Филипенков В. А.
ИУ7-22М

Отчет по **ЛР№ 2** по дисциплине
«Конструирование компиляторов»

ПРЕОБРАЗОВАНИЕ ГРАММАТИК

Вариант 1.

1. Задание

Постройте программу, которая в качестве входа принимает произвольную КС-грамматику G = (N, ∑, P, S) и преобразует ее в эквивалентную КС-грамматику G' = (N', ∑', P', S') без левой рекурсии и не содержащую недостижимых символов.

2. Текст программы (алгоритмов)

Устранение левой рекурсии

```
def LR_elimination(g: Grammar):
    new_rules: list[Rule] = g.rules.copy()

    nts = g.nonterminals.copy()

    for i, i_nt in enumerate(g.nonterminals):
        for j_nt in g.nonterminals[:i]:
            rules = filter(lambda x: x.left == i_nt and len(x.right) and
x.right[0] == j_nt, new_rules)
            for rule in rules:
                new_rules.remove(rule)
                m_rules = filter(lambda x: x.left == j_nt, new_rules)
                for m_rule in m_rules:
                    new_rules.append(
                        Rule(
                            i_nt, m_rule.right + rule.right[1:]
                        )
                    )
        rules = list(filter(lambda x: x.left == i_nt, new_rules))
        need_modify = False
        for rule in rules:
            if len(rule.right) and rule.right[0] == i_nt:
                need_modify = True
                break

        if need_modify:
```

```
            new_nt = i_nt + "'"
            nts += [new_nt]

            for rule in rules:
                new_rules.remove(rule)
                if len(rule.right) and rule.right[0] == i_nt:
                    new_rules += [
                        Rule(new_nt, rule.right[1:]),
                        Rule(new_nt, rule.right[1:] + [new_nt]),
                    ]
                else:
                    new_rules += [
                        Rule(rule.left, rule.right.copy()),
                        Rule(rule.left, rule.right + [new_nt]),
                    ]

    return Grammar(nts, g.terminals.copy(), new_rules, g.axiom)
```

Устранение недостижимых символов

```
def UnRS_elimination(g: Grammar):
    queue = [g.axiom]
    nts = [ g.axiom]

    new_rules = []

    while len(queue):
        nonterminals = queue.pop(0)

        for rule in filter(lambda x: x.left == nonterminals, g.rules):
            new_rules.append(rule)
            for symbol in rule.right:
                if symbol in g.nonterminals and symbol not in nts:
                    nts.append(symbol)
                    queue.append(symbol)

    return Grammar(nts, g.terminals, new_rules, g.axiom)
```

