Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий
институт
Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №4

Синтаксический анализ контекстно-свободных языков
тема

Преподаватель _____ А. С. Кузнецов
подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031830504 _____ Е.В. Железкин
номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2021

1 Цель работы

Исследование свойств универсальных алгоритмов синтаксического анализа контекстно-свободных языков.

2 Задача работы

Необходимо с использованием системы JFLAP, построить грамматику, определяющую заданный язык для анализа его методом Кока-Янгера-Касами, или формально доказать невозможность этого.

Вариант 1. Язык оператора присваивания, в правой части которого задано арифметическое выражение. Элементами выражений являются целочисленные константы в двоичной системе счисления, имена переменных из одного символа (от a до f), знаки операций и скобки для изменения порядка вычисления подвыражений. Операции (в сторону уменьшения приоритета): унарный минус, мультипликативные, аддитивные, присваивание.

2.1 Инструкция по запуску

Необходимо установить *python*, желательно версии 3 и выше (выполнено на версии 3.8.2):

- Страница загрузки для Windows: https://www.python.org/downloads/
- Для Linux есть несколько способов, один из них инструмент apt-get:

  $ sudo apt-get update

  $ sudo apt-get install python3.8
- Или загрузить, распаковать и установить образ:

  $ wget https://www.python.org/ftp/python/3.8.2/Python-3.8.2.tgz

  $ tar -xvf Python-3.8.2.tgz

Для следующего шага понадобится компилятор gcc, но, думаю, это не проблема. Переходим в распакованную папку и собираем+устанавливаем:

$ cd Python-3.8.2

$ ./configure

$ make

$ sudo make install

Далее на любой из двух систем перейти в каталог с распакованным архивом Lab_4 и выполнить:

$ python PyPDA/main.py

($ python main.py; если из папки *PyPDA)*

Ввести тестовую цепочку, нажать «ввод»

Для запуска тестов:

Установить библиотеку pytest:

$ pip install pytest

Запуск:

$ pytest CYK_tests.py

3 Ход работы

Был реализован МПА (согласно заданию первого варианта), допускающий входную цепочку по заключительному состоянию:

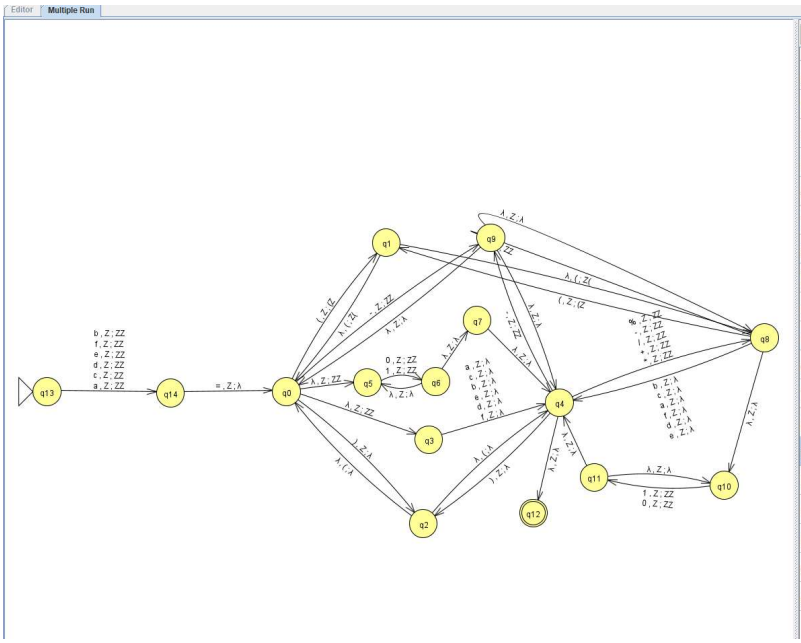

Table Text Size

Input	Result
a=a*b	Accept
b=(a*1010)	Accept
c=a*11111111111111...	Accept
d=(b*1)*c	Accept
e=a*(d*e)*f	Accept
f=a*(a)*a	Accept
a=(a*-1010)	Accept
b=a*-11111111111111...	Accept
c=(b*1)*c	Accept
d=1+((a-(b*c)/f)	Accept
e=a*-(d*e)*f	Accept
f=a*((-a))*a	Accept
a=a--b	Accept
reject_below	Reject
b=a**b	Reject
c=abc	Reject
d=a*-((de)*f	Reject
e=a*(-a))*a	Reject
a+b	Reject

Рисунок 1 – Полученный МПА (PDA.jff)

Предполагалось конвертировать МПА в грамматику и привести её к НФХ, но какое-то ограничение системы JFLAP не позволило этого сделать (предположительно недостаточно нетерминалов латинского алфавита для конвертации).

LHS		RHS
S	→	AB
A	→	a
A	→	b
A	→	c
A	→	d
A	→	e
A	→	f
B	→	CD
C	→	=
D	→	EF
E	→	{
F	→	DG
G	→	}
D	→	a
D	→	b
D	→	c
D	→	d
D	→	e
D	→	f
D	→	HI
H	→	0
H	→	1
I	→	HI
I	→	HK
O	→	*
O	→	/
O	→	%
O	→	+
O	→	-
D	→	JD
J	→	-

Рисунок 2 – Грамматика в НФХ, полученная вручную (файл grammar.jff)

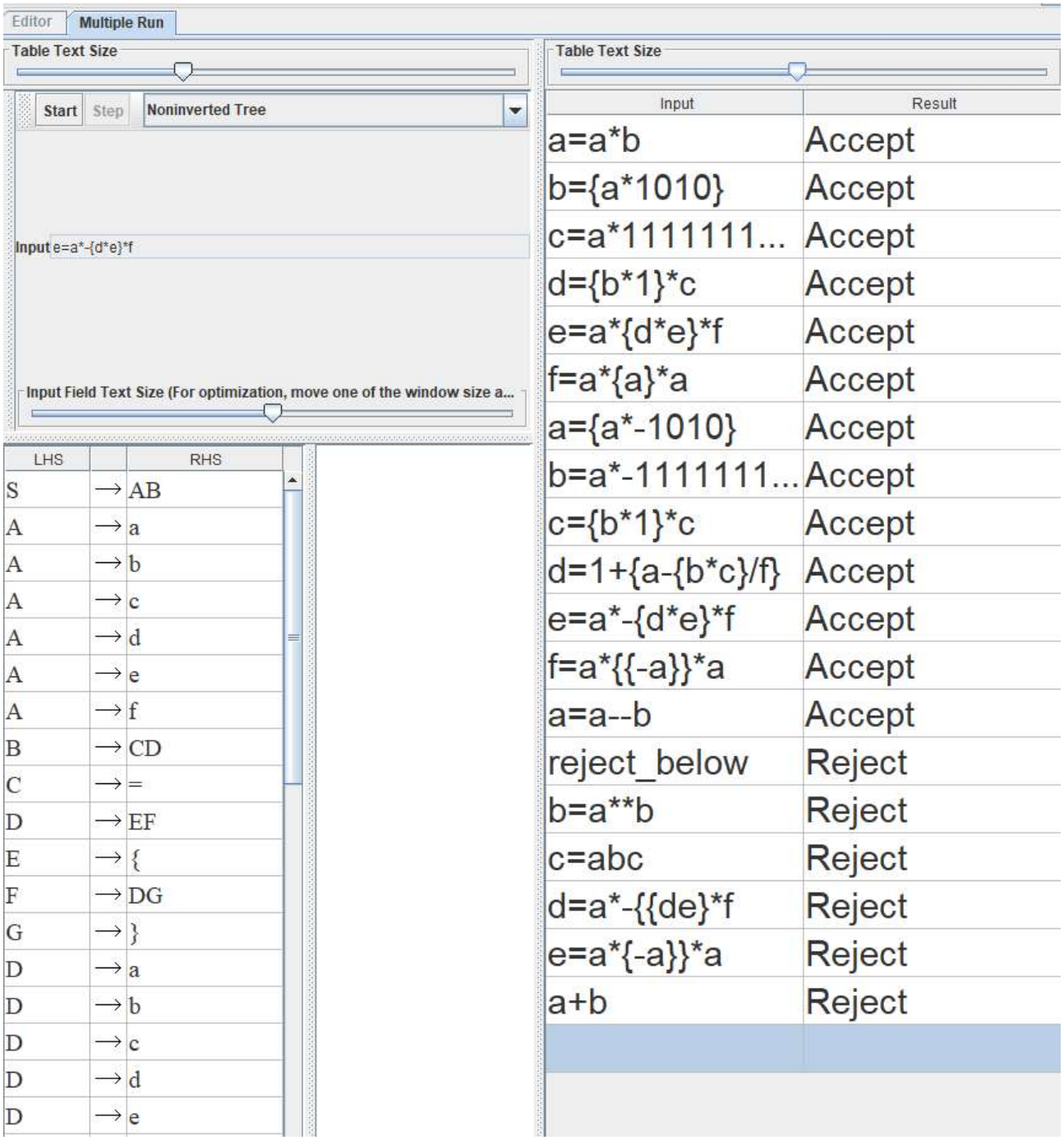

Рисунок 3 – Тестирование полученной грамматики с помощью алгоритма Кока-Янгера-Касами

