Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное
учреждение высшего образования «Национальный исследовательский
университет ИТМО»

Факультет инфокоммуникационных технологий

Математическая лингвистика

Практическая работа №4

Выполнили:

студент группы К34422

Малаев Степан Геннадьевич

Проверил:

доцент практики, КТН

Болгова Екатерина Владимировна

Санкт-Петербург

2025

Ход работы.

1. Постройте НКА, распознающие следующие множества цепочек:

а) abc, abd и aacd. Входным алфавитом считать {a, b, c, d};

б) 0101, 101 и 011;

в) ab, bc и ca. Входным алфавитом считать {a, b, c}.

а) Множество: {abc, abd, aacd}.

Алфавит: {a, b, c, d}.

1. q_0 – начальное состояние.
   Переход: q_0 — a → q_1.
2. q_1 – с этого состояния происходит недетерминированный выбор:
   a. При чтении b переходим в состояние q_2, ветвь для abc и abd.
   b. При чтении a переходим в состояние q_3, ветвь для aacd.
3. Ветвь q_1 — b → q_2:
   a. q_2 — c → q_4 – состояние q_4 является допускающим, распознает слово abc.
   b. q_2 — d → q_5 – состояние q_5 является допускающим, распознает слово abd.
4. Ветвь q_1 — a → q_3:
   a. q_3 — c → q_6.
   b. q_6 — d → q_7 – состояние q_7 является допускающим, распознает слово aacd.

б) Множество: {0101, 101, 011}.

Алфавит: {0, 1}.

1. q_0 – начальное состояние.
   Переходы:
   - q_0 — 0 → q_1 ветвь для цепочек, начинающихся с 0: 0101 и 011.
   - q_0 — 1 → q_5 ветвь для 101.

2. Ветвь для цепочек, начинающихся с 0:
    a. $q_1 — 1 \rightarrow q_2$.
    b. Из состояния q_2:
        - 0101: $q_2 — 0 \rightarrow q_3$, затем $q_3 — 1 \rightarrow q_4$, где q_4 – допускающее.
        - 011: $q_2 — 1 \rightarrow q_4$, общая допускающая для данной ветви.
3. Ветвь для цепочки 101:
    a. $q_5 — 0 \rightarrow q_6$.
    b. $q_6 — 1 \rightarrow q_7$, где q_7 является допускающим.

в) Множество: {ab, bc, ca}.

Алфавит: {a, b, c}.

1. q_0 – начальное состояние.

   Переходы:
    - $q_0 — a \rightarrow q_1$ – для ab.
    - $q_0 — b \rightarrow q_2$ – для bc.
    - $q_0 — c \rightarrow q_3$ – для ca.
2. Далее:
    a. q_1: $q_1 — b \rightarrow q_4$*, где q_4 допускающее, распознаёт ab.
    b. q_2: $q_2 — c \rightarrow q_5$*, где q_5 допускающее, распознаёт bc.
    c. q_3: $q_3 — a \rightarrow q_6$*, где q_6 допускающее, распознаёт ca.

2. Преобразуйте НКА из (1) в ДКА

При преобразовании НКА в ДКА, создается состояния ДКА, каждое из которых соответствует подмножеству состояний исходного НКА. При отсутствии ε-переходов достаточно взять замыкание по начальным состояниям равным самому состоянию. Для каждого нового подмножества определяются переходы по каждому символу алфавита как объединение переходов всех входящих в него состояний. При этом множество считается допускающим, если хотя бы одно из входящих в него состояний НКА является допускающим.

а)

1. Начальное состояние:
   $Q_0 = \{q_0\}$.
2. Переход из Q_0:
   a. По символу a:
      Из q_0 по a переходим в $q_1 \rightarrow Q_1 = \{q_1\}$.
   b. По символам b, c, d:
      Переходы не определены, пустое множество.
3. Переход из $Q_1 = \{q_1\}$:
   a. По a:
      Из q_1 по a получаем $q_2 \rightarrow Q_2 = \{q_2\}$.
   b. По b:
      Из q_1 по b получаем $q_3 \rightarrow Q_3 = \{q_3\}$.
4. Переход из $Q_2 = \{q_2\}$:
   a. По c:
      Из q_2 по c переходим в $q_4 \rightarrow Q_4 = \{q_4\}$.
   b. Остальные символы не дают перехода.

