Московский авиационный институт
(государственный технический университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа №4

по курсу
«Логическое программирование»

Выполнил: Баскаков О.А.
Группа: 08-306
№ по списку: 2
Руководитель: Левинская М.А.
Оценка:
Дата:

Москва
2011 г.

Задание

Познакомиться на практике с методами анализа естественно-языковых текстов в системах логического программирования, реализовать в соответствии с вариантом задания несложный фрагмент естественно-языкового интерфейса к модельной задаче и протестировать его на ряде примеров.
Вариант №3:
Реализовать синтаксический анализатор арифметического выражения и вычислить его числовое значение. В выражении допустимы операции +,-,*,/, степень ^. Учитывать приоритеты операций.

Реализация

Грамматика арифметического выражения:

```
Expr -> Term || Expr + Term || Expr – Term
Term -> Number || Term * Number || Term/Number
Term -> Power || Term * Power || Term/Power
Power -> Number ^ Number || Power ^ Number
```

Отсутствие скобок значительно упрощает расстановку приоритетов.
Для правильной работы необходимо реверсировать выражение и грамматику.

Программа на swi-prolog:

```
#! /usr/bin/prolog -s !#

calculate(Expr, Val) :- reverse(Expr, Expr1),
     a_expr(Val, Expr1).

a_number(NS, [NS]) :- number(NS).

a_pow(V,T) :- append(N,['^'|A],T),
     (a_pow(Vx,N);a_number(Vx,N)),
     a_number(Vy,A),
     pow(Vy,Vx,V).

a_term(V,T) :- (a_number(V,T); a_pow(V,T)).

a_term(V,T) :- append(X,['*'|Y],T),
     (a_number(Vx,X);a_pow(Vx,X)),
     a_term(Vy,Y),
     V is Vy*Vx.

a_term(V,T) :- append(X, ['/'|Y],T),
     (a_number(Vx,X);a_pow(Vx,X)),
     a_term(Vy,Y),
     V is Vy/Vx.

a_expr(V,T) :- a_term(V,T).
a_expr(V,T) :- append(X,['+'|Y],T),
     a_term(Vx, X),
     a_expr(Vy, Y),
     V is Vy + Vx.

a_expr(V,T) :- append(X,['-'|Y],T),
     a_term(Vx, X),
     a_expr(Vy, Y),
     V is Vy - Vx.
```

Тестирование:

```
oleg@debian:~/LP$ ./4.pro
% /home/oleg/LP/4.pro compiled 0.01 sec, 7,148 bytes
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.10.1)

?-  calculate([2,'^',2,'^',3],RES),!.
RES = 256.

?- calculate([6, '+',3,'*', 2, '^', 5,'/',4, '-', 1],RES), !.
RES = 29.

?- calculate([6, '+',2, '^', 5,'/',4, '-', 1],RES), !.
RES = 13.

?- calculate([2, '^', 5,'-', 1],RES), !.
RES = 31.

?- calculate([2, '*', 5,'/',4,'-', 1],RES), !.
RES = 1.5.

?-
% halt
```

Сравним с программой упрощения выражений:

```
simplity(X,X) :- atomic(X).
simplity(X,Y) :- X=..[Op, A, B],
          simplity(A,U),
          simplity(B,V),
          rule(Op, U, V, Y).

rule(*, _, 0,0).
rule(*, 0, _,0).
rule(+,X,0,X).
rule(+,0,X,X).
rule(*,X,1,X).
rule(*,1,X,X).

rule(+,X,Y,Z) :- number(X),number(Y), Z is X + Y.
rule(*,X,Y,Z) :- number(X),number(Y), Z is X * Y.
rule(^,X,Y,Z) :- number(X),number(Y), pow(X, Y,Z).

% finalize
rule(+,X,Y, X+Y).
rule(*,X,Y, X*Y).
rule('^',X,Y, X'^'Y).
```

Протестируем на операциях +,*,^:

```
?- simplity(11+13+15,X),!.
X = 39.
?- simplity(2^10,X),!.
X = 1024.
?- simplity(2^2^3,X),!.
X = 256.
?- simplity(2*3*5+3+5+7+1*a^b,X),!.
X = 45+a^b.
?- simplity(8+0*a+3*7+1*b*c^d*e,X),!.
X = 29+b*c^d*e.
```

Замечания

Процедура `simplity` дает очень хорошие результаты. Она способна применять правила только слева направо, поэтому проблематично будет упростить $3 + a + 7$ без грамматики. Для КС-грамматик в язык SWI-Prolog добавлен специальный механизм, позволяющий задавать их с оригинальным синтаксисом. Автомат разбора строится автоматически.

Выводы

Пролог очень удобен для описания анализа естественно-языковых текстов. Т.к. в нем встроены средства описания грамматик, SWI-Prolog часто используется для приложений Semantic Web в интернете, поддерживает литературное программирование, содержит реализацию веб-сервера, библиотеки для SGML, RDF, RDFS. Технологии Semantic Web, RDF, SGML и представляют информацию в виде, пригодном для машинной обработки. Решать задачу распознавания настоящего человеческого языка пока невозможно даже с прологом, главным образом потому, что для него нет определенной грамматики. Зато можно использовать некоторые подмножества, например, похожие на английский язык. Вообще КС-грамматики находят большое применение в информатике. Ими задаётся грамматическая структура большинства языков программирования, структурированных данных.

Перспектива развития всемирной Сети, известной как проект Веб 3.0 поддерживает особый интерес к семантике распределенных данных и их обработке декларативными методами. В связи с этим появились концепции языка OWL, основанного на концепции First-Order Logic, реализацию которой, в свою очередь, можно рассматривать как значительно расширенную технику классического Prolog.