Министерство образования и науки РФ

Белгородский государственный технологический университет

им. В.Г. Шухова

Кафедра Программного обеспечения вычислительной

техники и автоматизированных систем

**КУРСОВАЯ РАБОТА**

по дисциплине:

«Теория автоматов и формальных языков»

тема: «Обработка формальных языков»

Автор работы \_\_\_\_\_\_\_\_\_\_\_ Притчин Иван Сергеевич ПВ-31

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_ Рязанов Юрий Дмитриевич

(подпись)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Белгород

2018 г.

Содержание

[Введение 3](#_Toc515293641)

[Восходящая обработка контекстно-свободных языков методом «перенос-опознание» 4](#_Toc515293642)

[1.1 Краткие теоретические сведения 4](#_Toc515293643)

[1.2 Преобразование исходной КС – грамматики в грамматику без правой рекурсии 9](#_Toc515293644)

[1.3 Поиск множества первых и следующих 9](#_Toc515293645)

[1.4 Построение управляющей таблицы МП-распознавателя 10](#_Toc515293646)

[1.5 Процедура опознания в виде конечного автомата 11](#_Toc515293647)

[1.6 Построение набора тестовых данных 11](#_Toc515293648)

[1.7 Программа – распознаватель, реализующую построенный восходящий МП – распознаватель 12](#_Toc515293649)

[1.8 Результат работы программы 12](#_Toc515293650)

[Восходящая обработка контекстно-свободных языков методом «перенос-свертка» 13](#_Toc515293651)

[2.1 Пополнение грамматики 13](#_Toc515293652)

[2.2 Поиск множества следующих для грамматики 13](#_Toc515293653)

[2.3 Построение графа переходов на множестве SLR(1) – ситуаций 13](#_Toc515293654)

[2.4 Построение управляющей таблицы восходящего МП – распознавателя типа «перенос-свертка» 15](#_Toc515293655)

[2.5 Программа – распознаватель, обрабатывающая таблицу восходящего МП – распознавателя типа «перенос – свёртка» 16](#_Toc515293656)

[2.6 Построение набора тестовых данных 16](#_Toc515293657)

[2.7 Обработка цепочки из набора тестовых данных программой – распознавателем 16](#_Toc515293658)

[Выводы 17](#_Toc515293659)

[Список литературы 18](#_Toc515293660)

[Приложение А. Код программы - распознавателя типа «перенос – опознание». 19](#_Toc515293661)

[Приложение Б. Код программы для получения графа перехода 22](#_Toc515293662)

[Приложение В. Код программы для получения управляющей таблицы MP-распознавателя типа «перенос-свертка» 25](#_Toc515293663)

[Приложение Г. Код программы - распознавателя типа «перенос – свёртка» 26](#_Toc515293664)

# Введение

**Контекстно-свободной грамматикой** (англ. *сontext-free grammar*) называется грамматика, у которой в левых частях всех правил стоят только одиночные нетерминалы. **Контекстно-свободный язык** (англ. *context-free language*) — язык, задаваемый контекстно-свободной грамматикой. Смысл термина «контекстно-свободная» заключается в том, что есть возможность применить правило к нетерминалу, причём независимо от контекста этого нетерминала[1, 3].

Типы KC-грамматик[2]:

* LL-грамматика
* [LALR-грамматика](https://ru.wikipedia.org/w/index.php?title=LALR-%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0&action=edit&redlink=1) (см.: [LALR(1)](https://ru.wikipedia.org/wiki/LALR(1)))
* [LR-грамматика](https://ru.wikipedia.org/w/index.php?title=LR-%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0&action=edit&redlink=1)
* [SLR-грамматика](https://ru.wikipedia.org/w/index.php?title=SLR-%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0&action=edit&redlink=1) (см.: [SLR(1)](https://ru.wikipedia.org/wiki/SLR(1)))

Существуют два разных класса распознавателей, их названия связаны с порядком построения дерева вывода. Как правило, все распознаватели читают входную цепочку символов слева направо, поскольку предполагается такая нотация в написании исходного текста программ.

Распознаватели[2]:

* Нисходящие (порождают цепочки левостороннего вывода и строят дерево вывода сверху вниз)
* Восходящие (порождают цепочки правостороннего вывода и строят дерево вывода снизу вверх)

Основным ограничением возможностей КС грамматик является следующее: не все языки могут быть заданы с помощью КС-грамматик.

Применение КС-грамматик[2]:

* Контекстно-свободные грамматики хорошо зарекомендовали себя в задачах разбора многих западноевропейских языков, таких как английский.
* Контекстно-свободные грамматики описывают грамматическую структуру большинства языков программирования.
* Машинный разбор предложения с использованием контекстно-свободных грамматик реализуется при помощи [алгоритма Эрли](https://nlpub.ru/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%AD%D1%80%D0%BB%D0%B8&action=edit&redlink=1) или [алгоритма Кока — Янгера — Касами](https://nlpub.ru/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D0%BE%D0%BA%D0%B0_%E2%80%94_%D0%AF%D0%BD%D0%B3%D0%B5%D1%80%D0%B0_%E2%80%94_%D0%9A%D0%B0%D1%81%D0%B0%D0%BC%D0%B8).

# 1. Восходящая обработка контекстно-свободных языков методом «перенос-опознание»

## 1.1 Краткие теоретические сведения

Пусть язык задан грамматикой :

|  |
| --- |
| S → (AS) |
| S → (b) |
| A → (SaA) |
| A → (a) |

Применим последовательно ряд правил

Правая часть правила, которую применяем последней при правом выводе данной цепочки, называется основой, а правило основывающим

Правило – основывающее.

Пусть входная цепочка -|, где – обработанная часть, – необработанная часть. В магазине находится – выводится из . Тогда – промежуточная цепочка в правом выводе.

Обработаем цепочку из примера. В начале обработки в магазине – маркер дна (V).

Состояние магазина : ▼

Добавим первый символ «(»

Состояние магазина: *▼(*

Тогда .

Будем вносить в магазин символы до тех пор, пока в нём не окажется правая часть правил

Состояние магазина: *▼(((b)*

Выполним замену на нетерминал (произведем «свертку»)

Состояние магазина: *▼((S*

Опишем выполнение процедуры дальше:

|  |
| --- |
| ▼ |
| ▼ |
| ▼ |
| ▼ |
| ▼ |
| ▼ |
| ▼ |
| ▼ |
| ▼ |

Если входная цепочка закончилась, и в магазине находится начальный нетерминал – цепочка допускается

Вначале если первый символ принадлежит , то выполняем перенос. После свертки получим на вершине магазина нетерминал , а если следующий символ , то ошибка.

