Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №5**

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

# Цель работы

Исследование контекстно-свободных грамматик и алгоритмов синтаксического анализа контекстно-свободных языков.

# Задача работы

*Часть 1.* Необходимо с использованием системы JFLAP, построить LL(1)-грамматику, описывающую заданный язык, или формально доказать невозможность этого. Полученная грамматика не должна повторять SLR(1)- грамматику, конструируемую в части 3.

*Часть 2.* Предложить программную реализацию метода рекурсивного спуска для распознавания строк заданного языка. Представить формальное доказательство принадлежности к классу LL(1) грамматики, лежащей в основе синтаксического анализа заданного языка. Во всех случаях язык должен состоять из последовательностей выражений. В качестве разделителя может выступать символ новой строки, точка с запятой или любой другой символ, не задействованный в других лексемах. Результатом работы синтаксического анализатора является выдача сообщения «Accepted» или «Rejected».

*Часть 3.* Необходимо с использованием системы JFLAP, построить SLR(1)-грамматику, описывающую заданный язык, или формально доказать невозможность этого. Во всех случаях реализуется язык, состоящий из последовательностей операторов присваивания. В качестве разделителя может выступать символ новой строки, точка с запятой или любой другой символ, не задействованный в прочих лексемах. В качестве L-значения оператора присваивания выступает только имя переменной. В правой части оператора присваивания указывается выражение, элементы которых оговариваются в каждом варианте задания. Полученная грамматика не должна повторять LL(1)-грамматику, конструируемую в части 1.

*Вариант (1, 1, 1)*

*Часть 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.

*Часть 2:* Язык арифметических выражений, элементами которых являются целочисленные константы в двоичной, восьмеричной или десятичной системах счисления, имена переменных из 1-2 символов, знаки операций и скобки для изменения порядка вычисления подвыражений. Операции (в сторону уменьшения приоритета): унарный минус, мультипликативные, аддитивные, присваивание.

*Часть 3:* Элементами арифметического выражения являются целочисленные константы в 2- и 10-чной системах 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.

## Инструкция по запуску

Необходимо установить *python*, желательно версии 3 и выше (выполнено на версии 3.9.4):

* 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*](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*

# Ход работы

*Часть 1*

Реализована LL(1)-грамматика с помощью системы JFLAP:

