

# Лабораторная работа №1

Лиганкина Анна

Вариант 20

11 октября 2025 г.

## Содержание

<b>1</b>	<b>Задание</b>	<b>2</b>
<b>2</b>	<b>Исследование SRS</b>	<b>3</b>
2.1	Завершимость . . . . .	3
2.2	Классы эквивалентности . . . . .	3
2.3	Локальная конfluenceнтность . . . . .	3
2.4	Пополняемость по Кнуту-Бендиксу . . . . .	4
2.5	Построение SRS $\tilde{T}'$ . . . . .	5
<b>3</b>	<b>Тестирование</b>	<b>7</b>
3.1	Фазз-тестирование эквивалентности . . . . .	7
3.2	Метаморфное тестирование . . . . .	9

# 1 Задание

Дана SRS:

$cb \rightarrow ba$	$bba \rightarrow ba$	$cac \rightarrow cc$	$baca \rightarrow cabba$
$aaa \rightarrow aa$	$bbb \rightarrow b$	$bab \rightarrow cac$	$caab \rightarrow bb$
$aba \rightarrow ba$	$bbc \rightarrow c$	$ccc \rightarrow c$	$caac \rightarrow bc$
$ac \rightarrow cc$	$bcc \rightarrow cc$	$babb \rightarrow ba$	$aabcaa \rightarrow a$
$baa \rightarrow ba$	$ba \rightarrow cab$	$babc \rightarrow \varepsilon$	

По имеющейся SRS определить:

- завершимость;
- конечность классов эквивалентности по НФ (для построения эквивалентностей считаем, что правила могут применяться в обе стороны). Если их конечное число, то построить минимальную систему переписывания, им соответствующую;
- локальную конfluenceнтность и пополняемость по Кнуту-Бендиксу.

По SRS  $\mathcal{T}$  тем самым (исключая случай, когда она сразу локально конfluenceнтна или конечна и минимальна) строится другая SRS  $\mathcal{T}'$ , которая должна сохранять те же классы эквивалентности. Если исходная SRS завершима, то правила в  $\mathcal{T}'$  должны удовлетворять условию убывания левой части относительно правой по выбранному фундированному порядку  $\preceq$ .

Провести автоматическое тестирование предполагаемой эквивалентности указанных SRS.

**Фазз-тестирование эквивалентности:** строится случайное слово  $\omega$  и случайная цепочка переписываний его в  $\omega'$  по  $\mathcal{T}$ . Проверить, можно ли получить  $\omega'$  из  $\omega$  (или наоборот) в рамках правил  $\mathcal{T}'$ .

**Метаморфное тестирование:** выбрать инварианты, которые должны сохраняться (либо монотонно изменяться) при переписывании в рамках  $\mathcal{T}$ . Породить случайную цепочку переписываний над случайным словом в  $\mathcal{T}'$  и проверить выполнимость инвариантов. Как минимум два разных инварианта.

## 2 Исследование SRS

### 2.1 Завершимость

Исходная SRS  $\mathcal{T}$  незавершима, т.к. существует цикл:

$$caba \xrightarrow{aba \rightarrow ba} cba \xrightarrow{cb \rightarrow ba} baa \xrightarrow{ba \rightarrow cab} caba$$

### 2.2 Классы эквивалентности

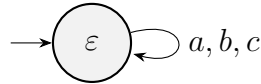
Нормальных форм в SRS  $\mathcal{T}$  бесконечное число. Для доказательства можно привести пример строки, состоящей лишь из подряд идущих подстрок  $abc$ , которые ни во что не редуцируются.

Рассмотрим, лежат ли все НФ длины 1 и пустая строка в одном классе эквивалентности. Т.к. стрелки рассматриваются в обе стороны, имеем такие цепочки:

$$\begin{aligned} \varepsilon &\xleftrightarrow{abc \leftrightarrow \varepsilon} abc \xleftrightarrow{aba \leftrightarrow ba} ababc \xleftrightarrow{abc \leftrightarrow \varepsilon} a \\ \varepsilon &\xleftrightarrow{abc \leftrightarrow \varepsilon} abc \xleftrightarrow{bba \leftrightarrow ba} bbabc \xleftrightarrow{abc \leftrightarrow \varepsilon} b \\ \varepsilon &\xleftrightarrow{abc \leftrightarrow \varepsilon} abc \xleftrightarrow{bab \leftrightarrow cac} cacc \xleftrightarrow{cac \leftrightarrow cc} ccc \xleftrightarrow{ccc \leftrightarrow c} c \end{aligned}$$

Т.е. НФ  $a, b, c$  лежат в одном классе эквивалентности  $\varepsilon$ . Т.к. алфавит состоит из этих букв, то все слова принадлежат одному классу эквивалентности ' $\varepsilon$ '.

