомат  $A = (Q, \Sigma, \delta, q_0, F)$  распознает (допускает) строку  $w = \overline{w_1 \dots w_n}$ , если:

- $w_i \in \Sigma, \forall i = \overline{1, n}$
- существует последовательность  $q_0 = r_0, r_1, \dots, r_n$  такая что:
  - $r_n \in F$
  - $\delta(r_i, w_i) = r_{i+1}, \forall i = \overline{0, n-1}$

Языком распознаваемым автоматом  $A$  называется  $L = \{w \mid A \text{ допускает } w\}$ .

## Определение

Язык называется регулярным, если его распознает некоторый конечный автомат.

Различают детерминированные и недетерминированные конечные автоматы. Разница в том, что в детерминированных автоматах не может быть переходов по пустой строке, и множественных переходов по некоторому слову (то есть переходов в несколько состояний).

По сути функция перехода для недетерминированного автомата выглядит следующим образом:

$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow B(Q)$ , где  $B(Q)$  — множество всех подмножеств  $Q$ ,  $\varepsilon$  — пустая строка.

Известна следующая теорема:

## Теорема

Для любого недетерминированного автомата существует эквивалентный ему детерминированный автомат (автоматы называются эквивалентными если языки, распознаваемые ими равны).

Регулярные языки замкнуты относительно следующих операций:

1. Объединение:  $A \cup B = \{w \mid w \in A \text{ или } w \in B\}$
2. Пересечение:  $A \cap B = \{w \mid w \in A \text{ и } w \in B\}$
3. Дополнение:  $\bar{A} = \{w \in \Sigma \cup \{\varepsilon\} \mid w \notin A\}$
4. Обращение:  $A^R = \{a_1 \dots a_k \mid a_k \dots a_1 \in A\}$
5. Конкатенация:  $A \circ B = \{vw \mid v \in A \text{ и } w \in B\}$
6. Замыкание Клини:  $A^* = \{w_1 \dots w_k \mid k \geq 0 \text{ и } w_i \in A \text{ для всех } i\}$

Соответственно регулярные выражения определяются рекурсивно:

1.  $\emptyset$  — регулярное выражение, обозначающее пустое множество
2.  $e$  — регулярное выражение, обозначающее множество  $\{e\}$
3.  $a$  — регулярное выражение, обозначающее множество  $\{a\}$
4. если  $p$  и  $q$  регулярные выражения, обозначающие регулярные языки  $P$  и  $Q$  соответственно, то:
  - a.  $pq = \{uw \mid u \in P \text{ и } w \in Q\}$
  - b.  $p|q = \{u \mid u \in P \text{ или } u \in Q\}$
  - c.  $p^*$  — минимальное надмножество множества  $P$ , замкнутое относительно конкатенации.

$L(R)$  – язык, которому соответствует регулярное выражение  $R$

Регулярное выражение допускает строку  $w$ , если  $w \in L(r)$ .

Будем также использовать обозначение  $p^+ = pp^*$

Работа конечного автомата представляет собой некоторую последовательность шагов, или тактов. Такт определяется текущим состоянием управляющего устройства и входным символом, обозреваемым в данный момент входной головкой. Сам шаг состоит из изменения состояния и, возможно, сдвига входной головки на одну ячейку вправо:



Конфигурацией будем называть пару  $(q, w) \in Q \times T^*$  ( $T^* = T \cup \{\varepsilon\}$ ).  $(q_0, w)$  – начальная конфигурация,  $(q, \varepsilon)$ , где  $q \in F$  – заключительная (допускающая). Таким образом после каждого такта наш автомат меняет конфигурацию. Более формально тактом определим следующим образом:

#### Определение

Тактом будем называть бинарное отношение  $\vdash$  на множестве конфигураций, такое что если  $q' \in \delta(q, a)$ , то  $(q, aw) \vdash (q', w)$ ,  $\forall w \in T^*$ .

Будем также рассматривать соответствующие транзитивное и рефлексивно-транзитивное замыкание нашего бинарного отношения  $\vdash^+$ ,  $\vdash^*$ .

Таким образом автомат распознает строку  $w$  тогда и только тогда, когда  $(q_0, w) \vdash^*(q, \varepsilon)$ , где  $q \in F$ .

