|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

по курсу «Конструирование компиляторов»

на тему «Преобразования грамматик»

Студент: Головнева Мария, ИУ7-22М

Преподаватель: Ступников А. А.

Москва

2021

**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.

**Общий вариант для всех:** Устранение левой рекурсии.

**Вариант 4**

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

**Текст программы**

|  |
| --- |
| import jsonfrom orderedset import OrderedSetdef prepare\_data():with open("g1.json") as json\_file:grammar = json.load(json\_file)return grammardef write\_answer(grammar):with open("answer.json", "w", encoding="utf-8") as file:json.dump(grammar, file, indent=4)def find\_production(productions, name):for production in productions:if production["lhs"]["-name"] == name:return productiondef eliminate\_left\_recursion(grammar):nonterms = grammar["grammar"]["nonterminalsymbols"]["nonterm"]productions = grammar["grammar"]["productions"]["production"]for i in range(len(nonterms)):ai = find\_production(productions, nonterms[i]["-name"])for j in range(i):aj = find\_production(productions, nonterms[j]["-name"])for ai\_production in ai["rhs"]:if ai\_production[0]["-name"] == aj["lhs"]["-name"]:for aj\_production in aj["rhs"]:new\_production = aj\_production.copy()new\_production[len(new\_production):len(ai\_production)] = ai\_production[1:]ai["rhs"].append(new\_production)ai["rhs"].remove(ai\_production)alpha = []beta = []for ai\_production in ai["rhs"]:if ai\_production[0]["-name"] == nonterms[i]["-name"]:alpha.append(ai\_production[1:])else:beta.append(ai\_production)if len(alpha) == 0:continuerhs = []for b in beta:b.append({'-type': 'nonterm', '-name': nonterms[i]["-name"] + str(1)})rhs.append(b)ai["rhs"] = rhsrhs = []for a in alpha:a.append({'-type': 'nonterm', '-name': nonterms[i]["-name"] + str(1)})rhs.append(a)rhs.append([{'-type': 'term', '-name': "eps"}])productions.append({'lhs': {'name': nonterms[i]["-name"] + str(1)}, 'rhs': rhs})print('')print('')def eliminate\_chain\_rules(grammar):nonterms = grammar["grammar"]["nonterminalsymbols"]["nonterm"]productions = grammar["grammar"]["productions"]["production"]n = []for i in range(len(nonterms)):na = OrderedSet()na.add(nonterms[i]["-name"])j = 0while j < len(na):a = find\_production(productions, na[j])for a\_production in a["rhs"]:if len(a\_production) == 1 and a\_production[0]["-type"] == "nonterm":na.add(a\_production[0]["-name"])j = j + 1n.append(na)for i in range(len(nonterms)):a = find\_production(productions, nonterms[i]["-name"])rhs = []for production in productions:if production["lhs"]["-name"] in n[i]:for p in production["rhs"]:if not(len(p) == 1 and p[0]["-type"] == "nonterm"):rhs.append(p)a["rhs"] = rhsdef main():grammar = prepare\_data()eliminate\_chain\_rules(grammar)# eliminate\_left\_recursion(grammar)write\_answer(grammar)if \_\_name\_\_ == "\_\_main\_\_":main() |

**Набор тестов**

|  |  |  |
| --- | --- | --- |
| Алгоритм | Привала входной грамматики | Результат |
| Устранение левой рекурсии | E → E + T | T  T → T \* F | F  F → a | (E) | E → TE1  E1 → + TE1 | ε  T → FT1  T1 → \*FT1 | ε  F → a | (E) |
| Устранение левой рекурсии | A → Ba | Aa | c  B → Bb | Ab | d | A → BaA1 | cA1  B → cA1bB1 | dB1  A1 → aA1 | ε  B1 → bB1 | aA1bB1 | ε |
| Устранение левой рекурсии | S → Aa | b  A → Ac | Sd | S → Aa | b  A → bdA1  A1 → cA1 | adA1 | ε |
| Устранение цепных правил | E → E + T | T  T → T \* F | F  F → a | (E) | E → E+T | T\*F | a | (E)  T → T \* F | a | (E)  F → a | (E) |
| Устранение цепных правил | S → ASB | C | BS  A → bA | B  B → C | a  C → D | AC  D → b | S → ASB | BS | AC | b  A → bA | a | AC | b  B → a | AC | b  C → AC | b  D → b |

**Вывод**

В данной лабораторной работе мы приобрели практические навыки реализации наиболее важных (но не всех) видов преобразований грамматик, чтобы удовлетворить требованиям алгоритмов синтаксического разбора.

Были выполнены следующие задачи:

1. Приняты к сведению соглашения об обозначениях, принятые в литературе по теории формальных языков и грамматик и кратко описанных в приложении.
2. Произошло ознакомление с основными понятиями и определениями теории формальных языков и грамматик.
3. Детально разобран алгоритм устранения левой рекурсии.
4. Разработана, протестирована и отлажена программа устранения левой рекурсии.
5. Разработана, протестирована и отлажена программа у преобразования грамматики в соответствии с предложенным вариантом.

**Список литературы**

1. АХО А.В, ЛАМ М.С., СЕТИ Р., УЛЬМАН Дж.Д. Компиляторы: принципы, технологии и инструменты. – М.: Вильямс, 2008.
2. АХО А., УЛЬМАН Дж. Теория синтаксического анализа, перевода и компиляции: В 2-х томах. Т.1.: Синтаксический анализ. - М.: Мир, 1978.