П – перенос (втолкнуть символ и выполнить сдвиг)

ОП – опознание (когда дошли до конца или встретился вверху магазина последний символ правых частей цепочек)

Можно вывести следующие правила:

Получим множество ПЕРВЫХ и СЛЕДУЮЩИХ (Таблица 1)

Таблица 1 — Множества первых и следующих для грамматики

|  |  |  |  |
| --- | --- | --- | --- |
|  | ПЕРВЫЕ | СЛЕДУЮЩИЕ | |
| S | S ( | ) a -| | ) a -| |
| A | A ( | ) ПЕРВ(S) | ) ( |
| ( |  | ПЕРВ(A), ПЕРВ(S),  a, b | ( b a |
| ) |  | СЛЕД(S), СЛЕД(А) | ) a ( -| |
| A |  | ПЕРВ(А) ) | ( ) |
| B |  | ) | ) |

Управляющая таблица представлена в таблице 2.

Таблица 2 — Управляющая таблица МП-распознавателя

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ( | ) | a | b | -| |
| S |  | П | П |  | ОП |
| A | П | П |  |  |  |
| ( | П |  | П | П |  |
| ) | ОП | ОП | ОП |  | ОП |
| A | П | П |  |  |  |
| B |  | П |  |  |  |
| ▼ | П |  | П |  |  |

Начальное состояние магазина: ▼

Граф опознания для грамматики представлен на рисунке 1.

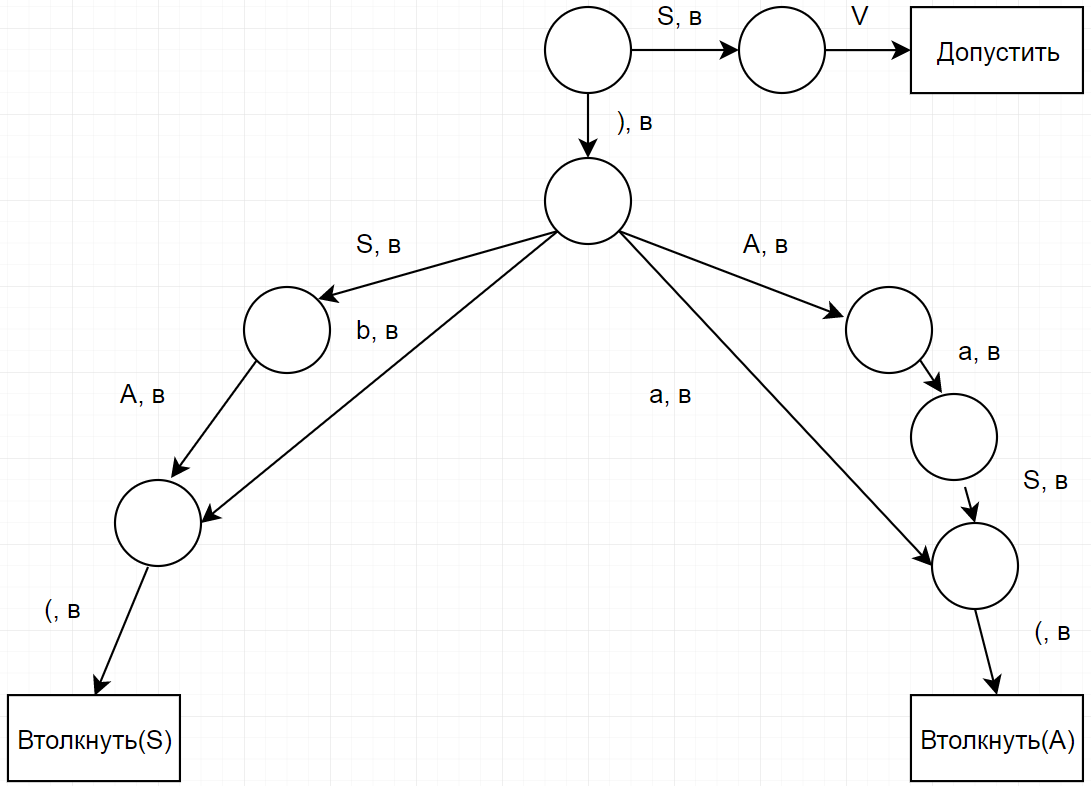


Рисунок 1 — Граф опознания для грамматики

Рассмотрим на других примерах.

Язык задан грамматикой

|  |
| --- |
| S → bASB |
| S → bA |
| A → dSca |
| A → e |
| B → cAa |
| B → c |

Найдем множество первых и следующих. Результат – таблица 3. Управляющая таблица для грамматики представлена на таблице 4.

Таблица 3 — Множества первых и следующих для грамматики

|  |  |  |  |
| --- | --- | --- | --- |
|  | ПЕРВЫЕ | СЛЕДУЮЩИЕ | |
| S | S b | ПЕРВ(B) c | c -| |
| A | A d e | ПЕРВ(S) СЛЕД(S) a | b a c -| |
| B | B c | СЛЕД(S) | c -| |
| A |  | СЛЕД(А) СЛЕД(B) | b a c -| |
| B |  | ПЕРВ(А) | d e |
| C |  | a  ПЕРВ(А) СЛЕД(B) | a d e c -| |
| D |  | ПЕРВ(S) | b |
| E |  | СЛЕД(А) | b a c -| |

Таблица 4 — Управляющая таблица МП-распознавателя (

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | a | B | c | d | e | -| |
| S |  |  | П |  |  | ОП |
| A | П | П | ОП |  |  | ОП |
| B |  |  | ОП |  |  | ОП |
| A | ОП | ОП | ОП |  |  | ОП |
| B |  |  |  | П | П |  |
| C | П |  | ОП | П | П | ОП |
| D |  |  |  | П |  |  |
| E | ОП | ОП | ОП |  |  | ОП |
| ▼ |  | П |  |  |  |  |

Начальное состояние магазина: ▼.

