Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА № 2

ТЕМА:

«Преобразования грамматик»

ВАРИАНТ №4

Группа: ИУ7-21М

Студент: Дубовицкая Ольга Николаевна

Дисциплина: Конструирование компиляторов

Преподаватель: Ступников Андрей Алексеевич

Москва, 2024 г.

ЛАБОРАТОРНАЯ РАБОТА №2

ПРЕОБРАЗОВАНИЯ ГРАММАТИК

Цель работы: приобретение практических навыков реализации наиболее важных (но не всех) видов преобразований грамматик, чтобы удовлетворить требованиям алгоритмов синтаксического разбора.

Задачи работы:

1) Принять к сведению соглашения об обозначениях, принятые в литературе по теории формальных языков и грамматик и кратко описанные в приложении.
2) Познакомиться с основными понятиями и определениями теории формальных языков и грамматик.
3) Детально разобраться в алгоритме устранения левой рекурсии.
4) Разработать, тестировать и отладить программу устранения левой рекурсии.
5) Разработать, тестировать и отладить программу преобразования грамматики в соответствии с предложенным вариантом.

Содержание работы:

Общий вариант для всех: Устранение левой рекурсии.

Постройте программу, которая в качестве входа принимает приведенную КС-грамматику G = (N, Σ, P, S) и преобразует ее в эквивалентную КС-грамматику G' без левой рекурсии.

Вариант 4. Устранение цепных правил.

Постройте программу, которая в качестве входа принимает произвольную КС-грамматику G = (N, Σ, P, S) без ε-правил и преобразует ее в эквивалентную КС-грамматику G' = (N, Σ, P', S) без ε-правил и без цепных правил.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Нетерминал A КС-грамматики G = (N, Σ, P, S) называется **рекурсивным**, если A =>+ αAβ для некоторых α и β.

Если α = ε, то A называется **леворекурсивным**.

Аналогично, если β = ε, то A называется **праворекурсивным**.

Грамматика, имеющая хотя бы один леворекурсивный нетерминал, называется **леворекурсивной**.

Аналогично определяется **праворекурсивная** грамматика.

Грамматика, в которой все нетерминалы, кроме, быть может, начального символа, рекурсивные, называется **рекурсивной**.

Алгоритм 4.8. Устранение левой рекурсии

ВХОД: грамматика G без циклов и ϵ-продукций.

ВЫХОД: эквивалентная грамматика без левой рекурсии.

МЕТОД: применить алгоритм, приведенный на рис. 4.11. Обратите внимание, что получающаяся грамматика без левых рекурсий может иметь ϵ-продукции. □

1) Расположить нетерминалы в некотором порядке $A_1, A_2, \ldots, A_n$.
2) **for** (каждое i от 1 до n) {
3) **for** (каждое j от 1 до $i-1$) {
4) заменить каждую продукцию вида $A_i \to A_j$ продукциями $A_i \to \delta_1\gamma \mid \delta_2\gamma \mid \cdots \mid \delta_k\gamma$, где $A_j \to \delta_1 \mid \delta_2 \mid \cdots \mid \delta_k$ — все текущие A_j-продукции
5) }
6) устранить непосредственную левую рекурсию среди A_i-продукций
7) }

Рис. 4.11. Алгоритм для устранения левой рекурсии из грамматики

Процедура на рис. 4.11 работает следующим образом. В первой итерации для $i = 1$ внешний цикл в строках 2–7 устраняет непосредственную левую рекурсию из A_1-продукций. Все остальные A_1-продукции вида $A_1 \to A_l\alpha$ должны, таким образом, иметь $l > 1$. После $i-1$-й итерации внешнего цикла все нетерминалы A_k, где $k < i$, оказываются "чистыми", т.е. все продукции $A_k \to A_l\alpha$ должны иметь $l > k$. В результате на i-й итерации внутренний цикл в строках 3–5 постепенно поднимает нижнюю границу всех продукций $A_i \to A_m\alpha$, пока мы не получим $m \geqslant i$. Затем устранение непосредственной левой рекурсии из A_i-продукций в строке 6 делает m больше i.

Алгоритм 4.10. Левая факторизация грамматики

ВХОД: грамматика G.

ВЫХОД: эквивалентная левофакторизованная грамматика.

