Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Разработка компиляторов

Домашнее задание №3

Вариант 13

Выполнил:

Маликов Глеб Игоревич

Группа № P3324

Преподаватель:

Лаздин Артур Вячеславович

Санкт-Петербург

2025

Оглавление

Задание

Для грамматики в соответствии с вариантом необходимо:

1. Устранить левую рекурсию (если необходимо).
2. Провести левую факторизацию грамматики (если необходимо).
3. Для полученной преобразованной грамматики построить множества FIRST и FOLLOW для нетерминальных символов грамматики.
4. Для преобразованной грамматики построить таблицу анализатора и разработать программную реализацию этого анализатора.

Примечание: если полученная грамматика окажется не LL(1) грамматикой — конфликты в таблице анализатора — то согласовать со мной изменения в грамматике с целью приведения её к LL(1) виду. Ввиду этого соображения вида: «но грамматика же не подходит под LL(1) анализатор – вот я ничего делать и не стал» — не работают.

Результатом работы является работающий код анализатора.

5. Отчет должен включать:
    a. Исходную грамматику;
    b. Отдельно (для каждого правила) действия по устранению прямой левой рекурсии и отдельно действия для левой факторизации.
    c. Преобразованную грамматику. Внесения изменений при обязательном согласовании со мной.
    d. Таблицы множеств FIRST и FOLLOW для нетерминалов;
    e. Таблица синтаксического анализатора;
    f. Реализацию синтаксического анализатора.
    g. Примеры корректных и ошибочных входных цепочек.
    h. Выводы.

Пункты 1 и 2 можно выполнять в любой последовательности и нужное количество раз.

+10 баллов — обработчик ошибок по крайней мере для двух промахов в таблице анализатора.

Вариант 13:

1. S → AABC
2. A → AAa | Aa | Ab | b
3. B → bA | bB | bC | b
4. C → aa | bb | cc

Выполнение

Преобразование грамматики

В правиле 2 видим левую рекурсию, которую нужно устранить. Также в правиле 3 можно провести левую факторизацию.

1. S → AABC
2. A → AAa | Aa | Ab | b
3. B → bA | bB | bC | b
4. C → aa | bb | cc

Устранение левой рекурсии A

1. S → AABC
2. A → bA’
3. A’ → AaA’ | aA’ | bA’ | ε
4. B → bA | bB | bC | b
5. C → aa | bb | cc

Левая факторизация B

1. S → AABC
2. A → bA’
3. A’ → AaA’ | aA’ | bA’ | ε
4. B → bX
5. X → A | B | C | ε
6. C → aa | bb | cc

Построение таблицы анализатора

VN	Nullable	FIRST	FOLLOW
S	False	b	$
A	False	b	a, b, c
A’	True	a, b	a, b, c
B	False	b	a, b, c
X	True	a, b, c	a, b, c
C	False	a, b, c	a, b, c, $

	a	b	c	$
S	-	S → AABC	-	-
A	-	A → bA’	-	-

A’	A’ → aA’| ε	A’ → AaA’ | bA’ | ε	A’ → ε	-
B	-	B → bX	-	-
X	X → C | ε	X → A | B | C | ε	X → C | ε	-
C	C → aa	C → bb	C → cc	$

Видно что грамматика не является LL(1), поэтому преобразуем грамматику:

1. S → AABC
2. A → bA’
3. A’ → AaA’ | aA’ | bA’ | ε
4. B → bX
5. X → A | B | C | ε
6. C → aa | bb | cc

Добавлено d и заменены A’ → AaA’, X → aA, X → cC и эпсилон правила

1. S → AABC
2. A → bA’
3. A’ → caA’ | aA’ | bA’ | d
4. B → bX
5. X → aA | B | cC | dd
6. C → aa | bb | cc

VN	Nullable	FIRST	FOLLOW
S	False	b	$
A	False	b	a, b, c
A’	False	a, b, c, d	a, b, c
B	False	b	a, b, c, d
X	False	a, b, c, d	a, b, c
C	False	a, b, c	a, b, c, $

	a	b	c	d	$
S	-	S → AABC	-	-	-
A	-	A → bA’	-	-	-
A’	A’ → aA’	A’ → bA’	A’ → caA’	A’ → d	-
B	-	B → bX	-	-	-
X	X → aA	X → B	X → cC	X → dd	-
C	C → aa	C → bb	C → cc	-	-

