

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)



ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №3

по дисциплине: Теория автоматов и формальных языков
тема: «Регулярные языки и конечные распознаватели»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
ст. пр. Рязанов Юрий Дмитриевич

Белгород 2024 г.

Лабораторная работа №3
Регулярные языки и конечные распознаватели
Вариант 8

Цель работы: изучить основные способы задания регулярных языков, способы построения, алгоритмы преобразования, анализа и реализации конечных распознавателей.

1. Язык L_1 в алфавите $\{0, 1\}$, представляющий собой множество цепочек, в которых на предпоследнем месте стоит единица, задан грамматикой:

$S \rightarrow A10$

$S \rightarrow A11$

$A \rightarrow 0A$

$A \rightarrow 1A$

$A \rightarrow \varepsilon$

Построить детерминированный конечный распознаватель языка L_1 .

Преобразуем заданную грамматику к автоматной правосторонней. Сейчас она является КС-грамматикой.

Приведём грамматику и устраним левую рекурсию.

Лишних символов в грамматике нет.

В грамматике есть ε -правило. Исключим его.

$S \rightarrow A10$

$S \rightarrow 10$

$S \rightarrow A11$

$S \rightarrow 11$

$A \rightarrow 0A$

$A \rightarrow 0$

$A \rightarrow 1A$

$A \rightarrow 1$

Цепных правил в грамматике нет.

Левой рекурсии в грамматике также нет.

Грамматика приведена, а также в ней нет левой рекурсии.

Преобразуем грамматику к такому виду, что каждое правило будет начинаться с терминала:

