

Лабораторная работа №1

Лиганкина Анна
Вариант 20

11 октября 2025 г.

Содержание

1 Задание	2
2 Исследование SRS	3
2.1 Завершимость	3
2.2 Классы эквивалентности	3
2.3 Локальная конфлюэнтность	3
2.4 Пополняемость по Кнуту-Бендикусу	4
2.5 Построение SRS \tilde{T}'	5
3 Тестирование	7
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 конфлюэнтность и пополняемость по Кнуту-Бендикусу.

По SRS \mathcal{T} тем самым (исключая случай, когда она сразу локально конфлюэнтна 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 ω и случайная цепочка переписываний его в ω' по \mathcal{T} . Проверить, можно ли получить ω' из ω (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 случайнм словом в \mathcal{T}' и проверить выполнимость инвариантов. Как минимум два разных инварианта.

2 Исследование SRS

2.1 Завершаемость

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

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

2.2 Классы эквивалентности

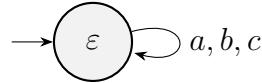
Нормальных форм в SRS \mathcal{T} бесконечное число. Для доказательства можно привести пример строки, состоящей лишь из подряд идущих подстрок abc , которые ни во что не редуцируются.

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

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

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

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 $babc$ в один шаг преобразований может быть переписана в слова $cacc$ (по правилу $bab \rightarrow cac$) и к слову ε , являющемуся НФ. При этом слово $cacc$ невозможно привести к ε .

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

2.4 Пополняемость по Кнуту-Бендику

Для применения алгоритма Кнута-Бендикуса 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:

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

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 $\langle abca, a \rangle$ 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 $babc \rightarrow \varepsilon$ 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:

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

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 $ac \rightarrow \varepsilon$, $cab \rightarrow ba$, $cabba \rightarrow baca$, $cac \rightarrow cc$, $cb \rightarrow ba$, $cc \rightarrow ac$, $ccc \rightarrow c$
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:

$$\begin{array}{l}
 a \rightarrow \varepsilon \\
 b \rightarrow \varepsilon \\
 c \rightarrow \varepsilon
 \end{array}$$

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

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

$$\begin{array}{l}
 a \rightarrow \varepsilon \\
 b \rightarrow \varepsilon \\
 c \rightarrow \varepsilon
 \end{array}$$

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 $\tilde{\mathcal{T}}$:

$cb \rightarrow ba$	$bba \Rightarrow \overline{ba}$	$cac \rightarrow cc$	$\overline{bac} \rightarrow \overline{cabba}$
$aaa \rightarrow aa$	$bbb \rightarrow b$	$bab \rightarrow cac$	$\overline{caab} \Rightarrow \overline{bb}$
$aba \rightarrow ba$	$bbc \rightarrow c$	$ccc \rightarrow c$	$\overline{caae} \Rightarrow \overline{bc}$
$ac \rightarrow cc$	$bce \Rightarrow \overline{cc}$	$babb \rightarrow ba$	$\overline{aab} \Rightarrow \overline{aa}$
$baa \rightarrow ba$	$ba \rightarrow cab$	$babc \rightarrow \varepsilon$	

В ней 2 класса эквивалентности: соответствующих ε и b .

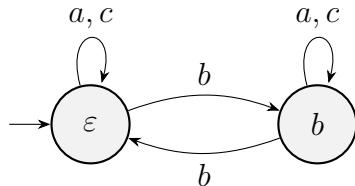
Доказательство.

Возьмем правило $babc \rightarrow \varepsilon$. Тогда слово $babc$ лежит в одном классе с ε ($babc \equiv \varepsilon$). 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 ω ;
2. строится случайная цепочка по правилам SRS $\tilde{\mathcal{T}}$ в ω' ;
3. выбираем наименьшее из ω и ω' , согласно выбранному фундированному порядку при построении \mathcal{T}' ;
4. смотрим множество слов, которые можно получить из меньшего слова в SRS $\tilde{\mathcal{T}}'$;
5. проверяем, можно ли из большего слова получить элемент ранее найденного множества по SRS $\tilde{\mathcal{T}}'$ — если да, то ω и ω' эквивалентны.

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

Ниже представлен псевдокод самых важных функций:

```
1  findAllOccurrences(pattern, text):
2      positions ← пустой список
3      for i ← 0 to |text| - |pattern| do
4          if подстрока text[i..] начинается с pattern then
5              добавить i в positions
6      return positions
7
7  replaceSubstring(idx, len, replacement, str):
8      before ← str[0 : idx]
9      after ← str[idx + len :]
10     return before + replacement + after
11
11 getNestStates(str, rules):
12     result ← ∅
13     foreach (lhs, rhs) ∈ srs do
14         indices ← findAllOccurrences(lhs, str)
15         foreach idx ∈ indices do
16             newStr ← replaceSubstring(idx, |lhs|, rhs, str)
17             добавить newStr в result
18     return result
```

```

1  findAllReachable(startStr, rules):
2      visited ← {startStr}
3      queue ← очередь со startStr
4      while queue ≠ ∅ do
5          current ← queue.Dequeue()
6          newStrings ← getNextStates(current, rules)
7          unvisited ← newStrings \ visited
8          visited ← visited ∪ unvisited
9          queue.Enqueue(unvisited)
10     return visited

11 checkIntersection(startStr, rules, targetSet):
12     if startStr ∈ targetSet then
13         return True
14     visited ← {startStr}
15     queue ← очередь со startStr
16     while queue ≠ ∅ do
17         current ← queue.Dequeue()
18         newStrings ← getNextStates(current, rules)
19         if ∃s ∈ newStrings, такое что s ∈ targetSet then
20             return True
21         unvisited ← newStrings \ visited
22         visited ← visited ∪ unvisited
23         queue.Enqueue(unvisited)
24     return False

25 main():
26     w ← generateRandomString(10, 20)
27     w' ← randomlyTransform(w)
28     if w ≺ w' then
29         shorterStr ← w; longerStr ← w'
30     else
31         shorterStr ← w'; longerStr ← w
32     reachableFromShorter ← findAllReachable(shorterStr, srsT')
33     areEquivalent ← checkIntersection(longerStr, srsT',
34                                         reachableFromShorter)
35     вывести areEquivalent

```

3.2 Метаморфное тестирование