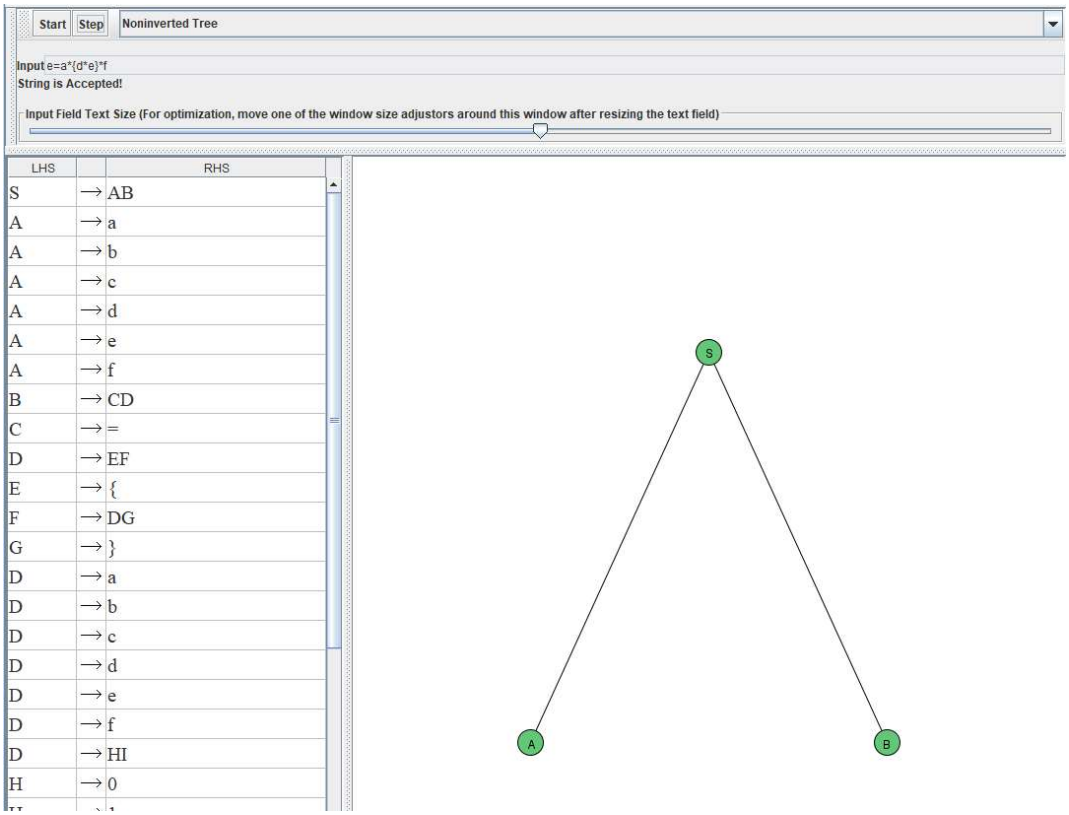

Рисунок 4 – Перехват экрана тестовых цепочек методом CYK

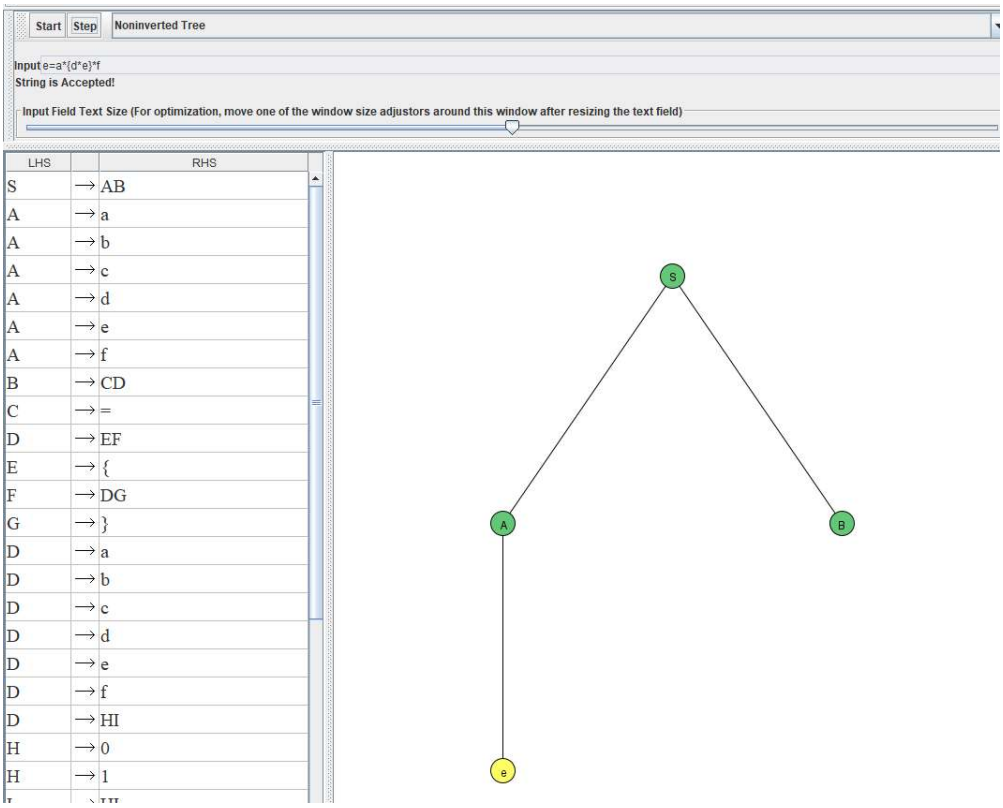

Рисунок 5 – Перехват экрана тестовых цепочек методом CYK

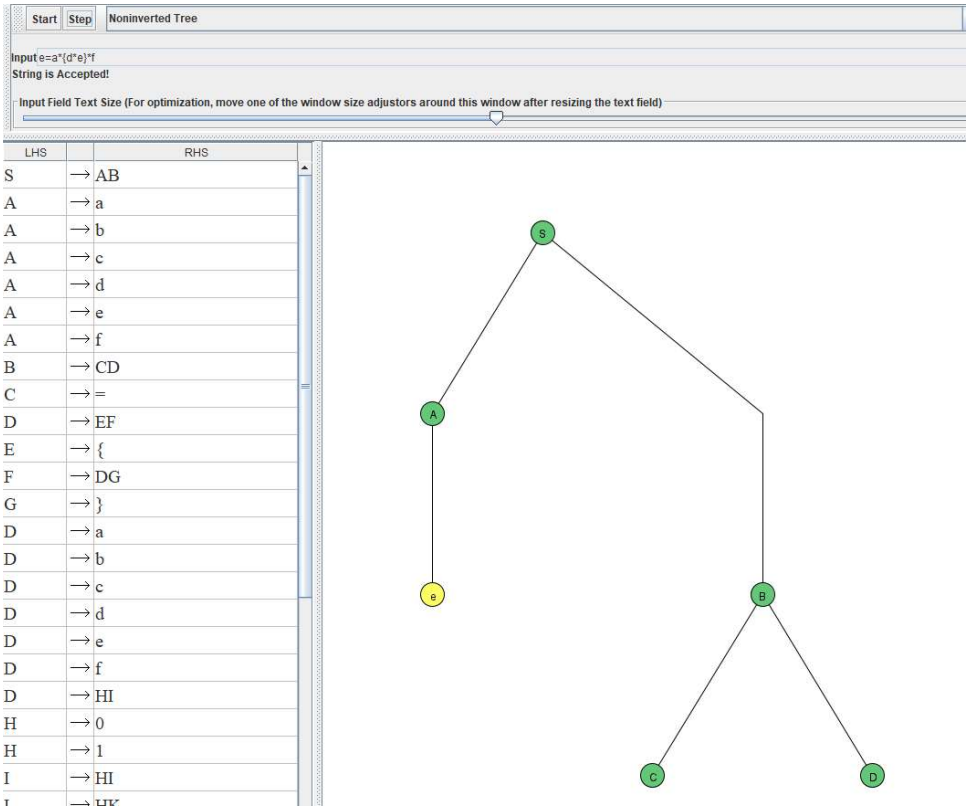

Рисунок 6 – Перехват экрана тестовых цепочек методом CYK

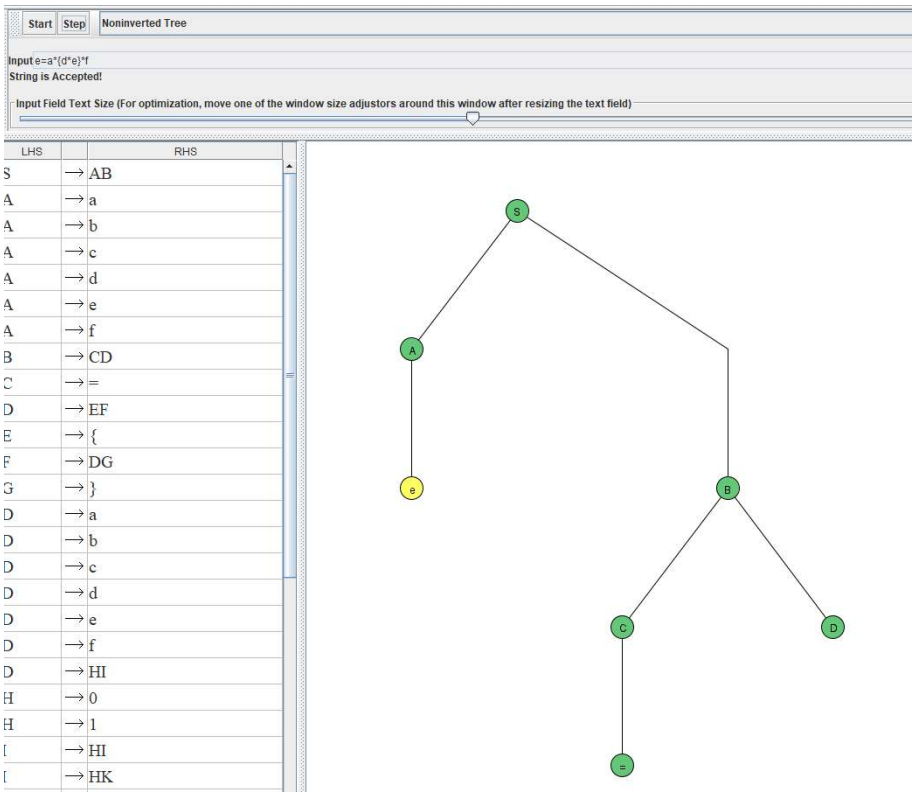