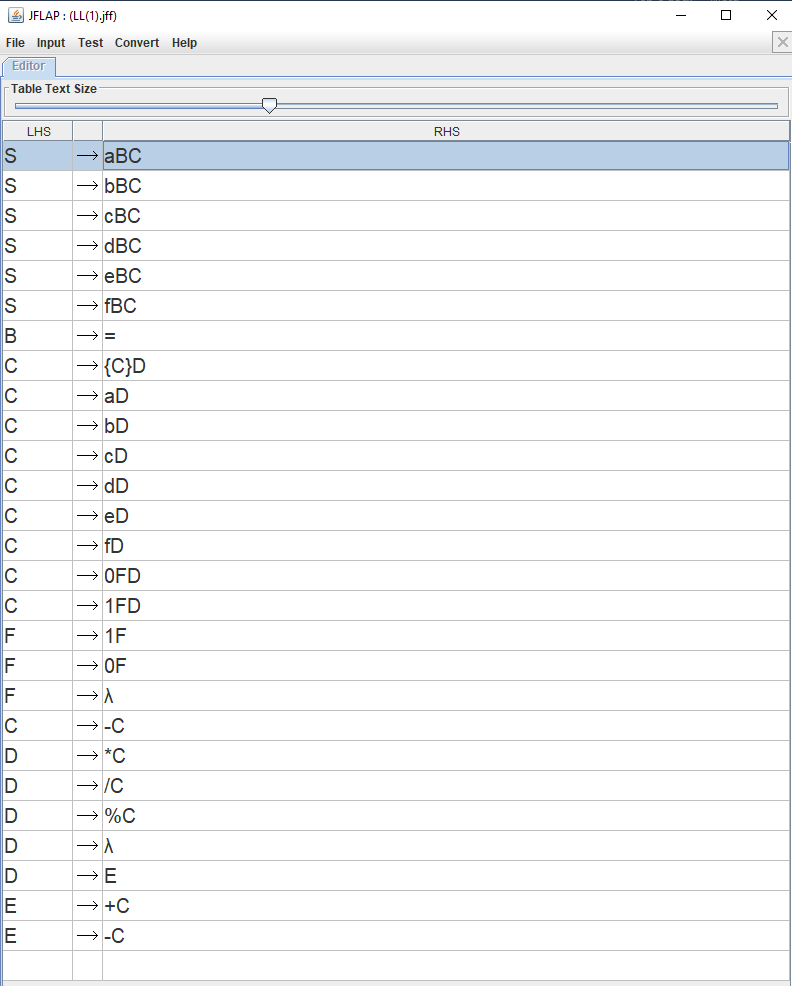


Рисунок 1 – LL(1)-грамматика (файл LL(1).jff)

