

МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА
ВЕЛИКОГО»**

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Отчёт по дисциплине «Математическая логика»

Лабораторная работа №2
«Регулярные выражения»
Вариант №14

Студент: _____

Салимли Айзек Мухтар Оглы

Преподаватель: _____

Востров Алексей Владимирович

«_____» _____ 20__ г.

Санкт-Петербург, 2025

Содержание

Введение	3
1 Постановка задачи	4
2 Математическое описание	5
2.1 Конечный автомат	5
2.2 Язык распознаваемый автоматом	6
2.3 Регулярное выражение	6
2.4 Граф НКА	7
2.5 Граф ДКА	8
3 Программная реализация	9
3.1 Lib.hs	9
3.2 Main.hs	13
Заключение	15
Список литературы	16

Введение

В данном отчете, описана реализация программы, распознающая ссылку на WEB страницу, по заданному варианту. Так же в отчете представлен конечный автомат, на основе которого спроектирована валидность ссылки, по регулярному выражению. Был спроектирован cabal проект. Программа была разделена на два .hs файла:

1. Lib.hs - Управляющая логика программы
2. Main.hs - Меню программы

Для реализации проекта, были выбраны:

- Cabal 3.0 - Сборщик проекта
- Haskell2010 - Спецификация языка
- Компилятор - GHC 9.12.1
- Haskell - Язык программирования
- VS Code - Среда разработки

Использованные библиотеки:

- base 4.19.2.0 - Стандартная библиотека Haskell
- random 1.2 - Для генерации ссылок
- text 1.2 - Для работы с текстом
- bytestring 0.10 - Для последующего использования в конечном автомате

1 Постановка задачи

По заданному варианту построить регулярное выражение, затем недетерминированный конечный автомат и детерминировать его (переходы можно задавать диапазонами). Реализовать программу, которая проверяет введенный текст через реализацию конечного автомата (варианты вывода: строка соответствует, не соответствует, символы не из алфавита). Также необходимо реализовать функцию случайной генерации верной строки по полученному конечному автомату.

Вариант №14:

- Проверка на соответствие ссылке яндекс диск

Яндекс Диск, работает по нескольким ссылкам:

- `disk.yandex.ru/d/...` – часто встречающийся вид ссылки, ведущий на конкретный файл/папку.
- `disk.yandex.ru/clients/disk/...` – менее распространенный формат.
- `yadi.sk/d/...` – укороченный вариант публичной ссылки.

Реализованы все вышеперечисленные варианты.

2 Математическое описание

2.1 Конечный автомат

Программа работает на основе детерминированного конечного автомата, который определяется кортежем:

$$M = \{S, \Sigma, \delta, s_\theta, F\}$$

, где:

- Q - Конечное множество состояний
- Σ - Входной алфавит (ASCII символы)
- $\delta : S \times \Sigma \rightarrow Q$ - Функция переходов
- $s_\theta \in S$ - Начальное состояние
- $F \subseteq S$ - Множество финальных состояний

В коде множество состояний задается алгебраическим типом ADT с помощью data, и содержит:

$$Q = \{Start, H, Ht, Htt, Htt\text{p}s, Htt\text{p}sColon, Htt\text{p}sColonSlash, Htt\text{p}sColonSlashSlash, \\ D, Di, Dis, Disk, DiskDot, DiskDotY, DiskDotYa, DiskDotYan, DiskDotYand, \\ DiskDotYande, DiskDotYandex, YandexDot, YandexDotR, YandexDotRu, \\ YandexDotRuSlash, YandexDotRuSlashD, YandexDotRuSlashDSlash, Final\}$$