Рисунок 7 – Перехват экрана тестовых цепочек методом CYK

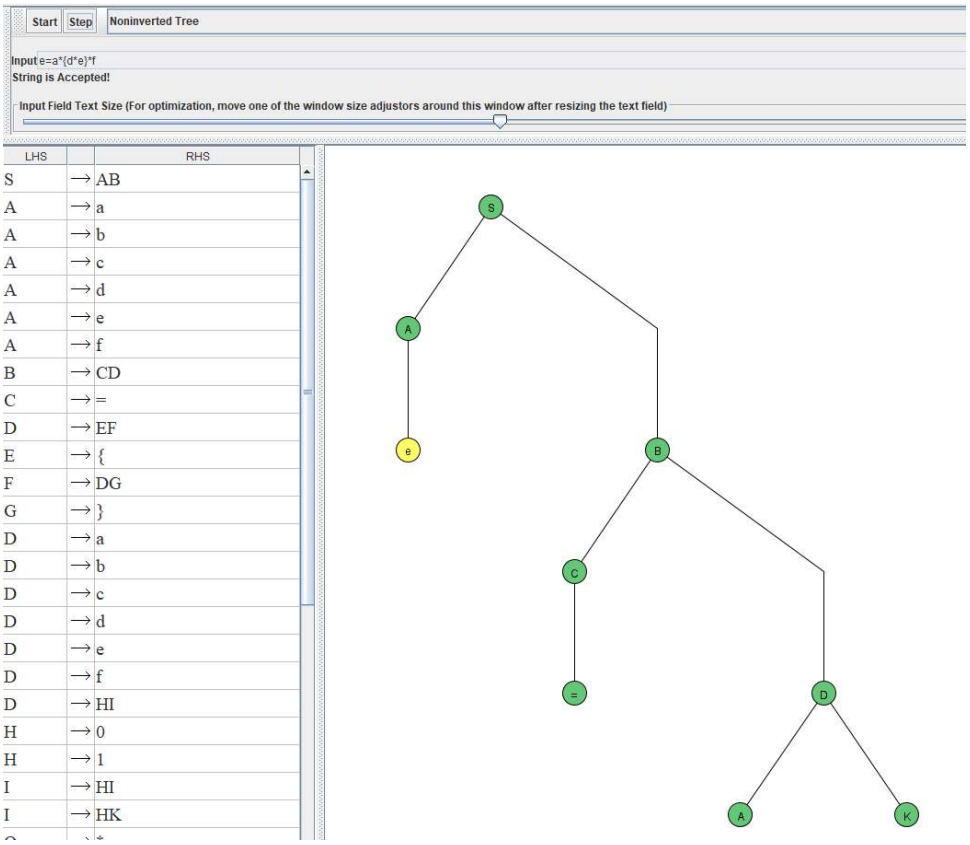

Рисунок 8 – Перехват экрана тестовых цепочек методом CYK

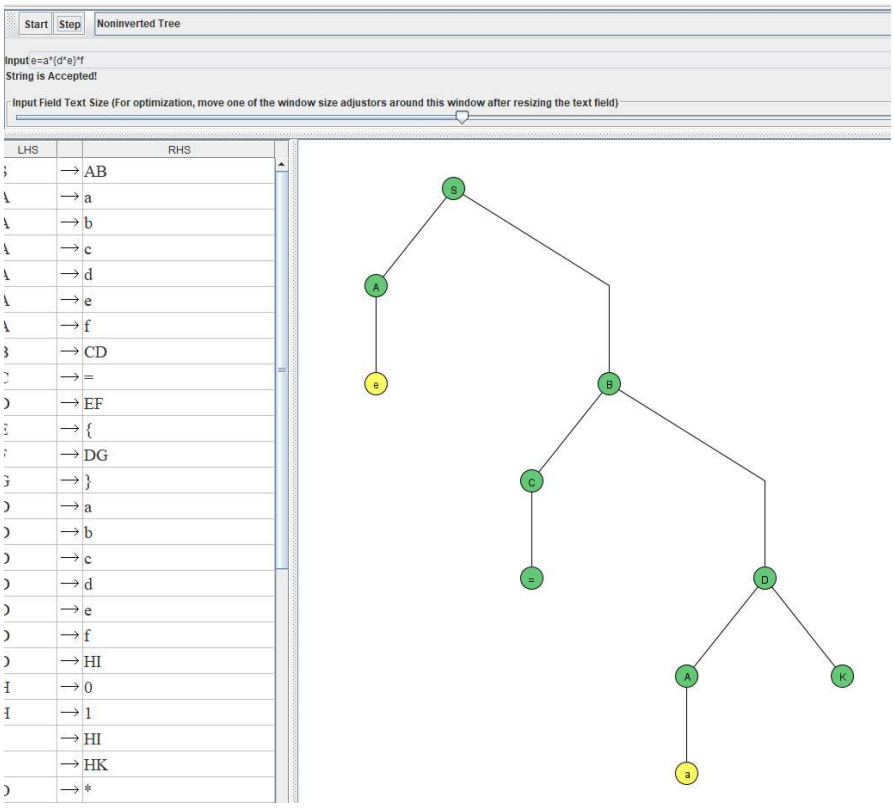

Рисунок 9 – Перехват экрана тестовых цепочек методом CYK

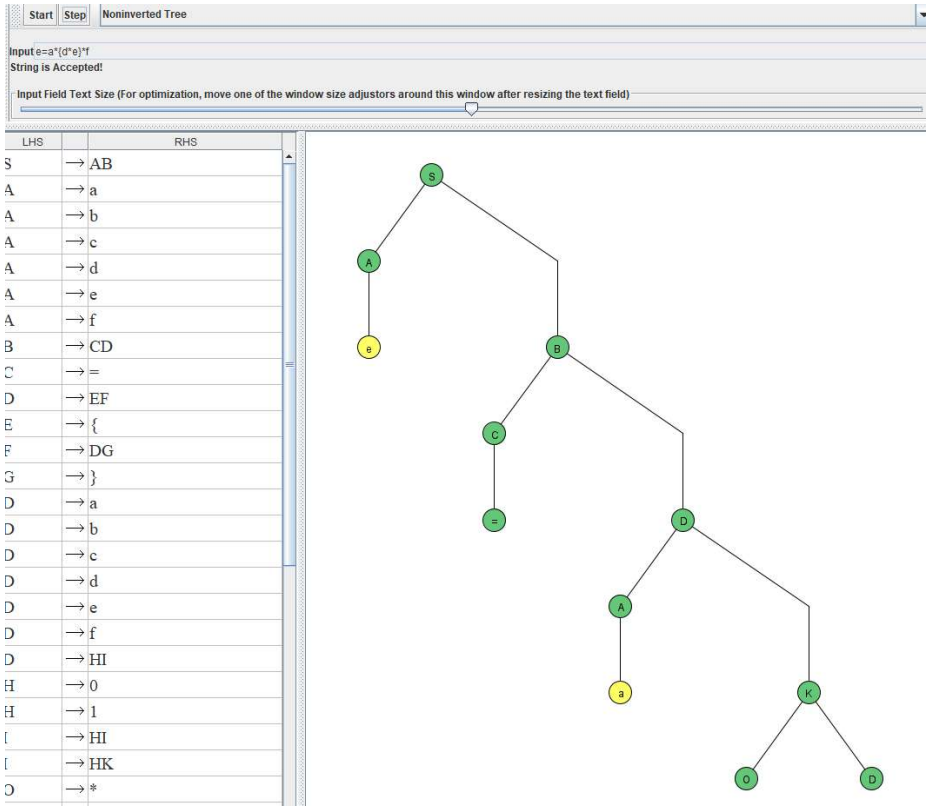

Рисунок 10 – Перехват экрана тестовых цепочек методом CYK

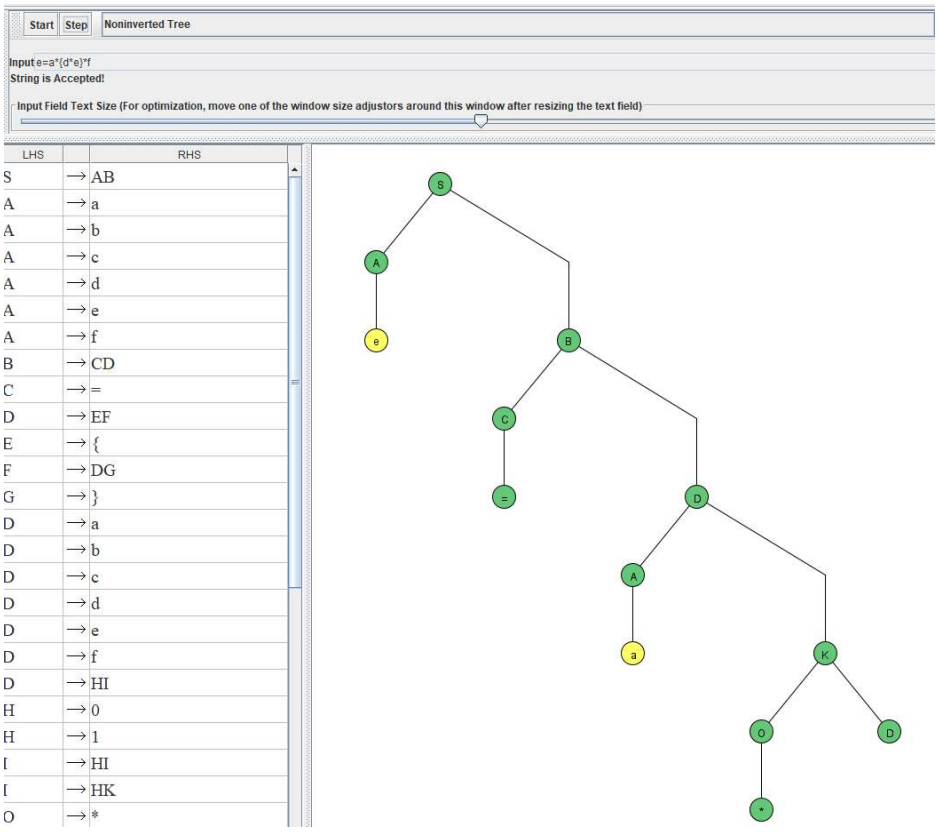