$S \rightarrow 0A10$

$S \rightarrow 1A10$

$S \rightarrow 010$

$S \rightarrow 110$

$S \rightarrow 0A11$

$S \rightarrow 1A11$

$S \rightarrow 011$

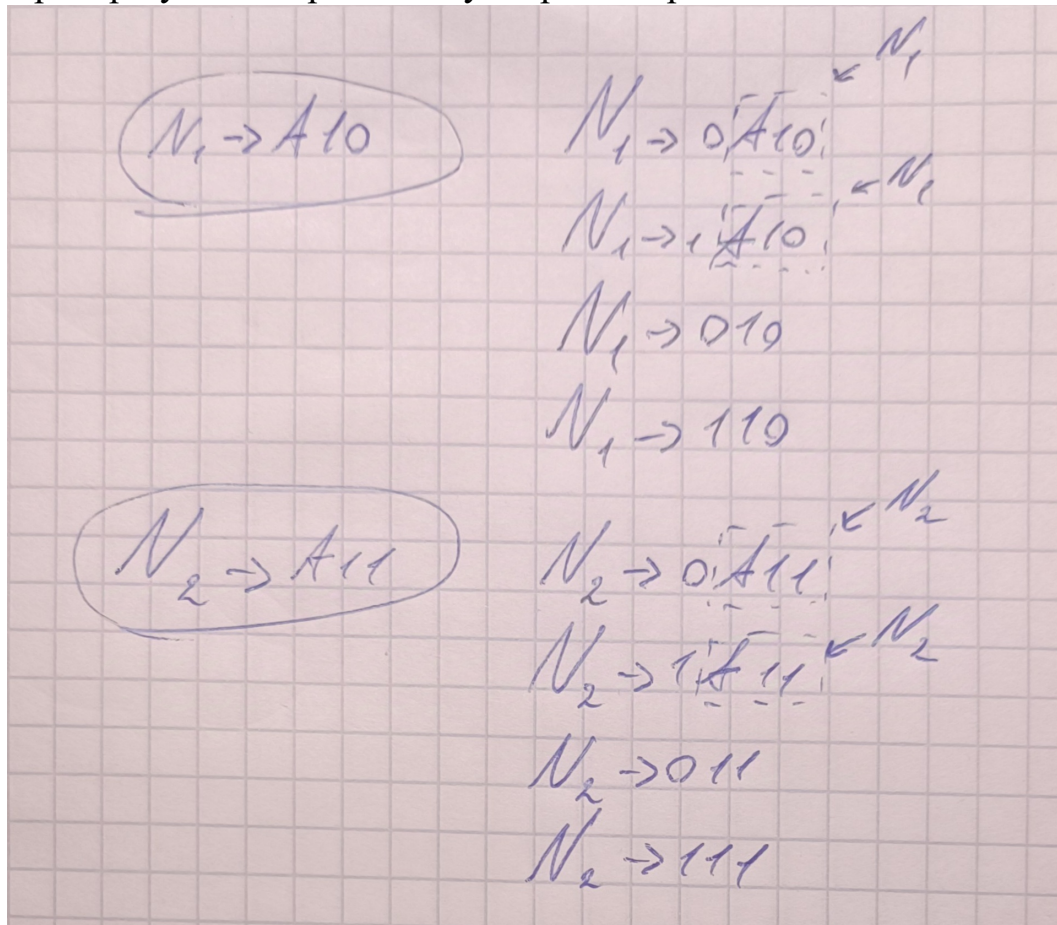
$S \rightarrow 111$

$A \rightarrow 0A$

$A \rightarrow 1A$

$A \rightarrow 0$
 $A \rightarrow 1$
 $S \rightarrow 10$
 $S \rightarrow 11$

Преобразуем КС-грамматику к правосторонней:



$S \rightarrow 0N_1$
 $S \rightarrow 1N_1$
 $S \rightarrow 010$
 $S \rightarrow 110$
 $S \rightarrow 0N_2$
 $S \rightarrow 1N_2$
 $S \rightarrow 011$
 $S \rightarrow 111$
 $A \rightarrow 0A$
 $A \rightarrow 1A$
 $A \rightarrow 0$
 $A \rightarrow 1$
 $S \rightarrow 10$
 $S \rightarrow 11$
 $N_1 \rightarrow 0N_1$
 $N_1 \rightarrow 1N_1$
 $N_1 \rightarrow 010$
 $N_1 \rightarrow 110$
 $N_2 \rightarrow 0N_2$

$$N_2 \rightarrow 1N_2$$

$$N_2 \rightarrow 011$$

$$N_2 \rightarrow 111$$

Исключим лишние символы:

$$S \rightarrow 0N_1$$

$$S \rightarrow 1N_1$$

$$S \rightarrow 010$$

$$S \rightarrow 110$$

$$S \rightarrow 0N_2$$

$$S \rightarrow 1N_2$$

$$S \rightarrow 011$$

$$S \rightarrow 111$$

$$S \rightarrow 10$$

$$S \rightarrow 11$$

$$N_1 \rightarrow 0N_1$$

$$N_1 \rightarrow 1N_1$$

$$N_1 \rightarrow 010$$

$$N_1 \rightarrow 110$$

$$N_2 \rightarrow 0N_2$$

$$N_2 \rightarrow 1N_2$$

$$N_2 \rightarrow 011$$

$$N_2 \rightarrow 111$$

Получили правостороннюю грамматику. Теперь преобразуем её к автоматной правосторонней грамматике. Введём нетерминалы: $N_3 \rightarrow 1$, $N_4 \rightarrow 0$, $N_5 \rightarrow 1N_3$, $N_6 \rightarrow 1N_4$, $N_7 \rightarrow \varepsilon$ и выполним замену там, где это требуется:

