



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ \_\_\_\_\_ «Информатика и системы управления»

КАФЕДРА \_\_\_\_\_ «Теоретическая информатика и компьютерные технологии»

## Лабораторная работа № 2

### по курсу «Теория формальных языков»

Студент группы ИУ9-51Б Винокурова Е. С.

Преподаватель Непейвода А.Н.

Москва 2025

# Содержание

<b>1</b>	<b>Задача</b>	<b>3</b>
<b>2</b>	<b>Минимальный ДКА</b>	<b>4</b>
2.1	Обоснование минимальности ДКА . . . . .	5
<b>3</b>	<b>Возможно малый НКА</b>	<b>5</b>
3.1	Частичное обоснование минимальности НКА . . . . .	6
<b>4</b>	<b>Возможно малый ПКА</b>	<b>6</b>
4.1	частичное обоснование минимальности ПКА . . . . .	7
<b>5</b>	<b>Расширенное регулярное выражение</b>	<b>8</b>
<b>6</b>	<b>Фазз-тестирование</b>	<b>8</b>

# 1 Задача

По имеющемуся академическому регулярному выражению построить:

- Минимальный ДКА, распознающий его язык (минимальность обосновать таблицей классов эквивалентности)
- Возможно малый НКА, распознающий его язык. Возможно малый переключаящийся (с конъюнкцией) КА, распознающий его язык. Частично обосновать таблицами множеств классов эквивалентности.
- Расширенное регулярное выражение, распознающее тот же язык. В расширенном выражении можно использовать:
  - wildcard-операцию  $.$  для замены произвольного символа алфавита;
  - положительную итерацию  $\tau^+$  и опцию  $\tau?$ .  $\tau^+ = \tau\tau^*$ ,  $\tau? = (\tau|\varepsilon)$ ;
  - операции предпросмотра  $\tau_0(? = \tau_1)\tau_2 \equiv \tau_0((\tau_1.*) \cap \tau_2)$  и ретроспективной проверки  $\tau_0(? \leq \tau_1)\tau_2 \equiv (\tau_0 \cap (\tau_1.*))\tau_2$ , а также их отрицательные версии  $\tau_0(?!\tau_1)\tau_2 \equiv \tau_0((\tau_1.*) \cap \tau_2)$  и  $\tau_0(? <!\tau_1)\tau_2 \equiv (\tau_0 \cap (\tau_1.*))\tau_2$
  - классы букв  $[c_1 \dots c_k] \equiv (c_1|c_2|\dots|c_k)$  и их дополнения  $[\hat{c}_1 \dots \hat{c}_k]$ .
  - (обязательно) маркеры начала и конца выражения  $\wedge$  и  $\$$ .

Провести автоматическое тестирование предполагаемой эквивалентности построенных распознавателей. Тем самым необходимо построить алгоритмы, определяющие принадлежность слова языку академического регулярного выражения, ДКА, НКА и ПКА.