Выполним построение графа опознания (рисунок 2):

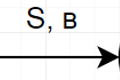
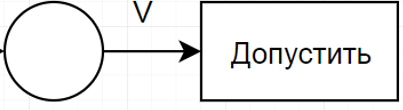
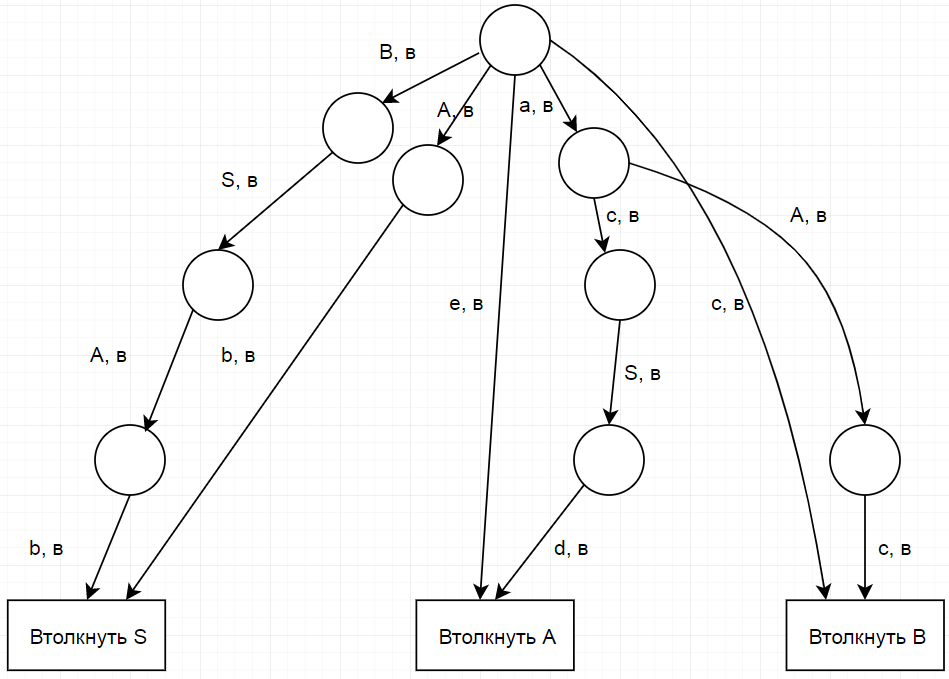
 

Рисунок 2 — Граф опознания для грамматики

**Описание алгоритма**

1. Преобразовать исходную КС-грамматику в грамматику без правой рекурсии
2. Определить множество ПЕРВЫХ для каждого символа грамматики
3. Определить множество СЛЕДУЮЩИХ для каждого символа грамматики
4. Построить управляющую таблицу восходящего МП-распознавателя. Если построение без конфликтов «перенос-опознание» невозможен – выполняем преобразование исходной грамматики, допускающую построение управляющей таблицы без конфликтов.
5. Построить процедуру опознания в виде конечного автомата.

## 1.2 Преобразование исходной КС – грамматики в грамматику без правой рекурсии

Рассмотрим язык заданный грамматикой (таблица 5).

Таблица 5 – Исходная и преобразованная грамматика

|  |  |
| --- | --- |
| Исходная грамматика | Преобразованная грамматика |
| S → OS | S → SO |
| S → O | S → O |
| O → {[S]Y} | O → {[S]Y} |
| O → {Y[S]} | O → {Y[S]} |
| O → a=E | O → a=E |
| Y → a=a | Y → a=a |
| Y → a<a | Y → a<a |
| Y → !(Y) | Y → !(Y) |
| E → +(E,E) | E → +(E,E) |
| E → \*(E,E) | E → \*(E,E) |
| E → b | E → b |

## 1.3 Поиск множества первых и следующих

Таблица 6 — Множества первых и следующих для грамматики

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | ПЕРВЫЕ | | СЛЕДУЮЩИЕ | | | |
| S | S ПЕРВ(О) | S O { a | ПЕРВ(О) ] | -| | { a ] | -| |
| O | O { a | O { a |  | СЛЕД(S) |  | { a -| ] |
| Y | Y a ! | Y a ! | } [ ) |  | } [ ) |  |
| E | E + \* b | E + \* b | , ) | СЛЕД(О) | , ) | { a -| ] |
| { |  |  | [ ПЕРВ(Y) |  | [ a ! |  |
| [ |  |  | ПЕРВ(S) |  | { a |  |
| ] |  |  | ПЕРВ(Y) } |  | a ! } |  |
| } |  |  |  | СЛЕД(O) |  | { a -| ] |
| = |  |  | ПЕРВ(E) |  | + \* b |  |
| < |  |  | a |  | a |  |
| ! |  |  | ( |  | ( |  |
| ( |  |  | ПЕРВ(Y) ПЕРВ(Е) |  | b a ! + \* |  |
| ) |  |  |  | СЛЕД(Y)  СЛЕД(E) |  | } [ ) , { a -| |
| + |  |  | ( |  | ( |  |
| , |  |  | ПЕРВ(E) |  | + \* b |  |
| \* |  |  | ( |  | ( |  |
| a |  |  | = < | СЛЕД(Y) | = < | } [ ) |
| b |  |  |  | СЛЕД(Е) |  | { a -| ] |

## 1.4 Построение управляющей таблицы МП-распознавателя

Выполним построение управляющей таблицы МП-распознавателя. Найдём множество ПОД.

Множество определяется как символы, находящиеся непосредственно перед нетерминалом. Для символа магазина , если после него в грамматике есть нетерминал – отмечаются ячейки на пересечении строки символа и нетерминалов из множества первых для .

Таблица 6 — Управляющая таблица МП-распознавателя

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | { | [ | ] | } | = | < | ! | ( | ) | + | , | \* | a | b | -| |  | S | O | Y | E |
| S | П |  | П |  |  |  |  |  |  |  |  |  | П |  | О |  |  | v |  |  |
| O | О |  | О |  |  |  |  |  |  |  |  |  | О |  | О |  |  |  |  |  |
| Y |  | П |  | П |  |  |  |  | П |  |  |  |  |  |  |  |  |  |  |  |
| E | О |  | О |  |  |  |  |  | П |  | П |  | О |  | О |  |  |  |  |  |
| { |  | П |  |  |  |  | П |  |  |  |  |  | П |  |  |  | v |  |  |  |
| [ | П |  |  |  |  |  |  |  |  |  |  |  | П |  |  |  | v |  |  |  |
| ] |  |  |  | П |  |  | П |  |  |  |  |  | П |  |  |  |  |  | v |  |
| } | О |  | О |  |  |  |  |  |  |  |  |  | О |  | О |  |  |  |  |  |
| = |  |  |  |  |  |  |  |  |  | П |  | П | П | П |  |  |  |  |  | v |
| < |  |  |  |  |  |  |  |  |  |  |  |  | П |  |  |  |  |  |  |  |
| ! |  |  |  |  |  |  |  | П |  |  |  |  |  |  |  |  |  |  |  |  |
| ( |  |  |  |  |  |  | П |  |  | П |  | П | П | П |  |  |  |  | v | v |
| ) | О | О |  | О |  |  |  |  | О |  | О |  | О |  | О |  |  |  |  |  |
| + |  |  |  |  |  |  |  | П |  |  |  |  |  |  |  |  |  |  |  |  |
| , |  |  |  |  |  |  |  |  |  | П |  | П | П | П |  |  |  |  |  | v |
| \* |  |  |  |  |  |  |  | П |  |  |  |  |  |  |  |  |  |  |  |  |
| a |  | О |  | О | П | П |  |  | О |  |  |  |  |  |  |  |  |  |  |  |
| b | О |  | О |  |  |  |  |  |  |  |  |  | О |  | О |  |  |  |  |  |
| ▼ | П |  |  |  |  |  |  |  |  |  |  |  | П |  |  |  | v | v |  |  |

