Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление программная инженерия
Образовательная программа системное и прикладное программное обеспечение
Специализация системное программное обеспечение

ОТЧЕТ ПО ДОМАШНЕЙ РАБОТЕ № 3

курса «Разработка компиляторов»

по теме: «Конструирование LL(1) анализатора для КС-грамматики»

Вариант № 9

Выполнил студент:
Тюрин Иван Николаевич
группа: P33102

Преподаватель:
Лаздин А. В.

Санкт-Петербург, 2024 г.

Содержание

Домашняя работа № 3
Конструирование LL(1) анализатора для КС-грамматики

1. Задание варианта № 9

Исходный вариант:

$$
\begin{aligned}
&S \to ABCC \\
&C \to cccA \mid ccBB \mid cC \mid c \\
&B \to BBb \mid BBa \mid b \\
&A \to aAa \mid c
\end{aligned}
$$

Упрощение (убраны все подряд идущие нетерминалы, которые раскрываются в бесконечные последовательности):

$$
\begin{aligned}
&S \to ABC \\
&C \to cccA \mid ccB \mid cC \mid c \\
&B \to Bb \mid Ba \mid b \\
&A \to aAa \mid c
\end{aligned}
$$

Факторизация C:

$$C \to cccA \mid ccB \mid cC \mid c$$

$$
\begin{aligned}
C &\to cC_1 \\
C_1 &\to ccA \mid cB \mid cC_1 \mid \varepsilon
\end{aligned}
$$

$$
\begin{aligned}
C &\to cC_1 \\
C_1 &\to cC_2 \mid \varepsilon \\
C_2 &\to cA \mid B \mid C_1
\end{aligned}
$$

$$
\begin{aligned}
C &\to cC_1 \\
C_1 &\to cC_2 \mid \varepsilon \\
C_2 &\to cA \mid B \mid cC_2 \mid \varepsilon
\end{aligned}
$$

$$
\begin{aligned}
C &\to cC_1 \\
C_1 &\to cC_2 \mid \varepsilon \\
C_2 &\to cC_3 \mid B \mid \varepsilon \\
C_3 &\to A \mid C_2
\end{aligned}
$$

$$
\begin{aligned}
C &\to cC_1 \\
C_1 &\to cC_2 \mid \varepsilon \\
C_2 &\to cC_3 \mid B \mid \varepsilon \\
C_3 &\to A \mid cC_3 \mid B \mid \varepsilon
\end{aligned}
$$

$$
\begin{aligned}
C &\to cC_1 \\
C_1 &\to cC_2 \mid \varepsilon \\
C_2 &\to cC_3 \mid B \mid \varepsilon \\
C_3 &\to c \mid aAa \mid cC_3 \mid B \mid \varepsilon
\end{aligned}
$$

$$
\begin{aligned}
C &\to cC_1 \\
C_1 &\to cC_2 \mid \varepsilon \\
C_2 &\to cC_3 \mid B \mid \varepsilon \\
C_3 &\to aAa \mid cC_4 \mid B \mid \varepsilon \\
C_4 &\to C_3 \mid \varepsilon
\end{aligned}
$$

Видно, что $C_4 = C_3$, поэтому уберем это правило:

$$
\begin{aligned}
C &\to cC_1 \\
C_1 &\to cC_2 \mid \varepsilon \\
C_2 &\to cC_3 \mid B \mid \varepsilon \\
C_3 &\to aAa \mid cC_3 \mid B \mid \varepsilon
\end{aligned}
$$

Факторизация B:

$$
B \to Bb \mid Ba \mid b
$$

$$
\begin{aligned}
B &\to BB_1 \mid b \\
B_1 &\to b \mid a
\end{aligned}
$$

Устраняем левую рекурсию B:

$$
\begin{aligned}
B &\to bB_2 \\
B_1 &\to b \mid a \\
B_2 &\to B_1B_2 \mid \varepsilon
\end{aligned}
$$

