**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Новгородский государственный университет имени Ярослава Мудрого»**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра ИТиС

Отчёт по

лабораторной работе

«ПРОСТЕЙШИЙ ИНТЕРФЕЙС НА ЕСТЕСТВЕННОМ ЯЗЫКЕ С ИСПОЛЬЗОВАНИЕМ КЛЮЧЕВЫХ СЛОВ»

дисциплина:

«Системы искусственного интеллекта»

Выполнил: ст. гр. 8091

\_\_\_\_\_\_\_ \_\_\_\_Шаклеин В.В.

Проверил: Преподаватель

\_\_\_\_\_\_\_\_\_\_\_Михайлов Д.В.

Великий Новгород

2021

1. **Формулировка цели и задач**

Целью работы является изучение лингвистических возможностей

языка Пролог.

# Данные, использованные в ЭС

Предметная область для ЭС – компьютерные вирусы

Вирусы я разбил на группы в зависимости от их основанного предназначения: вредительство и вымогательство.

Таблица 1. Описание предметной области

|  |  |  |
| --- | --- | --- |
| **Группа** | **Название вируса** | **Свойства** |
| **вредительство** | **Morris Worm** | отказ в обслуживании  самораспространение  повреждение данных |
| **ILOVEYOU** | отказ в обслуживании  самораспространение  заражение без участия пользователя |
| **SQL Slammer** | отказ в обслуживании  самораспространение  избирательность |
| **Stuxnet** | отказ в обслуживании  самораспространение  шифрование данных |
| **CryptoLocker** | отказ в обслуживании  самораспространение  повышение прав |
| **Conficker** | отказ в обслуживании  самораспространение  самозащита |
| **вымогательство** | **Sasser** | самораспространение  повреждение данных  избирательность |
| **Melissa** | самораспространение  повреждение данных  шифрование данных |
| **Zeus** | самораспространение  повреждение данных  повышение прав |
| **Storm Trojan** | самораспространение  повреждение данных  самозащита |
| **Michaelangelo** | самораспространение  повреждение данных  заражение без участия пользователя |
| **Code Red** | заражение без участия пользователя  избирательность  шифрование данных |

1. **Описание характеристик разработанных ЭС**

Наиболее известными являются следующие подходы к решению

задач ЕЯ-общения конечного пользователя с ЭВМ:

— Прагматический анализ - наиболее сложный, связан с изучением смысла предложения в связи с внеязыковой действительностью. Анализ ключевых слов — метод анализа ЕЯ-высказываний на предмет наличия ключевых слов, которые становятся значениями аргументов предикатов. При этом компьютер одинаково реагирует на различные варианты входного текста, наличие грамматической правильности предложений не является обязательным, роль играет лишь наличие ключевых слов. Применение -построение ЕЯ-интерфейсов к Базам Данных (БД).

— Грамматический анализ: контекстно-свободный и контекстно-зависимый. Контекстно-Свободный (КС) анализ — ЕЯ-фразы классифицируются в зависимости от их внутренней структуры вне зависимости от контекста в соответствии с грамматическими правилами, задающими порядок следования допустимых языком символов (слов). Здесь следует отметить синтаксический анализ предложений.

— Прагматический анализ — наиболее сложный, связан с изучением смысла предложения с учетом связи с внеязыковой действительностью.

В этой лабораторной интерфейс на естественном языке реализован с помощью ключевых слов. Введённая фраза разбивается на слова, после чего из этих слов выбираются перовое и последнее. Если первое слово начинается на “вых” или “вый” независимо от второго слова происходит выход из программы. Если первое слово начинается на “най”, а последнее слово – “вирус” делается вывод что пользователь хочет найти вирус и запускается процесс поиска по БЗ. Если ни одно из предыдущих условий не выполнилось пользователь получает сообщение об ошибке в формате “ Введённые вами ключевые слова ПЕРВОЕ\_СЛОВО\_ИЗ\_ФРАЗЫ и ПОСЛЕДНЕЕ\_СЛОВО\_ИЗ\_ФРАЗЫ не известны системе.”.