Начальное состояние магазина: ▼

{x | x ПОД S} = {{, [, ▼}

{x | x ПОД O} = {S, ▼}

{x | x ПОД Y} = {] (}

{x | x ПОД E} = {= ( ,}

## 1.5 Процедура опознания в виде конечного автомата

Построим граф опознания для грамматики (рисунок 3)

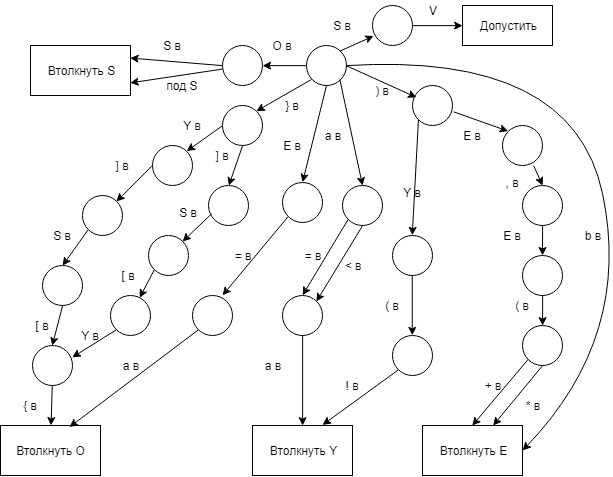


Рисунок 3 — Граф опознания для грамматики

## 1.6 Построение набора тестовых данных

В качестве тестовых данных возьмём цепочку, в выводе которой будут участвовать все правила нашей грамматики. Процесс вывода представлен в таблице 8.

Таблица 8 — Вывод цепочки тестовых данных

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S(1) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| S(2) | | | | | | | | | | | | | | | O(4) | | | | | | | | | | | | | | | | | |
| O(3) | | | | | | | | | | | | | | |
| { | [ | S(2) | | | | | | | | ] | Y(6) | | | } |
| S(2) | | | | | | | | a | = | a | { | Y(8) | | | | | | [ | S(2) | | | | | | | | ] | } |
| О(5) | | | | | | | | O(5) | | | | | | | |
| a | = | E(9) | | | | | | a | = | E(10) | | | | | |
| + | ( | E(11) | , | E(11) | ) | ! | ( | Y(7) | | | ) | \* | ( | E(11) | , | E(11) | ) |
| b | b | a | < | a | b | b |

## 1.7 Программа – распознаватель, реализующую построенный восходящий МП – распознаватель

Программа представлена в приложении А.

## 1.8 Результат работы программы

Скриншот работы экрана представлен на рисунке 4.

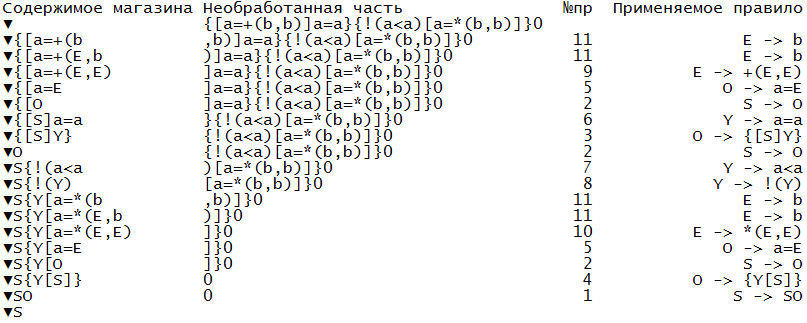


Рисунок 4 — Результат работы программы

# Восходящая обработка контекстно-свободных языков методом «перенос-свертка»

## 2.1 Пополнение грамматики

Рассмотрим грамматику . Пополним грамматику правилом . Пронумеруем полученные правила.

|  |  |
| --- | --- |
| S → OS | 1. R → S |
| S → O | 1. S → OS |
| O → a[S] | 1. S → O |
| O → a[S][S] | 1. O → a[S] |
| O → a=E | 1. O → a[S][S] |
| E → E+T | 1. O → a=E |
| E → T | 1. E → E+T |
| T → T\*P | 1. E → T |
| T → P | 1. T → T\*P |
| P → (E) | 1. T → P |
| P → a | 1. P → (E) |
|  | 1. P → a |

## 2.2 Поиск множества следующих для грамматики

Выполним поиск следующих для всех нетерминалов из грамматики (таблица 9). Их поиск необходим для построения графа переходов.

Таблица 9 — Множество следующих для грамматики

|  |  |  |
| --- | --- | --- |
|  | СЛЕДУЮЩИЕ | |
| S | СЛЕД(S) ] -| | ] -| |
| O | ПЕРВ(S)  СЛЕД(S) | a ] -| |
| T | СЛЕД(E) \* | a ] + ) \* -| |
| P | СЛЕД(T) | a ] + ) \* -| |
| E | СЛЕД(O) + ) | a ] + ) -| |

## 2.3 Построение графа переходов на множестве SLR(1) – ситуаций

Выполним построение графа переходов. Для решения данной задачи воспользуемся программой, написанной на языке программирования R. Код приложения и требования к программе представлены в приложении Б.

Результат работы программы представлен в таблице 10.