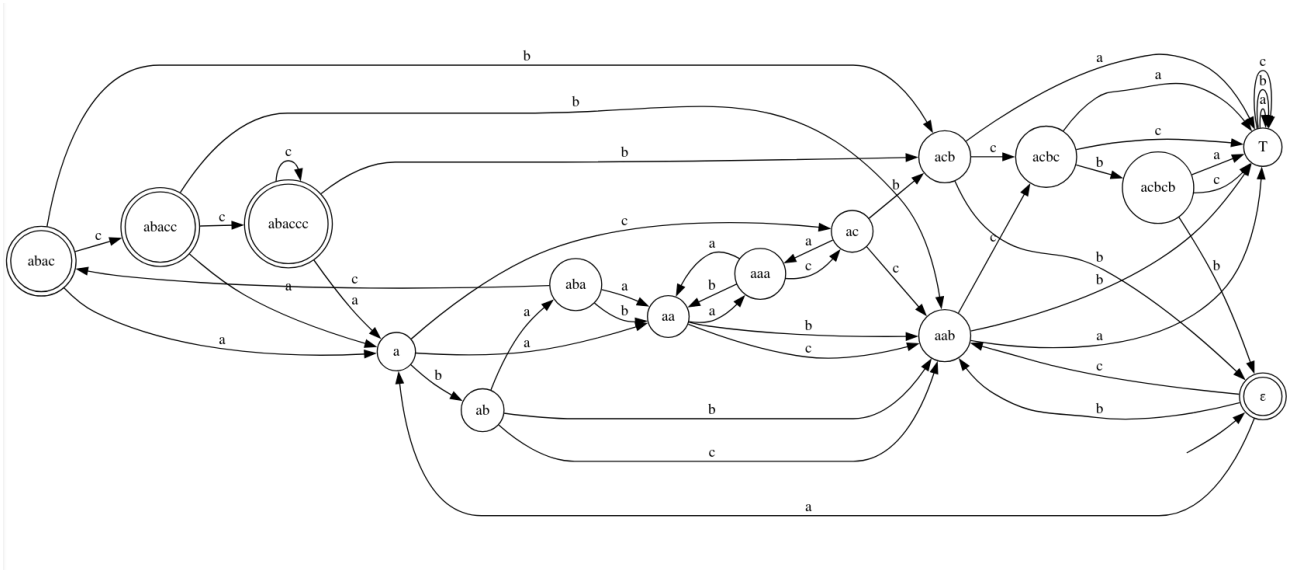
Требуется только фаза-тестирование эквивалентности: строится случайное слово  $\omega$  и проверяется, принадлежит ли он языкам регулярного выражения, ДКА, НКА и ПКА согласованно.

## Вариант 4

$$((aa|ab|ac)^*(bc|cc|ac)bb|abacc^*)^*$$

## 2 Минимальный ДКА

На основе академического регулярного выражения был построен недетерминированный конечный автомат, который затем детерминизирован и минимизирован; в результате получен минимальный детерминированный конечный автомат, распознающий данный язык. Минимальность ДКА доказана таблицей эквивалентности состояний.



