

**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)**

**ОТЧЕТ**

По лабораторной работе №2

По курсу «Конструирование компиляторов»

Тема: «Преобразования грамматик»

Вариант 5

Студент:

Ковалев М.С.

Группа:

ИУ7-22М

Преподаватель:

Ступников А.А.

Москва 2023

**Цель и задачи работы**

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**

Данную грамматику часто требуется модифицировать так, чтобы порождаемый ею язык приобрел нужную структуру. Общего алгоритмического метода, который придавал бы данному языку произвольную структуру, не существует.

Но с помощью ряда преобразований можно видоизменить грамматику, не испортив порождаемого грамматикой языка. В данной лабораторной работе рассматривается несколько преобразований такого рода. Для быстрого погружения в данную предметную область рекомендуется самостоятельно проработать главу 3 (параграфы 3.1, 3.2, 3.3) учебного пособия [3].

**Варианты заданий на лабораторную работу**

Общий вариант для всех:

Устранение левой рекурсии. Определение. Не терминал A КС-грамматики G = (Ν, 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 А 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. Некоторые из алгоритмов разбора не могут работать с леворекурсивными грамматиками. Можно показать, что каждый КС-язык определяется хотя бы одной не леворекурсивной грамматикой.

Постройте программу, которая в качестве входа принимает приведенную КС-грамматику G = (Ν, Σ, P, S) и преобразует ее в эквивалентную КС-грамматику G' без левой рекурсии. Указания. 1) Проработать самостоятельно п. 4.3.3. и п. 4.3.4. [2]. 2) Воспользоваться алгоритмом 2.13. При тестировании воспользоваться примером 2.27. [1]. 3) Воспользоваться алгоритмами 4.8 и 4.10. При тестировании воспользоваться примерами 4.7., 4.9. и 4.11. [2]. 4) Устранять надо не только непосредственную (immediate), но и косвенную (indirect) рекурсию. Этот вопрос подробно затронут в [4]. 5) После устранения левой рекурсии можно применить левую факторизацию.

Вариант 5.

Определение. КС-грамматика G = (Ν, Σ, P, S) называется грамматикой без циклов, если в ней нет выводов A −>+ А для А ∈ N. Грамматика G называется приведенной, если она без циклов, без ε-правил и без бесполезных символов. Грамматики с ε-правилами или циклами иногда труднее анализировать, чем грамматики без ε-правил и циклов. Кроме того, в любой практической ситуации бесполезные символы без необходимости увеличивают объем анализатора. Поэтому для некоторых алгоритмов синтаксического анализа, обсуждаемых в курсе, мы будем требовать, чтобы грамматики, фигурирующие в них, были приведенными. Постройте программу, которая в качестве входа принимает произвольную КС-грамматику и преобразует ее в эквивалентную приведенную КС-грамматику. Указания. Воспользоваться определением на стр. 175, алгоритмом 2.9. и алгоритмом 2.10. [1]. При тестировании воспользоваться упражнением 2.4.13. [1].

**Текст программы**

[**https://github.com/martel42/compilers/tree/master/Lab2**](https://github.com/martel42/compilers/tree/master/Lab2)

**Тесты**

Входной файл:

{  
 "terms": ["a", "b"],  
 "nonTerms": ["A", "B", "C"],  
 "prods": [  
 {"A": ["B", "C"]},  
 {"A": ["a"]},  
 {"B": ["C", "A"]},  
 {"B": ["A", "b"]},  
 {"C": ["A", "B"]},  
 {"C": ["C", "C"]},  
 {"C": ["a"]}  
 ],  
 "start": "A"  
}

Выходной файл:

{  
 "terms":["a","b"],  
 "nonTerms":["A","B","C","B1","C1"],  
 "prods":[  
 {"A":["B","C"]},  
 {"A":["a"]},  
 {"B":["C","A"]},  
 {"C":["a"]},  
 {"B":["a","b"]},  
 {"B1":["C","b"]},  
 {"B1":["C","b","B1"]},  
 {"B":["C","A","B1"]},  
 {"B":["a","b","B1"]},  
 {"C":["a","B"]},  
 {"C":["a","b","C","B"]},  
 {"C":["a","b","B1","C","B"]},  
 {"C1":["C"]},  
 {"C1":["C","C1"]},  
 {"C1":["A","C","B"]},  
 {"C1":["A","C","B","C1"]},  
 {"C1":["A","B1","C","B"]},  
 {"C1":["A","B1","C","B","C1"]},  
 {"C":["a","C1"]},  
 {"C":["a","B","C1"]},  
 {"C":["a","b","C","B","C1"]},  
 {"C":["a","b","B1","C","B","C1"]}],  
 "start":"A"  
}

**Вывод**

В результате выполнения данной лабораторной работы были разобраны правила приведения контекстно-свободных грамматик к приведённой форме, а именно: удаление бесполезных символов, недостижимых символов, eps-правил, цепных правил.

Также были получены навыки в удалении рекурсий, как непосредственной, так и произвольной.