Таблица 10 Граф переходов для грамматики

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № ст | № сост | Содер. | СП | Переход |
| 1 | 0 | R → .S | S | 6 |
| 2 | 0 | S → .OS | O | 4 |
| 3 | 0 | S → .O | O | 4 |
| 4 | 0 | O → .a[S] | a | 2 |
| 5 | 0 | O → .a[S][S] | a | 2 |
| 6 | 0 | O → .a=E | a | 2 |
| 7 | 0 | E → .E+T | E | 3 |
| 8 | 0 | E → .T | T | 7 |
| 9 | 0 | T → .T\*P | T | 7 |
| 10 | 0 | T → .P | P | 5 |
| 11 | 0 | P → .(E) | ( | 1 |
| 12 | 0 | P → .a | a | 2 |
| 13 | 1 | P → (.E) | E | 8 |
| 14 | 1 | E → .E+T | E | 8 |
| 15 | 1 | E → .T | T | 7 |
| 16 | 1 | T → .T\*P | T | 7 |
| 17 | 1 | T → .P | P | 5 |
| 18 | 1 | P → .(E) | ( | 1 |
| 19 | 1 | P → .a | a | 9 |
| 20 | 2 | P → a. | СЛЕД P | -12 |
| 21 | 2 | O → a.[S] | [ | 10 |
| 22 | 2 | O → a.[S][S] | [ | 10 |
| 23 | 2 | O → a.=E | = | 11 |
| 24 | 3 | E → E.+T | + | 12 |
| 25 | 4 | S → O. | СЛЕД S | -3 |
| 26 | 4 | S → O.S | S | 13 |
| 27 | 4 | S → .OS | O | 4 |
| 28 | 4 | S → .O | O | 4 |
| 29 | 4 | O → .a[S] | a | 2 |
| 30 | 4 | O → .a[S][S] | a | 2 |
| 31 | 4 | O → .a=E | a | 2 |
| 32 | 5 | T → P. | СЛЕД T | -10 |
| 33 | 6 | R → S. | СЛЕД R | -1 |
| 34 | 7 | E → T. | СЛЕД E | -8 |
| 35 | 7 | T → T.\*P | \* | 14 |
| 36 | 8 | P → (E.) | ) | 15 |
| 37 | 8 | E → E.+T | + | 12 |
| 38 | 9 | P → a. | СЛЕД P | -12 |
| 39 | 10 | O → a[.S] | S | 16 |
| 40 | 10 | O → a[.S][S] | S | 16 |
| 41 | 10 | S → .OS | O | 4 |
| 42 | 10 | S → .O | O | 4 |
| 43 | 10 | O → .a[S] | a | 2 |
| 44 | 10 | O → .a[S][S] | a | 2 |
| 45 | 10 | O → .a=E | a | 2 |
| 46 | 11 | O → a=.E | E | 17 |
| 47 | 11 | E → .E+T | E | 17 |
| 48 | 11 | E → .T | T | 7 |
| 49 | 11 | T → .T\*P | T | 7 |
| 50 | 11 | T → .P | P | 5 |
| 51 | 11 | P → .(E) | ( | 1 |
| 52 | 11 | P → .a | a | 9 |
| 53 | 12 | E → E+.T | T | 18 |
| 54 | 12 | T → .T\*P | T | 18 |
| 55 | 12 | T → .P | P | 5 |
| 56 | 12 | P → .(E) | ( | 1 |
| 57 | 12 | P → .a | a | 9 |
| 58 | 13 | S → OS. | СЛЕД S | -2 |
| 59 | 14 | T → T\*.P | P | 19 |
| 60 | 14 | P → .(E) | ( | 1 |
| 61 | 14 | P → .a | a | 9 |
| 62 | 15 | P → (E). | СЛЕД P | -11 |
| 63 | 16 | O → a[S.] | ] | 20 |
| 64 | 16 | O → a[S.][S] | ] | 20 |
| 65 | 17 | O → a=E. | СЛЕД O | -6 |
| 66 | 17 | E → E.+T | + | 12 |
| 67 | 18 | E → E+T. | СЛЕД E | -7 |
| 68 | 18 | T → T.\*P | \* | 14 |
| 69 | 19 | T → T\*P. | СЛЕД T | -9 |
| 70 | 20 | O → a[S]. | СЛЕД O | -4 |
| 71 | 20 | O → a[S].[S] | [ | 21 |
| 72 | 21 | O → a[S][.S] | S | 22 |
| 73 | 21 | S → .OS | O | 4 |
| 74 | 21 | S → .O | O | 4 |
| 75 | 21 | O → .a[S] | a | 2 |
| 76 | 21 | O → .a[S][S] | a | 2 |
| 77 | 21 | O → .a=E | a | 2 |
| 78 | 22 | O → a[S][S.] | ] | 23 |
| 79 | 23 | O → a[S][S]. | СЛЕД O | -5 |

## 2.4 Построение управляющей таблицы восходящего МП – распознавателя типа «перенос-свертка»

Построение таблицы восходящего МП-распознавателя типа «перенос-свертка» является процессов, в ходе которого легко допустить ошибку. Во избежание таких проблем опишем программу на языке R (Приложение В). Результат работы программы представлен в таблице 11.

Таблица 11 — Управляющая таблица MP-распознавателя

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ( | ) | \* | [ | ] | + | = | 0 | a | E | O | P | S | T |
| 0 | P1 |  |  |  |  |  |  |  | P2 | V3 | V4 | V5 | V6 | V7 |
| 1 | P1 |  |  |  |  |  |  |  | P9 | V8 |  | V5 |  | V7 |
| 2 |  | C12 | C12 | P10 | C12 | C12 | P11 | C12 | C12 |  |  |  |  |  |
| 3 |  |  |  |  |  | P12 |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  | C3 |  |  | C3 | P2 |  | V4 |  | V13 |  |
| 5 |  | C10 | C10 |  | C10 | C10 |  | C10 | C10 |  |  |  |  |  |
| 6 |  |  |  |  | C1 |  |  | C1 |  |  |  |  |  |  |
| 7 |  | C8 | P14 |  | C8 | C8 |  | C8 | C8 |  |  |  |  |  |
| 8 |  | P15 |  |  |  | P12 |  |  |  |  |  |  |  |  |
| 9 |  | C12 | C12 |  | C12 | C12 |  | C12 | C12 |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  | P2 |  | V4 |  | V16 |  |
| 11 | P1 |  |  |  |  |  |  |  | P9 | V17 |  | V5 |  | V7 |
| 12 | P1 |  |  |  |  |  |  |  | P9 |  |  | V5 |  | V18 |
| 13 |  |  |  |  | C2 |  |  | C2 |  |  |  |  |  |  |
| 14 | P1 |  |  |  |  |  |  |  | P9 |  |  | V19 |  |  |
| 15 |  | C11 | C11 |  | C11 | C11 |  | C11 | C11 |  |  |  |  |  |
| 16 |  |  |  |  | P20 |  |  |  |  |  |  |  |  |  |
| 17 |  |  |  |  | C6 | P12 |  | C6 | C6 |  |  |  |  |  |
| 18 |  | C7 | P14 |  | C7 | C7 |  | C7 | C7 |  |  |  |  |  |
| 19 |  | C9 | C9 |  | C9 | C9 |  | C9 | C9 |  |  |  |  |  |
| 20 |  |  |  | P21 | C4 |  |  | C4 | C4 |  |  |  |  |  |
| 21 |  |  |  |  |  |  |  |  | P2 |  | V4 |  | V22 |  |
| 22 |  |  |  |  | P23 |  |  |  |  |  |  |  |  |  |
| 23 |  |  |  |  | C5 |  |  | C5 | C5 |  |  |  |  |  |

## 2.5 Программа – распознаватель, обрабатывающая таблицу восходящего МП – распознавателя типа «перенос – свёртка»

Программа представлена в приложении Г текущей курсовой работы.