Итоговая грамматика:

$$
\begin{aligned}
S &\to ABC \\
C &\to cC_1 \\
C_1 &\to cC_2 \mid \varepsilon \\
C_2 &\to cC_3 \mid B \mid \varepsilon \\
C_3 &\to A \mid cC_3 \mid B \mid \varepsilon \\
B &\to bB_2 \\
B_1 &\to b \mid a \\
B_2 &\to B_1B_2 \mid \varepsilon \\
A &\to aAa \mid c
\end{aligned}
$$

FIRST	**FOLLOW**	**Name**	**c**	**a**	**b**	**$**
$\{a, c\}$	$\{\$\}$	S	$S \to ABC$	$S \to ABC$		
$\{c\}$	$\{\$\}$	C	$C \to cC_1$			
$\{c, \varepsilon\}$	$\{\$\}$	C_1	$C_1 \to cC_2$			$C_1 \to \varepsilon$
$\{c, \varepsilon, b\}$	$\{\$\}$	C_2	$C_2 \to cC_3$		$C_2 \to B$	$C_2 \to \varepsilon$
$\{a, c, \varepsilon, b\}$	$\{\$\}$	C_3	$C_3 \to cC_3$	$C_3 \to aAa$	$C_3 \to B$	$C_3 \to \varepsilon$
$\{b\}$	$\{c, \$\}$	B			$B \to bB_2$	
$\{b, a\}$	$\{b, a, c, \$\}$	B_1		$B_1 \to a$	$B_1 \to b$	
$\{b, a, \varepsilon\}$	$\{c, \$\}$	B_2	$B_2 \to$	$B_2 \to B_1B_2$	$B_2 \to B_1B_2$	$B_2 \to \varepsilon$
$\{a, c\}$	$\{b, a\}$	A	$A \to c$	$A \to aAa$		

Таблица 1.1: Сконструированный СА

Для сконструированного СА разработаем программный код выполняющий аналогичную функцию, см. 1.1. Программный код составляет код на языке DOT для визуализации в среде Graphviz.

```
class SA:
    id = 0
    s: str = None
    idx = 1
    cur_char: chr = None


    def __init__(self, string: str):
        self.s = string
        self.cur_char = self.s[0]


    def get_next_char(self):
        self.s += "$"
        self.idx += 1
        return self.s[self.idx - 1]

```

```
    def preambula(self, from_id, name):
        local_id = self.id
        print(f'{local_id} [label = "{name}"];')
        self.id += 1
        print(f"{from_id} -> {local_id};")

        return local_id


    def B1(self, from_id):
        self.preambula(from_id, "B1")

        if self.cur_char in ["b", "a"]:
            self.cur_char = self.get_next_char()
        else:
            raise ValueError


    def B2(self, from_id):
        local_id = self.preambula(from_id, "B2")

        if self.cur_char in ["c", "$"]:
            pass
        elif self.cur_char in ["b", "a"]:
            self.B1(local_id)
            self.B2(local_id)
        else:
            raise ValueError


    def B(self, from_id):
        local_id = self.preambula(from_id, "B")

        if self.cur_char != "b":
            raise ValueError

        self.cur_char = self.get_next_char()
        self.B2(local_id)


    def A(self, from_id):
        local_id = self.preambula(from_id, "A")

        if self.cur_char == "c":
            self.cur_char = self.get_next_char()
        elif self.cur_char == "a":
            self.cur_char = self.get_next_char()
            self.A(local_id)
            if self.cur_char != "a":
                raise ValueError
            self.cur_char = self.get_next_char()
        else:
            raise ValueError


    def C3(self, from_id):
        local_id = self.preambula(from_id, "C3")

        if self.cur_char == "c":
            self.cur_char = self.get_next_char()
```

```
            self.C3(local_id)
        elif self.cur_char == "b":
            self.B(local_id)
        elif self.cur_char == "a":
            self.cur_char = self.get_next_char()
            self.A(local_id)
            if self.cur_char != "a":
                raise ValueError
            self.cur_char = self.get_next_char()
        elif self.cur_char == "$":
            pass
        else:
            raise ValueError


    def C2(self, from_id):
        local_id = self.preambula(from_id, "C2")

        if self.cur_char == "c":
            self.cur_char = self.get_next_char()
            self.C3(local_id)
        elif self.cur_char == "b":
            self.B(local_id)
        elif self.cur_char == "$":
            pass
        else:
            raise ValueError


    def C1(self, from_id):
        local_id = self.preambula(from_id, "C1")

        if self.cur_char == "c":
            self.cur_char = self.get_next_char()
            self.C2(local_id)
        elif self.cur_char == "$":
            pass
        else:
            raise ValueError


    def C(self, from_id):
        local_id = self.preambula(from_id, "C")

        if self.cur_char == "c":
            self.cur_char = self.get_next_char()
            self.C1(local_id)
        else:
            raise ValueError


    def S(self):
        local_id = self.id
        print(f'{local_id} [label = "S"];')
        self.id += 1
        self.A(local_id)
        self.B(local_id)
        self.C(local_id)

if __name__ == "__main__":
```

```
    sa = SA(input())
    print("digraph {")
    sa.S()
    print("}")
```