$$S \rightarrow 0N_1$$

$$S \rightarrow 1N_1$$

$$S \rightarrow 0N_6$$

$$S \rightarrow 1N_6$$

$$S \rightarrow 0N_2$$

$$S \rightarrow 1N_2$$

$$S \rightarrow 0N_5$$

$$S \rightarrow 1N_5$$

$$S \rightarrow 1N_4$$

$$S \rightarrow 1N_3$$

$$N_1 \rightarrow 0N_1$$

$$N_1 \rightarrow 1N_1$$

$$N_1 \rightarrow 0N_6$$

$$N_1 \rightarrow 1N_6$$

$$N_2 \rightarrow 0N_2$$

$$N_2 \rightarrow 1N_2$$

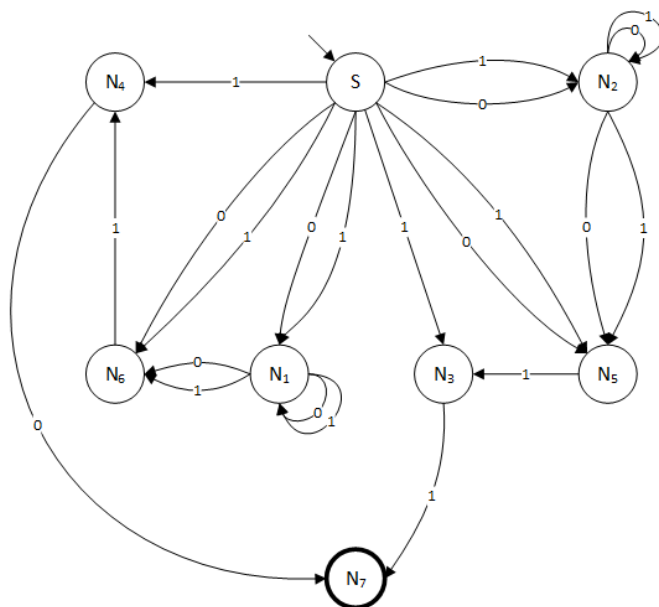
$$N_2 \rightarrow 0N_5$$

$N_2 \rightarrow 1N_5$
 $N_3 \rightarrow 1N_7$
 $N_4 \rightarrow 0N_7$
 $N_5 \rightarrow 1N_3$
 $N_6 \rightarrow 1N_4$
 $N_7 \rightarrow \varepsilon$

Теперь можем построить распознаватель по КС-грамматике:

| | \downarrow | | | | | | | 1 |
|---|--------------------------------|------------|------------|-------|-------|-------|-------|-------|
| | S | N_1 | N_2 | N_3 | N_4 | N_5 | N_6 | N_7 |
| 1 | $N_1, N_2, N_3, N_4, N_5, N_6$ | N_1, N_6 | N_2, N_5 | N_7 | | N_3 | N_4 | |
| 0 | N_1, N_2, N_5, N_6 | N_1, N_6 | N_2, N_5 | | N_7 | | | |

Получение
недетерминированного
конечного распознавателя:

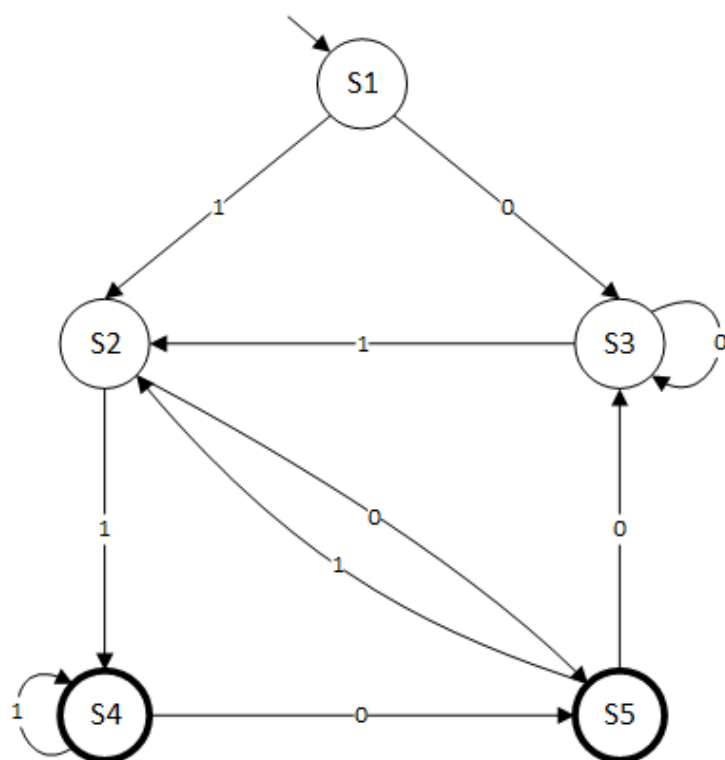


Расознаватель недетерминированный, преобразуем его к детерминированному.
 ε -переходов в распознавателе нет.
 Преобразуем недетерминированный конечный распознаватель в детерминированный:

| | | | | | |
|---|------------------------------------|---|------------------------------------|---|------------------------------------|
| | | | | | |
| | $\{S\}$ | $\{N_1, N_2, N_3, N_4, N_5, N_6\}$ | $\{N_1, N_2, N_5, N_6\}$ | $\{N_1, N_2, N_3, N_4, N_5, N_6, N_7\}$ | $\{N_1, N_2, N_5, N_6, N_7\}$ |
| 1 | $\{N_1, N_2, N_3, N_4, N_5, N_6\}$ | $\{N_1, N_2, N_3, N_4, N_5, N_6, N_7\}$ | $\{N_1, N_2, N_3, N_4, N_5, N_6\}$ | $\{N_1, N_2, N_3, N_4, N_5, N_6, N_7\}$ | $\{N_1, N_2, N_3, N_4, N_5, N_6\}$ |
| 0 | $\{N_1, N_2, N_5, N_6\}$ | $\{N_1, N_2, N_5, N_6, N_7\}$ | $\{N_1, N_2, N_5, N_6\}$ | $\{N_1, N_2, N_5, N_6, N_7\}$ | $\{N_1, N_2, N_5, N_6\}$ |

| | | | | | |
|---|----|----|----|----|----|
| | ↓ | | | 1 | 1 |
| | S1 | S2 | S3 | S4 | S5 |
| 1 | S2 | S4 | S2 | S4 | S2 |
| 0 | S3 | S5 | S3 | S5 | S3 |

Переход к
детерминированному
распознавателю



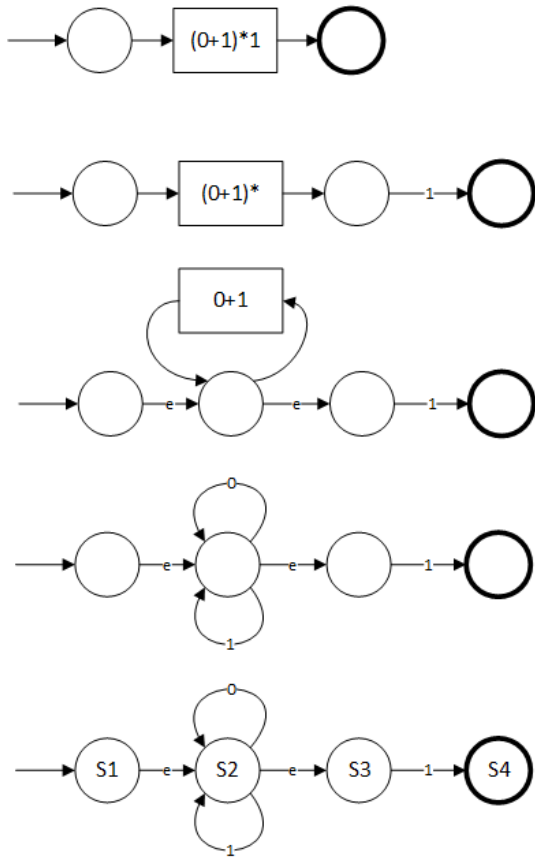
Построили детерминированный конечный распознаватель языка L_1 .

2. Язык L_2 в алфавите $\{0, 1\}$, представляющий собой множество цепочек, в которых на последнем месте стоит единица, задан регулярным выражением:
 $(0+1)^*1$

Построить детерминированный конечный распознаватель языка L_2 .

Для начала построим конечный недетерминированный распознаватель языка:

Получение
недетерминированного
конечного распознавателя:



Данный распознаватель языка не является детерминированным, так как он содержит ϵ -переходы. Преобразуем данный конечный распознаватель языка в детерминированный:

| | \downarrow | | | 1 |
|------------|--------------|----|----|----|
| | S1 | S2 | S3 | S4 |
| 1 | | S2 | S4 | |
| 0 | | S2 | | |
| ϵ | S2 | S3 | | |

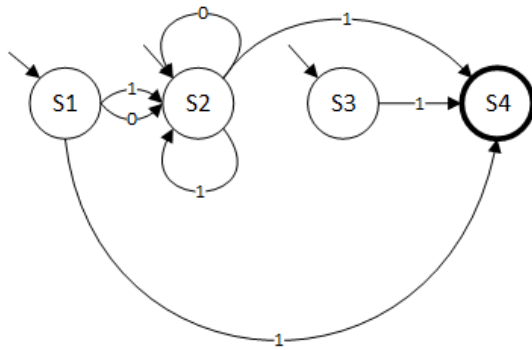
Удалим ϵ -переходы:

ϵ -замыкания: $\epsilon(S1) = \{S1, S2, S3\}$, $\epsilon(S2) = \{S2, S3\}$, $\epsilon(S3) = \{S3\}$, $\epsilon(S4) = \{S4\}$

| | \downarrow | \downarrow | \downarrow | 1 |
|---|------------------------------------|--------------------------------|----------------------------|----------------------------|
| | $\epsilon(S1)$ $\{S1, S2, S3\}$ | $\epsilon(S2)$ $\{S2, S3\}$ | $\epsilon(S3)$ $\{S3\}$ | $\epsilon(S4)$ $\{S4\}$ |
| 1 | $\epsilon(S2), \epsilon(S4)$ | $\epsilon(S2), \epsilon(S4)$ | $\epsilon(S4)$ | |
| 0 | $\epsilon(S2)$ | $\epsilon(S2)$ | | |

| | \downarrow | \downarrow | \downarrow | 1 |
|---|--------------|--------------|--------------|----|
| | S1 | S2 | S3 | S4 |
| 1 | S2, S4 | S2, S4 | S4 | |
| 0 | S2 | S2 | | |

Устранение ϵ -переходов



Преобразуем недетерминированный конечный распознаватель в детерминированный:

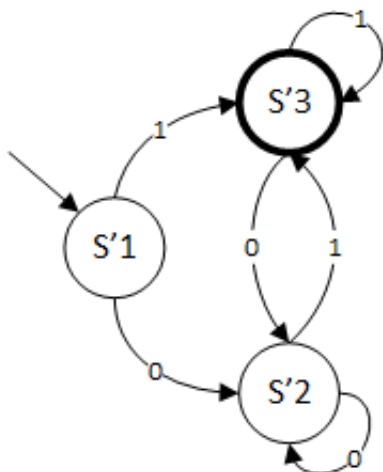
| | $\{S1, S2, S3\}$ | $\{S2\}$ | $\{S2, S4\}$ |
|---|------------------|--------------|--------------|
| 1 | $\{S2, S4\}$ | $\{S2, S4\}$ | $\{S2, S4\}$ |
| 0 | $\{S2\}$ | $\{S2\}$ | $\{S2\}$ |

Обозначим множества состояний как $S'1, S'2, S'3...$

$S'1$ обозначим как начальное состояние, согласно алгоритму, а $S'3$ обозначим как допускающее состояние, так как множество $\{S2, S4\}$ включает в себя допускающее состояние $S4$.

| | \downarrow | | 1 |
|---|--------------|-------|-------|
| | $S'1$ | $S'2$ | $S'3$ |
| 1 | $S'3$ | $S'3$ | $S'3$ |
| 0 | $S'2$ | $S'2$ | $S'2$ |

Переход к
детерминированному
распознавателю



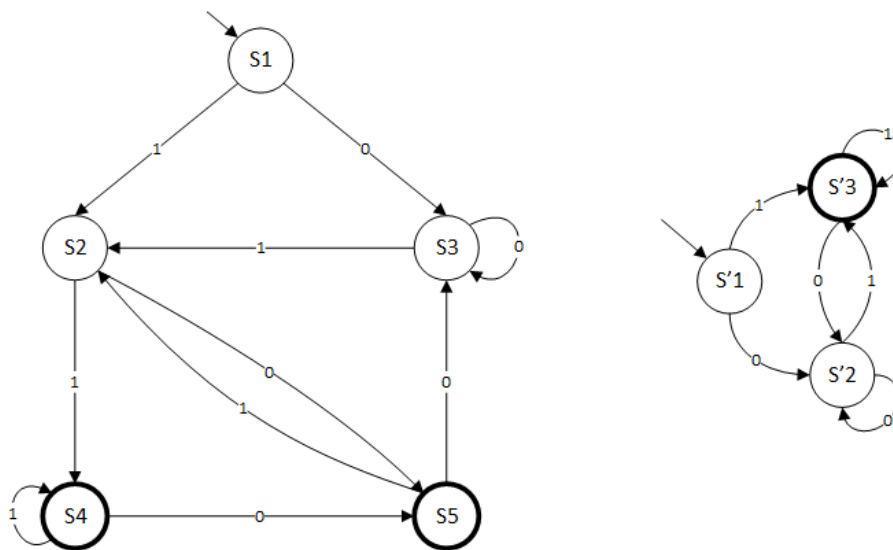
Построили детерминированный конечный распознаватель языка L_2 .