## 2.6 Построение набора тестовых данных

В качестве тестовых данных возьмём цепочку, в выводе которой будут участвовать все правила нашей грамматики. Процесс вывода представлен в таблице 12.

Таблица 12 — Вывод цепочки

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S(2) | | | | | | | | | | | | | | | | | | | | | | | | |
| O(5) | | | | | | | | | | | | | | | | | | | S(4) | | | | | |
| a | [ | S(6) | | | | | | | ] | [ | S(6) | | | | | | | ] | a | [ | S(6) | | | ] |
| a | = | E(7) | | | | | a | = | E(8) | | | | | a | = | E(8) |
| E(8) | + | T(9) | | | T(10) | | | | | T(10) |
| T(10) | T(10) | \* | P(12) | P(12) | | | | | P(12) |
| P(12) | P(12) | a | ( | E(7) | | | ) | a |
| a | a | E(8) | + | T(10) |
| T(10) | P(12) |
| P(12) | a |
| a |

## 2.7 Обработка цепочки из набора тестовых данных программой – распознавателем

Пример работы программы представлен на рисунке 4.

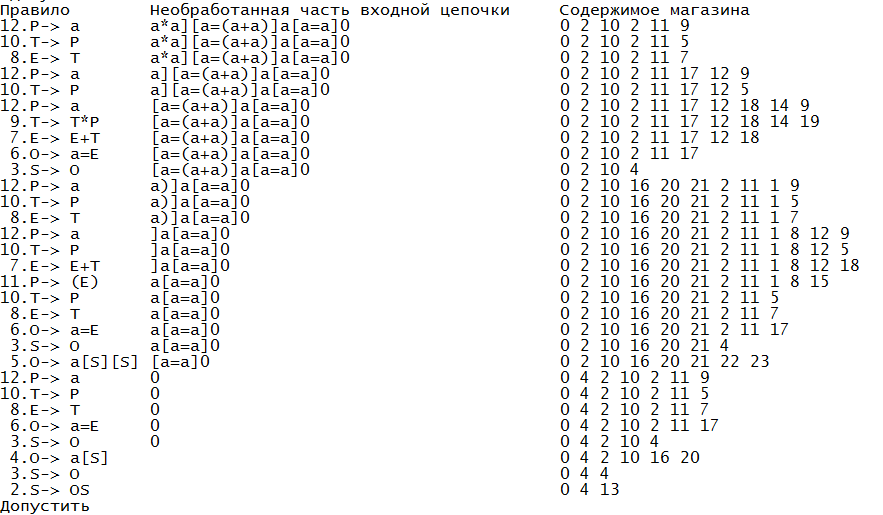


Рисунок 4 — Результат работы программы

# Выводы

Вы ходе выполнения курсовой работы были описаны алгоритмы восходящей обработки контекстно-свободных языков. Написаны

* Программа-распознаватель типа «перенос-опознание»
* Программа-распознаватель типа «перенос-свертка»
* Программа для получения графа перехода по исходной грамматике
* Программа для построения управляющей таблицы восходящего МП – распознавателя типа «перенос-свертка» по графу перехода и множеству следующий для нетерминалов грамматики

# Список литературы

1. **Хопкрофт Д., Мотвани Р., Ульман Д.** — Введение в теорию автоматов, языков и вычислений, 2-е изд. : Пер. с англ. — Москва, Издательский дом «Вильямс», 2002. — 528 с. : ISBN 5-8459-0261-4 (рус.)
2. **Wikipedia** [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki /Контекстно-свободная\_грамматика (дата обращения 28.05.2018)
3. **Университет ИТМО** [Электронный ресурс] URL: <https://neerc.ifmo.ru/wiki> /index/php?title= Контекстно-свободные грамматики, вывод, лево- и правосторонний вывод, дерево разбора (дата обращения 28.05.2018)

# Приложение А. Код программы - распознавателя типа «перенос – опознание».

class Stack:

def \_\_init\_\_(self):

self.\_data = []

def \_\_len\_\_(self):

return len(self.\_data)

def isempty(self):

return len(self.\_data) == 0

def top(self):

return self.\_data[-1]

def push(self, e):

self.\_data.append(e)

def pop(self):

return self.\_data.pop()

def str(self):

res = ""

for el in self.\_data:

res = res + str(el)

return res

def clear(self):

self.\_data = []

def inTopAndPop(values):

return mag.top() in values and mag.pop();

def inTop(values):

return mag.top() in values;

def inTopAndPopArray(values):

i = 0

while i < len(values) and inTopAndPop(values[i]):

i += 1

return i == len(values)

def identification():

if inTopAndPop("S"):

if inTop("▼"):

return True

elif inTopAndPop("O"):

if inTopAndPop("S"):

return 1

elif inTop("{[▼"):

return 2

elif inTopAndPop("}"):

if inTopAndPop("Y"):

if inTopAndPopArray(["]", "S", "[", "{"]):

return 3

elif inTopAndPop("]"):

if inTopAndPopArray(["S", "[", "Y", "{"]):

return 4

elif inTopAndPopArray(["E", "=", "a"]):

return 5

elif inTopAndPop("a"):

if inTopAndPopArray(["=", "a"]):

return 6

elif inTopAndPopArray(["<", "a"]):

return 7

elif inTopAndPop(")"):

if inTopAndPop("Y"):

if inTopAndPopArray(["(", "!"]):

return 8

elif inTopAndPop("E"):

if inTopAndPopArray([",", "E", "("]):

if inTopAndPop("+"):

return 9

if inTopAndPop("\*"):

return 10

elif(inTopAndPop("b")):

return 11

return False

manipulateTable = {

"П": {

"{": "S[▼",

"[": "Y{",

"]": "S",

"}": "Y]",

"=": "a",

"<": "a",

"!": "{](",

"(": "!+\*",

")": "YE",

"+": "=(,",

",": "E",

"\*": "=(,",

"a": "S{[]=<(,▼",

"b": "=(,"

},

"ОП": {

"{": "O})b",

"[": ")a",

"]": "OE}b)",

"}": ")a",

")": ")ab",

",": ")b",

"a": "OE})b",

"0": "SOE})b"