Т.к. число классов эквивалентности конечно, то построим автомат по данной SRS:



Минимальной SRS, описывающей этот автомат, будет  $a \rightarrow \varepsilon$ ,  $b \rightarrow \varepsilon$ ,  $c \rightarrow \varepsilon$ .

### 2.3 Локальная конфлюэнтность

SRS  $\mathcal{T}$  не является локально конфлюэнтной, т.к. для слова  $bab$  в один шаг преобразований может быть переписана в слова  $cacc$  (по правилу  $bab \rightarrow cac$ ) и к слову  $\varepsilon$ , являющемуся НФ. При этом слово  $cacc$  невозможно привести к  $\varepsilon$ .

Таким образом, система  $\mathcal{T}$  не является локально конфлюэнтной.

## 2.4 Пополняемость по Кнуту-Бендиксу

Для применения алгоритма Кнута-Бендикса введем фундированный порядок — length-lexicographic order  $\prec$  — и, в соответствии с ним, переориентируем правила (что сохранит классы эквивалентности исходной SRS  $\mathcal{T}$ ).

Теперь наша SRS выглядит следующим образом:

$$\begin{array}{llll}
 aaa \rightarrow aa & abc \rightarrow \varepsilon & caab \rightarrow bb & cac \rightarrow bab \\
 aabcaa \rightarrow a & bba \rightarrow ba & caac \rightarrow bc & cb \rightarrow ba \\
 aba \rightarrow ba & bbb \rightarrow b & cab \rightarrow ba & cc \rightarrow ac \\
 baa \rightarrow ba & bbc \rightarrow c & cabba \rightarrow baca & ccc \rightarrow c \\
 babb \rightarrow ba & bcc \rightarrow cc & cac \rightarrow cc & 
 \end{array}$$

Найдем множество всех критических пар:

1. Правила  $aaa \rightarrow aa$  и  $aabcaa \rightarrow a$  порождают критическую пару  $\langle aabcaa, aa \rangle$  из слов  $aaabcaa$  и  $aabcaaa$ . НФ для этой пары равны соответственно  $\langle a, aa \rangle$ . Поэтому, т.к.  $a \prec aa$ , введем правило  $aa \rightarrow a$ . Теперь можно редуцировать правила  $aaa \rightarrow a$  и  $baa \rightarrow ba$ ;
2. Правила  $aa \rightarrow a$  и  $aabcaa \rightarrow a$  порождают критические пары  $\langle abcaa, a \rangle$  и  $\langle aabca, a \rangle$  из слова  $aabcaa$ . НФ для этой пары равны соответственно  $\langle abca, a \rangle$  и  $\langle abca, a \rangle$ . Поэтому, т.к.  $a \prec abca$ , введем правило  $abca \rightarrow a$ . Теперь можно редуцировать правило  $aabcaa \rightarrow a$ ;
3. Правила  $aa \rightarrow a$  и  $caab \rightarrow bb$  порождают критическую пару  $\langle cab, bb \rangle$  из слова  $caab$ . НФ для этой пары равны соответственно  $\langle ba, bb \rangle$ . Поэтому, т.к.  $ba \prec bb$ , введем правило  $bb \rightarrow ba$ . Теперь можно редуцировать правила  $bba \rightarrow ba$  (т.к.  $bba \xrightarrow{bb \rightarrow ba} baa \xrightarrow{aa \rightarrow a} ba$ ) и  $babb \rightarrow ba$  (т.к.  $babb \xrightarrow{bb \rightarrow ba} baba \xrightarrow{aba \rightarrow ba} bba \xrightarrow{\text{переход выше}} ba$ );
4. Т.к. существуют 2 различных правила переписывания строки  $cac$  рассмотрим критическую пару  $\langle cc, bab \rangle$ . НФ для этой пары равны  $ac$  и  $bab$ . Поэтому, т.к.  $cc \prec bab$ , введем правило  $bab \rightarrow ac$ . Теперь можно редуцировать правило  $cac \rightarrow bab$ ;
5. Правила  $bab \rightarrow ac$  и  $abc \rightarrow \varepsilon$  порождают критическую пару  $\langle acc, \varepsilon \rangle$  из слова  $babc$ . НФ для этой пары равны соответственно  $\langle ac, \varepsilon \rangle$ . Поэтому, т.к.  $\varepsilon \prec ac$ , введем правило  $ac \rightarrow \varepsilon$ . Теперь можно редуцировать правило  $babc \rightarrow \varepsilon$ , т.к. можно переписать  $babc \xrightarrow{bab \rightarrow ac} acc \xrightarrow{cc \rightarrow ac} aac \xrightarrow{aa \rightarrow a} ac \rightarrow \varepsilon$ ;