Входной алфавит можно определить как:

$$\Sigma = \{ASCII / (Control \cup Space)\}$$

Функция переходов определена в transition(в коде), можно описать: $\delta : S \times \Sigma \rightarrow S$

$$\delta(q, a) = \begin{cases} H & \text{если } q = Start, a = 'h' \\ Ht & \text{если } q = H, a = 't' \\ Htt & \text{если } q = Ht, a = 't' \\ Htt\text{p}s & \text{если } q = Htt, a = 'p' \\ Htt\text{p}sColon & \text{если } q = Htt\text{p}s, a = 's' \\ Htt\text{p}sColonSlash & \text{если } q = Htt\text{p}sColon, a = ':' \\ Htt\text{p}sColonSlashSlash & \text{если } q = Htt\text{p}sColonSlash, a = '/' \\ D & \text{если } q = Htt\text{p}sColonSlashSlash, a = '/' \\ \dots & \text{(по аналогии, вплоть до Final)} \\ Final & \text{если } q = YandexDotRuSlashDSlash, a \in \Sigma_v \\ q & \text{если } a \in Space \\ Start & \text{иначе} \end{cases}$$

где $\Sigma_v = \{c \in \Sigma \mid c \notin Control \cup Space\}$ — допустимые символы пути.

- $s_\theta = Start$ — начальное состояние
- $F = \{Final\}$ — множество финальных состояний

Начальное состояние: $s_0 = Start$

Финальное состояние: $F = Final$

2.2 Язык распознаваемый автоматом

Автомат распознаёт язык $L \subseteq \Sigma^*$, содержащий строки следующего вида:

$$L = \{w = p \cdot s \mid p \in P, s \in \Sigma_v^+\}$$

, где $P = \text{https://disk.yandex.ru/d/...}, \text{https://disk.yandex.ru/clients...}, \text{https://yadi.sk/d/...}$

То есть, строка должна начинаться с допустимого префикса и содержать непустую последовательность валидных символов после него.

2.3 Регулярное выражение

Регулярные выражения представляют собой формальный способ задания регулярных языков.

Случаи:

- \emptyset - не содержит ни одной строки
- ϵ - пустая строка (строка длины 0)
- $\forall(a) : a \in \Sigma, a -$

Операции:

- Конкатенация: Если r и s - регулярные выражения, то их конкатенация задает язык $xy \mid x \in L(r), y \in L(s)$
- Объединение: Если r и s - регулярные выражения, то $r \cup s$ задает язык $L(r) \cup L(s)$
- Звезда Клини: r - регулярное выражение, то r^* задает язык: $x_1 x_2 \dots x_n \mid n \geq 0, x_i \in L(r)$

Множество регулярных выражений: Множество над Σ - определяется рекурсивной грамматикой:

- \emptyset, ϵ и $a \in \Sigma$ - регулярные выражения
- r, s - регулярные выражения $\Rightarrow (r \cup s), (rs), (r)^*$ - тоже регулярные выражения
- Множество - счетное.

Регулярное выражение:

- Вариант 1: `https://disk.yandex.ru/d/ + id`
- Вариант 2: `https://disk.yandex.ru/clients/disk/ + id`
- Вариант 3: `https://yadi.sk/d/ + id`

$id = \{a, b, \dots, z, 0, 1, \dots, 9, -, \}$, а длина генерируется случайно от 12 до 20 символов.

$$let A = [a - z 0 - 9 - _] \{12, 20\}$$

Тогда регулярное выражение можно записать как:

`^(https://disk\.yandex\.ru/d/|https://disk\.yandex\.ru/clients/disk/|https://yadi\.sk/d/)[a-z0-9_]{12,20}`