}

}

rules = {

1 : "S -> SO",

2 : "S -> O",

3 : "O -> {[S]Y}",

4 : "O -> {Y[S]}",

5 : "O -> a=E",

6 : "Y -> a=a",

7 : "Y -> a<a",

8 : "Y -> !(Y)",

9 : "E -> +(E,E)",

10 : "E -> \*(E,E)",

11 : "E -> b"

}

mag = Stack()

def check(chain):

chain = chain + "0"

mag.clear()

mag.push("▼")

i = 0

imax = len(chain)

print(" %-19s %-35s %3s %20s" %("Содержимое магазина", "Необработанная часть",

"№пр", "Применяемое правило"))

print(" %-19s %-35s" %(mag.str(), chain[i:imax]))

while i < imax:

sym = chain[i]

if sym in manipulateTable["П"] and mag.top() in manipulateTable["П"][sym]:

mag.push(sym)

i += 1

elif sym in manipulateTable["ОП"] and mag.top() in

manipulateTable["ОП"][sym]:

old\_mag = mag.str()

res = identification()

if type(res) == type(True):

print(" %-19s" %("▼S"))

print(" Допустить ")

break

elif res == False:

print(" Отвергнуть ")

break

else:

print(" %-19s %-35s %3s %20s" %(old\_mag, chain[i:imax], res,

rules[res]))

mag.push(rules[res][0])

else:

print("Отвергнуть")

break

# Приложение Б. Код программы для получения графа перехода

library(readxl)

library(rstudioapi)

library(magrittr)

library(openxlsx)

library(dplyr)

library(stringr)

options(tibble.print\_max = 500, tibble.print\_min = 300)

(function() {

current\_path <- getActiveDocumentContext()$path

setwd(dirname(current\_path))

})()

data <- read\_excel("Исходные данные.xlsx")

init <- function(data) {

data$`Номер состояния` <- 0

data$`Символ перехода` <-

str\_extract(data$Содержимое, "\\..") %>%

str\_replace(".", "")

data$Переход <- dense\_rank(data$`Символ перехода`)

data

}

data %<>% init()

del\_point <- function(string) {

str\_replace(string, "\\.", "")

}

get\_all\_transition\_symbol <- function(res) {

str\_extract(res$Содержимое, "\\..") %>% del\_point()

}

getSetPointNonterminal <- function(rules) {

uni <-

rules %>%

str\_extract(., "\\.[:alpha:]") %>%

del\_point() %>%

unique() %>%

.[!is.na(.)]

uni

}

getTable <- function(new\_rules) {

n <- length(new\_rules)

tibble(

`Номер состояния` = rep(NA, n),

`Содержимое` = new\_rules,

`Символ перехода` = rep(NA, n),

`Переход`= rep(NA, n)

)

}

get\_rules <- function(sum\_data, i, native\_data) {

native\_data\_copy <- native\_data

uni <- getSetPointNonterminal(native\_data$Содержимое)

# Получение правил

rules <- filter(sum\_data, Переход == i)

rules <- filter(rules, rules$`Номер состояния` == rules$`Номер состояния`[1])

new\_rules <-

str\_replace(rules$Содержимое, "\\..", str\_c(rules$`Символ перехода`, '\\.'))

nonTermRules <- getSetPointNonterminal(new\_rules)

while (str\_detect(nonTermRules, "[:upper:]") %>% any(na.rm = TRUE)) {

append\_rules <-

native\_data\_copy$Содержимое[str\_extract(native\_data\_copy$Содержимое, '.') %in% nonTermRules]

native\_data\_copy %<>% filter(!(`Содержимое` %in% append\_rules))

new\_rules %<>% c(append\_rules)

nonTermRules <- getSetPointNonterminal(append\_rules)

}

res <- getTable(new\_rules)

res$`Номер состояния` <- i

return(res)

}

pr\_rules\_with\_sv <- function(local\_res, native\_rules) {

rules <- local\_res$Содержимое

native\_rules %<>% del\_point()

rules %<>% del\_point()

for (i in 1:nrow(local\_res)) {

local\_res$`Символ перехода`[i] <-

local\_res$Содержимое[i] %>%

str\_sub(1, 1) %>%

str\_c("СЛЕД ", .)

print(rules[i])

print(native\_rules)

rule\_num <- which(rules[i] == native\_rules)

local\_res$Переход[i] <- -rule\_num

}

local\_res

}

### Получние таблицы

get\_table <- function(data) {

native\_data <- sum\_data <- data

new\_state\_index <- max(data$Переход) + 1

i = 1;

while (TRUE) {

res <- get\_rules(sum\_data, i, native\_data)

res$`Символ перехода` <- get\_all\_transition\_symbol(res)

##Получение правил с символом перехода NA

rules\_with\_svert <- filter(res, is.na(res$`Символ перехода`))

rules\_without\_svert <- filter(res, !is.na(res$`Символ перехода`))

plus\_data <- filter(native\_data, FALSE)

if (nrow(rules\_with\_svert) > 0) {

procRWS <- pr\_rules\_with\_sv(rules\_with\_svert, native\_data$Содержимое)

plus\_data %<>% bind\_rows(procRWS)

}

sum\_data %<>% bind\_rows(plus\_data)

all\_symbol\_per <- unique(rules\_without\_svert$`Символ перехода`)

res <- rules\_without\_svert

for (sym in all\_symbol\_per) {

rools\_with\_one\_transition\_symbol <- filter(res, `Символ перехода` == sym)

search\_rules <- rools\_with\_one\_transition\_symbol$Содержимое

searched\_rules <- filter(sum\_data, `Содержимое` %in% search\_rules)

all\_trans\_number <- sum\_data$Переход[sum\_data$Содержимое %in%

rools\_with\_one\_transition\_symbol] %>% unique()

all\_rules\_with\_trans\_number <- filter(sum\_data, `Переход` %in%

all\_trans\_number)

delta <- dplyr::setdiff(all\_rules\_with\_trans\_number, searched\_rules)

if (nrow(delta)) {

delta\_numbers <- unique(delta$`Переход`)

searched\_rules %<>% filter(!(`Переход` %in% delta\_numbers))

}

if (nrow(searched\_rules)) {

searched\_rules %<>%

select(., -"Номер состояния") %>%

unique() %>%

cbind(`Номер состояния` = 1:nrow(.), .)