Рисунок 11 – Перехват экрана тестовых цепочек методом CYK

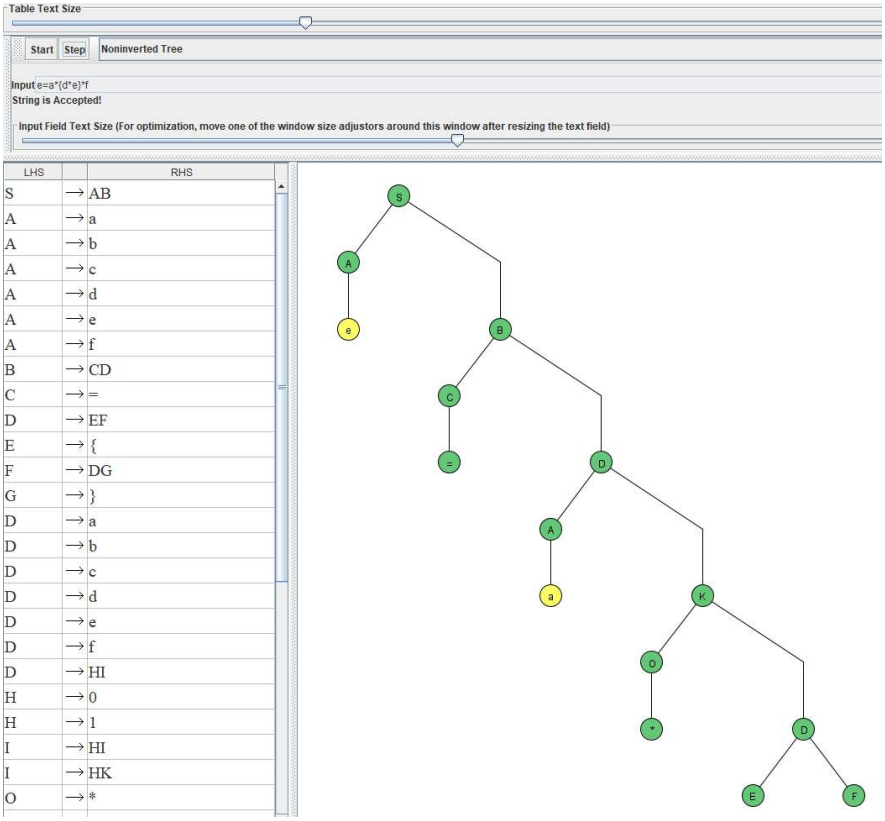

Рисунок 12 – Перехват экрана тестовых цепочек методом CYK

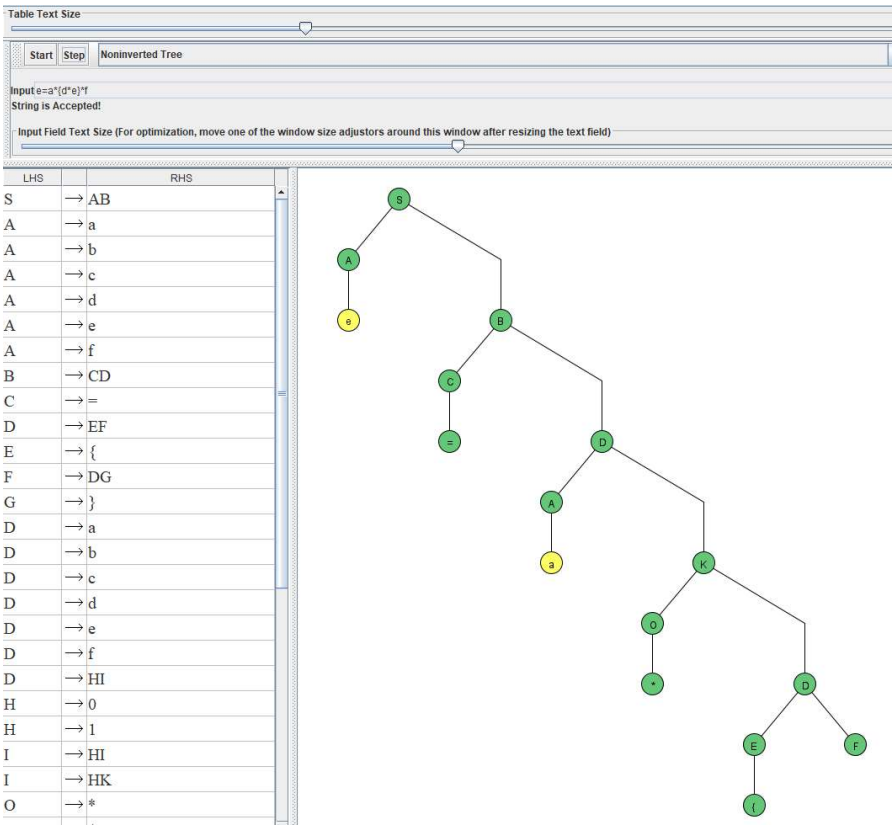

Рисунок 13 – Перехват экрана тестовых цепочек методом CYK

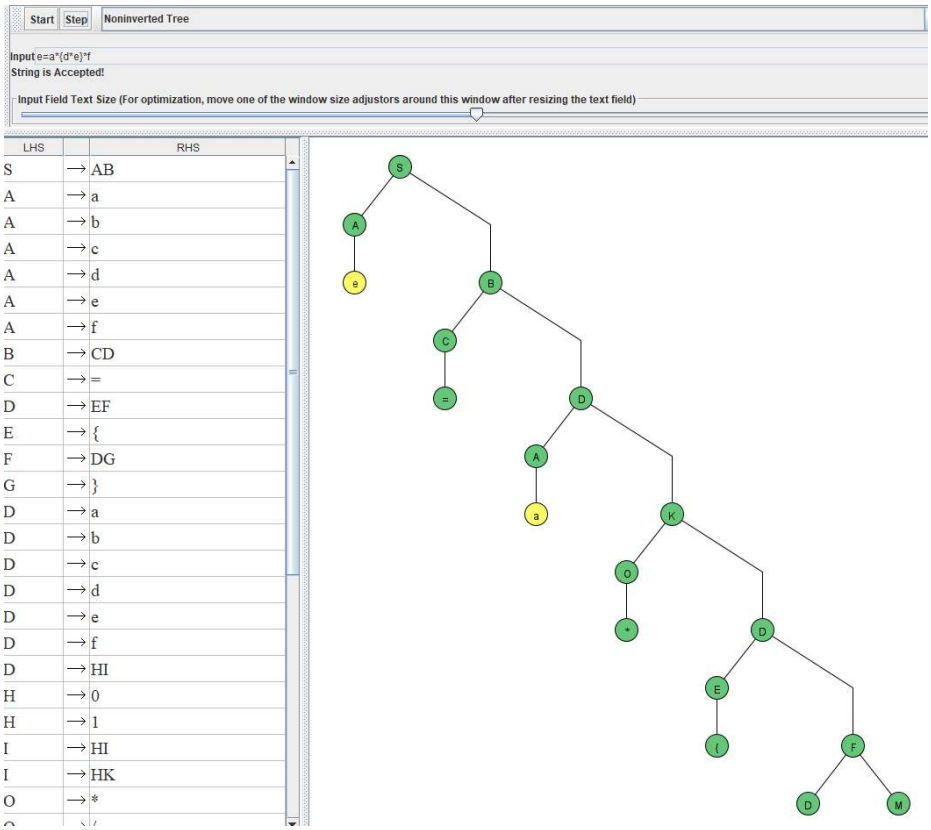

Рисунок 14 – Перехват экрана тестовых цепочек методом CYK

