Лабораторная работа № 4 по курсу Логическое программирование

Выполнил студент группы М8О-307Б *Лопатин Александр.*

Задание

Написать программу на Прологе, запросы к которой будут выглядеть следующим образом:
Вариант №3:
Реализовать синтаксический анализатор арифметического выражения и вычислить его числовое значение. В выражении допустимы операции +,-,*,/, степень ^. Учитывать приоритеты операций.

Введение

2 основных подхода к обработке языков – статистический и лингвистический. В основе статистического подхода лежит предположение, что содержание текста отражается наиболее часто встречающимися словами. Суть этого анализа заключается в подсчете количества вхождений слов в текст. Лингвистический подход основан на лингвистическом анализе, который представляет собой 4 уровня анализа входных данных: графематический (отдельные слова), морфологический (морфологические характеристики слов), синтаксический (зависимости слов в предложении), семантический (смысл высказывания).

Принцип решения

Обходим список, проверяя, встретилась ли комбинация "операнд + оператор + операнд". Если встретилась, то считаем значение этого выражения и заменяем его полученным числом. Для операций сложения, вычитания, деления и умножения можно идти слево направо, но для операции возведения в степень нужно проходить справа налево (т.к. выражения x^y^z считается как x^(y^z)). Для вычисления значения ввсего выражения проходим по списку и вычисляем сначала все операции возведения в степень, потом операция деления/умножения, и уже потом операции сложения и вычитания. Для того, чтобы учесть проход справа налево для операции возведения степень я для простоты реверсирую список с помощью предиката reverse, который я написал для лабораторной работы №1. Предикат вычисления значения выражения выглядит следующим образом:

```
calculate(L, X):-
    reverse(L, L1, []),
    calc_pows([], L1, L2),
    reverse(L2, L3, []),
```

```
    calc_divs([], L3, L4),
    calc_mults([], L4, L5),
    calc_zero(L5, X).
```

Предикаты вычисления для каждого оператора очень похожи друг для друга, поэтому приведу пример только предиката вычисления возведения в степень:

```
calc_pows(L, [X,Y,Z|T], C):-
    pow(Y), R is Z**X, calc_pows(L, [R|T], C).
calc_pows(L, [X, Y, Z|T], C):-
    append(L, [X, Y], L1), calc_pows(L1, [Z|T], C).
calc_pows(L, [X], L1):-
        append(L, [X], L1).
```

Результат работы программы

Пример вычисления некоторых выражений:

```
?- calculate([5, '+', 3, '^',2], X).
X = 14
?- calculate([1, '-', 3, '^',3, '^',2, +, 12356], X).
X = -7326
?- calculate([1, '+', 2, '*', 4, '/', 3, '-', 8], X).
X = -4.333333333333334
```

Вывод

На прологе гораздо проще писать синтаксические и морфологические разборы естественно-языковых текстов, чем на императивных языках программирования. Пролог предлагает совершенно иные возможности для написания программ, нежели популярные императивные и функциональные языки программирования. Из минусов решения подобных задач искусственного интеллекта можно выделить тот факт, что для их решения требуется некоторая база данных, которую приходится набивать руками.