МЕТОД: для каждого нетерминала A находим самый длинный префикс α, общий для двух или большего числа альтернатив. Если $\alpha \neq \epsilon$, т.е. имеется нетривиальный общий префикс, заменим все продукции $A \to \alpha\beta_1 \mid \alpha\beta_2 \mid \cdots \mid \alpha\beta_n \mid \gamma$, где γ представляет все альтернативы, не начинающиеся с α, продукциями

$$\begin{aligned} A &\to \alpha A' \mid \gamma \\ A' &\to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n \end{aligned}$$

Здесь A' — новый нетерминал. Выполняем это преобразование до тех пор, пока никакие две альтернативы нетерминала не будут иметь общий префикс. □

Правила вида A –> B, где A ∈ N и B ∈ N, будем называть **цепными**.

Алгоритм 2.11. Устранение цепных правил.

Вход. КС-грамматика G без e-правил.

Выход. Эквивалентная КС-грамматика G' без e-правил и без цепных правил.

Метод.

(1) Для каждого $A \in N$ построить $N_A = \{B \mid A \Rightarrow^* B\}$ следующим образом:

(а) Положить $N_0 = \{A\}$ и $i = 1$.

(б) Положить $N_i = \{C \mid B \to C$ принадлежит P и $B \in N_{i-1}\} \cup N_{i-1}$.

(в) Если $N_i \neq N_{i-1}$, положить $i = i + 1$ и повторить шаг (б). В противном случае положить $N_A = N_i$.

(2) Построить P' так: если $B \to \alpha$ принадлежит P и не является цепным правилом, включить в P' правило $A \to \alpha$ для всех таких A, что $B \in N_A$.

(3) Положить $G' = (N, \Sigma, P', S)$. □

РЕЗУЛЬТАТЫ РАБОТЫ

УДАЛЕНИЕ ЛЕВОЙ РЕКУРСИИ

Пример 1

example.txt

```
S E
i t e a
S -> i E t S | i E t S e S | a
E -> b
S
```

input_grammar.txt

```
2
S E
4
i t e a
4
S -> i E t S
S -> i E t S e S
S -> a
E -> b
S
```

output_grammar.txt

```
3
S E S1
4
i t e a
5
S -> i E t S S1
S -> a
E -> b
S1 -> e S
S1 -> eps
S
```

Пример 2

example1.txt

```
A S
a b
A -> S a | A a
S -> A b
A
```

input_grammar1.txt

```
2
A S
2
a b
3
A -> S a
A -> A a
S -> A b
A
```

output_grammar1.txt

```
4
A S A1 S1
2
a b
6
A -> S a A1
S -> S1
A1 -> a A1
A1 -> eps
S1 -> a A1 b S1
S1 -> eps
A
```

Пример 3

example2.txt

```
E T F
+ * ( ) a
E -> E + T | T
T -> T * F | F
F -> a | (E)
E
```

input_grammar2.txt

```
3
E T F
5
+ * ( ) a
6
E -> E + T
E -> T
T -> T * F
T -> F
F -> a
F -> (E)
E
```

output_grammar2.txt

```
5
E T F E1 T1
5
+ * ( ) a
8
E -> T E1
T -> F T1
F -> a
F -> (E)
E1 -> + T E1
E1 -> eps
T1 -> * F T1
T1 -> eps
E
```

ЦЕПНЫЕ ПРАВИЛА

Пример 1

example2.txt

```
E T F
+ * ( ) a
E -> E + T | T
T -> T * F | F
F -> a | (E)
E
```

chain_rules21.txt

```
3
E T F
5
+ * ( ) a
6
E -> E + T
E -> T
T -> T * F
T -> F
F -> a
F -> (E)
E
```

output_grammar21.txt

```
3
E T F
5
+ * ( ) a
9
E -> E + T
E -> T * F
E -> a
E -> (E)
T -> T * F
T -> a
T -> (E)
F -> a
F -> (E)
E
```

Пример 2

example3.txt

```
A B C D
a b c
A -> B | a
B -> C | b
C -> D D | c
A
```

chain_rules31.txt

```
4
A B C D
3
a b c
6
A -> B
A -> a
B -> C
B -> b
C -> D D
C -> c
A
```

output_grammar31.txt

```
4
A B C D
3
a b c
9
A -> a
A -> b
A -> D D
A -> c
B -> b
B -> D D
B -> c
C -> D D
C -> c
A
```

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы была реализована программа на языке Python, позволяющая преобразовывать грамматики.

В программе было реализовано устранение левой рекурсии совместно с устранением непосредственной и косвенной рекурсии, после чего применялась левая факторизация. Также было реализовано устранение цепных правил.

Таким образом, в результате выполнения лабораторной работы были приобретены практические навыки реализации наиболее важных (но не всех) видов преобразований грамматик, чтобы удовлетворить требованиям алгоритмов синтаксического разбора.