5. Переход из $Q_3 = \{q_3\}$:
    a. По c:

       Из q_3 по c $\rightarrow$ q_6 $\rightarrow$ $Q_6 = \{q_6\}$, принимающее, так как q_6 – финальное для abc.
    b. По d:

       Из q_3 по d $\rightarrow$ q_7 $\rightarrow$ $Q_7 = \{q_7\}$, принимающее, для abd.
6. Переход из $Q_4 = \{q_4\}$:
    a. По d:

       Из q_4 по d $\rightarrow$ q_5 $\rightarrow$ $Q_5 = \{q_5\}$, принимающее, для aacd.

Принимающими будут множества, содержащие:

- $Q_5 = \{q_5\}$.
- $Q_6 = \{q_6\}$.
- $Q_7 = \{q_7\}$.

б)

1. Начальное состояние:

   $Q_0 = \{q_0\}$.
2. Переходы из Q_0:
    a. По 0:

       Из q_0 по 0 могут быть переходы в два состояния, так как в НКА для 0101 и 011 оба начинаются с 0.

       $Q_1 = \{q_1, q_8\}$.
    b. По 1:

       Из q_0 по 1 переходим в q_5 $\rightarrow$ $Q_5 = \{q_5\}$.
3. Переход из $Q_1 = \{q_1, q_8\}$:
    a. По 1:

       Из q_1 по 1 $\rightarrow$ q_2, из q_8 по 1 $\rightarrow$ q_9, таким образом $Q_2 = \{q_2, q_9\}$.
    b. По символам 0 или 1 — пустое множество.

4. Переход из $Q_2 = \{q_2, q_9\}$:
    a. По 0:

       Из q_2 по 0 → q_3; если для q_9 по 0 перехода нет, то $Q_3 = \{q_3\}$.
    b. Остальные переходы не дают дополнительных состояний.
5. Переход из $Q_3 = \{q_3\}$:
    a. По 1:

       Из q_3 по 1 → q_4, получаем $Q_4 = \{q_4\}$, которое является принимающим, цепочка 0101.
6. Обработка ветви для цепочки 101:
    a. $Q_5 = \{q_5\}$, из Q_0 по 1.
    b. Из Q_5 по 0:

       Из q_5 по 0 → q_6, получаем $Q_6 = \{q_6\}$.
    c. Из Q_6 по 1:
    d. Из q_6 по 1 → q_7, получаем $Q_7 = \{q_7\}$, принимающее, для 101.
7. Отдельная обработка для цепочки 011:

   Уже рассмотренная ветвь через Q_1 и Q_2 привела к конечному состоянию $Q_4 = \{q_4\}$ для 0101, но цепочка 011 должна обрабатываться отдельно. Если рассматривать, что переходы из Q_0 по 0 могли быть неоднозначны, то можно выделить:

   - Из Q_0 по 0 → $Q_1 = \{q_1, q_8\}$.
   - Из Q_1 по 1: $Q_2 = \{q_2, q_9\}$.
   - Из Q_2:

     Если по 1 для некоторого элемента переход существует, то:

     Рассмотрим, что для q_9 по 1 → q_{10}, тогда дополнительно из Q_2 по 1 дополнительно получаем $Q_{10}' = \{q_{10}\}$.

Таким образом, для цепочки 011:

- Из Q_0 по 0 → $\{q_1, q_8\}$.
- Из $\{q_1, q_8\}$ по 1 → $\{q_2, q_9\}$.
- Из $\{q_2, q_9\}$ по 1 → получаем $\{q_{10}\}$, если только q_9 имеет переход по 1.

Тогда $Q_{10} = \{q_{10}\}$ будет принимающим состоянием для 011.

Принимающими будут множества, содержащие:

- $Q_4 = \{q_4\}$.
- $Q_7 = \{q_7\}$.
- $Q_{10} = \{q_{10}\}$.

