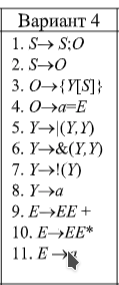
Восходящая обработка контекстно-свободных языков методом «перенос-опознание»

Пункт 1

Преобразовать исходную КС-грамматику (см. варианты заданий) в грамматику без правой рекурсии.



Пункт 2

Найдем множество первых

FIRST(S) = S O { a

FIRST(O) = O { a

FIRST(Y) = Y | & ! a

FIRST(E) = E a

Пункт 3

Найдем множество следующих

FOLLOW(S) = -| ; ]

FOLLOW(O) = -| ; ]

FOLLOW(Y) = [ , )

FOLLOW(E) = -| ; ] + \* a

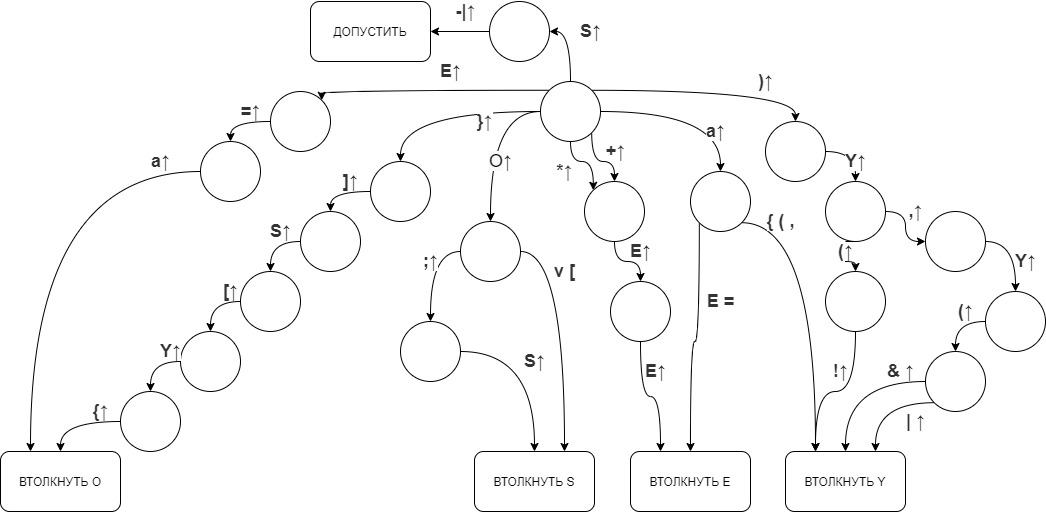
Пункт 4

Построить управляющую таблицу восходящего МП-распознавателя типа «перенос-опознание». Если не удается построить управляющую таблицу МП-распознавателя без конфликтов типа «перенос-опознание», то преобразовать грамматику в грамматику, допускающую построение МП-распознавателя без конфликтов типа «перенос-опознание» и выполнить п.п. 2 — 4.

| НАД |  | { | [ | ] | } | a | = | ( | | | ) | ! | & | + | \* | ; | , | -| |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S |  |  | П |  |  |  |  |  |  |  |  |  |  | П |  | ОП |
|  | O |  |  | ОП |  |  |  |  |  |  |  |  |  |  | ОП |  | ОП |
|  | Y |  | П |  |  |  |  |  |  | П |  |  |  |  |  | П |  |
| E | E |  |  | ОП |  | П |  |  |  |  |  |  | П | П | ОП |  | ОП |
| Y | { |  |  |  |  | П |  |  | П |  | П | П |  |  |  |  |  |
| S O | [ | П |  |  |  | П |  |  |  |  |  |  |  |  |  |  |  |
|  | ] |  |  |  | П |  |  |  |  |  |  |  |  |  |  |  |  |
|  | } |  |  | ОП |  |  |  |  |  |  |  |  |  |  | ОП |  | ОП |
|  | a |  | ОП | ОП |  | ОП | П |  |  | ОП |  |  | ОП | ОП | ОП | ОП | ОП |
| E | = |  |  |  |  | П |  |  |  |  |  |  |  |  |  |  |  |
| Y | ( |  |  |  |  | П |  |  | П |  | П | П |  |  |  |  |  |
|  | | |  |  |  |  |  |  | П |  |  |  |  |  |  |  |  |  |
|  | ) |  | ОП |  |  |  |  |  |  | ОП |  |  |  |  |  | ОП |  |
|  | ! |  |  |  |  |  |  | П |  |  |  |  |  |  |  |  |  |
|  | & |  |  |  |  |  |  | П |  |  |  |  |  |  |  |  |  |
|  | + |  |  | ОП |  | ОП |  |  |  |  |  |  | ОП | ОП | ОП |  | ОП |
|  | \* |  |  | ОП |  | ОП |  |  |  |  |  |  | ОП | ОП | ОП |  | ОП |
| O | ; | П |  |  |  | П |  |  |  |  |  |  |  |  |  |  |  |
| Y | , |  |  |  |  | П |  |  | П |  | П | П |  |  |  |  |  |
| S O | V | П |  |  |  | П |  |  |  |  |  |  |  |  |  |  |  |