# Тестовые данные

Подается на ввод: Выйти из программы

Результат: Программа завершает свою работу.

Подается на ввод: Найти смысл

Результат: Введённые вами ключевые слова найти и смысл не известны системе.

Подается на ввод: Найти вирус  
Входные данные: Нет, Да, Да, Да, Да

Результат: Вероятнее всего Ваш вирус: Morris Worm.

Подается на ввод: Найди вирус

Входные данные: Да, Да, Да, Да

Результат: Вероятнее всего Ваш вирус: Sasser.

Подается на ввод: Найди пожалуйста мой вирус

Входные данные: Нет, Нет

Результат: Информация об интересующем Вас вирусе отсутствует в БЗ.

1. **Вывод**

Я на практике научился использовать лингвистические возможности языка Пролог для построения интерфейса на естественном языке (пускай и в максимально примитивной форме) для взаимодействия с пользователем.

# Приложение

## Текст программы

predicates

  task\_win\_eh : EHANDLER

     rev(word\_list,word\_list,word\_list).

     reverse(word\_list,word\_list).

     nondeterm rule(rule\_number,category,category,conditions)

     nondeterm cond(cond\_number,condition)

     nondeterm topic(condition)

     assert\_database

     do\_consulting

%      nondeterm info

     erase

     clear

     do\_answer(cond\_number,integer)

%      keyword(category)

     do\_right\_form(string,category)

     func\_keyword(string)

     last(word\_list, string)

     member(char,char\_list).

     nondeterm symbol\_counter(string,integer).

     nondeterm del\_front\_space(string,string).

     nondeterm fronttoken\_cyr(string,string,string).

     nondeterm convers(string,word\_list)

     nondeterm upper\_lower\_cyr(string,string).

     nondeterm upper\_lower\_cyr\_convers(char\_list,char\_list).

     nondeterm str\_char\_list(string,char\_list).

     pack(char\_list,string).

     goes(string,category)

     nondeterm go(history,category)

     nondeterm check(rule\_number,history,conditions)

     ask\_question(cond\_number,condition)

constants

%BEGIN Task Window, CreateParms, 21:26:22-22.12.2007, Code automatically updated!

  task\_win\_Flags = [wsf\_SizeBorder,wsf\_TitleBar,wsf\_Close,wsf\_Maximize,wsf\_Minimize,wsf\_ClipSiblings]

  task\_win\_Menu  = res\_menu(idr\_task\_menu)

  task\_win\_Title = "Экспертная система, базирующаяся на логике"

  task\_win\_Help  = idh\_contents

%END Task Window, CreateParms

clauses

/\* База знаний. \*/

/\* Размещение в резидентной БД информации из утверждений БЗ ЭС \*/

     assert\_database:-

           rule(Rule\_number,Category,Type\_of\_virus,Conditions),

           assertz(d\_rule(Rule\_number,Category,Type\_of\_virus,Conditions)),fail.

     assert\_database:-

           cond(Cond\_number,Condition),

           assertz(d\_cond(Cond\_number,Condition)),fail.

     assert\_database:-

           topic(Condition),

           assertz(d\_topic(Condition)),fail.

     assert\_database:-!.

/\* Условия-характеристики различных вирус.\*/

     cond(1,"цель вредительство").

     cond(2,"цель вымогательство").

     cond(3,"отказ в обслуживании").

     cond(4,"самораспространение").

     cond(5,"повреждение данных").

     cond(6,"заражение без участия пользователя").

     cond(7,"избирательность").

     cond(8,"шифрование данных").

     cond(9,"повышение прав").

     cond(10,"самозащита").

/\* Данные о типах вирусы \*/

     topic("вымогательство").

     topic("вредительство").