в)

1. Начальное состояние:

   $Q_0 = \{q_0\}$.
2. Переходы из Q_0:
   a. По a:

      $Q_1 = \{q_1\}$, для цепочки ab.
   b. По b:

      $Q_3 = \{q_3\}$, для bc.
   c. По c:

      $Q_5 = \{q_5\}$, для ca.
3. Дальнейшие переходы:
   a. Из $Q_1 = \{q_1\}$ по b:

      Переход в $\{q_2\}$ → $Q_2 = \{q_2\}$, принимающее для ab.
   b. Из $Q_3 = \{q_3\}$ по c:

      Переход в $\{q_4\}$ → $Q_4 = \{q_4\}$, принимающее для bc.
   c. Из $Q_5 = \{q_5\}$ по a:

      Переход в $\{q_6\}$ → $Q_6 = \{q_6\}$, принимающее для ca.

Принимающими будут множества Q_2, Q_4 и Q_6, поскольку они содержат состояния, являющиеся принимающими в исходном НКА.

3. Преобразуйте следующий НКА в эквивалентный ДКА и опишите неформально язык, который он допускает

	0	1
→p	{p,q}	{p}
q	{r,s}	{t}
r	{p,r}	{t}
*s	Ø	Ø
*t	Ø	Ø

Обозначим состояния для исходного НКА:

- p — начальное.
- q, r — промежуточные.
- s, t — конечные.

Начальное состояние ДКА: {p}.

Переходы:

1. Из {p}:
    a. по 0 → {p, q}.
    b. по 1 → {p}.
2. Из {p, q}:
    a. по 0 → {p, q, r, s}, объединяем переходы из p и q.
    b. по 1 → {p, t}.
3. Из {p, q, r, s}:
    a. по 0 → {p, q, r, s}, само в себя.
    b. по 1 → {p, t}.
4. Из {p, t}:
    a. по 0 → {p, q}.
    b. по 1 → {p}.

Состояния ДКА:

- S_0 = {p}, начальное.
- S_1 = {p, q}
- S_2 = {p, q, r, s}
- S_3 = {p, t}

Конечные состояния:

- S_1 = {p, q, r, s}, содержит s.
- S_3 = {p, t}, содержит t

Неформальное описание языка:

Автомат начинает в p. Пока читаем 1, остаёмся в p. При первом 0 мы добавляем в множество и состояние q.

Далее два варианта:

- Если в q, или в r, который возникает по дальнейшим 0, считываем 1, то переходим в t.
- Если в q считываем ещё 0, то переходим в s.

Состояния s и t не имеют исходящих переходов, поэтому слово должно заканчиваться ровно тогда, когда НКА попал в s или t. Язык — это все строки, в которых после некоторой последовательности единиц обязательно встречается хотя бы один ноль, и в итоге завершиться либо на том 0, который привёл к s, либо на 1, которая привела к t.

ДКА содержит четыре непустых подмножества, а язык — это все строки, заканчивающиеся ровно на переходе в s когда двойной 0 из q или t когда 1 из q или r.

4. Рассмотрите следующий ε-НКА

	ε	a	b	c
→p	∅	{p}	{q}	{r}
q	{p}	{q}	{r}	∅
*r	{q}	{r}	∅	{p}

а) найдите ε-замыкание каждого из состояний;

б) выпишите все цепочки, длина которых не более 3, допустимые данным автоматом;

а)

Замыкания:

1. ε — замыкание(p):

   ε — замыкание(p) = {p}, переходов по ε из p нет.
2. ε — замыкание(q):

   ε — замыкание(q) = {q, p}, из q по ε можно перейти в p.
3. ε — замыкание(r):

   ε — замыкание(r) = {r, q, p}, из r по ε → q, а из q по ε → p.

б)

Состояние r – конечное, значит итоговое множество состояний, с учётом ε-замыканий, должно содержать r.

Все принимаемые цепочки для заданной длины:

1. Длина 0:

   ε-строка не принимается, начинаем в p, ε — замыкание(p) не содержит r.