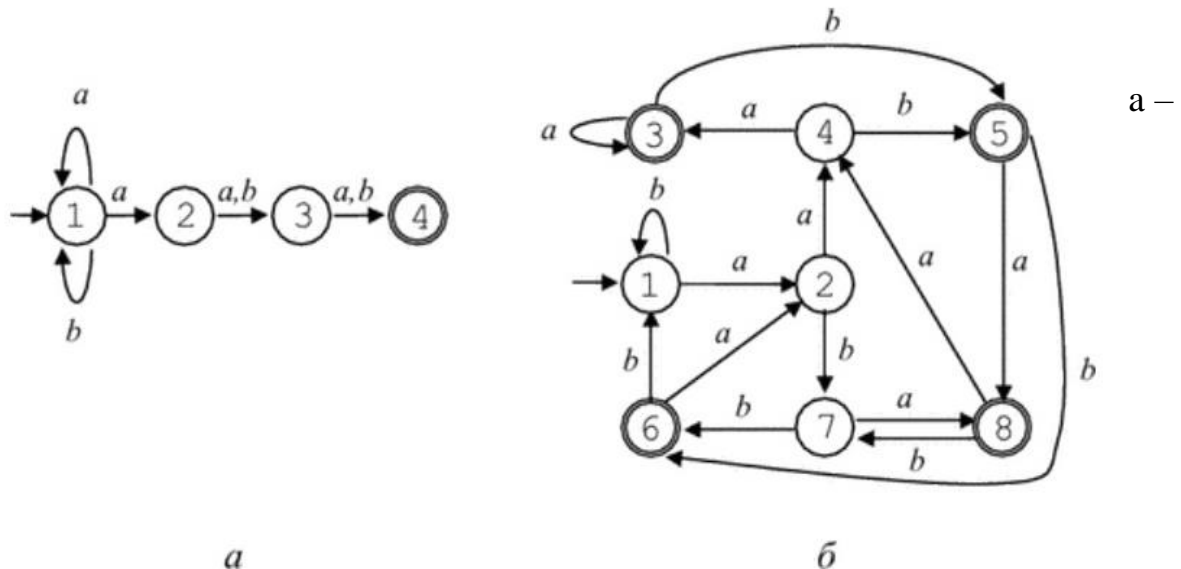
Конечные автоматы удобно представлять в виде ориентированного графа, в котором вершинами будут состояния нашего автомата, а между вершинами

будет существовать ребро с меткой из алфавита  $\Sigma \cup \{\varepsilon\}$ , если соответствующий переход есть в функции переходов.

### Пример

Рассмотрим регулярное выражение  $R = (a|b) * a(a|b)(a|b)$

Конечные автоматы, соответствующие данному регулярному выражению:



недетерминированный конечный автомат, б – детерминированный.

- Попробуем проанализировать цепочку  $w = ababa$  с помощью автомата из пункта а.

Наш автомат может сделать следующую цепочку тактов:

$$(1, ababa) \vdash (1, baba) \vdash (1, aba) \vdash (2, ba) \vdash (3, a) \vdash (4, e).$$

Поскольку состояние 4 является заключительным, то наш автомат распознает строку  $w$ .

- Попробуем проанализировать цепочку  $w = ababab$  с помощью автомата из пункта б.

Поскольку автомат детерминированный то цепочка тактов определяется однозначно, и она равна:

$$(1, ababab) \vdash (2, babab) \vdash (7, abab) \vdash (8, bab) \vdash (7, ab) \vdash (8, b) \vdash (7, e)$$

Поскольку состояние 7 не является заключительным, то наш автомат не распознает строку  $w$ .

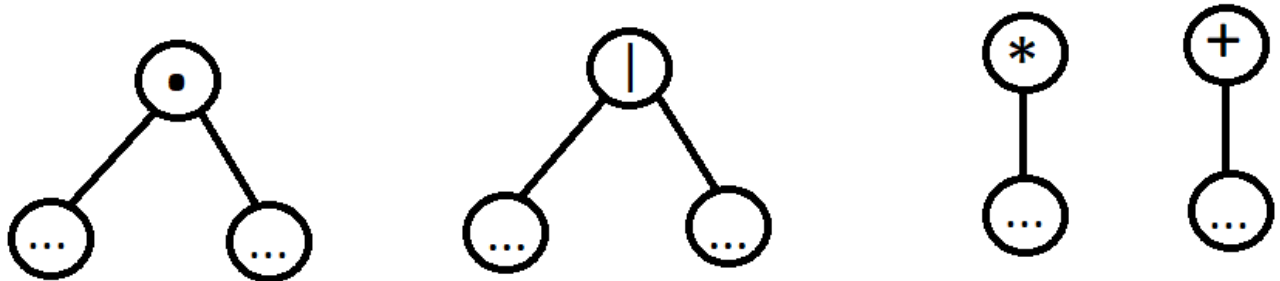
## Абстрактные синтаксические деревья

Абстрактное синтаксическое дерево есть представление какого-либо вычисляемого выражения в виде дерева. Это может быть какое-нибудь арифметическое выражение, или же последовательность команд в языке программирования. В нашем случае дерево будет представлять регулярное выражение.