/\* Данные о конкретных вирусах \*/

     rule(1,"вирус","вымогательство",[1]).

     rule(2,"вирус","вредительство",[2]).

     rule(3,"вредительство","Morris Worm",[3,4,5]).

     rule(4,"вредительство","ILOVEYOU",[3,4,6]).

     rule(5,"вредительство","SQL Slammer",[3,4,7]).

     rule(6,"вредительство","Stuxnet",[3,4,8]).

     rule(7,"вредительство","CryptoLocker",[3,4,9]).

     rule(8,"вредительство","Conficker",[3,4,10]).

     rule(9,"вымогательство","Sasser",[4,5,6]).

     rule(10,"вымогательство","Melissa",[4,5,7]).

     rule(11,"вымогательство","Zeus",[4,5,8]).

     rule(12,"вымогательство","Storm Trojan",[4,5,9]).

     rule(13,"вымогательство","Michaelangelo",[4,5,10]).

     rule(14,"вымогательство","Code Red",[5,6,7]).

     do\_consulting:-

           goes(\_,\_),!.

      %      go([],First\_keyword),!.

     do\_consulting:-

           not(dummy),

           dlg\_Error("Информация об интересующем Вас вирусе отсутствует в БЗ."),

           clear.

/\* Выдача подсказки \*/

%      info:-

%            findall(Virus\_type,topic(Virus\_type),Virus\_type\_list),

%            term\_str(slist,Virus\_type\_list,Virus\_type\_list\_str\_repr),

%            str\_char\_list(Virus\_type\_list\_str\_repr,[\_|Virus\_type\_list\_char\_repr]),

%            reverse(Virus\_type\_list\_char\_repr,[\_|Virus\_type\_list\_char\_repr\_rev]),

%            reverse(Virus\_type\_list\_char\_repr\_rev,Virus\_type\_list\_char\_repr\_rev1),

%            pack(Virus\_type\_list\_char\_repr\_rev1,Virus\_type\_list\_str\_repr1),

%            dlg\_Note("База знаний содержит информацию о типах вирусов : ",Virus\_type\_list\_str\_repr1),

%            assertz(dummy).

/\* Запрос и получение ответов yes и no от пользователя \*/

     ask\_question(Virus\_cond,Text):-

           concat("Вопрос : ",Text,Temp),

           concat(Temp," ",Temp1),

           concat(Temp1,"?",Quest),

           Response1=dlg\_Ask("Консультация",Quest,["Да","Нет"]),

           Response=Response1+1,

           do\_answer(Virus\_cond,Response).

/\* Предикаты ЕЯ-интерфейса \*/

/\* Реверсирование списка \*/

     rev([],Init,Init).

     rev([H|T],Init,Res):-

        rev(T,[H|Init],Res).

     reverse(Arg,Res):-

        rev(Arg,[],Res).

/\* Принадлежность элемента списку \*/

     member(Head,[Head|\_]):-!.

     member(Elem,[\_|T]):-

             member(Elem,T).

/\* Подсчет символов в строке до конца строки, либо ближайшего пробела,

   символа возврата каретки, перевода строки, !,",#,$ \*/

     symbol\_counter(Str,0):-

             frontchar(Str,'\32',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\10',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\13',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\33',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\34',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\35',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\36',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\40',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\41',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\44',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\45',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\46',\_),!.

     symbol\_counter(Str,0):-

             frontchar(Str,'\59',\_),!.

     symbol\_counter("",0).

     symbol\_counter(Str,Number):-

             frontchar(Str,\_Char,Rest\_of\_string),

             symbol\_counter(Rest\_of\_string,Number1),

             Number=Number1+1.

/\* Выделение подстроки до первого разделителя \*/

     fronttoken\_cyr(Str,Token,Rest\_of\_string):-

             symbol\_counter(Str,Number),

             frontstr(Number,Str,Token,Rest\_of\_string).

