Белорусский государственный университет информатики и радиоэлектроники

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Логические основы интеллектуальных систем

Отчёт по лабораторной работе №1

Студенты:	Макаренко А. И.
	Корсакова С. К.
Группа:	121701
Проверил:	Ивашенко В. П.

Минск 2023

Тема: Представление и синтаксическая обработка формул сокращённого языка логики высказываний.

Цель: Приобрести навыки программирования алгоритмов синтаксического разбора формул сокращенного языка логики высказываний.

Вариант (H): Определить количество подформул заданного уровня в формуле сокращенного языка логики высказываний.

Теоретические сведения:

Лексическая грамматика сокращённого языка логики высказываний:

<логическая константа> ::=1|0
<латинская заглавная буква> ::=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<отрицание> ::=!
<конъюкция> ::=/\
<дизъюнкция> ::=\/
<импликация> ::=->
<эквиваленция> ::=~
<открывающаяся скобка> ::=(
<закрывающаяся скобка> ::=)
<бинарная связка> ::=<эквиваленция>|<конъюкция>|<дизъюнкция>|<импликация>

Грамматика сокращённого языка логики высказываний:

<атомарная формула> ::=<латинская заглавная буква>
<унарная сложная формула> ::=<открывающаяся скобка><отрицание><формула><закрывающаяся скобка>
<бинарная сложная формула> ::=<открывающаяся скобка><формула><бинарная связка><формула> <закрывающаяся скобка>
<сложная формула> ::=<унарная сложная формула>|<бинарная сложная формула>
<формула> ::=<логическая константа>|<атомарная формула>|<сложная формула>

Подформула – подстрока формулы, которая тоже является формулой.

Принцип работы программы:

1. Пользователь вводит формулу сокращённого языка логики высказываний. Формула сохраняется в виде строки.

2. Вызывается функция *verify*, которая проверяет строку формулы на корректность.

   (a) Функция *check-symbols* проверяет, чтобы все символы строки формулы являлись либо атомарными формулами сокращенного языка логики высказываний, либо логическими связками, либо круглыми скобками.

   (b) Функция *syntax* проверяет, чтобы все подформулы первого уровня строки формулы соответствовали одному из паттернов:

   - <открывающаяся скобка><формула> <бинарная связка><формула> <закрывающаяся скобка>
   - <открывающаяся скобка> <отрицание><формула> <закрывающаяся скобка>

   (c) Если подформула соответсвует одному из паттернов, то она заменяется на атомарную формулу "A"(символ).

   (d) Проверка строки формулы происходит столько раз, сколько всего подформул в исхоной формуле.

   (e) Если в итоге строка формулы преобразовалась в строку "A"из одного символа, то все подформулы формулы корректные, следовательно и сама формула корректна.

   (f) На экран выводится сообщение о корректности или некорректности формулы.

3. Если строка формулы корректна, то происходит поиск подформул. Если некорректна, то происходит завершение программы.

4. Создаётся вспомогательный вектор *helping*, заполненный нулями; длина вектора равна длине строки формулы.

5. Далее функция *added-zero-level* находит все атомарные подформулы из строки формулы и сохраняет их индексы в строке формулы в отдельном векторе *data*. Вместе с этим сохраняет в *data* и уровень атомарных формул, а именно уровень 0.

6. Функция *get-number-of-brackets* подсчитывает количество открывающихся и закрывающихся скобок в строке формулы, чтобы определить необходимое количество итераций для поиска всех подформул. Данное количество сохраняется в переменную *number-of-brackets*.

7. Вызывается функция *fulfill-data*, в которой с помощью цикла while происходит поиск подформул. Цикл повторяется *number-of-brackets*-раз

   (a) В цикле происходит поиск такого символа строки формулы, чтобы он был равен открывающейся скобке и в соответсвующей ячейке вспомогательного вектора *helping* был равен 0 (означает, что скобка еще не была найдена). Если такой символ найден, то во вспомогательном векторе *helping* ячейка с индексом символа помечается как 1.

   (b) Далее происходит поиск такого символа строки формулы, чтобы он был равен закрывающейся скобке и во вспомогательном векторе *helping* был равен 0 (означает, что скобка еще не была найдена). Если такой символ найден, то во вспомогательном векторе *helping* ячейка с индексом символа помечается как 1.

(c) Происходит проверка на наличие строки подформулы в векторе *data*, чтобы предотвратить ненужное дублирование одинаковых подформул.

(d) Вызывается функция *define-level*, которая определяет уровень строки подформулы.

i. Если подформула ещё не была добавлена в *data*, то индексы открывающейся и закрывающейся скобок сохраняются в *data* вместе с уровнем строки подформулы.

8. Пользователь вводит необходимый уровень подформул.

9. Вызывается функция *get-number-of-subformulas*. Её результат и есть количество подформул с заданным уровнем.

10. Вывод ответа в консоль.

11. Конец программы.

Вывод: Выполнение задачи требовало понимания грамматики сокращенного языка логики высказываний и подформулы формулы. Разработанный алгоритм позволяет отыскать все подформулы формулы сокращённого языка логики высказываний и определить их уровень.

```
Input the formula
((((A/\B)\/(B\/C))/\((!C)\/((1\/A)\/(B->C))))/\((!D)->(Q/\A)))

Level: 0   A
Level: 0   B
Level: 0   C
Level: 0   1
Level: 0   D
Level: 0   Q
Level: 1   (A/\B)
Level: 1   (B\/C)
Level: 1   (!C)
Level: 1   (1\/A)
Level: 1   (B->C)
Level: 1   (!D)
Level: 1   (Q/\A)
Level: 2   ((A/\B)\/(B\/C))
Level: 2   ((1\/A)\/(B->C))
Level: 2   ((!D)->(Q/\A))
Level: 3   ((!C)\/((1\/A)\/(B->C)))
Level: 4   (((A/\B)\/(B\/C))/\((!C)\/((1\/A)\/(B->C))))
Level: 5   ((((A/\B)\/(B\/C))/\((!C)\/((1\/A)\/(B->C))))/\((!D)->(Q/\A)))

Input the nessesary level of subformula: 1
Number of subformulas with the 1 level is 7
```

Рис. 1: Пример выполнения программы

```
Input the formula
((A)~B)

Syntax error
Try again
```

Рис. 2: Пример выполнения программы