Листинг 1.1: Python-код синтаксического анализатора

Пример работы программы для строки `acabbc` можно видеть на листинге 1.2. И получившийся граф можно видеть на рисунке 1.1.

```
digraph {
    0 [label = "S"];
    1 [label = "A"];
    0 -> 1;
    2 [label = "A"];
    1 -> 2;
    3 [label = "B"];
    0 -> 3;
    4 [label = "B2"];
    3 -> 4;
    5 [label = "B1"];
    4 -> 5;
    6 [label = "B2"];
    4 -> 6;
    7 [label = "C"];
    0 -> 7;
    8 [label = "C1"];
    7 -> 8;
}
```

Листинг 1.2: Пример вывода программы для строки acabbc

2. Вывод

Изучили принципы построения синтаксических анализаторов, способы устранения левой рекурсии и выполнения левой факторизации. Построили синтаксический анализатор для грамматики выданной по варианту и реализовали его поведение в программном коде.

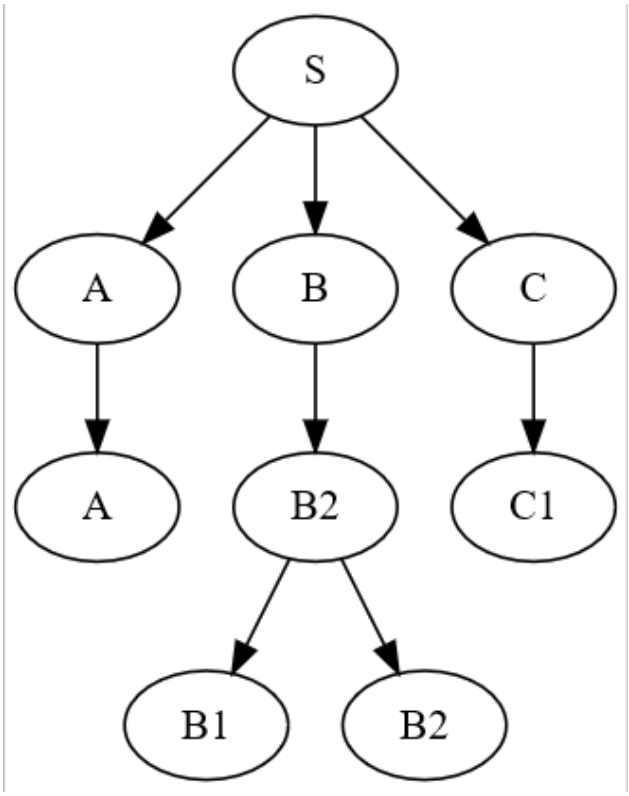

Рис. 1.1: Получившийся граф разбора для строки acabbc