Внутренние вершины нашего дерева будут помечены некоторыми операндами нашего регулярного выражения. Варианты операндов:

- (“.”) – конкатенация двух регулярных выражений.
- (“|”) – дизъюнкция двух регулярных выражений.
- (“\*”) – замыкание Клини некоторого регулярного выражения
- (“+”) – замыкание Клини без пустых строк некоторого регулярного выражения.

Листьями будет вершина обозначающая некоторый символ алфавита (“sym”).



По сути абстрактное синтаксическое дерево для регулярного выражения задает в каком порядке можно проверять распознает ли регулярное выражение некоторую строку.

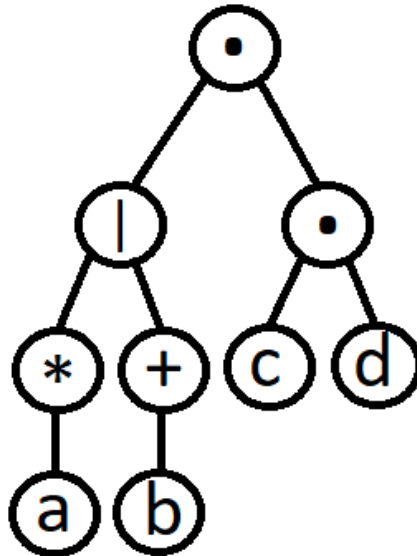
Нетрудно проанализировать распознавание некоторой строки регулярным выражением в случае если в синтаксическом дереве нету вершин (“.”). Однако даже для простейшего случай конкатенации (например  $a^+b^+$ ) пришлось бы для строки находить множество префиксов, удовлетворяющих левой части конкатенации, множество суффиксов, удовлетворяющих правой части конкатенации и проверять что есть префикс и суффикс, составляющие всю строку, и распознаваемые соответствующими частями конкатенации. Это представляется довольно трудоемким и сложным процессом.

Поэтому использовать синтаксические деревья для распознавания регулярных выражений мы не будем. Вместо этого синтаксические деревья будут

использоваться для построения конечных автоматов, с помощью которых уже можно легко проверить распознавание некоторой строки.

### Пример

Построим абстрактное синтаксическое дерево для регулярного выражения  $R = (a^*|b^+)(cd)$ .



## Алгоритмы

Пускай у нас есть некоторое регулярное выражение  $R$ . Для распознавания строк этим регулярным выражением нам необходимо построить для него конечный автомат (неважно детерминированный или недетерминированный).

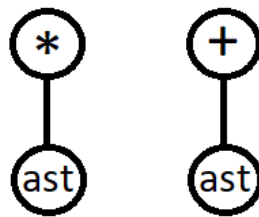
Однако прежде чем построить автомат нам необходимо построить абстрактное синтаксическое дерево.

### Алгоритм для построения абстрактного синтаксического дерева

Для начала нам необходимо определиться с приоритетом операций. Наивысшим приоритетом у нас будут обладать операции " $*$ ", " $+$ ". Потом идет операция конкатенации, а последним идет операция дизъюнкции.

Алгоритм выглядит следующим образом:

- Храним последовательности АСД, которые будут объединены конкатенацией. Например для регулярного выражения  $R = (a|b)cd^*|f(g^*e^+)^+$  мы имеем две последовательности:  $s_1 = \{\text{АСД}((a|b)), \text{АСД}(c), \text{АСД}(d^*)\}$ ,  $s_2 = \{\text{АСД}(f), \text{АСД}((g^*e^+)^+)\}$
- Идем по строке и смотрим текущий символ. Если символ равен:
  - " $+$ " или " $*$ ", то мы берем последнее АСД из последней существующей последовательности АСД (назовем его  $ast$ ) и заменяем его на АСД следующего вида (в зависимости от случая):



- если мы встретили открывающую скобку, то мы рекурсивно строим АСД для выражения в скобках и добавляем его в последнюю последовательность.
- если мы встретили закрывающую скобки, то мы объединяем последовательности дизъюнкцией и возвращаем полученное АСД из функции.
- если мы встретили обычный символ, то добавляем в последнюю последовательность вершину с одним символом.
- если мы встретили " $|$ ", то начинаем новую последовательность АСД и заполняем ее.

- если закончилась строка, то мы объединяем последовательности дизъюнкцией и возвращаем полученное АСД из функции.

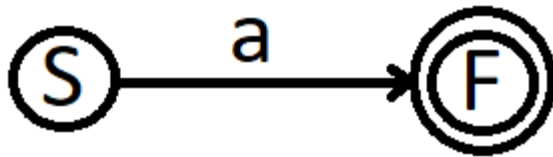
После отработки данного алгоритма мы получим АСД для нашего регулярного выражения.