2. Длина 1:
    a. a: p → p, нет r.
    b. b: p → q, ε — замыкание {q, p}, нет r
    c. c: p → r, ε — замыкание {r, q, p}, содержит r, следовательно принимается.
3. Длина 2:

    Обозначим:

    - A = {p}, начальное, непринимающее.
    - B = {q, p}, непринимающее.
    - C = {r, q, p}, принимающее.

    Из A по a остаёмся в A, по b переходим в B, по c → C.

    Из B по a → B, по b → C, по c → C.

    Из C по a,b,c → C.

    Две буквы приводят в C, если: $A \xrightarrow{x} A \mid B \mid C \xrightarrow{y} C$.

    Все пары:

    - ac: A → A → C
    - bb: A → B → C
    - bc: A → B → C
    - ca, cb, cc: уже первый c переводит в C, второй символ оставляет в C.

    Тогда принимаются: {ac, bb, bc, ca, cb, cc}.
4. Длина 3:

    Если мы в C, принимающем, на любом шаге, то оставшиеся символы удерживают в C.

Следовательно:

1. Первый символ c → сразу C, любые два символа дальше (a,b,c) → остаёмся в C. Строки вида cxx, их 9 вариантов.
2. Переходим в C на втором символе, тогда третий символ любой → 3 * 3 = 9 вариантов.
3. Переходим в C только на третьем символе.

Получаем 23 принимаемые строки из 27 возможных:

- c: caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc.
- ac: aca, acb, acc.
- bb: bba, bbb, bbc.
- bc: bca, bcb, bcc.
- aa → c: aac.
- ab → b или c: abb, abc.
- ba → b или c: bab, bac.

5. Опишите обычными словами языки следующих регулярных выражений:
а) (1 + ε)(00*1)*0*
б) (0*1*)*000(0 + 1)*
в) (0 + 10)*1*

а) (1 + ε)(00*1)*0*
Язык, где строки могут начинаться с 1 или без неё, затем содержать повторяющиеся конструкции как 0 с дополнительными нулями и 1, а завершаться цепочкой 0.

Структура:

1. Строка может начинаться с 1 или сразу с пустой строки.
2. Затем может идти любое количество повторений блока, где блок выглядит так:
    a. Сначала идёт 0,
    b. Далее следует любое количество дополнительных 0
    c. Затем 1.
3. Строка завершается произвольной последовательностью нулей, включая пустую.

б) (0*1*)*000(0 + 1)*

Язык включает все строки, в которых обязательно присутствует последовательность трёх нулей, а до неё строка состоит из упорядоченных блоков нулей и единиц.

Структура:

1. Сначала идёт любая последовательность блоков, где каждый блок представляет собой последовательность нулей, возможно пустую, и за ней последовательность единиц, также возможно пустая. Заметим, что в каждом таком блоке все нули располагаются перед единицами, то есть никогда 1 не следует за 0 внутри одного блока.
2. Затем должна встретиться подстрока 000.
3. После неё может идти любая последовательность символов 0 и 1, включая пустую.

в) (0 + 10)*1*

Строки начинаются с комбинаций 0 и 10, то есть, 1 никогда не встречается без следующего 0, а завершаются цепочкой единиц, которая может отсутствовать.

Структура:

1. Начинается с любой, возможно пустой, последовательности, составленной из символа 0 и подстроки 10.
2. Затем следует произвольное, возможно пустое, количество единиц.

6. Напишите регулярное выражение для описания телефонных номеров. Мобильных и городских.

Для мобильных номеров: \+7\d{10}.
Номер мобильного телефона в России начинается с +7, за которым следует 10 цифр.

Для городских номеров: (?:\+7|8)\s*\(\d{3}\)\s*\d{3}-\d{2}-\d{2}
Номер городского телефона может начинаться с +7 или 8. Далее, возможно, идут пробелы, затем код города в круглых скобках, после чего – опциональные пробелы и сам номер.
Примеры:

- +7 (495) 123-45-67
- 8(495)123-45-67

7. Дано регулярное выражение (0 + 1)*1(0 + 1) + (0 + 1)*1(0 + 1)(0 + 1). С помощью дистрибутивных законов преобразуйте его в два различных, более простых, эквивалентных выражения.