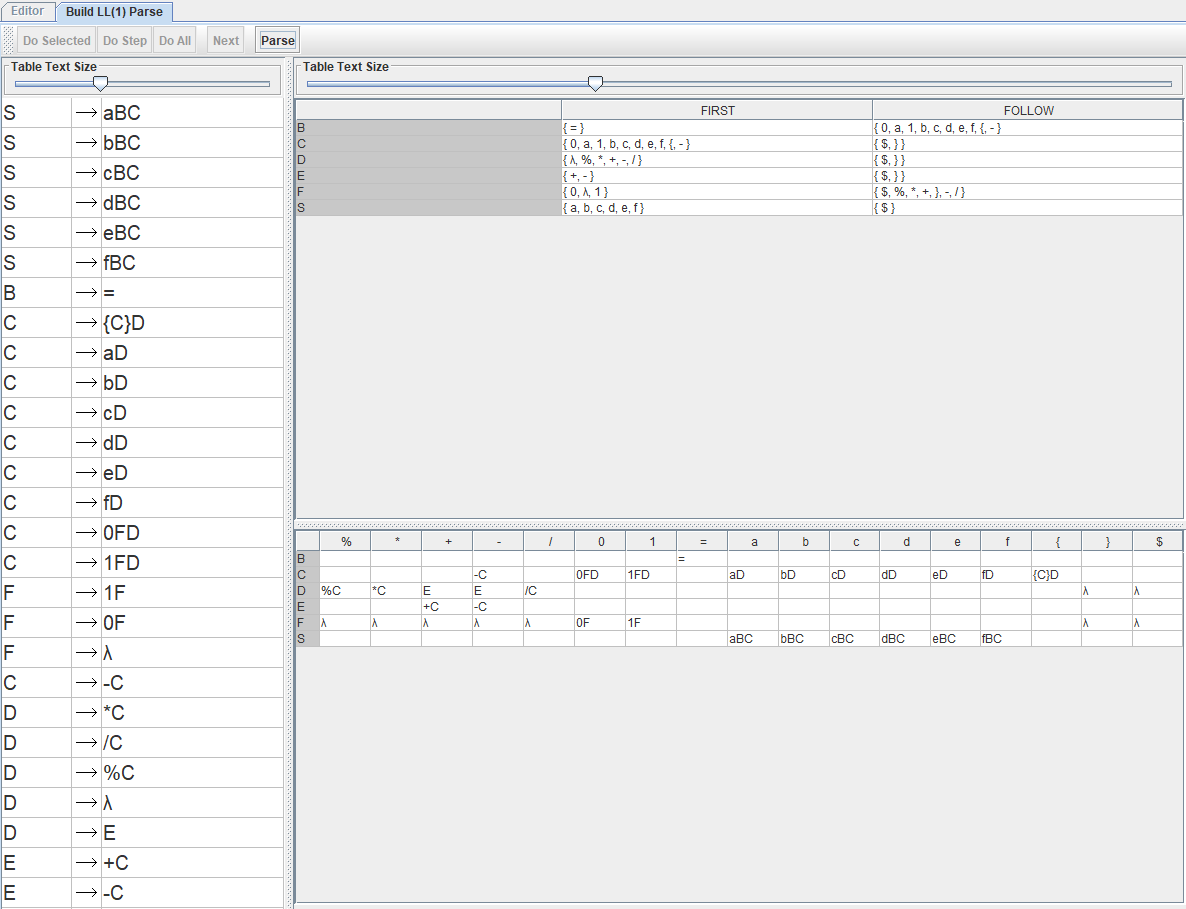


Рисунок 2 – таблица синтаксического LL(1)-анализа

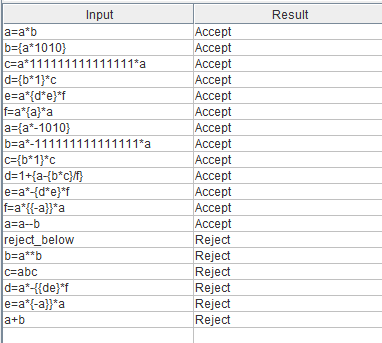


Рисунок 3 – Тестирование полученной грамматики

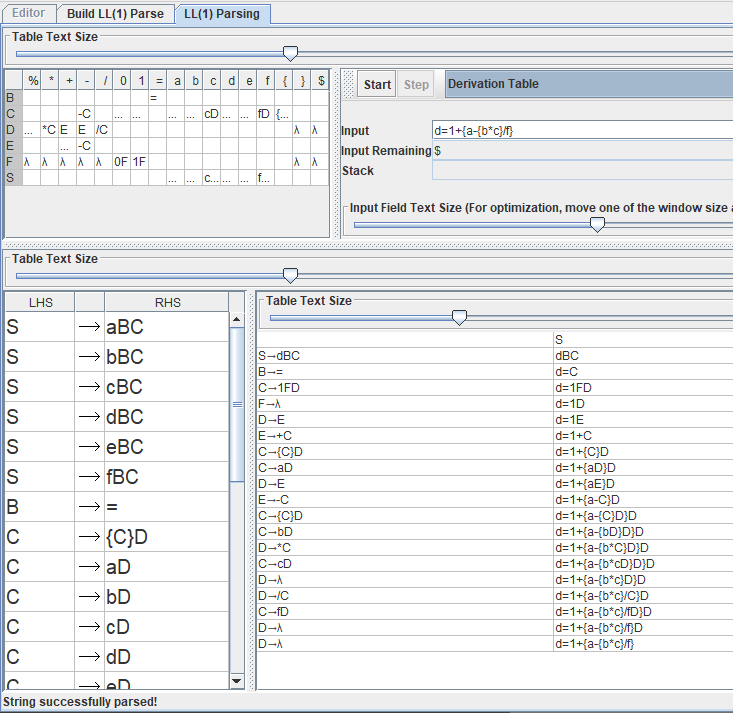


Рисунок 4 – Перехват экрана распознавания