3. Построить минимальный детерминированный конечный распознаватель языка L_3 в алфавите $\{0,1\}$, представляющий собой множество цепочек, в которых хотя бы на одной из последних двух позиций стоит единица.

Можно отметить, что язык L_1 в алфавите $\{0, 1\}$ содержит цепочки, которые оканчиваются на 10 или 11. Язык не эквивалентен L_3 , так как он не учитывает цепочки, которые содержат только 1 или оканчиваются на 01. А язык L_2 в алфавите $\{0, 1\}$ содержит цепочки, которые содержат на последней позиции 1 в том числе и цепочку 1, а значит учитывает ещё и цепочки, оканчивающиеся на 01. Язык не эквивалентен L_3 , так как он не учитывает цепочки, которые заканчиваются на 10. Однако если объединим языки, можем получить язык, который содержат все необходимые из обоих языков цепочки, а значит и получим искомым язык L_3 .

| | | | | | | | | |
|---|----|----|----|----|----|-----|-----|-----|
| | ↓ | | | 1 | 1 | ↓ | | 1 |
| | S1 | S2 | S3 | S4 | S5 | S'1 | S'2 | S'3 |
| 1 | S2 | S4 | S2 | S4 | S2 | S'3 | S'3 | S'3 |
| 0 | S3 | S5 | S3 | S5 | S3 | S'2 | S'2 | S'2 |

Исходный
недетерминированный
распознаватель

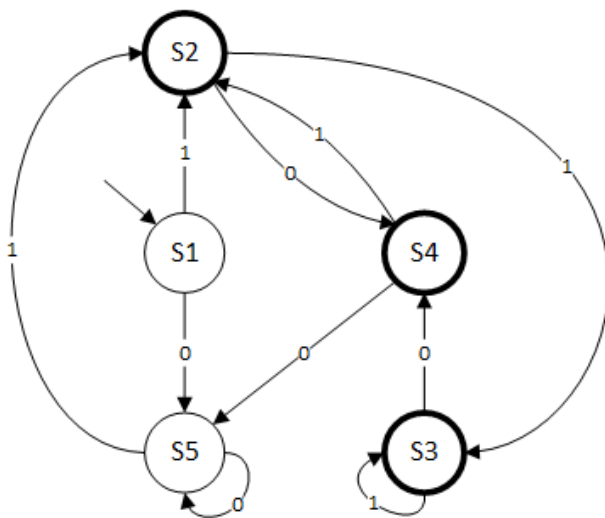


Преобразуем недетерминированный распознаватель в детерминированный:

| | | | | | |
|---|-----------|-----------|-----------|-----------|-----------|
| | | | | | |
| | {S1, S'1} | {S2, S'3} | {S4, S'3} | {S5, S'2} | {S3, S'2} |
| 1 | {S2, S'3} | {S4, S'3} | {S4, S'3} | {S2, S'3} | {S2, S'3} |
| 0 | {S3, S'2} | {S5, S'2} | {S5, S'2} | {S3, S'2} | {S3, S'2} |

| | | | | | |
|---|----|----|----|----|----|
| | ↓ | 1 | 1 | 1 | |
| | S1 | S2 | S3 | S4 | S5 |
| 1 | S2 | S3 | S3 | S2 | S2 |
| 0 | S5 | S4 | S4 | S5 | S5 |

Переход к
детерминированному
распознавателю



Полученный распознаватель является детерминированным, однако является ли он минимальным?

В распознавателе нет состояний, недостижимых из начального.