2.4 Граф НКА

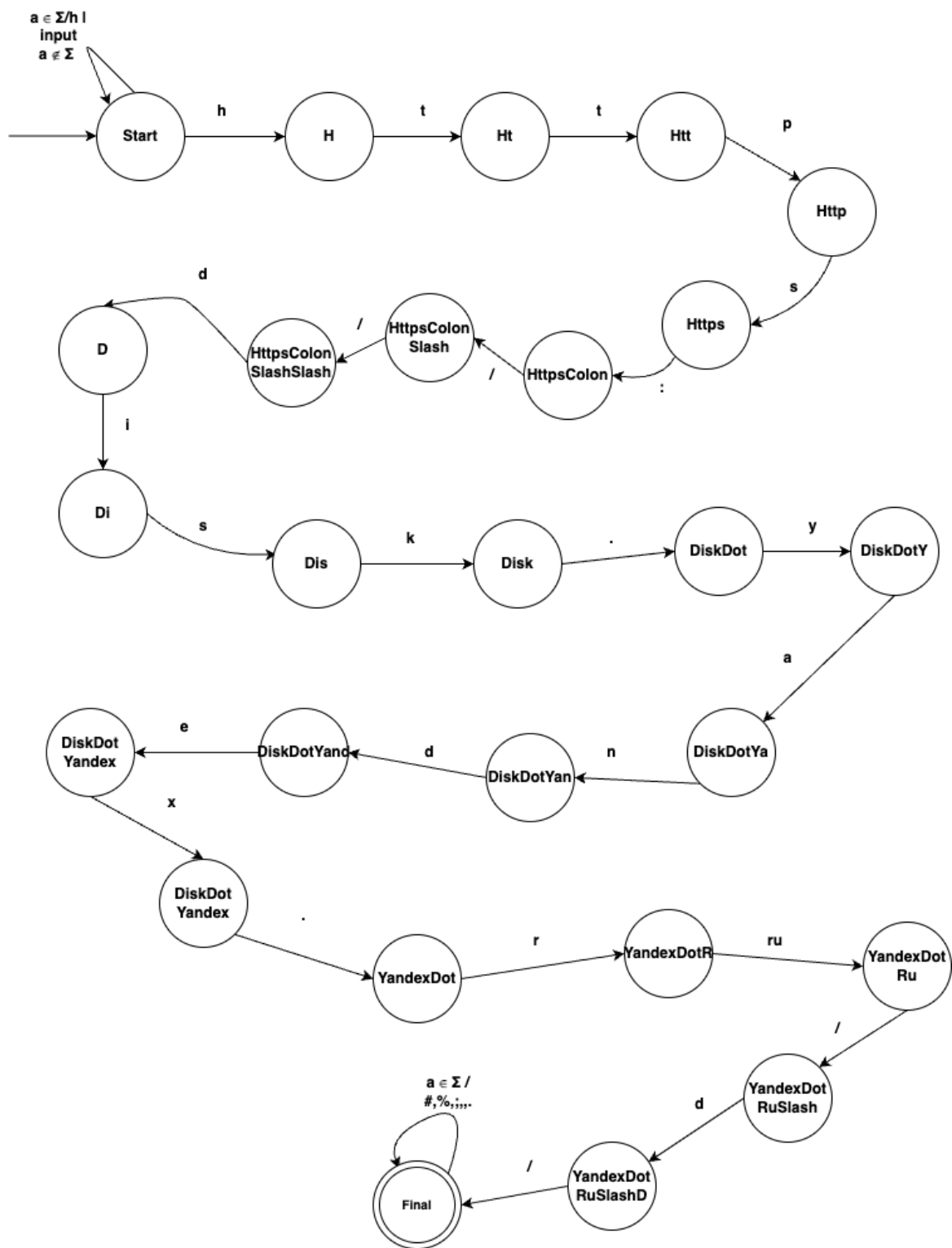


Рис. 1: Граф конечного автомата

2.5 Граф ДКА

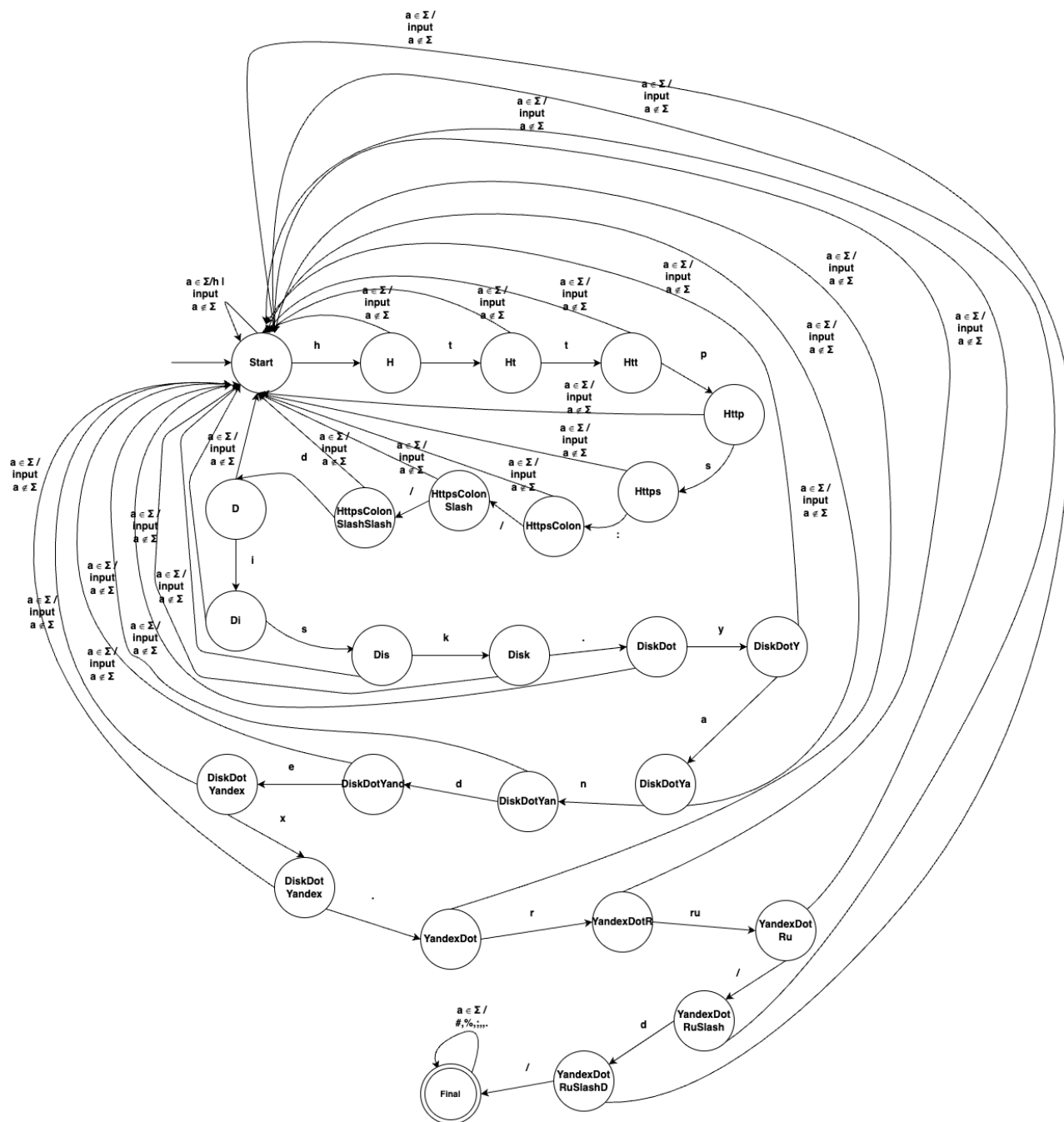


Рис. 2: Граф детерминированного конечного автомата

3 Программная реализация

В качестве языка программирования был выбран Haskell
В качестве стандарта языка, был выбран Haskell2010
Для реализации программы, был собран cabal - проект, который включает в себя два .hs файла:

- Lib.hs - Отвечающий за меню программы
- Main.hs - Отвечающий за управляющую логику

В листинге 1, представлен код Lib.hs. В листинге 2, представлен код Main.hs:

3.1 Lib.hs

Управляющая логика, основана на работе конечного автомата, на вход которого поступают введенные символы, далее функция переходов, переходит в следующее состояние, и так пока состояние не будет Final, либо пока не будет введен некорректный символ. В качестве состояний автомата, был создан тип данных - State, а функции переходов transition определяются как State -> Char -> State. То есть: состояние, входной символ, следующее состояние.