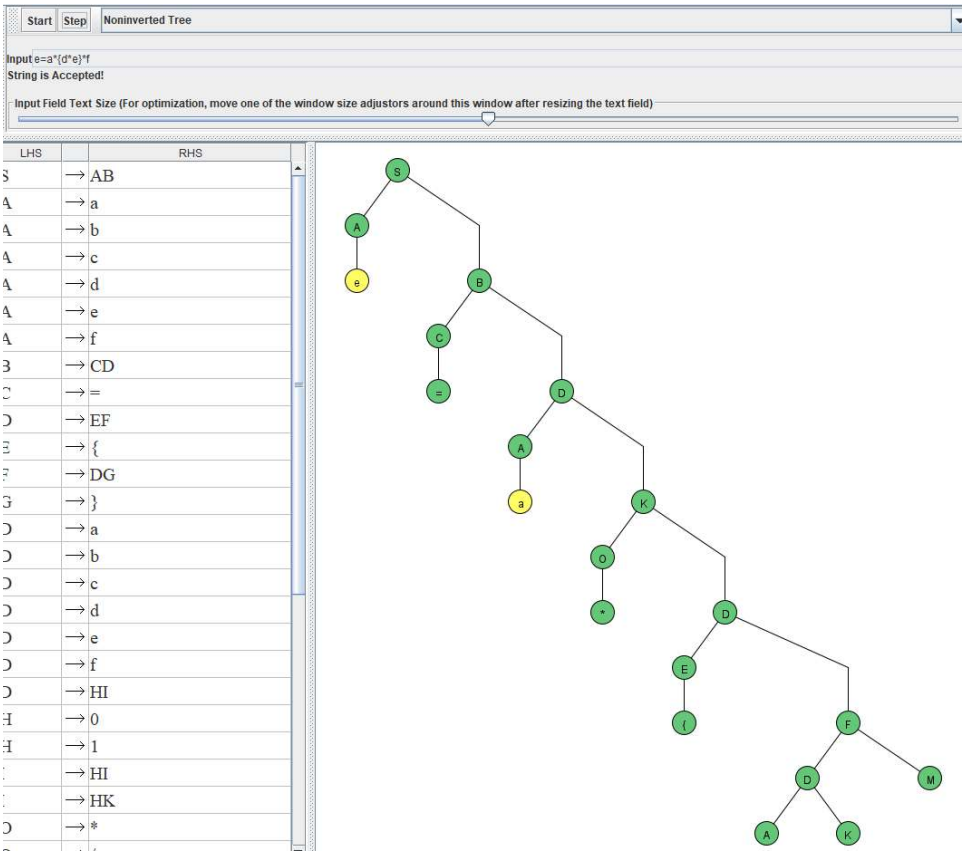

Рисунок 15 – Перехват экрана тестовых цепочек методом CYK

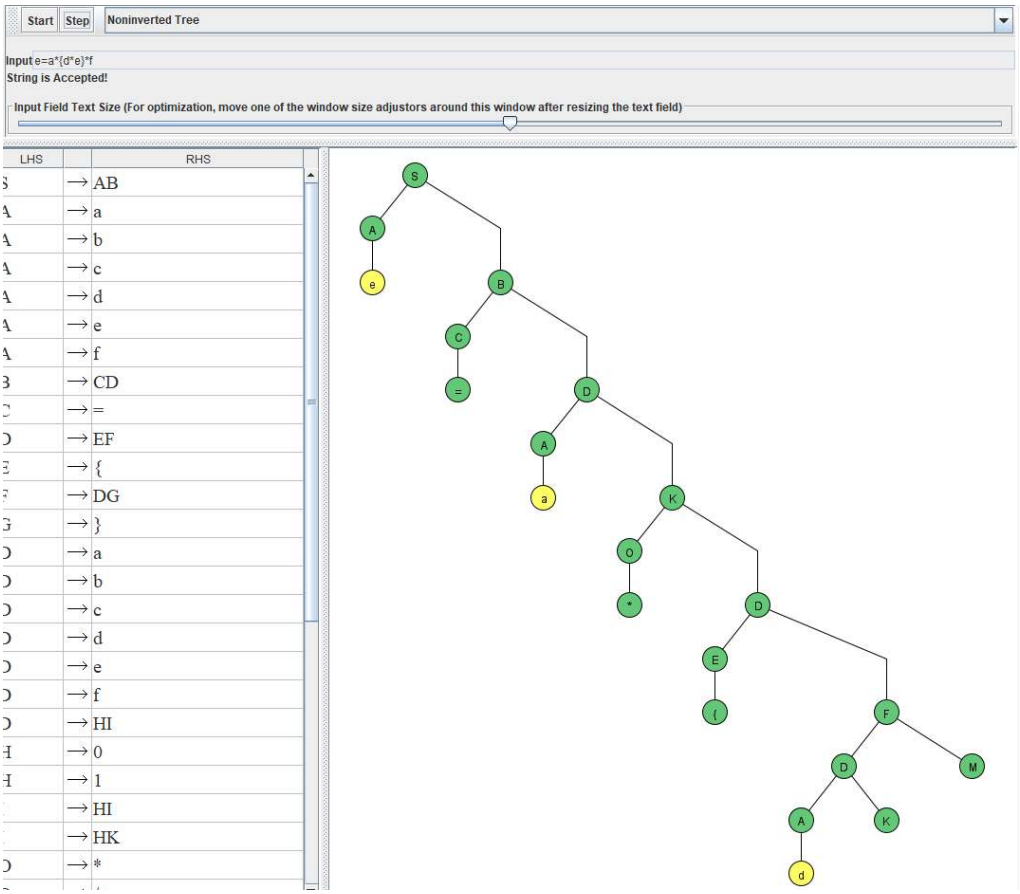

Рисунок 16 – Перехват экрана тестовых цепочек методом CYK

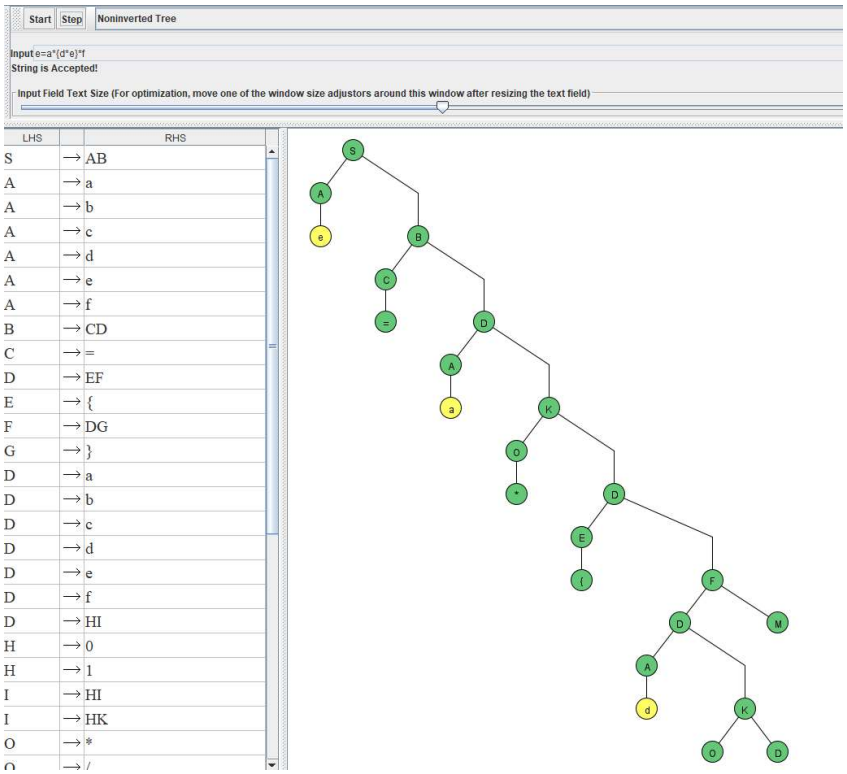

Рисунок 17 – Перехват экрана тестовых цепочек методом CYK

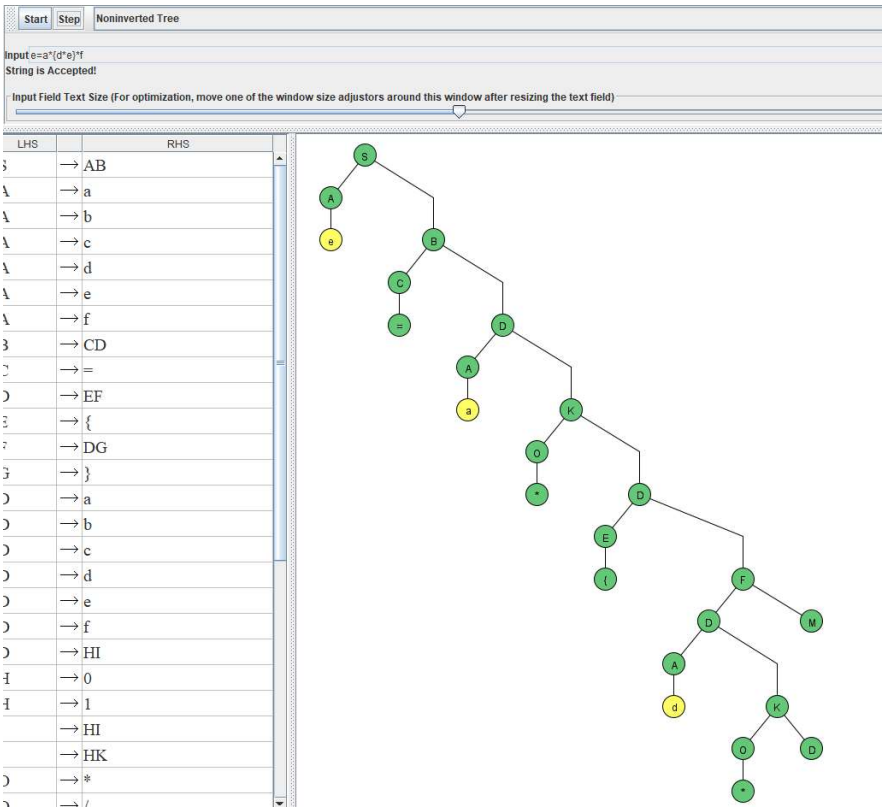

Рисунок 18 – Перехват экрана тестовых цепочек методом CYK