Промежуточная SRS:

$aa \rightarrow a$	$bb \rightarrow ba$	$caac \rightarrow bc$	$cc \rightarrow ac$
$ac \rightarrow \varepsilon$	$bbb \rightarrow b$	$cab \rightarrow ba$	$ccc \rightarrow c$
$abca \rightarrow a$	$bbc \rightarrow c$	$cabba \rightarrow bac$	
$aba \rightarrow ba$	$bcc \rightarrow cc$	$cac \rightarrow cc$	
$bab \rightarrow ac$	$caab \rightarrow bb$	$cb \rightarrow ba$	

6. Правила  $aa \rightarrow a$  и  $ac \rightarrow \varepsilon$  порождают критическую пару  $\langle ac, a \rangle$  из слова  $aac$ . НФ для этой пары равны соответственно  $\langle \varepsilon, a \rangle$ . Поэтому, т.к.  $\varepsilon \prec a$ , введем правило  $a \rightarrow \varepsilon$ . Теперь можно редуцировать правила  $aa \rightarrow a$ ,  $aba \rightarrow ba$ ,  $caab \rightarrow bb$  и заменить  $bb \rightarrow ba$  и  $bbb \rightarrow b$  на  $bb \rightarrow b$ ;
7. Правила  $a \rightarrow \varepsilon$  и  $ac \rightarrow \varepsilon$  порождают критическую пару  $\langle c, \varepsilon \rangle$  из слова  $ac$ . Добавляем правило  $c \rightarrow \varepsilon$  и редуцируем правила  $ac \rightarrow \varepsilon$ ,  $cab \rightarrow ba$ ,  $cabba \rightarrow bac$ ,  $cac \rightarrow cc$ ,  $cb \rightarrow ba$ ,  $cc \rightarrow ac$ ,  $ccc \rightarrow c$
8. Правила  $bcc \rightarrow cc$  и  $c \rightarrow \varepsilon$  порождают критическую пару  $\langle cc, bc \rangle$  из слова  $bcc$ . НФ для этой пары равны соответственно  $\langle \varepsilon, b \rangle$ , поэтому добавим правило  $b \rightarrow \varepsilon$ . Теперь можно редуцировать все правила, кроме  $a \rightarrow \varepsilon$ ,  $b \rightarrow \varepsilon$  и  $c \rightarrow \varepsilon$ .

Таким образом, пополненная SRS имеет вид:

$$\begin{aligned} a &\rightarrow \varepsilon \\ b &\rightarrow \varepsilon \\ c &\rightarrow \varepsilon \end{aligned}$$

## 2.5 Построение SRS $\tilde{\mathcal{T}}'$

На основе исследования SRS  $\mathcal{T}$ , эквивалентная ей SRS  $\mathcal{T}'$ :

$$\begin{aligned} a &\rightarrow \varepsilon \\ b &\rightarrow \varepsilon \\ c &\rightarrow \varepsilon \end{aligned}$$

т.к. она сохраняет те же классы эквивалентности.

Т.к. в исходной SRS  $\mathcal{T}$  один класс эквивалентности, то задача фазз-тестирования и метаморфного тестирования тривиальна. Поэтому необходимо выкинуть одно или несколько правил, которые сводят любые два слова в один класс. Рассмотрим такую систему переписываний  $\tilde{\mathcal{T}}$ :

$cb \rightarrow ba$	<del><math>bba \rightarrow ba</math></del>	$cac \rightarrow cc$	<del><math>baca \rightarrow cabba</math></del>
$aaa \rightarrow aa$	$bbb \rightarrow b$	$bab \rightarrow cac$	<del><math>caab \rightarrow bb</math></del>
$aba \rightarrow ba$	$bbc \rightarrow c$	$ccc \rightarrow c$	<del><math>caac \rightarrow bc</math></del>
$ac \rightarrow cc$	<del><math>bce \rightarrow cc</math></del>	$babb \rightarrow ba$	<del><math>aabeaa \rightarrow a</math></del>
$baa \rightarrow ba$	$ba \rightarrow cab$	$babc \rightarrow \varepsilon$	

В ней 2 класса эквивалентности: соответствующих  $\varepsilon$  и  $b$ .

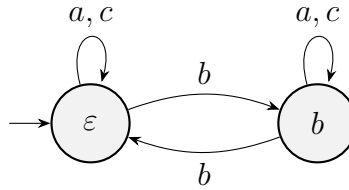
**Доказательство.**

Возьмем правило  $babc \rightarrow \varepsilon$ . Тогда слово  $babc$  лежит в одном классе с  $\varepsilon$  ( $babc \equiv \varepsilon$ ). Т.к.  $bab \rightarrow cac$ , то  $cacc \equiv \varepsilon$ . Далее,  $cac \rightarrow cc$ , поэтому  $ccc \equiv \varepsilon$ . А тогда  $c \equiv \varepsilon$  (т.к.  $ccc \rightarrow c$ ).