С готовой таблицей синтаксического анализатора, был реализован

синтаксический анализатор в Питоне.

```
import logging
import colorlog

logger = logging.getLogger()

handler = logging.StreamHandler()

formatter = colorlog.ColoredFormatter(
    "%(log_color)s%(message)s",
    log_colors={
        'INFO': 'green',
        'DEBUG': 'white',
        'ERROR': 'red',
    }
)
handler.setFormatter(formatter)
logger.addHandler(handler)
logger.setLevel(logging.DEBUG)

grammar = {
    'S': {
        'b': ['A', 'A', 'B', 'C']
    },
    'A': {
        'b': ['b', "A'"]
    },
    "A'": {
        'a': ["a", "A'"],
        'b': ["b", "A'"],
        'c': ["c", "a", "A'"],
        'd': ["d"],
    },
    'B': {
        'b': ['b', 'X']
    },
    'X': {
        'a': ['a', 'A'],
        'b': ['B'],
        'c': ['c', 'C'],
        'd': ['d', 'd']
    },
    'C': {
        'a': ['a', 'a'],
        'b': ['b', 'b'],
        'c': ['c', 'c']
    }
}

def parse(input_tokens):
    input_tokens.append('$')
    stack = ['$', 'S']
    index = 0

    while stack:
        logging.debug(f"Стек: {''.join(stack)}, вход:
{''.join(input_tokens[index:])}, позиция: {index + 1} ")
        top = stack.pop()
        current = input_tokens[index]
        if top == current == '$':
            logging.info("Разбор завершён успешно!")
            return True
        # Если верх стека - терминал
```

```
        if top in ['a','b','c','d','$']:
            if top == current:
                index += 1
                logging.debug(f"Совпадение терминала '{top}'")
            else:
                logging.error(f"Ошибка на позиции {index + 1}: ожидалось
'{top}', а найдено '{current}'")
                return False
        else:
            # Нетерминал - ищем правило по таблице
            production = grammar.get(top, {}).get(current)
            if not production:
                logging.error(f"Ошибка на позиции {index + 1}: нет правила
для '{top}' при токене '{current}'")
                return False
            if production != ['ε']:
                logging.debug(f"Применено правило {top} ->
{''.join(production)}")
                # Правило записано как список символов - добавляем их в стек
в обратном порядке
                for symbol in reversed(production):
                    stack.append(symbol)
    return False


tokens = list('bdbdbddcc')
parse(tokens) # правильно
tokens = list('adbabddacc')
parse(tokens) # неправильно
tokens = list('bdbdbccbad')
parse(tokens) # неправильно
tokens = list('bbabcadbdbbddaa')
parse(tokens) # правильно
```