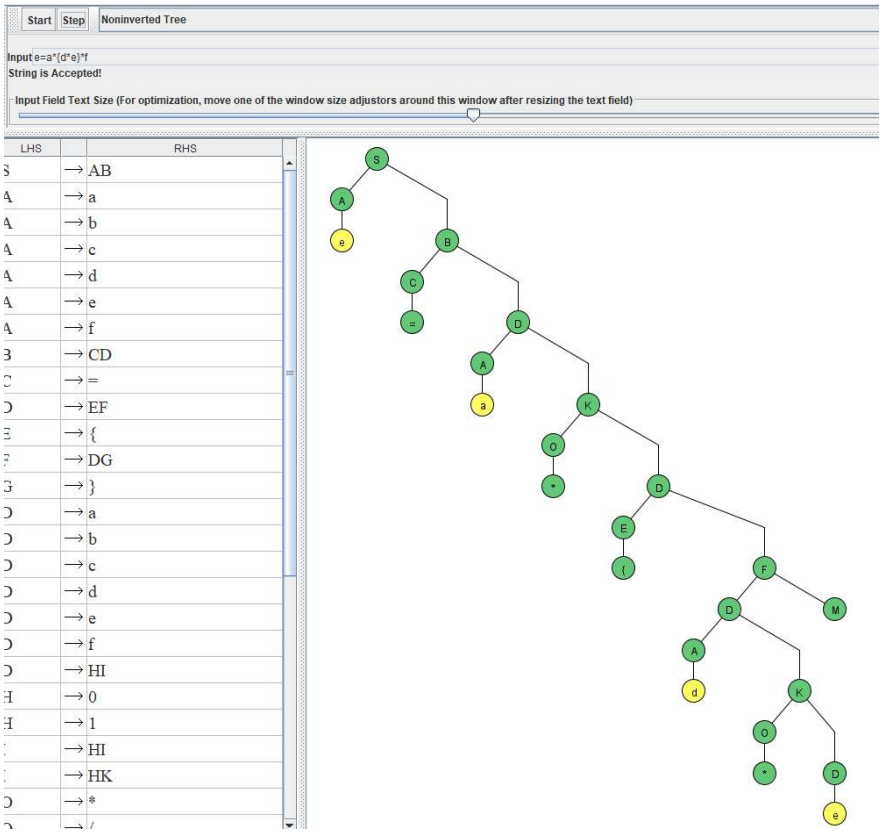

Рисунок 19 – Перехват экрана тестовых цепочек методом CYK

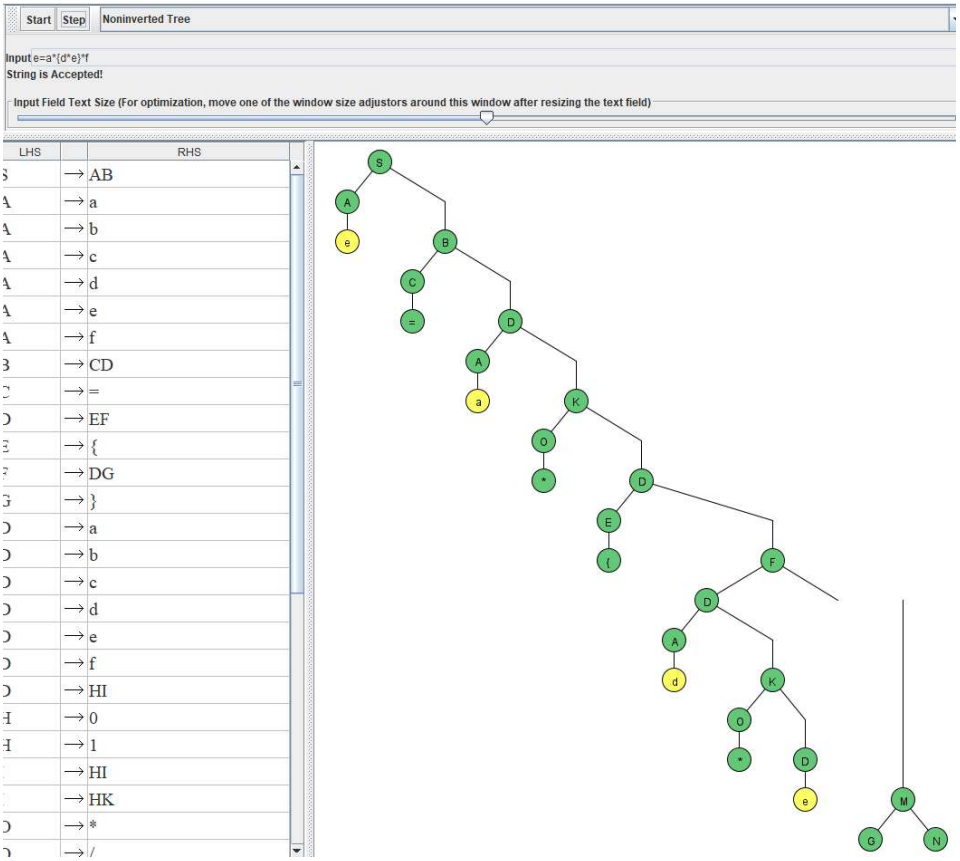

Рисунок 20 – Перехват экрана тестовых цепочек методом CYK

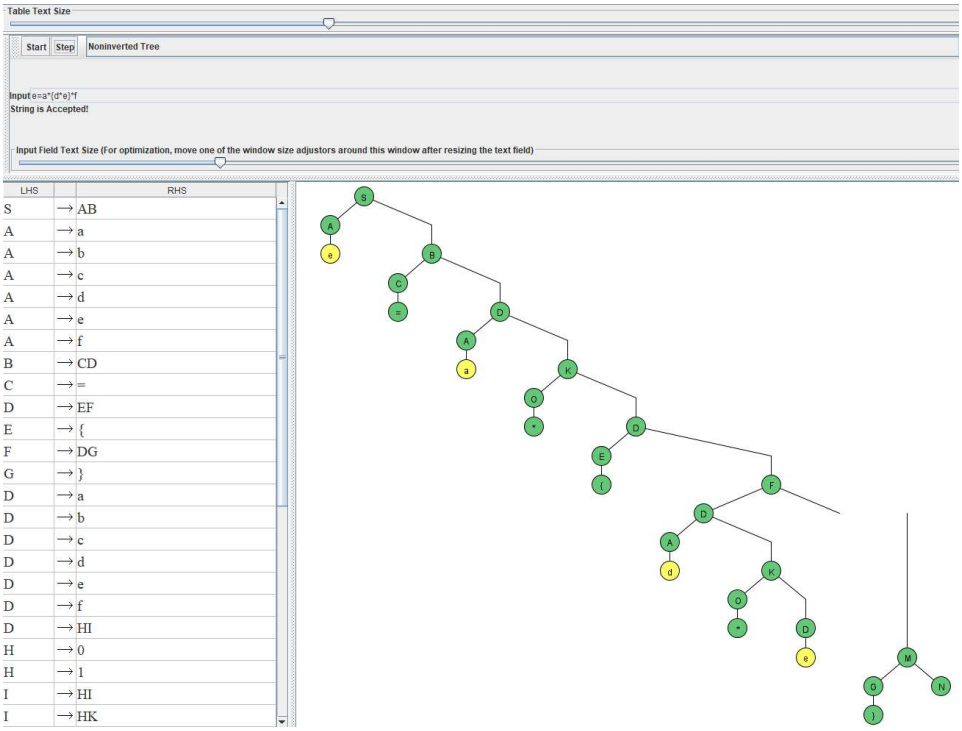

Рисунок 21 – Перехват экрана тестовых цепочек методом CYK

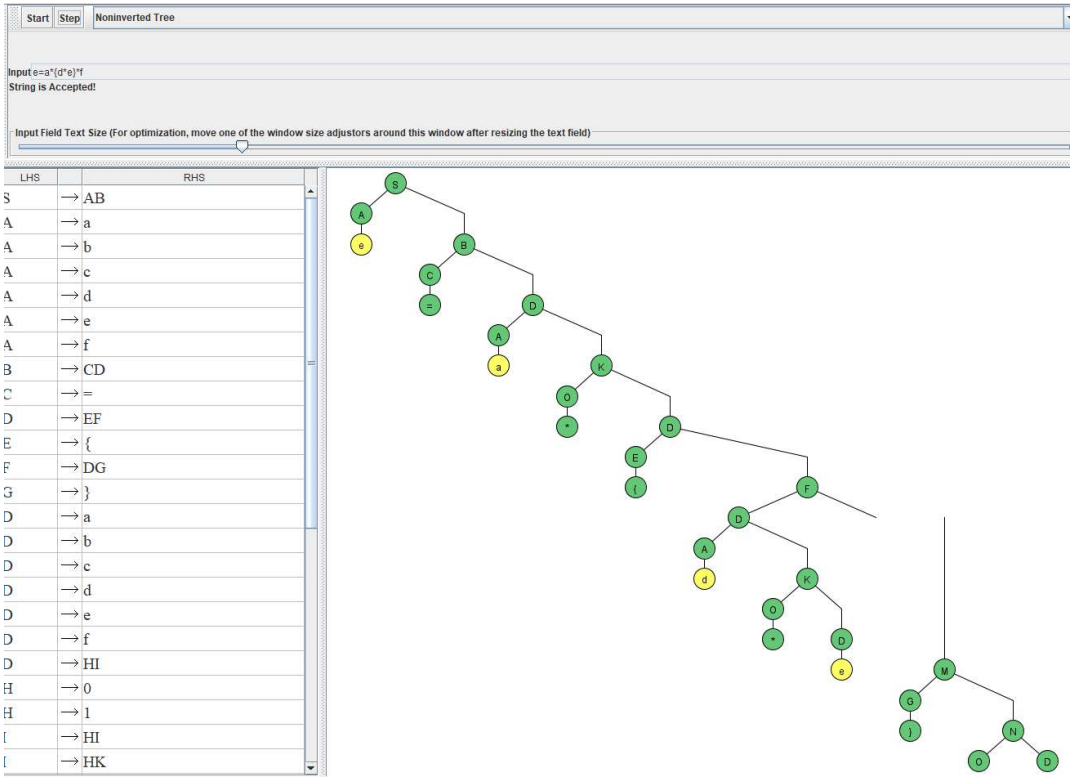