Листинг 1: Lib.hs

```
1      module Lib
2      ( checkYandexDiskLink
3      , generateValidLink
4      ) where
5
6  import Data.Char (isAscii, isSpace, isControl)
7  import System.Random (randomRIO)
8  import Control.Monad (replicateM)
9  import Data.List (isInfixOf)
10
11  validChars :: String
12  validChars = ['a'..'z'] ++ ['0'..'9'] ++ "-_"
13
14  preprocess :: String -> String
15  preprocess = filter (not . isSpace) . map toLower
16      where
17          toLower c =
18              if c >= 'A' && c <= 'Z'
19              then toEnum (fromEnum c + 32)
20              else c
21
22  data State = Start
23              | H | Ht | Htt | HttS
24              | HttSColon | HttSColonSlash | HttSColonSlashSlash
25              | D | Di | Dis | Disk
26              | DiskDot | DiskDotY | DiskDotYa | DiskDotYan | DiskDotYand
27              | DiskDotYande | DiskDotYandex
28              | YandexDot | YandexDotR | YandexDotRu
29              | YandexDotRuSlash
30              | YandexDotRuSlashD
31              | YandexDotRuSlashDSlash
32              | YandexDotRuSlashC
33              | YandexDotRuSlashCl
34              | YandexDotRuSlashCli
35              | YandexDotRuSlashClie
36              | YandexDotRuSlashClieN
37              | YandexDotRuSlashClient
38              | YandexDotRuSlashClients
39              | YandexDotRuSlashClientsSlash
```

```

40         | YandexDotRuSlashClientsSlashD
41         | YandexDotRuSlashClientsSlashDi
42         | YandexDotRuSlashClientsSlashDis
43         | YandexDotRuSlashClientsSlashDisk
44         | YandexDotRuSlashClientsSlashDiskSlash
45         | Y | Ya | Yad | Yadi
46         | YadiDot | YadiDotS | YadiDotSk
47         | YadiDotSkSlash
48         | YadiDotSkSlashD
49         | YadiDotSkSlashDSLash
50         | Final
51     deriving (Eq, Ord, Show, Enum, Bounded)
52
53 isValidChar :: Char -> Bool
54 isValidChar c = not (isSpace c || isControl c)
55
56 transition :: State -> Char -> State
57 transition Start 'h' = H
58 transition H 't' = Ht
59 transition Ht 't' = Htt
60 transition Htt 'p' = Https
61 transition Https 's' = HttpsColon
62 transition HttpsColon ':' = HttpsColonSlash
63 transition HttpsColonSlash '/' = HttpsColonSlashSlash
64 transition HttpsColonSlashSlash 'd' = D
65 transition HttpsColonSlashSlash 'y' = Y
66 transition Start 'd' = D
67 transition Start 'y' = Y
68 transition D 'i' = Di
69 transition Di 's' = Dis
70 transition Dis 'k' = Disk
71 transition Disk '.' = DiskDot
72 transition DiskDot 'y' = DiskDotY
73 transition DiskDotY 'a' = DiskDotYa
74 transition DiskDotYa 'n' = DiskDotYan
75 transition DiskDotYan 'd' = DiskDotYand
76 transition DiskDotYand 'e' = DiskDotYande
77 transition DiskDotYande 'x' = DiskDotYandex
78 transition DiskDotYandex '.' = YandexDot
79 transition YandexDot 'r' = YandexDotR
80 transition YandexDotR 'u' = YandexDotRu
81 transition YandexDotRu '/' = YandexDotRuSlash
82 transition YandexDotRuSlash 'd' = YandexDotRuSlashD
83 transition YandexDotRuSlash 'c' = YandexDotRuSlashC
84 transition YandexDotRuSlashD ' ' = YandexDotRuSlashD
85 transition YandexDotRuSlashD '/' = YandexDotRuSlashDSLash
86 transition YandexDotRuSlashDSLash _ = Final
87 transition YandexDotRuSlashC 'l' = YandexDotRuSlashCl
88 transition YandexDotRuSlashCl 'i' = YandexDotRuSlashCli
89 transition YandexDotRuSlashCli 'e' = YandexDotRuSlashClie
90 transition YandexDotRuSlashClie 'n' = YandexDotRuSlashClien
91 transition YandexDotRuSlashClien 't' = YandexDotRuSlashClient
92 transition YandexDotRuSlashClient 's' = YandexDotRuSlashClients
93 transition YandexDotRuSlashClients '/' = YandexDotRuSlashClientsSlash
94 transition YandexDotRuSlashClientsSlash 'd' = YandexDotRuSlashClientsSlashD
95 transition YandexDotRuSlashClientsSlashD 'i' =
    YandexDotRuSlashClientsSlashDi
96 transition YandexDotRuSlashClientsSlashDi 's' =
    YandexDotRuSlashClientsSlashDis
97 transition YandexDotRuSlashClientsSlashDis 'k' =
    YandexDotRuSlashClientsSlashDisk