### Алгоритм для построения недетерминированного конечного автомата (НКА) из АСД

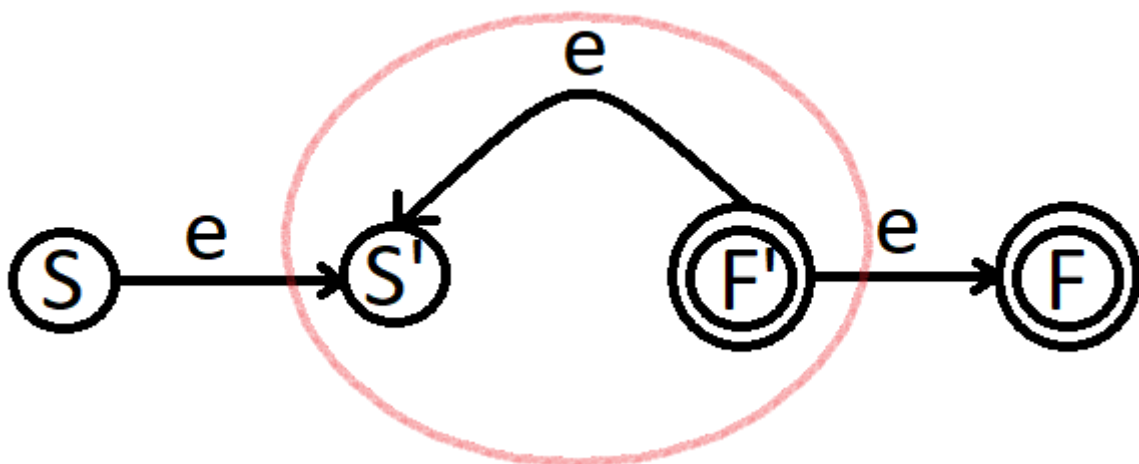
У построенных нами автоматов будет лишь одна заключительная вершина, поэтому будем обозначать начальную и заключительную вершину  $S$  и  $F$  соответственно.

Проходимся по АСД и в зависимости от типа вершины делаем следующее:

- если у нас обычная символьная вершина, то строим автомат следующего вида:



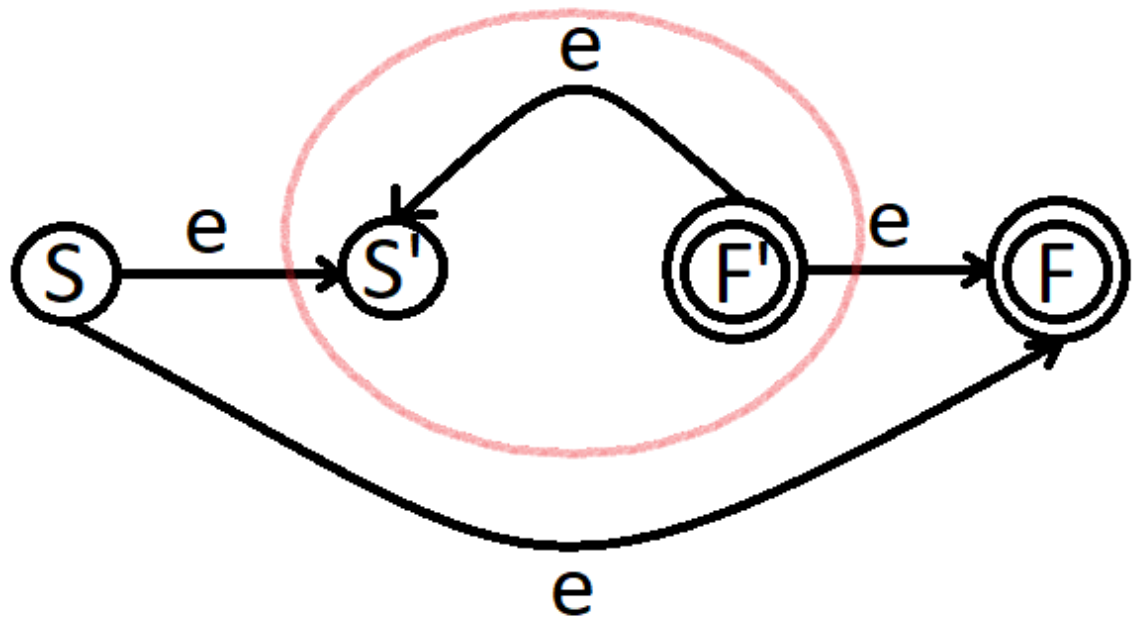
- если у нас вершина типа (" + "), то мы строим для внутреннего выражения автомат, а потом достраиваем его до автомата вида:





где  $S'$  и  $F'$  соответственно начальное и заключительное состояние для внутреннего регулярного выражения.

- если у нас вершина типа (" $*$ "), то мы строим для внутреннего выражения автомат, а потом достраиваем его до автомата вида:



где  $S'$  и  $F'$  соответственно начальное и заключительное состояние для внутреннего регулярного выражения.

- если у нас вершина типа дизъюнкции