}

n\_searched\_rules <- length(search\_rules)

trans <-

count(searched\_rules, `Переход`) %>%

filter(n == n\_searched\_rules) %>%

pull(`Переход`)

if (length(trans)) {

rools\_with\_one\_transition\_symbol$Переход <- trans[1]

} else {

rools\_with\_one\_transition\_symbol$Переход <- new\_state\_index

new\_state\_index <- new\_state\_index + 1

}

sum\_data %<>% bind\_rows(rools\_with\_one\_transition\_symbol)

plus\_data %<>% bind\_rows(rools\_with\_one\_transition\_symbol)

}

i = i + 1

if (i == new\_state\_index) {

break

}

}

sum\_data

}

graph <- get\_table(data)

Исходные данные: excel-файл с именем “Исходные данные”

Содержание файла:

|  |  |  |  |
| --- | --- | --- | --- |
| Номер состояния | Содержимое | Символ перехода | Переход |
|  | R → .S |  |  |
|  | S → .OS |  |  |
|  | S → .O |  |  |
|  | … | … |  |

# Приложение В. Код программы для получения управляющей таблицы MP-распознавателя типа «перенос-свертка»

getDirectTable <-function(graph, nextSet) {

initDirectTable <- function(graph) {

columns <- graph$`Символ перехода` %>% unique() %>% .[!str\_detect(., "СЛЕД")]

columns <- c(sort(columns), "0")

max <- graph$`Номер состояния`[nrow(graph)] + 1

directTable <- data.frame(rep("", max), stringsAsFactors = F)

for (i in 2:length(columns))

directTable %<>% cbind(data.frame(rep("", max), stringsAsFactors = F))

colnames(directTable) <- columns

rownames(directTable) <- 0:(max-1)

return(directTable)

}

dirTable <- initDirectTable(graph)

for (i in 1:nrow(graph)) {

s <- graph$`Номер состояния`[i]

p <- graph$Переход[i]

sp <- graph$`Символ перехода`[i]

#отрицательные переходы

if (p < 0) {

sp <- str\_sub(sp, str\_length(sp))

nextEl <- nextSet$Следующие[nextSet$Нетерминал == sp] %>% str\_split(' ')

nextEl <- nextEl[[1]]

for (j in 1:length(nextEl)) {

dirTable[s+1, nextEl[j]] = str\_c("C", -p)

}

}

#большие буквы

else if (sp %>% str\_detect("[:upper:]")) {

dirTable[s+1, sp] = str\_c("V", p)

}

#остальное

else {

dirTable[s+1, sp] = str\_c("P", p)

}

}

return(dirTable[sort(colnames(dirTable))])

}

dirTable <- getDirectTable(res, read\_excel(path = "Множество следующиx.xlsx"))

Исходными данными для работы программы являются выходные данные из программы, описанной в приложении Б. Также программа требует Excel-файл следующего содержания

|  |  |
| --- | --- |
| Нетерминал | Следующие |
| R | ] 0 |
| S | ] 0 |
| O | a ] 0 |
| … | …. |

# Приложение Г. Код программы - распознавателя типа «перенос – свёртка»

class Stack:

...

...

def str(self):

res = ""

for el in self.\_data:

res = res + str(el) + “”

return res

...

...

rules = {

1 : ("X", "S"),

2 : ("S", "OS"),

3 : ("S", "O"),

4 : ("O", "a[S]"),

5 : ("O", "a[S][S]"),

6 : ("O", "a=E"),

7 : ("E", "E+T"),

8 : ("E", "T"),

9 : ("T", "T\*P"),

10 : ("T", "P"),

11 : ("P", "(E)"),

12 : ("P", "a")

}

table = {

"a" :

["P2", "P9", "C12", "", "P2", "C10", "", "C8", "", "C12", #0

"P2", "P9", "P9", "", "P9", "C11", "", "C16", "C7", "C9", #10

"C4", "P2", "", "C5"], #20

"[" :

["", "", "P10", "", "", "", "", "", "", "", #0

"", "", "", "", "", "", "", "", "", "", #10

"P21", "", "", ""], #20

"]" :

["", "", "C12", "", "C3", "C10", "", "C8", "", "C12", #0

"", "", "", "C2", "", "C11", "P20", "C6", "C7", "C9", #10

"C4", "", "P23", "C5"], #20

"+" :

["", "", "C12", "P12", "", "C10", "", "С8", "P12", "C12", #0

"", "", "", "", "", "C11", "", "P12", "C7", "C9"], #10

"\*" :

["", "", "C12", "", "", "C10", "", "P14", "", "C12", #0

"", "", "", "", "", "C11", "", "", "P14", "C9"], #10

"=" :

["", "", "P11"], #0

"(" :

["P1", "P1", "", "", "", "", "", "", "", "", #0

"", "P1", "P1", "", "P1", "", "", "", "", "", #10

"", "", "", "", ""], #20

")" :

["", "", "C12", "", "", "C10", "", "С8", "P15", "C12", #0

"", "", "", "", "", "C11", "", "", "C7", "C9"], #20

"0" :

["", "", "C12", "", "C3", "C10", "D", "С8", "", "C12", #0

"", "", "", "C2", "", "C11", "", "C6", "C7", "C9", #10

"C4", "", "", "C5"],

"S" :

["V6", "", "", "", "V13", "", "", "", "", "", #0

"V16", "", "", "", "", "", "", "", "", "", #10

"", "V22"],

"O" :

["V4", "", "", "", "V4", "", "", "", "", "", #0

"V4", "", "", "", "", "", "", "", "", "", #10

"", "v4"],

"T" :

["V7", "V7", "", "", "", "", "", "", "", "", #0

"", "V7", "V18", "", "", "", "", "", "", "", #10

"", "", "", "", ""],

"P" :

["V5", "V5", "", "", "", "", "", "", "", "", #0

"", "V5", "V5", "", "V19"],

"E" :

["V3", "V8", "", "", "", "", "", "", "", "", #0

"", "V17"]

}

magNum = Stack()

def check(chain):

magNum.clear()

magNum.push(0)

chain += "0"

imax = len(chain)

currentCond = 0

nonTetm = ""

i = 0

imax = len(chain)

print("%-14s %-40s %-25s" %("Правило", "Необработанная часть входной цепочки",

"Содержимое магазина"))

try:

while (i < imax):

sym = chain[i]

action = table[sym][currentCond]

type = action[0]

if action == "D":

print("Допустить")

break

number = int(action[1:])

if type == "P":

magNum.push(number)

currentCond = number

i += 1

elif type == "C" or type == "С":

rightPart = rules[number][1]

nLet = len(rightPart)

nonTerm = rules[number][0]

print("%2s.%-1s-> %-7s %-40s %-25s" %(number, nonTerm, rightPart,

chain[i+1:], magNum.str()))

for k in range(nLet):

magNum.pop()

currentCond = magNum.top()

V = table[nonTerm][currentCond]

if len(V) == 0:

break

else:

currentCond = int(V[1:])

magNum.push(currentCond)

except Exception:

print("Отвергнуть")