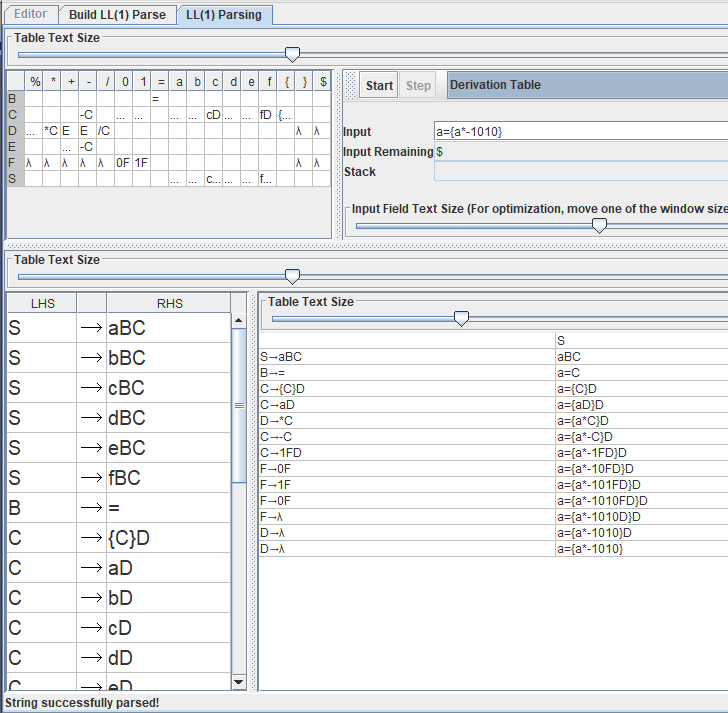


Рисунок 5 – Перехват экрана распознавания

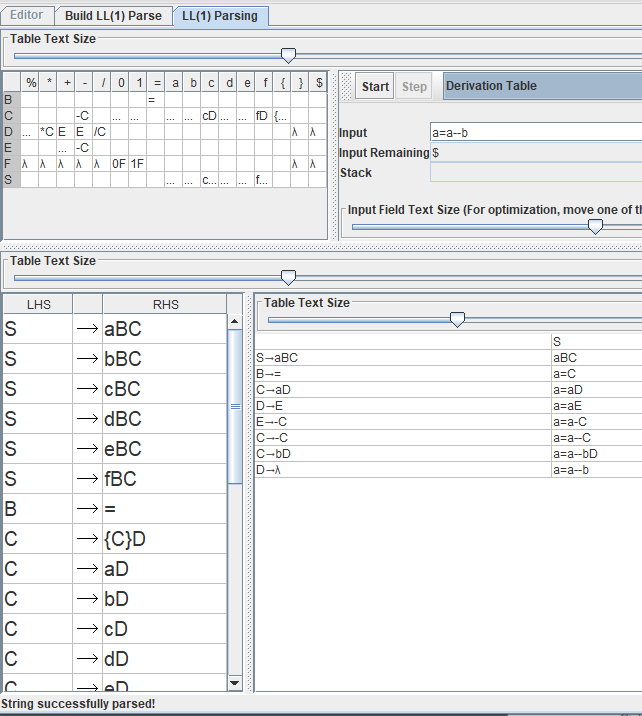


Рисунок 6 – Перехват экрана распознавания

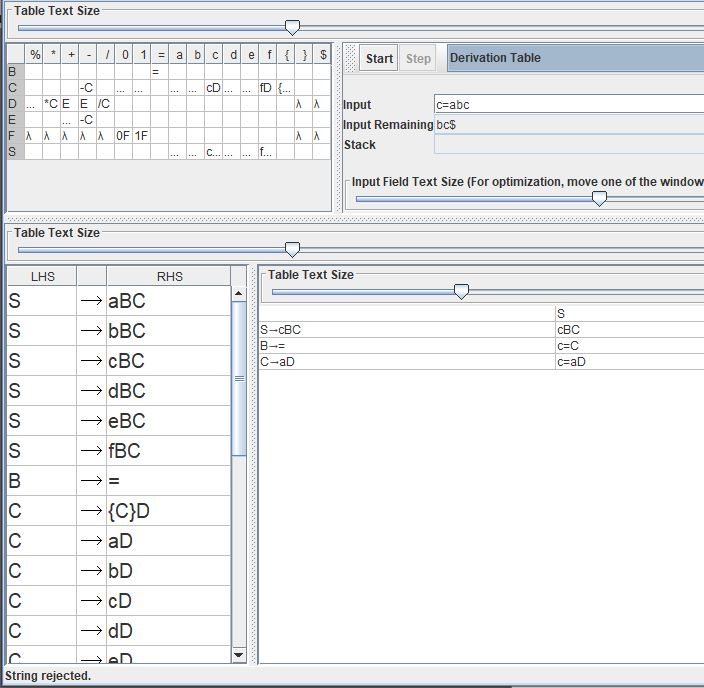


Рисунок 7 – Перехват экрана распознавания (для неверной строки)

