**Лабораторная работа № 1**

**Продукционные системы. Семантические сети. Prolog**

***Формирование рекурсивных структур***

Процедура предок формирует цепочку родственных связей.

родитель(коля,толя).

родитель(аня, коля).

родитель(вася, аня).

предок(X,Y,род(X,Y)):- родитель(X,Y).

предок(X,Y,род(X,A)):- родитель(X,Z),предок(Z,Y,A).

Запрос: Является ли Вася предком Толи ?

?-предок(вася,толя,X)

X = род(вася, род(аня, род(коля, толя)))

***Процедура, которая учится у пользователя***

Предикат «место» связывает название места в городе с улицами и номерами домов. По заданному названию места процедура пытается определить его адрес, просматривая базу данных «адрес». Процедура «место» действует в соответствии с предположением об открытости мира в том смысле, что она не просто завершается неудачей, если не может найти название места в базе данных. Вместо этого процедура переключатся на другую стратегию и получает сведения от пользователя, выступающего в роли альтернативного источника знаний. Процедура «место» учится на своем опыте, добавляя новые ответы в текущую программу.

:-dynamic(адрес/3).

адрес("БГТУ", "Костюкова", 46).

адрес("БГУ", "Победы", 85).

адрес("Художественный музей", "Победы", 77).

место(X, Улица, Дом) :- адрес(X,Улица, Дом), !.

место(X, Улица, Дом) :- nonvar(X),var(Улица), var(Дом),

write('Это место '), write(X), nl,

write('на какой улице расположено ?'), read(Улица),

write('какой номер дома?'), read(Дом), assert(адрес(X,Улица,Дом)).

Предикат отсечения ! в конце первого предложения процедуры **место** означает, что если решение найдено по первому предложению (т.е. найдено в базе данных), то второе предложение рассматриваться не будет.

Предикат assert добавляет новый факт адрес в базу данных.

Запрос:

?- место(“БГТУ”,X,Y).

X = “Костюкова”

Y = 46.

***Алгоритм поиска с возвратом, в котором используется предикат «assert»***

В следующем запросе пользователь опрашивается о знании языков.

Данный запрос добавляет факты со сведениями о пользователе в базу данных.

язык(итальянский).

язык(немецкий).

язык(французский).

язык(английский).

? - write('введите Ваше имя:'), read(Имя),

язык(Язык),

write('Знаете ли Вы '), write(Язык), write('язык'), nl,

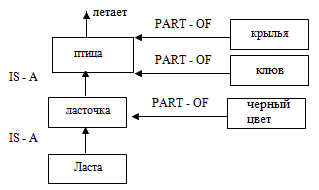
read('да'),

assert(владеет(Имя, Язык),

fail.

Подцель **fail** заставляет интерпретатор возвратиться назад и взять из базы фактов для рассмотрения новый язык. Этот процесс будет продолжаться до тех пор, пока не исчерпается перечень языков, после чего весь составной запрос потерпит неудачу.

***Представление семантических сетей***

****

is\_a(ласточка, птица).

is\_a(ласта, ласточка).

летает(птица).

имеет(крылья, птица).

имеет(клюв, птица).

имеет(‘черный цвет’,ласточка).

является(X,Y):-is\_a(X,Y).

является(X,Y):-is\_a(Z,Y), является(X,Z).

имеет(X,Y):-является(Y,Z),!,имеет(X,Z).

летает(X):-является(X,Y),!,летает(Y).

Запрос:

?-имеет(X, ласта)

X = "черный цвет";

X = крылья;

X = клюв;

PySwip

Позволяет выполнять запросы SWI-пролог из программ на Python.

Установка:

Pip install Pyswip

Пример:

from pyswip import Prolog

prolog = Prolog()

prolog.consult("knowledge\_base.pl")

print(list(prolog.query("имеет(X, ласта)")))

[{‘X’:’черный цвет’}, {‘X’:’крылья’}, {‘X’:’клюв’}]

**Задание.**

1. Выполните рассмотренные примеры.
2. Постройте и реализуйте средствами SWI Prolog семантическую сеть для какой-либо предметной области.
3. Напишите python программу посылающую запросы к семантической сети.