Ниже приведён результат анализа четырёх строк.

```
Стек: $S, вход: bdbdbddcc$, позиция: 1

Применено правило S -> AABC

Стек: $CBAA, вход: bdbdbddcc$, позиция: 1

Применено правило A -> bA'

Стек: $CBAA'b, вход: bdbdbddcc$, позиция: 1

Совпадение терминала 'b'

Стек: $CBAA', вход: dbdbddcc$, позиция: 2

Применено правило A' -> d

Стек: $CBAd, вход: dbdbddcc$, позиция: 2

Совпадение терминала 'd'

Стек: $CBA, вход: bdbddcc$, позиция: 3

Применено правило A -> bA'

Стек: $CBA'b, вход: bdbddcc$, позиция: 3

Совпадение терминала 'b'

Стек: $CBA', вход: dbddcc$, позиция: 4

Применено правило A' -> d
```

```
Стек: $CBd, вход: dbddcc$, позиция: 4
Совпадение терминала 'd'
Стек: $CB, вход: bddcc$, позиция: 5
Применено правило B -> bX
Стек: $CXb, вход: bddcc$, позиция: 5
Совпадение терминала 'b'
Стек: $CX, вход: ddcc$, позиция: 6
Применено правило X -> dd
Стек: $Cdd, вход: ddcc$, позиция: 6
Совпадение терминала 'd'
Стек: $Cd, вход: dcc$, позиция: 7
Совпадение терминала 'd'
Стек: $C, вход: cc$, позиция: 8
Применено правило C -> cc
Стек: $cc, вход: cc$, позиция: 8
Совпадение терминала 'c'
Стек: $c, вход: c$, позиция: 9
Совпадение терминала 'c'
Стек: $, вход: $, позиция: 10
Разбор завершён успешно!
Стек: $S, вход: adbabddacc$, позиция: 1
Ошибка на позиции 1: нет правила для 'S' при токене 'a'
Стек: $S, вход: bdbdbccbad$, позиция: 1
Применено правило S -> AABC
Стек: $CBAA, вход: bdbdbccbad$, позиция: 1
Применено правило A -> bA'
Стек: $CBAA'b, вход: bdbdbccbad$, позиция: 1
Совпадение терминала 'b'
Стек: $CBAA', вход: dbdbccbad$, позиция: 2
Применено правило A' -> d
Стек: $CBAd, вход: dbdbccbad$, позиция: 2
Совпадение терминала 'd'
Стек: $CBA, вход: bdbccbad$, позиция: 3
Применено правило A -> bA'
Стек: $CBA'b, вход: bdbccbad$, позиция: 3
Совпадение терминала 'b'
```

```
Стек: $CBA', вход: dbccbad$, позиция: 4
Применено правило A' -> d
Стек: $CBd, вход: dbccbad$, позиция: 4
Совпадение терминала 'd'
Стек: $CB, вход: bccbad$, позиция: 5
Применено правило B -> bX
Стек: $CXb, вход: bccbad$, позиция: 5
Совпадение терминала 'b'
Стек: $CX, вход: ccbad$, позиция: 6
Применено правило X -> cC
Стек: $CCc, вход: ccbad$, позиция: 6
Совпадение терминала 'c'
Стек: $CC, вход: cbad$, позиция: 7
Применено правило C -> cc
Стек: $Ccc, вход: cbad$, позиция: 7
Совпадение терминала 'c'
Стек: $Cc, вход: bad$, позиция: 8
Ошибка на позиции 8: ожидалось 'c', а найдено 'b'
Стек: $S, вход: bbabcadbdbbddaa$, позиция: 1
Применено правило S -> AABC
Стек: $CBAA, вход: bbabcadbdbbddaa$, позиция: 1
Применено правило A -> bA'
Стек: $CBAA'b, вход: bbabcadbdbbddaa$, позиция: 1
Совпадение терминала 'b'
Стек: $CBAA', вход: babcadbdbbddaa$, позиция: 2
Применено правило A' -> bA'
Стек: $CBAA'b, вход: babcadbdbbddaa$, позиция: 2
Совпадение терминала 'b'
Стек: $CBAA', вход: abcadbdbbddaa$, позиция: 3
Применено правило A' -> aA'
Стек: $CBAA'a, вход: abcadbdbbddaa$, позиция: 3
Совпадение терминала 'a'
Стек: $CBAA', вход: bcadbdbbddaa$, позиция: 4
Применено правило A' -> bA'
Стек: $CBAA'b, вход: bcadbdbbddaa$, позиция: 4
Совпадение терминала 'b'
```

```
Стек: $CBAA', вход: cadbdbbddaa$, позиция: 5
Применено правило A' -> caA'
Стек: $CBAA'ac, вход: cadbdbbddaa$, позиция: 5
Совпадение терминала 'c'
Стек: $CBAA'a, вход: adbdbbddaa$, позиция: 6
Совпадение терминала 'a'
Стек: $CBAA', вход: dbdbbddaa$, позиция: 7
Применено правило A' -> d
Стек: $CBAd, вход: dbdbbddaa$, позиция: 7
Совпадение терминала 'd'
Стек: $CBA, вход: bdbbddaa$, позиция: 8
Применено правило A -> bA'
Стек: $CBA'b, вход: bdbbddaa$, позиция: 8
Совпадение терминала 'b'
Стек: $CBA', вход: dbbddaa$, позиция: 9
Применено правило A' -> d
Стек: $CBd, вход: dbbddaa$, позиция: 9
Совпадение терминала 'd'
Стек: $CB, вход: bbddaa$, позиция: 10
Применено правило B -> bX
Стек: $CXb, вход: bbddaa$, позиция: 10
Совпадение терминала 'b'
Стек: $CX, вход: bddaa$, позиция: 11
Применено правило X -> B
Стек: $CB, вход: bddaa$, позиция: 11
Применено правило B -> bX
Стек: $CXb, вход: bddaa$, позиция: 11
Совпадение терминала 'b'
Стек: $CX, вход: ddaa$, позиция: 12
Применено правило X -> dd
Стек: $Cdd, вход: ddaa$, позиция: 12
Совпадение терминала 'd'
Стек: $Cd, вход: daa$, позиция: 13
Совпадение терминала 'd'
Стек: $C, вход: aa$, позиция: 14
Применено правило C -> aa
```

```
Стек: $aa, вход: aa$, позиция: 14
Совпадение терминала 'a'
Стек: $a, вход: a$, позиция: 15
Совпадение терминала 'a'
Стек: $, вход: $, позиция: 16
Разбор завершён успешно!
```

Заключение

В ходе выполнения лабораторной работы были реализованы этапы построения синтаксического анализатора для заданной грамматики. Была проведена обработка исходной грамматики: устранение левой рекурсии и выполнение левой факторизации. Далее были сформированы множества FIRST и FOLLOW для всех нетерминальных символов. Была переделано грамматика из-за конфликтов и были внесены изменения в грамматику. Результатом работы стала разработка синтаксического анализатора на языке Python.

Таким образом, данная лабораторная работа позволила глубже изучить методы преобразования грамматик, построения таблиц FIRST и FOLLOW, а также практические аспекты разработки LL(1)-анализатора.