Изображение этого ДКА находится в файле DKA.png

## 2.1 Обоснование минимальности ДКА

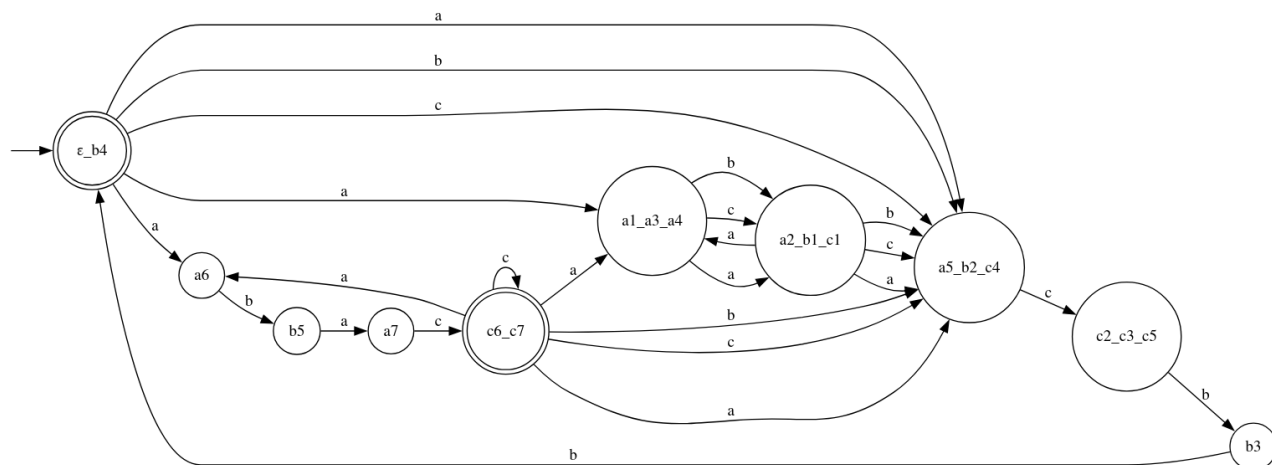
Таблица префиксов и суффиксов для ДКА

-	$\varepsilon$	$b$	$bb$	$c$	$ac$	$bcbb$	$cbb$	$bac$	$accbb$
$abac$	1	0	1	1	0	1	0	0	0
$abaccc$	1	0	1	1	0	1	1	0	0
$abacc$	1	0	0	1	0	1	1	0	0
$\varepsilon$	1	0	0	0	0	1	0	0	0
$acb$	0	1	0	0	0	0	1	0	0
$acbc$	0	1	0	0	0	0	0	0	0
$aba$	0	0	0	1	0	0	1	0	1
$ac$	0	0	1	0	0	1	0	0	0
$acbc$	0	0	1	0	0	0	0	0	0
$ab$	0	0	0	0	1	1	0	0	0
$a$	0	0	0	0	0	0	1	1	1
$aaa$	0	0	0	0	0	0	1	0	1
$aab$	0	0	0	0	0	0	1	0	0
$T$	0	0	0	0	0	0	0	0	0
$aa$	0	0	0	0	0	1	0	0	0

Построенный ДКА является минимальным, поскольку все его состояния различимы. Это подтверждается таблицей префиксов и суффиксов, так как все строки в ней различны.

## 3 Возможно малый НКА

На основе академического регулярного выражения с использованием линеаризации был построен, а затем частично минимизирован недетерминированный конечный автомат распознающий его язык.



Изображение этого НКА находится в файле NKA.png

### 3.1 Частичное обоснование минимальности НКА

Таблица префиксов и суффиксов для НКА

-	<i>bac</i>	<i>ac</i>	<i>b</i>	<i>c</i>	$\varepsilon$	<i>bb</i>	<i>cbb</i>
<i>a</i>	1	0	0	0	0	0	1
<i>ab</i>	0	1	0	0	0	0	0
<i>bcb</i>	0	0	1	0	0	0	0
<i>aba</i>	0	0	0	1	0	0	1
$\varepsilon$	0	0	0	0	1	0	0
<i>bc</i>	0	0	0	0	0	1	0
<i>b</i>	0	0	0	0	0	0	1