Рисунок 22 – Перехват экрана тестовых цепочек методом CYK

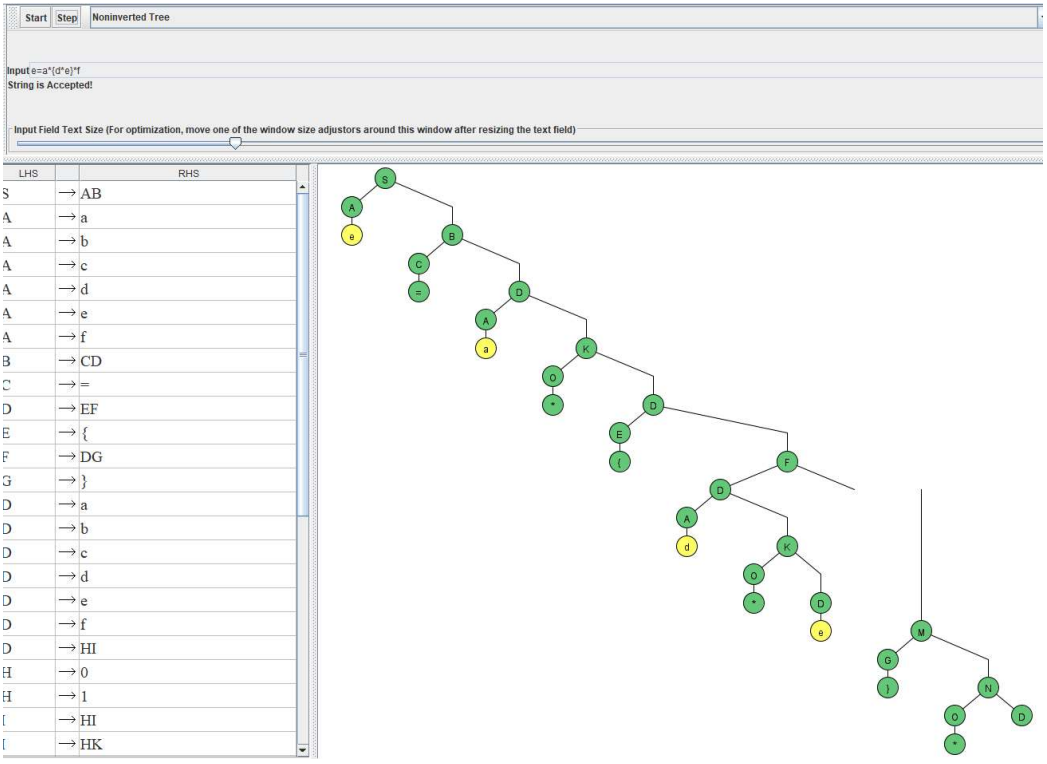

Рисунок 23 – Перехват экрана тестовых цепочек методом CYK

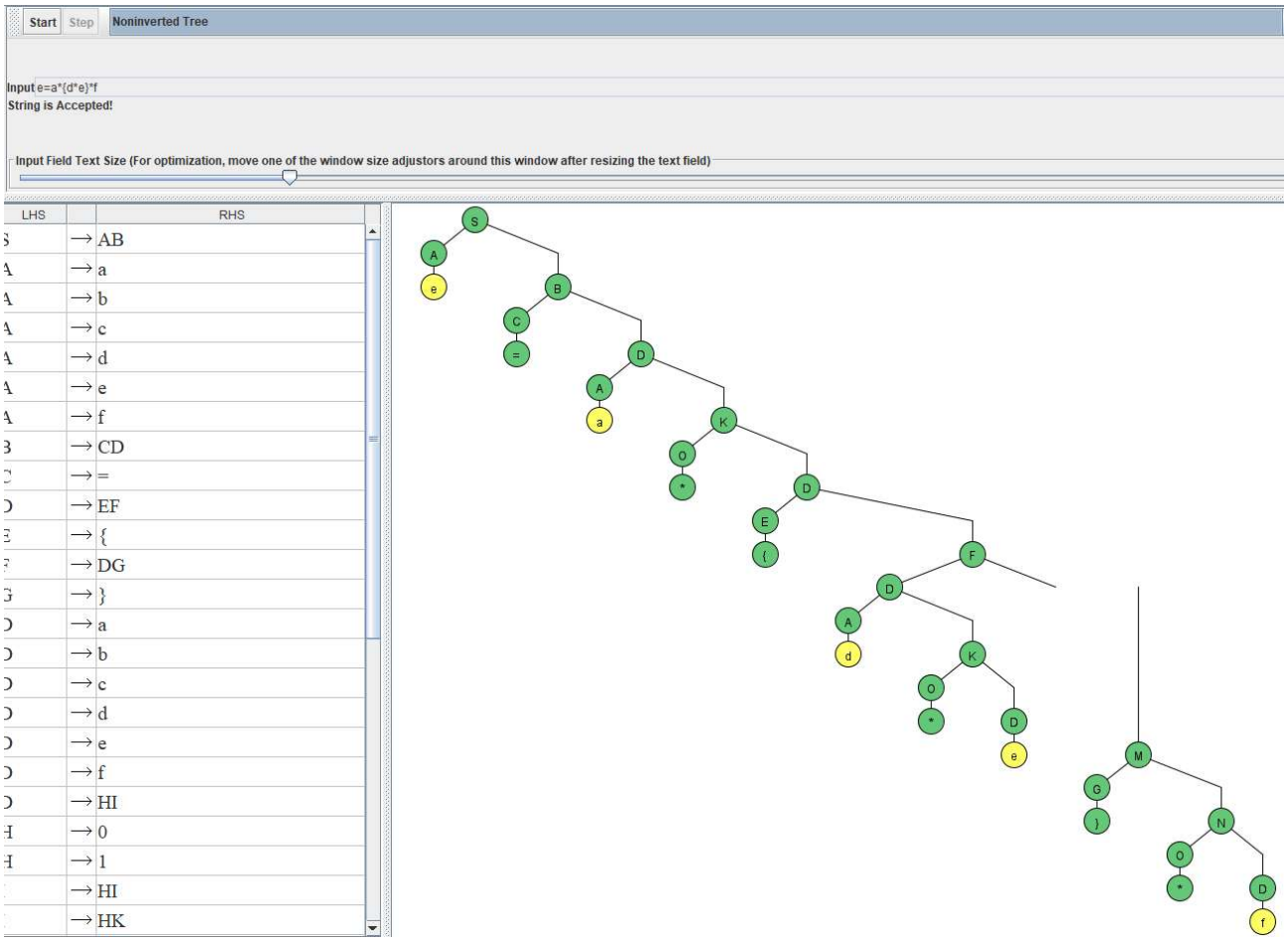

Рисунок 24 – Перехват экрана тестовых цепочек методом CYK

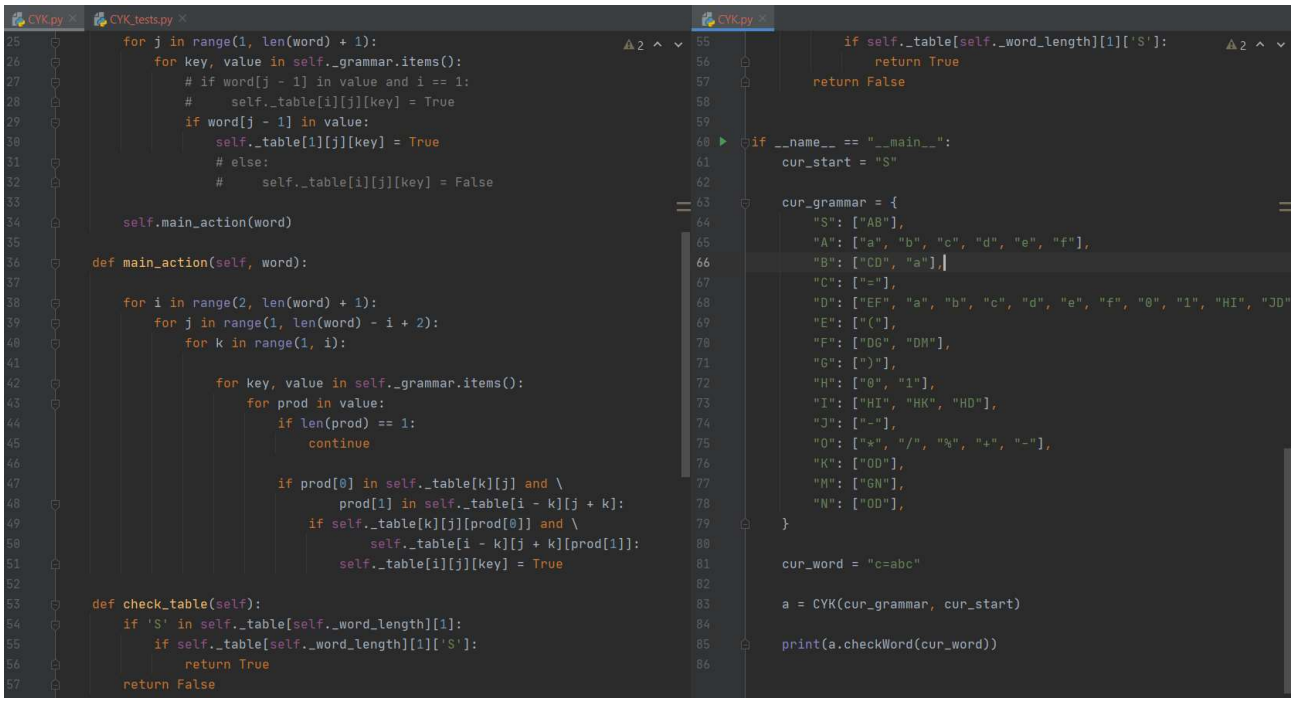

Рисунок 25 – Программная реализация алгоритма CYK (файл CYK.py)