Из правил  $ac \equiv cc$  и  $c \equiv \varepsilon$  следует, что  $a \equiv \varepsilon$ . А из  $babc \equiv \varepsilon$ ,  $a \equiv \varepsilon$  и  $c \equiv \varepsilon$  следует, что  $bb \equiv \varepsilon$ .

Т.к. ни в одном правиле четность вхождений  $b$  не меняется, то  $b$  является отдельным классом эквивалентности, в который входят все слова с нечетным количеством вхождений  $b$ .

Минимальный автомат, соответствующий этой SRS:



Минимальная SRS  $\tilde{T}'$  по этому автомату:

$$\begin{aligned}
 a &\rightarrow \varepsilon \\
 bb &\rightarrow \varepsilon \\
 c &\rightarrow \varepsilon
 \end{aligned}$$

## 3 Тестирование

### 3.1 Фазз-тестирование эквивалентности

Фазз-тестирование проводится следующим образом:

1. генерируется случайная строка  $\omega$ ;
2. строится случайная цепочка по правилам SRS  $\tilde{T}$  в  $\omega'$ ;
3. выбираем наименьшее из  $\omega$  и  $\omega'$ , согласно выбранному фундированному порядку при построении  $T'$ ;
4. смотрим множество слов, которые можно получить из меньшего слова в SRS  $\tilde{T}'$ ;
5. проверяем, можно ли из большего слова получить элемент ранее найденного множества по SRS  $\tilde{T}'$  — если да, то  $\omega$  и  $\omega'$  эквивалентны.

Исходный код программы представлен в файле «fuzzer.hs».

Ниже представлен псевдокод самых важных функций:

---

```
1 findAllOccurrences(pattern, text):
2   positions  $\leftarrow$  пустой список
3   for  $i \leftarrow 0$  to  $|text| - |pattern|$  do
4       if подстрока  $text[i..]$  начинается с  $pattern$  then
5           добавить  $i$  в positions
6   return positions

7 replaceSubstring(idx, len, replacement, str):
8   before  $\leftarrow str[0 : idx]$ 
9   after  $\leftarrow str[idx + len :]$ 
10  return before + replacement + after

11 getNextStates(str, rules):
12  result  $\leftarrow \emptyset$ 
13  foreach  $(lhs, rhs) \in srs$  do
14      indices  $\leftarrow$  findAllOccurrences(lhs, str)
15      foreach  $idx \in indices$  do
16          newStr  $\leftarrow$  replaceSubstring(idx, |lhs|, rhs, str)
17          добавить newStr в result
18  return result
```

---

---

```

1 findAllReachable(startStr, rules):
2   visited  $\leftarrow \{startStr\}$ 
3   queue  $\leftarrow$  очередь со startStr
4   while queue  $\neq \emptyset$  do
5     current  $\leftarrow$  queue.Dequeue()
6     newStrings  $\leftarrow$  getNextStates(current, rules)
7     unvisited  $\leftarrow$  newStrings  $\setminus$  visited
8     visited  $\leftarrow$  visited  $\cup$  unvisited
9     queue.Enqueue(unvisited)
10  return visited

11 checkIntersection(startStr, rules, targetSet):
12  if startStr  $\in$  targetSet then
13    return True
14  visited  $\leftarrow \{startStr\}$ 
15  queue  $\leftarrow$  очередь со startStr
16  while queue  $\neq \emptyset$  do
17    current  $\leftarrow$  queue.Dequeue()
18    newStrings  $\leftarrow$  getNextStates(current, rules)
19    if  $\exists s \in newStrings$ , такое что  $s \in targetSet$  then
20      return True
21    unvisited  $\leftarrow$  newStrings  $\setminus$  visited
22    visited  $\leftarrow$  visited  $\cup$  unvisited
23    queue.Enqueue(unvisited)
24  return False

25 main():
26   $\omega \leftarrow$  generateRandomString(10, 20)
27   $\omega' \leftarrow$  randomlyTransform( $\omega$ )
28  if  $\omega \prec \omega'$  then
29    shorterStr  $\leftarrow \omega$ ; longerStr  $\leftarrow \omega'$ 
30  else
31    shorterStr  $\leftarrow \omega'$ ; longerStr  $\leftarrow \omega$ 
32  reachableFromShorter  $\leftarrow$  findAllReachable(shorterStr, srsT')
33  areEquivalent  $\leftarrow$  checkIntersection(longerStr, srsT',
    reachableFromShorter)
34  вывести areEquivalent

```

---



## 3.2 Метаморфное тестирование