3. Тесты

Устранение левой рекурсии

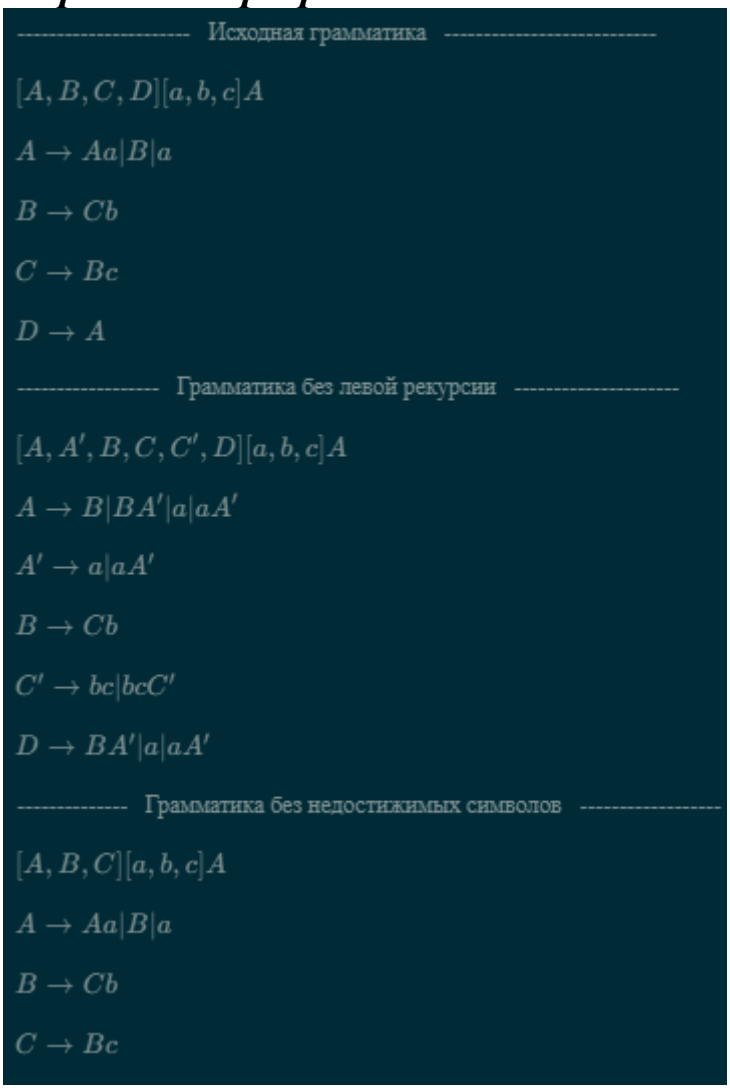
Входная грамматика	*Результат*
$[A][a]A$ $A \to aA$	$[A][a]A$ $A \to aA$

$[A, B][a, b]A$ $A \to Ba \mid a$ $B \to Ab \mid b$	$[A, B, B'][a, b]A$ $A \to Ba \mid a$ $B \to ab \mid abB' \mid b \mid bB'$ $B' \to ab \mid abB'$
$[A][+, -, a]A$ $A \to A + A \mid A - A \mid a$	$[A, A'][+, -, a]A$ $A \to a \mid aA'$ $A' \to +A \mid +AA' \mid -A \mid -AA'$
$[A, B, C][a, b, c]A$ $A \to Aa \mid a$ $B \to Cb$ $C \to Bc$	$[A, A', B, C, C'][a, b, c]A$ $A \to a \mid aA'$ $A' \to a \mid aA'$ $B \to Cb$ $C' \to bc \mid bcC'$

Устранение недостижимых символов

Входная грамматика	*Результат*
$[A][a]A$ $A \to aA$	$[A][a]A$ $A \to aA$
$[A, B, C][a, b, c]A$ $A \to Aa \mid a$ $B \to Cb$ $C \to Bc$	$[A][a, b, c]A$ $A \to Aa \mid a$

4. *Результаты работы программы*



5. *Ответы на контрольные вопросы*

1) Как может быть определён формальный язык?

- Перечисление слов языка.
- Слова, порождённые некоторой формальной грамматикой
- Слова, порождённые регулярным выражением.
- Слова, распознаваемые некоторым КА

2) Какими характеристиками определяется грамматика?

$\sum$ – множество терминальных символов
N – множество нетерминальных символов
P – множество правил

- слева – непустая последовательность (не)терминалов, содержащая хотя бы один нетерминал
- справа – любая последовательность (не)терминалов)

S – начальный символ из множества нетерминалов

3) Дайте описания грамматик по иерархии Хомского.

- *Неограниченные* – грамматики с фразовой структурой
- *Контекстно-зависимые* – КЗ и неукорачивающие грамматики
- *Контекстно-свободные* – грамматика допускает появление в левой части правила только нетерминального символа
- *Регулярные* – КС грамматики с ограничениями

4) Какие абстрактные устройства используются для разбора грамматик?

- Распознающие грамматики – устройства, принимающие цепочку языка, и выводящие условное «ОК», если цепочка принадлежит языку, и «ERROR» – в противном случае.
- Порождающие грамматики - устройства для порождения цепочек языков по требованию.

5) Оцените временную и емкостную сложность предложенного вам алгоритма.

Устранение левой рекурсии: t – $O(N^2)$, m – O(N)
Устранение недостижимых символов: t – $O(N^2)$, m – O(N)