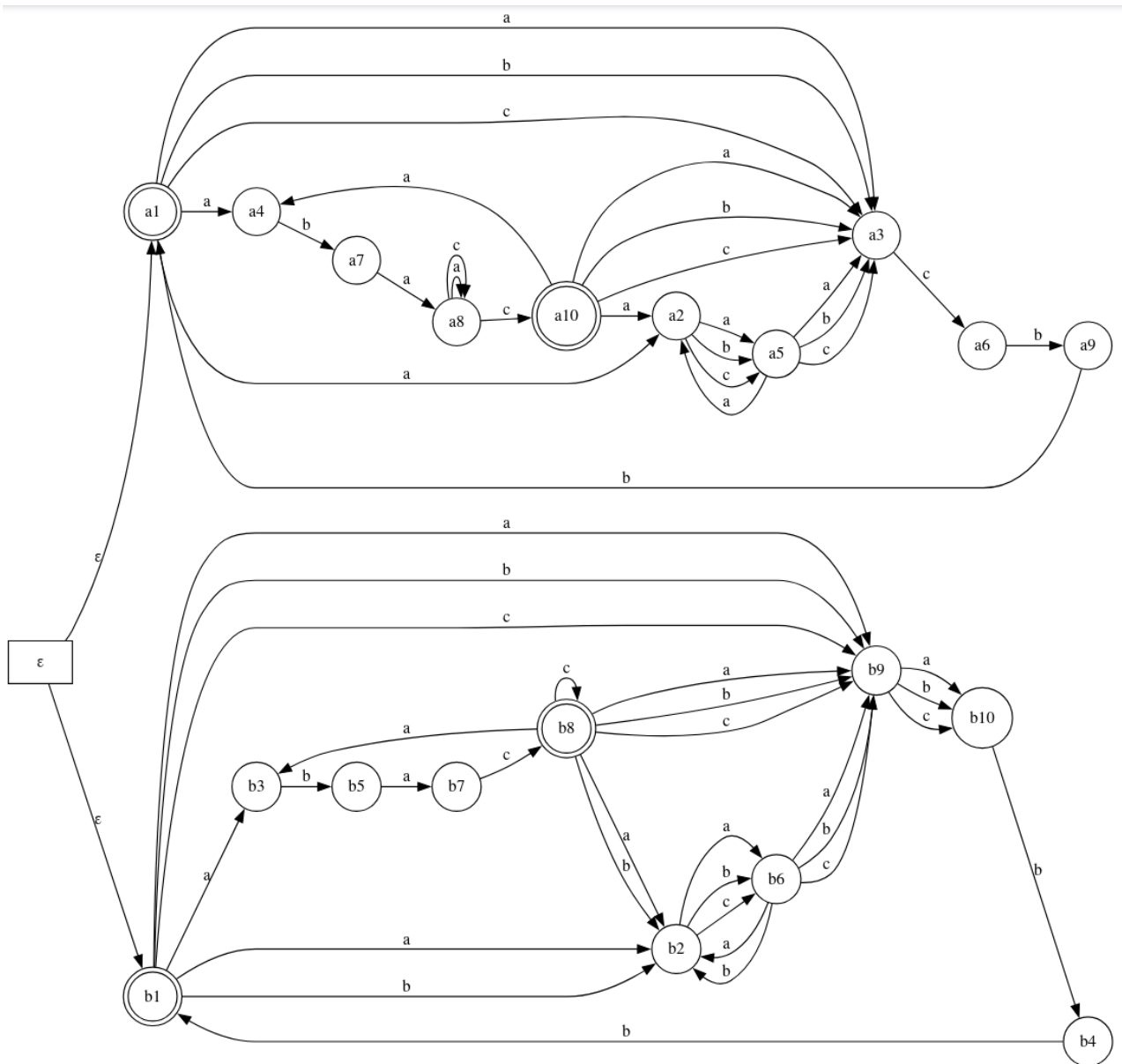
## 4 Возможно малый ПКА

На основе академического регулярного выражения был построен переключающийся конечный автомат распознающий его язык.

Каждая ветка ПКА распознает одно из регулярных выражений  $L_1$  и  $L_2$ , которые в пересечении дают исходную регулярку.

$$L_1 = ((aa|ab|ac)^*(bc|cc|ac)bb|aba(a|c)^*c)^*$$

$$L_2 = (((a|b)(a|b|c))^*(a|b|c)(a|b|c)bb|abacc^*)^*$$



Изображение этого ПКА находится в файле РКА.png

## 4.1 частичное обоснование минимальности ПКА

Таблица префиксов и суффиксов для ПКА

-	<i>b</i>	<i>bb</i>	<i>bcbb</i>	<i>c</i>	<i>cbb</i>
<i>abaccc</i>	0	1	1	1	1
<i>abacc</i>	0	0	1	1	1
<i>aba</i>	0	0	0	1	1
<i>aaa</i>	0	0	0	0	1
<i>aa</i>	0	0	1	0	0

## 5 Расширенное регулярное выражение

Исходное регулярное выражение

$$((aa|ab|ac)^*(bc|cc|ac)bb|abacc^*)^*$$

описывает произвольную (возможно пустую) последовательность блоков двух форм.

В первой форме блока подвыражение  $(aa|ab|ac)$  представляет собой все пары символов, начинающиеся с  $a$  и продолжающиеся любой буквой из множества  $\{a, b, c\}$ . Это эквивалентно записи  $(aa|ab|ac) \equiv a[abc]$

Повторение данной группы даёт  $(aa|ab|ac)^* \equiv (a[abc])^*$

Аналогично, подвыражение  $(bc|cc|ac)$  всегда имеет вид  $xc$ , где  $x \in \{a, b, c\}$ , то есть  $(bc|cc|ac) \equiv [abc]c$

Таким образом, первая форма блока становится  $(a[abc])^*[abc]cbb$

Вторая форма блока имеет вид  $abacc^*$ . Поскольку после обязательного символа  $c$  следует  $c^*$ , общее число букв  $c$  в конце не менее одного, следовательно,  $abacc^* \equiv abac^+$ .

Объединяя обе формы, а также сохраняя внешнюю звезду, получаем эквивалентное расширенное регулярное выражение:

$$^((a[abc])^*[abc]cbb|abac^+)^*\$.$$

## 6 Фазз-тестирование

Для проверки эквивалентности построенных распознавателей было написано автоматическое тестирование.

Программа генерирует случайные слова над алфавитом  $a, b, c$  как полностью случайные, так и случайные, которые соответствуют регулярному выражению. Для каждого слова выполняется проверка принадлежности языку с использованием четырёх методов: регулярного выражения, минимального ДКА, построенного НКА, переключающегося конечного автомата.

Код программы представлен в `test.cpp`.