/\* Удаление разделителя в начале строки \*/

     del\_front\_space("","").

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\32',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\10',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\13',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\33',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\34',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\35',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\36',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\44',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\40',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\41',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\45',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\46',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Res):-

             frontchar(Arg,Char,Res1),Char='\59',!,

             del\_front\_space(Res1,Res).

     del\_front\_space(Arg,Arg):-frontchar(Arg,Char,\_),

                               not(member(Char,['\32','\10','\13','\33',

                                                '\34','\35','\36','\40','\41',

                                                '\44','\45','\46','\59'])).

/\* Модифицированное правило преобразования строки в список

   слов \*/

     convers("",[]):-!.

     convers(Str,[Head1|Tail]):-

             fronttoken\_cyr(Str,Head,Str2),

             upper\_lower\_cyr(Head,Head1),

             del\_front\_space(Str2,Str1),

             convers(Str1,Tail).

/\* Предикат upper\_lower для кирилицы Windows \*/

     upper\_lower\_cyr(InString,OutString):-

             str\_char\_list(InString,Char\_List\_for\_InString),

             upper\_lower\_cyr\_convers(Char\_List\_for\_InString,Char\_List\_for\_OutString),

             pack(Char\_List\_for\_OutString,OutString).

     upper\_lower\_cyr\_convers([],[]).

     upper\_lower\_cyr\_convers([Char|Char\_List],[Char1|Char\_List1]):-

             char\_int(Char,ASCII\_code),

             ASCII\_code>=192,ASCII\_code<=223,!,

             ASCII\_code\_new=ASCII\_code+32,

             char\_int(Char1,ASCII\_code\_new),

             upper\_lower\_cyr\_convers(Char\_List,Char\_List1).

     upper\_lower\_cyr\_convers([Char|Char\_List],[Char1|Char\_List1]):-

             char\_int(Char,ASCII\_code),

             ASCII\_code=168,!,

             ASCII\_code\_new=ASCII\_code+16,

             char\_int(Char1,ASCII\_code\_new),

             upper\_lower\_cyr\_convers(Char\_List,Char\_List1).

     upper\_lower\_cyr\_convers([Char|Char\_List],[Char1|Char\_List1]):-

             char\_int(Char,ASCII\_code),

             ASCII\_code>=65,ASCII\_code<=90,!,

             ASCII\_code\_new=ASCII\_code+32,

             char\_int(Char1,ASCII\_code\_new),

             upper\_lower\_cyr\_convers(Char\_List,Char\_List1).

     upper\_lower\_cyr\_convers([Char|Char\_List],[Char|Char\_List1]):-

             upper\_lower\_cyr\_convers(Char\_List,Char\_List1).

/\* Преобразование строки в список символов \*/

     str\_char\_list("",[]).

     str\_char\_list(Word,[Char|Char\_List]):-

             frontchar(Word,Char,WordRest),

             str\_char\_list(WordRest,Char\_List).

/\* Превращение списка символов в строку \*/

     pack([],"").

     pack([H|T],Res):-

             str\_char(Str\_H,H),

             pack(T,Res1),

             concat(Str\_H,Res1,Res).

/\* Проверка правильности ключевого слова \*/

%      keyword(Keyword):-

%              rule(\_,Keyword,\_,\_),!.

%      keyword(Keyword):-

%              rule(\_,\_,Keyword,\_),!.