Перейдём к поиску и исключению эквивалентных состояний:

| | | | | | | |
|---|----|----|----|----|----|--|
| | ↓ | 1 | 1 | 1 | | |
| | S1 | S2 | S3 | S4 | S5 | |
| 1 | S2 | S3 | S3 | S2 | S2 | |
| 0 | S5 | S4 | S4 | S5 | S5 | |

Отвергающие состояния {S1, S5} объединим в класс K1 0-эквивалентных состояний, а допускающие состояния {S2, S3, S4} - в класс K2.

| | | | | | | |
|---|----|----|----|----|----|----|
| | K1 | | | K2 | | |
| | S1 | S5 | | S2 | S3 | S4 |
| 1 | K2 | K2 | K1 | K2 | K2 | K2 |
| 0 | K1 | K1 | K1 | K2 | K2 | K1 |

Получили таблицу переходов в классы 0-эквивалентных состояний. На основе этой таблицы можем построить таблицу переходов в классы 1-эквивалентных состояний.

| | | | | | | |
|---|----|----|----|----|----|----|
| | K1 | | K2 | K3 | | K4 |
| | S1 | S5 | | S2 | S3 | S4 |
| 1 | K3 | K3 | K2 | K3 | K3 | K3 |
| 0 | K1 | K1 | K2 | K4 | K4 | K1 |

По таблице видно, что классы 2-эквивалентных состояний совпадают с классами 1-эквивалентных состояний, следовательно, классы 1-эквивалентных состояний представляют собой классы эквивалентных состояний.

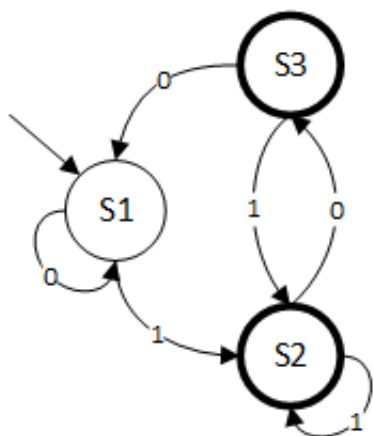
Таблица переходов минимального распознавателя:

| | | | | |
|---|----|----|----|----|
| | ↓ | | 1 | 1 |
| | K1 | K2 | K3 | K4 |
| 1 | K3 | K2 | K3 | K3 |
| 0 | K1 | K2 | K4 | K1 |

Переобозначим K1 как S1, K3 как S2, K4 как S3:

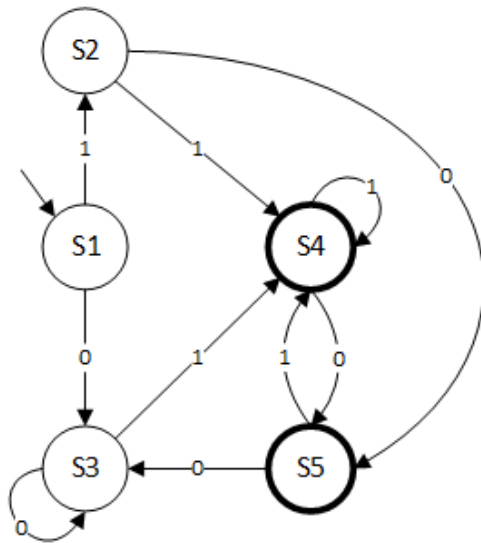
| | | | |
|---|----|----|----|
| | ↓ | 1 | 1 |
| | S1 | S2 | S3 |
| 1 | S2 | S2 | S2 |
| 0 | S1 | S3 | S1 |

Переход к минимальному
детерминированному
распознавателю



Построили минимальный детерминированный конечный распознаватель языка L_3 .
Сравним полученный распознаватель с распознавателем, полученным в прошлой
версии лабораторной работы:

Детерминированный распознаватель



Во-первых, исходная версия предполагала обязательное наличие хотя бы двух символов в цепочках, следовательно распознаватели будут формировать разные языки, такое различие возникло из-за неверной интерпретации словесного описания языка.

Во-вторых, в новой версии гораздо меньше состояний и переходов, что положительно влияет на производительность, понижает сложность отладки и сокращает набор тестовых данных.