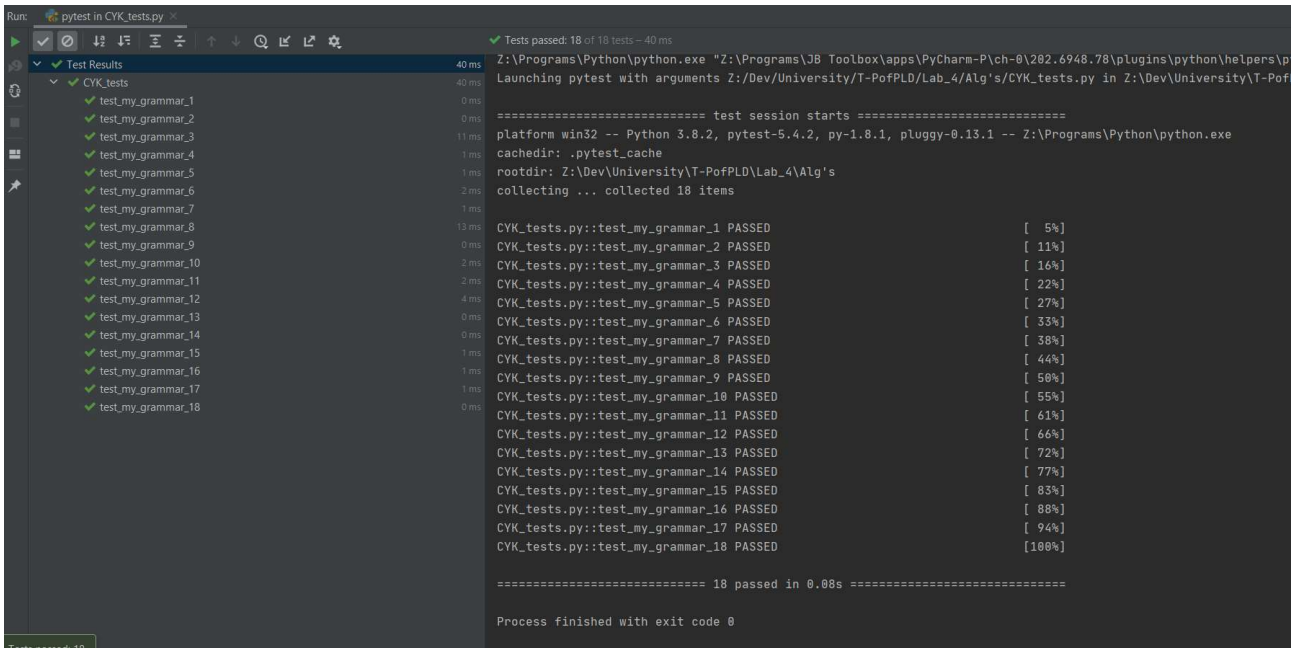

Рисунок 26 – Тестирование реализации (те же входные цепочки, что и для грамматики)

Для запуска тестов необходимо установить библиотеку *pytest* (pip install pytest) для python и выполнить *pytest CYK_tests.py*

4 Вывод

В ходе данной лабораторной работы были исследованы свойства универсальных алгоритмов синтаксического анализа контекстно-свободных языков.

ПРИЛОЖЕНИЕ А

Листинг 1 – файл CYK.py

```
UNKNOWN_SYMBOL_ERR = 0
NOT_REACHED_FINAL_STATE = 1
REACHED_FINAL_STATE = 2


class CYK:
    _word_length = 0

    def __init__(self, grammar, start):
        self._grammar = grammar
        self._start = start
        self._table = [[{}]]  # Формат хранения:
[индекс][индекс][ключ(нетерминальный символ)-значение]

    def checkWord(self, word):
        self._word_length = len(word)
        self.init_step(word)
        return self.check_table()

    def init_step(self, word):
        self._table = [[{}  # r
                        for j in range(len(word) + 1)]  # n
                       for k in range(len(word) + 1)]  # n

        # for i in range(1, len(word) + 1):
        for j in range(1, len(word) + 1):
            for key, value in self._grammar.items():
                # if word[j - 1] in value and i == 1:
                #     self._table[i][j][key] = True
                if word[j - 1] in value:
                    self._table[1][j][key] = True
                    # else:
                    #     self._table[i][j][key] = False

        self.main_action(word)

    def main_action(self, word):

        for i in range(2, len(word) + 1):
            for j in range(1, len(word) - i + 2):
                for k in range(1, i):

                    for key, value in self._grammar.items():
                        for prod in value:
                            if len(prod) == 1:
                                continue
```

```
                                if prod[0] in self._table[k][j] and \
                                        prod[1] in self._table[i -
k][j + k]:
                                    if self._table[k][j][prod[0]] and
\
                                            self._table[i - k][j +
k][prod[1]]:
                                        self._table[i][j][key] = True

    def check_table(self):
        if 'S' in self._table[self._word_length][1]:
            if self._table[self._word_length][1]['S']:
                return True
        return False


if __name__ == "__main__":
    cur_start = "S"

    cur_grammar = {
        "S": ["AB"],
        "A": ["a", "b", "c", "d", "e", "f"],
        "B": ["CD", "a"],
        "C": ["="],
        "D": ["EF", "a", "b", "c", "d", "e", "f", "0", "1", "HI",
"JD", "AK", "HK"],
        "E": ["("],
        "F": ["DG", "DM"],
        "G": [")"],
        "H": ["0", "1"],
        "I": ["HI", "HK", "HD"],
        "J": ["-"],
        "O": ["*", "/", "%", "+", "-"],
        "K": ["OD"],
        "M": ["GN"],
        "N": ["OD"],
    }

    cur_word = "c=abc"

    a = CYK(cur_grammar, cur_start)

    print(a.checkWord(cur_word))
```

Листинг 2 – файл CYK_tests.py

```
from CYK import CYK

cur_start = "S"
```

```
cur_grammar = {
    cur_start: ["AB"],
    "A": ["a", "b", "c", "d", "e", "f"],
    "B": ["CD", "a"],
    "C": ["="],
    "D": ["EF", "a", "b", "c", "d", "e", "f", "0", "1", "HI",
"JD", "AK", "HK"],
    "E": ["("],
    "F": ["DG", "DM"],
    "G": [")"],
    "H": ["0", "1"],
    "I": ["HI", "HK", "HD"],
    "J": ["-"],
    "O": ["*", "/", "%", "+", "-"],
    "K": ["OD"],
    "M": ["GN"],
    "N": ["OD"],
}

cyk = CYK(cur_grammar, cur_start)


def test_my_grammar_1():

    cur_word = "a=a*b"

    assert cyk.checkWord(cur_word)


def test_my_grammar_2():

    cur_word = "b=(a*1010)"

    assert cyk.checkWord(cur_word)


def test_my_grammar_3():

    cur_word = "c=a*111111111111111*a"

    assert cyk.checkWord(cur_word)


def test_my_grammar_4():

    cur_word = "d=(b*1)*c"

    assert cyk.checkWord(cur_word)


def test_my_grammar_5():
```

```
    cur_word = "e=a*(d*e)*f"
    assert cyk.checkWord(cur_word)

def test_my_grammar_6():
    cur_word = "f=a*(a)*a"
    assert cyk.checkWord(cur_word)

def test_my_grammar_7():
    cur_word = "a=(a*-1010)"
    assert cyk.checkWord(cur_word)

def test_my_grammar_8():
    cur_word = "b=a*-111111111111111*a"
    assert cyk.checkWord(cur_word)

def test_my_grammar_9():
    cur_word = "c=(b*1)*c"
    assert cyk.checkWord(cur_word)

def test_my_grammar_10():
    cur_word = "d=1+(a-(b*c)/f)"
    assert cyk.checkWord(cur_word)

def test_my_grammar_11():
    cur_word = "e=a*-(d*e)*f"
    assert cyk.checkWord(cur_word)

def test_my_grammar_12():
    cur_word = "f=a*((-a))*a"
```

```
    assert cyk.checkWord(cur_word)

def test_my_grammar_13():
    cur_word = "a=a--b"
    assert cyk.checkWord(cur_word)

def test_my_grammar_14():
    cur_word = "b=a**b"
    assert not cyk.checkWord(cur_word)

def test_my_grammar_15():
    cur_word = "c=abc"
    assert not cyk.checkWord(cur_word)

def test_my_grammar_16():
    cur_word = "d=a*-((de)*f"
    assert not cyk.checkWord(cur_word)

def test_my_grammar_17():
    cur_word = "e=a*(-a))*a"
    assert not cyk.checkWord(cur_word)

def test_my_grammar_18():
    cur_word = "a+b"
    assert not cyk.checkWord(cur_word)
```