/\* Правило для проверки ключевых слов \*/

      do\_right\_form (Kname, Lname):-

            func\_keyword (Kname),

            rule (\_,Lname,\_,\_),

            go([],Lname), !.

      do\_right\_form (Kname,\_):-

            frontstr (3, Kname, Word, \_),

            upper\_lower\_cyr (Word, Key),

            Key="вый",!,

            exit.

      do\_right\_form (Kname,\_):-

            frontstr (3, Kname, Word, \_),

            upper\_lower\_cyr (Word, Key),

            Key="вых",!,

            exit.

      do\_right\_form (Kname,\_):-  upper\_lower\_cyr (Kname, Word),

            Word="exit", !,

            exit.

      do\_right\_form (Kname,\_):-

            upper\_lower\_cyr (Kname, Word), Word="quit", !,

            exit.

      do\_right\_form (Kname, Lname):-

            concat ("Введенные Вами ключевые слова : ", Kname, Temp),

            concat (Temp, " и ",Temp1),

            concat (Temp1, Lname, Temp2),

            concat (Temp2, " не известны системе. ", Msg),

            dlg\_Note (Msg), !.

/\* Пример проверки допустимости ключевого слова-приказа \*/

      func\_keyword (Word):-

            frontstr (3, Word, Key, \_),

            upper\_lower\_cyr (Key, KeyTr),

            KeyTr="най", !.

/\* Поиск первого ключевого слова в высказывании пользователя \*/

%      first\_keyword\_in\_sentence([Head|\_],Head):-

%              keyword(Head),!.

%      first\_keyword\_in\_sentence([\_|Tail],First\_keyword):-

%              first\_keyword\_in\_sentence(Tail,First\_keyword).

%      first\_keyword\_in\_sentence([],\_):-!,info,fail.

/\* Последний элемент списка \*/

      last([X], X):-!.

      last([\_Head|Tail], Element):-

            last(Tail, Element).

/\* Ввод запроса на естественном (русском) языке \*/

     goes(Mygoal,First\_keyword):-

            Mygoal=dlg\_GetStr("Консультация","Введите Ваш запрос : ","Текст Вашего запроса"),

            convers(Mygoal,Word\_list),

         /\*  do\_right\_form (Word\_list, First\_keyword), \*/

            last(Word\_list, Lname),

            last(Word\_list, First\_keyword),

            reverse(Word\_list, RWrod\_list),

            last(RWrod\_list, Kname),

            do\_right\_form(Kname, Lname),!.

            % first\_keyword\_in\_sentence(Word\_list,First\_keyword),!.

/\* Механизм вывода \*/

/\* Начальное правило механизма вывода \*/

     go(\_,Mygoal):-

           not(rule(\_,Mygoal,\_,\_)),!,

           concat("Вероятнее всего Ваш вирус : ",Mygoal,Temp),

           concat(Temp,".",Result),

           dlg\_Note("Экспертное заключение : ",Result).

     go(History,Mygoal):-

           rule(Rule\_number,Mygoal,Type\_of\_virus,Conditions),

           check(Rule\_number,History,Conditions),

           go([Rule\_number|History],Type\_of\_virus).

/\* Сопоставление входных данных пользователя со списками атрибутов

   отдельных типов вирусов \*/

     check(Rule\_number,History,[Virus\_cond|Rest\_virus\_cond\_list]):-

           yes(Virus\_cond),!,

           check(Rule\_number,History,Rest\_virus\_cond\_list).

     check(\_,\_,[Virus\_cond|\_]):-

           no(Virus\_cond),!,fail.

     check(Rule\_number,History,[Virus\_cond|Rest\_virus\_cond\_list]):-

           cond(Virus\_cond,Text),

           ask\_question(Virus\_cond,Text),

           check(Rule\_number,History,Rest\_virus\_cond\_list).

     check(\_,\_,[]).

     do\_answer(Cond\_number,1):-!,

           assertz(yes(Cond\_number)).

     do\_answer(Cond\_number,2):-!,

           assertz(no(Cond\_number)),fail.

/\* Исключение данных из базы знаний

   после завершения цикла "Распознавание-действие" \*/

     erase:-retract(\_),fail.

     erase.

/\* Уничтожение в базе данных всех ответов yes (да) и no (нет) \*/

     clear:-retract(yes(\_)),retract(no(\_)),fail,!.

     clear.