Оба слагаемых содержат общий префикс (0 + 1)*1(0 + 1). Можно заметить что, в первом слагаемом после общего префикса ничего не остаётся, что можно записать как ε, а во втором слагаемом остается (0 + 1).
Получается: (0 + 1)*1(0 + 1)(ε + (0 + 1)).

Также можно вынести не всю общую часть, а (0 + 1)*1. Тогда можно упростить к: (0 + 1)*1((0 + 1) + (0 + 1)(0 + 1)).

8. Постройте конечный автомат для распознавания в тексте переменных. Для простоты будем считать: имена переменных, функций, методов должны содержать строчные и прописные буквы латинского алфавита, цифры от 0 до 9, знак _. При этом начинаться имя должно с буквы. Для расширения: используйте также выделение специальных зарезервированных “магических имён”, которые играют особую роль в Python (возьмем,__init__, __import__, __file__).

Автомат для обычных идентификаторов:

- q0, начальное:
  При входном символе из диапазона A–Z или a–z → переход в q1.
- q1, принимающее:
  При входном символе из диапазона A–Z, a–z, цифры (0–9) или символ _ → остаёмся в q1.

Регулярное выражение для обычного идентификатора: [A-Za-z][A-Za-z0-9_]*

Распознавание магических имён:
Если из q0 первым символом является _, проверяем, что сразу после него идёт ещё _. Далее, с помощью ветвления, сравниваем последовательность символов с одним из зарезервированных имён:

- __init__
- __import__
- __file__

Если последовательность полностью совпадает с одним из этих паттернов, автомат переходит в специальное принимающее состояние.

9. Постройте конечный автомат для распознавания в тексте зарезервированных слов. Для простоты возьмем следующие слова: class, for, while, if, elif, else, function, procedure.

Схема:

1. Начальное состояние:

   q0.
2. Из q0 по первому символу осуществляется ветвление:
   a. Если символ = c:

      Распознаётся слово class: q0 — c → q1 — l → q2 — a → q3 — s → q4 — s → q5, конечное.
   b. Если символ = f:

      Из q0 переход по f ведёт к ветке, где различают:
      i. for: q0 — f → q6 — o → q7 — r → q8, конечное.
      ii. function: q0 — f → q6 — u → q9 — n→ q10 — c→ q11 — t → q12 — i → q13 — o → q14 — n → q15, конечное.
   c. Если символ = w:

      while: q0 — w → q16 — h → q17 — i → q18 — l → q19 — e → q20, конечное.
   d. Если символ = i:

      if: q0 — i → q21 — f → q22, конечное.
   e. Если символ = e:

      Из q0 переход по e ведёт к ветке для слов, начинающихся с el: q0 — e → q23 — l → q24, затем:
      i. elif: q24 — i → q25 — f → q26, конечное.
      ii. else: q24 — s → q27 — e → q28, конечное.
   f. Если символ = p:

      procedure: q0 — p → q29 — r → q30 — o → q31 — c → q32 — e → q33 — d → q34 — u → q35 — r → q36 — e → q37, конечное.

10. Напишите регулярное выражение для проверки корректности использования скобок — все скобки должны быть в паре: (), [], {}.

Регулярное выражение:
^\s*(?:(?:\((?>[^()]*|\g<0>)*\))|(?:\[(?>[^\[\]]*|\g<0>)*\])|(?:\{(?>[^{}]*|\g<0>)*\}))*\s*$

Для каждой группы используется конструкция вида \((?>[^()]*|\g<0>)*\), где:

- \(– открывающая скобка.
- (?>[^()]*|\g<0>)* – либо последовательность символов, не являющихся скобками, либо рекурсивный вызов того же шаблона.
- \) – закрывающая скобка.

Вывод.

В ходе выполнения данной практической работы были исследованы методы построения конечных автоматов для распознавания цепочек и их преобразование в ДКА.

Также вычислены ε-замыкания, построены регулярные выражения для проверки корректности вложенности скобок, форматов телефонных номеров и идентификаторов.

Результаты демонстрируют практическое применение теории автоматов и регулярных выражений в лексическом анализе.