Пункт 5

Построить процедуру опознания в виде конечного автомата.



Пунтк 6

Написать программу-распознаватель, реализующую построенный восходящий МП-распознаватель. После выполнения свертки программа должна выводить номер правила, по которому выполняется свертка, и строку, состоящую из содержимого магазина и необработанной части входной цепочки.

Код программы

G = [(**"S"**, **"S;O"**),

(**"S"**, **"O"**),

(**"O"**, **"{Y[S]}"**),

(**"O"**, **"a=E"**),

(**"Y"**, **"|(Y,Y)"**),

(**"Y"**, **"&(Y,Y)"**),

(**"Y"**, **"!(Y)"**),

(**"Y"**, **"a"**),

(**"E"**, **"EE+"**),

(**"E"**, **"EE\*"**),

(**"E"**, **"a"**), ]

T = {

**"S"**: {

**"]"**: **"П"**,

**";"**: **"П"**,

**"-"**: **"П"**,

},

**"O"**: {

**"]"**: **"ОП"**,

**";"**: **"ОП"**,

**"-"**: **"ОП"**,

},

**"Y"**: {

**"["**: **"П"**,

**")"**: **"П"**,

**","**: **"П"**,

},

**"E"**: {

**"-"**: **"ОП"**,

**";"**: **"ОП"**,

**"\*"**: **"П"**,

**"+"**: **"П"**,

**"a"**: **"П"**,

**"]"**: **"ОП"**,

},

**"{"**: {

**"&"**: **"П"**,

**"!"**: **"П"**,

**"|"**: **"П"**,

**"a"**: **"П"**,

},

**"["**: {

**"a"**: **"П"**,

**"{"**: **"П"**,

},

**"]"**: {

**"}"**: **"П"**,

},

**"}"**: {

**"-"**: **"ОП"**,

**";"**: **"ОП"**,

**"]"**: **"ОП"**,

},

**"a"**: {

**"-"**: **"ОП"**,

**","**: **"ОП"**,

**";"**: **"ОП"**,

**"\*"**: **"ОП"**,

**"+"**: **"ОП"**,

**")"**: **"ОП"**,

**"="**: **"П"**,

**"a"**: **"ОП"**,

**"]"**: **"ОП"**,

**"["**: **"ОП"**,

},

**"="**: {

**"a"**: **"П"**,

},

**"("**: {

**"&"**: **"П"**,

**"!"**: **"П"**,

**"|"**: **"П"**,

**"a"**: **"П"**,

},

**"|"**: {

**"("**: **"П"**,

},

**")"**: {

**","**: **"ОП"**,

**")"**: **"ОП"**,

**"["**: **"ОП"**,

},

**"!"**: {

**"("**: **"П"**,

},

**"&"**: {

**"("**: **"П"**,

},

**"+"**: {

**"-"**: **"ОП"**,

**";"**: **"ОП"**,

**"\*"**: **"ОП"**,

**"+"**: **"ОП"**,

**"a"**: **"ОП"**,

**"]"**: **"ОП"**,

},

**"\*"**: {

**"-"**: **"ОП"**,

**";"**: **"ОП"**,

**"\*"**: **"ОП"**,

**"+"**: **"ОП"**,

**"a"**: **"ОП"**,

**"]"**: **"ОП"**,

},

**";"**: {

**"a"**: **"П"**,

**"{"**: **"П"**,

},

**","**: {

**"&"**: **"П"**,

**"!"**: **"П"**,

**"|"**: **"П"**,

**"a"**: **"П"**,

},

**"V"**: {

**"a"**: **"П"**,

**"{"**: **"П"**,

},

}

P = {

**"S"**: **"[V"**,

**"O"**: **"[;V"**,

**"Y"**: **"{(,"**,

**"E"**: **"E="**,

}

class UpstreamRecognizer:

def \_\_init\_\_(self, \_G, \_T, \_P):

self.G = \_G

self.T = \_T

self.P = \_P

def recognize(self, string: str):

string += **"-"**

stack = **"V"**

rule\_seq = []

while string:

t\_row = self.T[stack[-1]]

t\_value = t\_row[string[0]]

if t\_value == **"П"**:

stack += string[0]

string = string[1:]

elif t\_value == **"ОП"**:

if stack[-1] == **"S"** and string == **"-"**:

break

for index\_rule, rule in enumerate(self.G):

len\_stack\_without\_rule = len(stack) - len(rule[1])

if rule[1] == stack[len\_stack\_without\_rule:] and \

stack[len\_stack\_without\_rule-1] in self.P[rule[0]]:

stack = stack[:len\_stack\_without\_rule] + rule[0]

rule\_seq.append(**f"**{index\_rule+1}**)**{rule[0]} **->** {rule[1]}**"**)

print(**f"Свертка по правилу** {index\_rule+1}**)**{rule[0]} **->** {rule[1]}\n**Магазин:**{stack}\n**Цепочка:**{string}\n**"**)

break

else:

raise self.RecognizeException(**f"Ошибка:**\n**Магазине:** {stack}\n**Цепочка:** {string}\n**"**)

rule\_seq.reverse()

print(**"Правый вывод:"**)

for rule in rule\_seq:

print(rule)

print(**"Допустить"**)

class RecognizeException(Exception):

pass

if \_\_name\_\_ == **"\_\_main\_\_"**:

print(**"Введите цепочку: "**)

string = input()

recognizer = UpstreamRecognizer(G, T, P)

try:

recognizer.recognize(string)

except Exception as e:

print(**"Опровергнуть"**)

print(e)

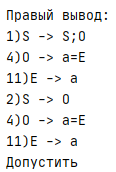
Пункт 7

Сформировать наборы тестовых данных. Тестовые данные должны содержать цепочки, принадлежащие языку, заданному грамматикой, (допустимые цепочки) и цепочки, не принадлежащие языку. Для каждой допустимой цепочки построить дерево вывода и правый вывод. Каждое правило грамматики должно использоваться в выводах допустимых цепочек хотя бы один раз.