*Часть 2*

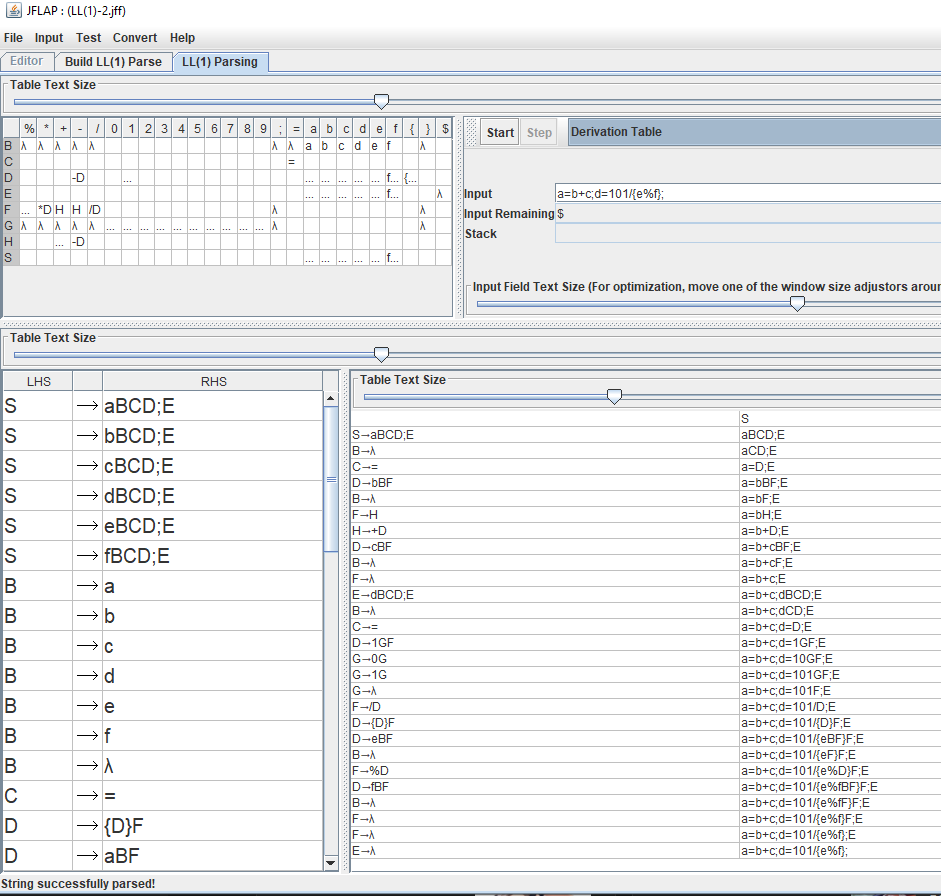


Рисунок 8 – Перехват экрана распознавания

Рисунок 9 – Перехват экрана распознавания

Рисунок 10 – Перехват экрана распознавания

Рисунок 11 – Перехват экрана распознавания

Рисунок 12 – Перехват экрана распознавания

*Часть 3*

Рисунок 3 – Полученный МПА (PDA.jff)

Рисунок 4 – Полученный МПА (PDA.jff)

Рисунок 5 – Полученный МПА (PDA.jff)

Рисунок 6 – Полученный МПА (PDA.jff)

Рисунок 7 – Полученный МПА (PDA.jff)

Рисунок 8 – Полученный МПА (PDA.jff)

Рисунок 9 – Полученный МПА (PDA.jff)

Рисунок 1 – Полученный МПА (PDA.jff)

Рисунок 1 – Полученный МПА (PDA.jff)

Рисунок 1 – Полученный МПА (PDA.jff)

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

# Вывод

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

**ПРИЛОЖЕНИЕ А**

Листинг 1 – файл CYK.py