4. Написать программу компиляционного типа для реализации минимального детерминированного конечного распознавателя языка L_3 .

```
MESSAGES = {
    -1: "Отвергнуть, невалидный входной символ",
    0: "Отвергнуть, цепочка не содержит 1 на одной из двух последних позиций",
    1: "Допустить",
    2: "Допустить"
}

def l3_validator(l3_input):
    original_input = l3_input
    s = 0
    while len(l3_input) > 0 and s >= 0:
        current_symbol = l3_input[0]
        if s == 0:
            if current_symbol == "1":
                s = 1
            elif current_symbol == "0":
                s = 0
        else:
            s = -1
        break
```

```

elif s == 1:
    if current_symbol == "1":
        s = 1
    elif current_symbol == "0":
        s = 2
    else:
        s = -1
        break
elif s == 2:
    if current_symbol == "1":
        s = 1
    elif current_symbol == "0":
        s = 0
    else:
        s = -1
        break

l3_input = l3_input[1:]

print(original_input, MESSAGES[s])
return s

```

5. Написать программу интерпретационного типа для реализации минимального детерминированного конечного распознавателя языка L_3 .

```

MESSAGES = {
    -1: "Отвергнуть, невалидный входной символ",
    0: "Отвергнуть, цепочка не содержит 1 на одной из двух последних позиций",
    1: "Допустить",
}

PERMITTING = [1, 2]

MATRIX = {
    "1": [1, 1, 1],
    "0": [0, 2, 0]
}

def l3_validator(l3_input):
    original_input = l3_input
    s = 0
    while len(l3_input) > 0 and s >= 0:
        current_symbol = l3_input[0]
        if current_symbol in MATRIX:
            s = MATRIX[l3_input[0]][s]
        else:
            s = -1
            break

    l3_input = l3_input[1:]

```

```

if s in PERMITTING:
    s = 1

print(original_input, MESSAGES[s])
return s

```

6. Подобрать наборы тестовых данных так, чтобы в процессе тестирования сработал каждый переход конечного распознавателя.

(a) 0110100 - использование всех переходов

(b) 21 - переход в ошибку

Тесты для компиляционного варианта программы:

```

# Тестовые данные для всех переходов
assert l3_validator("0110100") == 0
assert l3_validator("21") == -1

```

Тесты для интерпретационного варианта программы:

```

# Тестовые данные для всех переходов
assert l3_validator("0110100") == 0
assert l3_validator("21") == -1

```

7. Подобрать наборы тестовых данных так, чтобы в процессе тестирования распознаватель закончил обработку цепочек в каждом состоянии конечного распознавателя.

(a) <пустая строка> - состояние S1

(b) 1 - состояние S2

(c) 10 - состояние S3

(d) 21 - состояние ошибки

Тесты для компиляционного варианта программы:

```

# Тестовые данные для всех состояний
assert l3_validator("") == 0
assert l3_validator("1") == 1
assert l3_validator("10") == 2
assert l3_validator("21") == -1

```

Тесты для интерпретационного варианта программы:

```
# Тестовые данные для всех состояний
assert l3_validator("") == 0
assert l3_validator("1") == 1
assert l3_validator("10") == 1
assert l3_validator("21") == -1
```

8. Выполнить тестирование программ для реализации минимального детерминированного конечного распознавателя языка L_3 .

Результаты выполнения компиляционного варианта программы:

```
0110100 Отвергнуть, цепочка не содержит 1 на одной из двух последних позиций
21 Отвергнуть, невалидный входной символ
    Отвергнуть, цепочка не содержит 1 на одной из двух последних позиций
1 Допустить
10 Допустить
21 Отвергнуть, невалидный входной символ

Process finished with exit code 0
```

Результаты выполнения интерпретационного варианта программы:

```
0110100 Отвергнуть, цепочка не содержит 1 на одной из двух последних позиций
21 Отвергнуть, невалидный входной символ
    Отвергнуть, цепочка не содержит 1 на одной из двух последних позиций
1 Допустить
10 Допустить
21 Отвергнуть, невалидный входной символ

Process finished with exit code 0
```

Оба варианта программы завершились без ошибок, а значит проверки в проверках истинные, следовательно программа написана верно.

Вывод: в ходе лабораторной работы изучили основные способы задания регулярных языков, способы построения, алгоритмы преобразования, анализа и реализации конечных распознавателей.