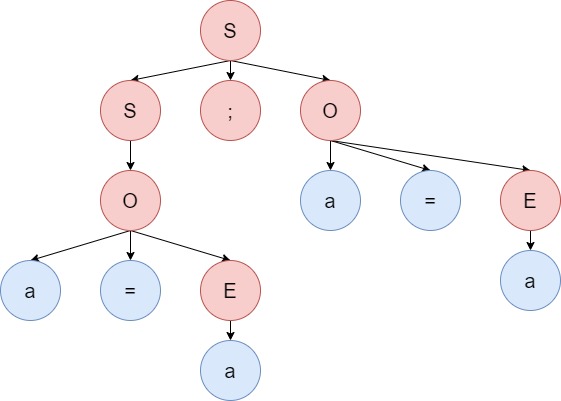
Допустимые цепочки

a=a;a=a

Правый вывод

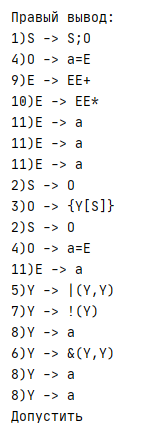


Дерево вывода

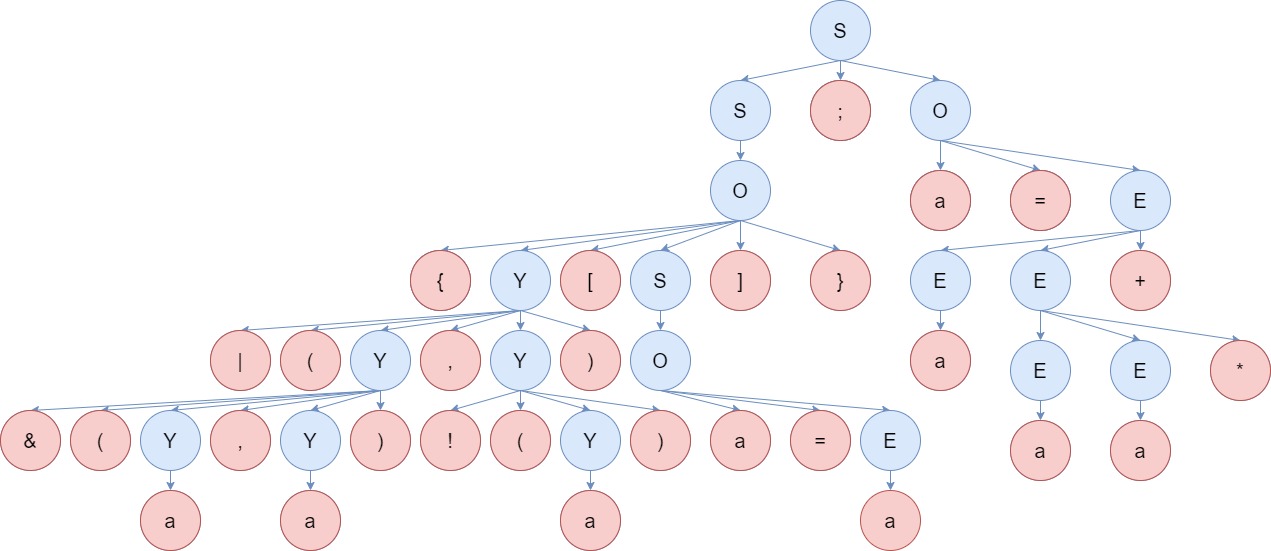


{|(&(a,a),!(a))[a=a]};a=aaa\*+

Правый вывод



Дерево вывода



Недопустимые

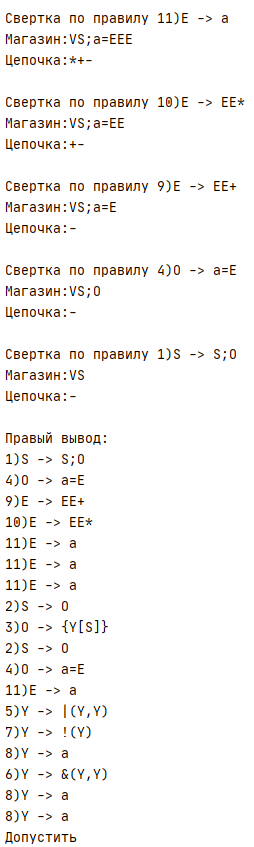
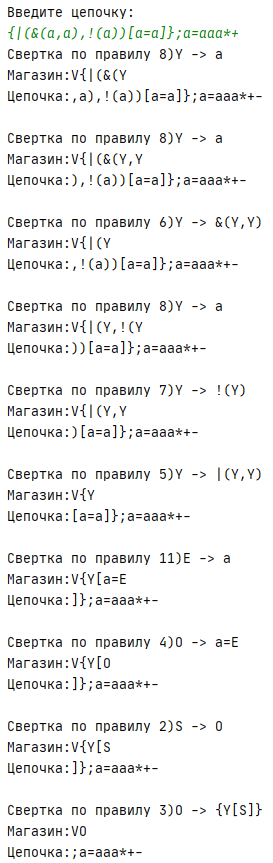
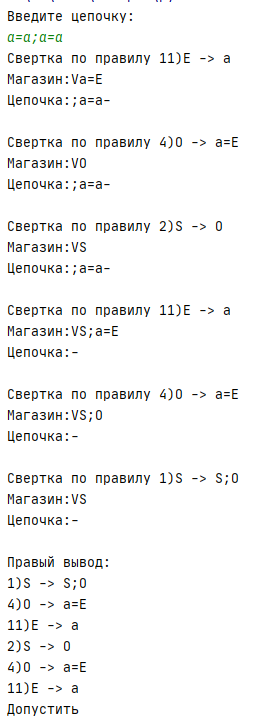
{|(&(a,a),!(a))

+{|(&(a,a),!(a))[a=a]};a=aaa

Пункт 8

Обработать цепочки из набора тестовых данных программой-распознавателем.

Допустимые



Недопустимые