```

```

98 transition YandexDotRuSlashClientsSlashDisk '/' =
    YandexDotRuSlashClientsSlashDiskSlash
99 transition YandexDotRuSlashClientsSlashDiskSlash _ = Final
100 transition Y 'a' = Ya
101 transition Ya 'd' = Yad
102 transition Yad 'i' = Yadi
103 transition Yadi '.' = YadiDot
104 transition YadiDot 's' = YadiDotS
105 transition YadiDotS 'k' = YadiDotSk
106 transition YadiDotSk '/' = YadiDotSkSlash
107 transition YadiDotSkSlash 'd' = YadiDotSkSlashD
108 transition YadiDotSkSlashD '/' = YadiDotSkSlashDSlash
109 transition YadiDotSkSlashDSlash _ = Final
110
111 transition Final c
112     | isValidChar c = Final
113     | otherwise = Start
114 transition s ' '
115     | s /= Final = s
116     | otherwise = Final
117 transition _ _ = Start
118
119 checkYandexDiskLink :: String -> IO ()
120 checkYandexDiskLink input = do
121     if not (all isAscii input)
122     then putStrLn "Oops! Only ASCII symbols!"
123     else do
124         let processed = preprocess input
125         if isMainPage processed
126         then putStrLn "Yandex Disk main page"
127         else do
128             let result = processString processed Start
129             if result == Final && isValidPath processed
130             then putStrLn "String - is a Yandex Disk link"
131             else putStrLn "Oops! Not a Yandex Disk link"
132 where
133     isMainPage :: String -> Bool
134     isMainPage s =
135         let normalized = preprocess s
136         in normalized == "https://disk.yandex.ru"
137         || normalized == "disk.yandex.ru"
138         || normalized == "https://disk.yandex.ru/"
139         || normalized == "disk.yandex.ru/"
140         || normalized == "disk.yandex.ru/clients/"
141         || normalized == "disk.yandex.ru/clients/disk/"
142         || normalized == "yadi.sk"
143         || normalized == "yadi.sk/"
144
145 processString :: String -> State -> State
146 processString [] st = st
147 processString (c:cs) st = processString cs (transition st c)
148
149 isValidPath :: String -> Bool
150 isValidPath s
151     | "disk.yandex.ru/d/" `isInfixOf` s = True
152     | "disk.yandex.ru/clients/disk/" `isInfixOf` s = True
153     | "yadi.sk/d/" `isInfixOf` s = True
154     | otherwise = False
155
156 generateValidLink :: IO String
157 generateValidLink = do

```

```

158 idLength <- randomRIO (12, 20)
159 idPart   <- replicateM idLength (randomChar validChars)
160 patternIndex <- randomRIO (1 :: Int, 3)
161 return $ case patternIndex of
162   1 -> "https://disk.yandex.ru/d/" ++ idPart
163   2 -> "https://disk.yandex.ru/clients/disk/" ++ idPart
164   3 -> "https://yadi.sk/d/" ++ idPart
165   _ -> "https://disk.yandex.ru/d/" ++ idPart
166 where
167   randomChar :: String -> IO Char
168   randomChar chars = do
169     idx <- randomRIO (0, length chars - 1)
170     return (chars !! idx)

```

Функция preprocess

- Тип: `String -> String`
- На входе: строка исходного текста (например, введённая ссылка)
- На выходе: строка без пробелов (и приведённая к нижнему регистру)
- Нужна для: очистки входной строки перед проверкой или генерацией ссылки

Функция checkYandexDiskLink

- Тип: `String -> IO ()`
- На входе: строка, содержащая ссылку
- На выходе: IO-экшн, который выводит сообщение о результате проверки
- Нужна для: проверки, является ли введённая строка валидной ссылкой Яндекс Диска

Функция generateValidLink

- Тип: `IO String`
- На входе: отсутствует (генерируется случайным образом)
- На выходе: сгенерированная строка, представляющая валидную ссылку Яндекс Диска
- Нужна для: автоматической генерации валидных ссылок

