Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

Кафедра «Вычислительная и прикладная математика»

**Отчет  
по лабораторной работе №2  
по дисциплине**   
«Теория автоматов и формальных языков» **на тему**«Построение конечно автомата по регулярной грамматике»

Выполнили: студенты гр. 243  
Титова П.Ю.  
Попов А.Н

Проверил:  
доц. Филатов И.Ю

Рязань 2024

**Цель:**

1. Закрепить понятия «регулярная грамматика», «недетерминированный и детерминированный конечный автомат»;
2. Сформировать умения и навыки построения конечного автомата по регулярной грамматике и преобразования недетерминированного конечного автомата к детерминированному конечному автомату.

**Ход работы**

1. Грамматика:

G = ({S, A, B, C}, {a, b, c}, P, S), где P:

1) S -> aA | bB | aC;

2) A -> bA | bB | c;

3) B -> aA | cC | b;

4) C -> bB | bC | a.

2. Отобразим таблицу недетерминированной грамматики:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| F | S | A | B | C | N |
| a | A, C | - | A | N | - |
| b | B | A, B | N | B, C | - |
| c | - | N | C | - | - |

3. Объединим неоднозначности:  
**(A, C) = :**

* (A, N) =
* (C, N) =

**(A, B) = :**

* (A, N) =
* (B, N) =

**(B, C) = :**

* (B, N) =
* (C, N) =

4. Обновим таблицу:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| F | S | A | B | C |  |  |  |  |  |  | N |
| a |  | - | A | N |  |  | - | A | N | N | - |
| b | B |  | N |  | - |  |  | N | B | N | - |
| c | - | N | C | - |  | - |  | N | N | C | - |

5. Алгоритмизация. Листинг:

def generate\_undetermined\_grammar\_sheet(nonterminals\_count, terminals\_count):

#Первая строка - F, S, нетерминалы

field\_names = (

["F"] +

["S"] +

[nonterminal for nonterminal in ascii\_uppercase[:nonterminals\_count] if nonterminal not in {"F", "S", "N"}]

)

table = pd.DataFrame(columns=field\_names)

#Первая колонка - терминалы, остальные ячейки - пустые строки

for i in range(terminals\_count):

row = (

[ascii\_lowercase[i]] +

["" for \_ in range(len(field\_names) - 1)]

)

table.loc[i] = row

#Вывод: незаполненная таблица

print("Ваша незаполненная таблица:")

print(format\_table(table), end="\n\n")

#Ручной ввод грамматики в таблицу

for lin in range(terminals\_count):

for col in range(1, nonterminals\_count + 2):

table.iloc[lin, col] = "INPUT"

print(format\_table(table.iloc[[lin]]))

user\_input = input("Введите заместо 'INPUT': ")

while not re.match(r"^[A-Z](,[A-Z])\*$" , user\_input):

print("Неверный ввод!")

user\_input = input("Введите заместо 'INPUT': ")

table.iloc[lin, col] = user\_input

#Добавим столбец N после ввода грамматики

table["N"] = "-"

return table

def put\_undetermined\_units(table=pd.DataFrame()):

units = {} # Словарь неопределённостей: ключ - переменная, значение - неопределённости

for lin in range(table.shape[0]):

for col in range(1, table.shape[1]):

str\_cell = "".join(table.iloc[lin, col])

if len(str\_cell) > 1:

table.iloc[lin, col] = (str(table.columns[col]) +

str(table.iloc[lin, 0]) +

"\_" +

table.iloc[lin, col]) # Заполняем ячейку переменной из неопределённостей

units[table.iloc[lin, col]] = str\_cell.split(",")

return table, units

def create\_and\_add\_helpers(table=pd.DataFrame(), units=dict):

if units is None:

units = {}

for key in units.keys():

table[key] = "-"

for key, value in units.items():

for i in range(len(units[key])):

terminal = units[key][i]

units[key][i] = terminal + "h"

helper = units[key][i]

if helper not in table.columns:

table[helper] = "N"

non\_terminal = terminal.lower()

table.loc[table['F'] == non\_terminal, helper] = terminal

table.loc[table['F'] == non\_terminal, key] = helper

table.pop("N")

table["N"] = "-"

return table, units

def draw\_graph(table=pd.DataFrame()):

#Создаём граф и заполняем его вершинами названий колонок (кроме "F")

column\_names\_list = table.columns.to\_list()[1:]

grammar\_graph = nx.DiGraph()

grammar\_graph.add\_nodes\_from(column\_names\_list)

for i in range(table.shape[0]):

for j in range(1, table.shape[1]):

source = table.iloc[i, j]

if source == "-":

continue

else:

grammar\_graph.add\_edge(table.columns[j], table.iloc[i, j], type=table.iloc[i, 0])

#Рисуем граф в MatPlotLib

pos = nx.spring\_layout(grammar\_graph)

nx.draw(grammar\_graph,

pos,

with\_labels=True,

node\_color='lightblue',

node\_size=1000,

font\_size=10)

edge\_labels = nx.get\_edge\_attributes(grammar\_graph, 'type')

nx.draw\_networkx\_edge\_labels(grammar\_graph, pos, edge\_labels=edge\_labels)

plt.show()

def format\_table(table):

#Формирование любого вида таблицы (даже строки) с "шапкой"

return tabulate(table, headers="keys", tablefmt="grid")

def format\_dict(d):

# Форматируем вывод

formatted\_output = []

for key, values in d.items():

formatted\_output.append(f"{key} = ({', '.join(values)})")

return "\n".join(formatted\_output)