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 «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 2

Дисциплина: Конструирование компиляторов

Студент: Платонова Ольга

Вариант: 2

Группа: ИУ7-22М

Преподаватель: Ступников А. А.

Москва, 2023 г.

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.

Теоретическая часть

Нетерминал A КС-грамматики G = (N, E, P, S) называется *рекурсивным*, если A ->+ aAb для некоторых a и b. Если a = eps, то A называется *леворекурсивным*. Аналогично, если b = eps, 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

Назовем символ X∈ N U E *бесполезным* в КС-грамматике G = (N, E, P, S), если в ней нет вывода вида S =>* wXy =>* wxy, где w, x, y принадлежат E*.

Алгоритм 2.9. Устранение бесполезных символов.

Вход. КС-грамматика $G = (N, \Sigma, P, S)$, у которой $L(G) \neq \varnothing$.

Выход. КС-грамматика $G' = (N', \Sigma', P', S)$, у которой $L(G') = L(G)$ и в $N' \cup \Sigma'$ нет бесполезных символов.

Метод.

(1) Применив к G алгоритм 2.7, получить N_e. Положить $G_1 = (N \cap N_e, \Sigma, P_1, S)$, где P_1 состоит из правил множества P, содержащих только символы из $N_e \cup \Sigma$.

(2) Применив к G_1 алгоритм 2.8, получить $G' = (N', \Sigma', P', S)$. □

На шаге (1) алгоритма 2.9 из G устраняются все нетерминалы, которые не могут порождать терминальных цепочек. Затем на шаге (2) устраняются все недостижимые символы. Каждый символ X результирующей грамматики должен появиться хотя бы в одном выводе вида $S \Rightarrow^* wXy \Rightarrow^* wxy$. Заметим, что если сначала применить алгоритм 2.8, а потом алгоритм 2.7, то не всегда результатом будет грамматика, не содержащая бесполезных символов.

Результаты работы

Устранение левой рекурсии:

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


Left recursion
--------------
S -> i E t S S1 | a
E -> b
S1 -> e S | eps
```

Устранение косвенной рекурсии:

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


Indirect recursion
---------------
A -> S a A1
S -> S1
A1 -> a A1 | eps
S1 -> a A1 b S1 | eps
```

Устранение бесполезных символов:

```
Grammar
----------------
S -> A B | C D
A -> E F
G -> A D
C -> c


Useless symbols
----------------
S -> A B | C D
A -> E F
C -> c
```