Функция transition

- Тип: `State -> Char -> State`
- На входе: текущее состояние автомата и входной символ (`Char`)
- На выходе: новое состояние автомата
- Нужна для: реализации функции переходов конечного автомата, определяющей, в какое состояние перейти для данного символа

Функция processString

- Тип: `String -> State -> State`
- На входе: строка для обработки и начальное состояние автомата
- На выходе: конечное состояние автомата после обработки всей строки
- Нужна для: последовательного применения функции `transition` ко всем символам входной строки

3.2 Main.hs

Листинг 2: Main.hs

```
1 module Main where
2 import Lib
3
4     main :: IO ()
5     main = do
6         putStrLn "Select an option:"
7         putStrLn ""
8         putStrLn "1. Check the link"
9         putStrLn ""
10        putStrLn "2. Generate a valid link"
11        putStrLn ""
12        putStrLn "3. Show link pattern"
13        putStrLn ""
14        putStrLn "4. Exit"
15        putStrLn ""
16        putStrLn "Enter the number and press Enter:"
17        choice <- getLine
18
19        case choice of
20            "1" -> do
21                putStrLn ""
22                putStrLn "Enter the link to check:"
23                putStrLn ""
24                input <- getLine
25                checkYandexDiskLink input
26                main
27            "2" -> do
28                putStrLn ""
29                link <- generateValidLink
30                putStrLn $ "Generated link: " ++ link
31                putStrLn ""
32                main
33            "3" -> do
34                putStrLn ""
35                putStrLn $ "Pattern /d/: " ++ "https://disk."
36                putStrLn $ "Pattern /clients/: " ++ "https"
37                putStrLn $ "Pattern /clients/disk/: " ++ "https://disk.yandex.ru/clients/disk/..."
38                putStrLn ""
39                main
40            "4" -> putStrLn "Goodbye!"
41            _ -> do
42                putStrLn ""
43                putStrLn ""
44                putStrLn "Invalid choice, please try again"
45                putStrLn ""
46                putStrLn ""
47                main
```

- Монада IO, выводит меню
- Обертка do, вызывает кейс для выбора подпрограмм программы, где:

1. Проверка строки на соответствие ссылки Яндекс Диска

Вызывает функцию checkYandexDiskLink

2. Генератор валидной ссылки Яндекс Диска

Вызывает функцию `generateValidLink` и перечисляет его в лист

3. Демонстрация шаблонов

Выводит шаблоны в консоль среды разработки

4. Завершение программы

Завершает программу и выводит сообщение

При ином входе в `main` функцию, выводится сообщение о неверном выборе, затем происходит возврат в меню.

Заключение

В результате выполнения лабораторной работы, был реализован cabal-проект, содержащий код программы, работающий на логике конечного автомата, который проверяет регулярное выражение - ссылку на Яндекс Диск. Были реализованы три варианта обращения к сайту:

- <https://disk.yandex.ru/d/...>
- <https://disk.yandex.clients/d/...>
- <https://yadisk.ru/d/...>

Плюсы:

- ДКА обрабатывает входную строку за линейное время относительно её длины.
- Язык Haskell, позволяет реализовать переходы в ДКА, как чистые функции. Позволяет избежать побочных эффектов.
- Была реализована функция преобразования ссылки в нормальную форму (без пробелов).

Минусы:

- Большое количество состояний и переходов сказывается на скорости работы программы.
- Любое изменение формата ссылки требует правок во многих правилах переходов, что увеличивает риск ошибок.
- Отсутствие явного логгирования.

Масштабируемость:

- Добавить явный вывод ошибок в входной строке.
- Реализация проверок иных сайтов Яндекса.
- Разбиение обработок на модули для разных ссылок.

Список литературы

1. Востров, А. В. Математическая логика [Электронный ресурс]. Режим доступа: <https://tema.spbstu.ru/compiler/> (последний визит: 01.04.2025).
2. Сети, Р.; Ахо, А. Компиляторы: принципы, технологии и инструменты / Р. Сети, А. Ахо. – М.: Издательство «Наука», 2006. – С. 104.