МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе

О.Г. Локтионова

«__» _____ 2020 г.

СИСТЕМЫ ЕСТЕСТВЕННО-ЯЗЫКОВОГО ОБЩЕНИЯ

Методические указания по выполнению лабораторной работы по дисциплине «Экспертные системы» для студентов направления подготовки 09.04.04 «Программная инженерия»

Курск 2020

УДК 004.65

Составители: В.Г. Белов, Т.М. Белова

Рецензент

Кандидат технических наук, доцент кафедры программной инженерии ЮЗГУ И.Н. Ефремова

Системы естественно–языкового общения: методические указания по выполнению лабораторной работы по дисциплине " Экспертные системы" для студентов направления подготовки 09.04.04 "Программная инженерия" / Юго-Зап. гос. ун-т; сост.: В.Г. Белов, Т.М. Белова, – Курск, 2017. – 24 с.

Изложены основные процедуры естественно-языкового интерфейса.

Материал предназначен для студентов направления подготовки 09.04.04 «Программная инженерия» (профиль "Разработка информационно-вычислительных систем").

Подписано в печать 15.12.17. Формат 60x84 1/16.
Усл. печ. л. 1,4. Уч.-изд. л. 1,3. Тираж 100 экз. Заказ 4431. Бесплатно.
Юго-Западный государственный университет

305040, Курск, ул.50 лет Октября, 94.

Содержание

1 Цель работы

Цель работы: изучить возможности естественно-языкового интерфейса (ЕЯИ) и получить практические навыки в реализации систем ЕЯИ на основании сведений, предоставленных в [1-4].

2 Общие сведения

Программа GEOBASE принимает запросы, сделанные на естественном языке, если пользователь спрашивает о географии Соединенных Штатов. Эта программа является базой данных, ведущей диалог на естественном языке, и одним из первых примеров широкомасштабного применения технологии естественного языка.

Структура GEOBASE основана на идее просмотра ключевых слов. Фактически вместо ключевых слов программа ищет некие образцы более высокого уровня, которые содержат информацию о специфике вопроса (т.е. о том, что именно спрашивается) и о его форме. Эти образцы находятся посредством передачи запросу шаблонов. Каждый образец имеет свою структуру, встроенную в один из шаблонов. Когда шаблон совпадает с образцом, производится анализ структуры вопроса.

Шаблон подобен вырезанной перфокарте с ячейками. Если он налагается на соответствующую ему форму, ячейки заполняются полезной информацией. Если же такого соответствия нет, ячейки либо заполняются "мусором", либо остаются пустыми.

Каждый шаблон является правилом Пролога, и процесс отождествления на этом языке соответствует физическому наложению реального шаблона на печатную форму. Если правило не срабатывает, значит, шаблон не был использован с той формой (типом вопроса), для которой он был создан. Срабатывание правила свидетельствует о том, что форма вопроса понятна, так

как каждое правило соответствует только одному типу вопроса. Аргументы, экземпляры которых создаются в этих правилах, эквивалентны ячейкам, заполняемым в настоящем шаблоне, и точно определяют смысл вопроса.

Вся операция выполнения запроса в GEOBASE может быть разбита на три части:

1. Синтаксический анализ – программа выясняет, что именно пользователь хочет узнать посредством нахождения и заполнения шаблона, который соответствует вопросу.
2. Создание подходящего запроса с использованием свойства Пролога комбинировать образцы.
3. Выбор данных по этому запросу с помощью средств Пролога как обработчика базы данных.

Система GEOBASE содержит следующую информацию:

- Данные о штатах: общая площадь; численность населения; столица; штаты, граничащие с данным; реки; города; самая высокая и низкая географические точки.
- Данные о реках: длина реки.
- Данные о городах: численность населения.
- Пользователь может осуществлять выборку всей информации или ее части путем постановки правильно сформулированных вопросов на естественном языке. Приведем несколько примеров запросов, которые система GEOBASE может обработать:

- Штаты (states)
- Перечислить города в Калифорнии (give me the cities in California)
- Какой самый большой город в Калифорнии? (what is the biggest city in California?)
- Какова самая длинная река в США? (what is the longest river in the USA?)
- Какие реки имеют длину более 1000 км? (which rivers are longer than 1 thousand kilometers?)
- Каково название штата с наименьшей географической высотой? (what is the name of the state with the lowest point?)
- С какими штатами граничит Алабама? (which states border Alabama?)
- Какие реки не протекают через Техас? (which rivers do not run through Texas?)
- Какие реки протекают через штаты, граничащие со штатом, столицей которого является Остин? (which rivers run through states that border the state with the capital Austin?)

В основу организации базы данных, где хранится географическая информация, положены семь типов главных фактов. Любой факт или любое отношение обычно имеет множество аргументов, и существует множество экземпляров каждого факта.

Главные факты: штат, город, река, граница, гора, дорога, озеро.
Пример одного из фактов о штатах:

state("washington","wa","olympia",41.132e6,68.139e3,42, "seattle","spokane","tacoma","bellevue")

Вся информация, относящаяся к штату, компонуется в этот один факт со многими аргументами. Города, указанные в конце, являются крупнейшими по численности населения.

Факт по городу выглядит следующим образом:

city("california","ca","san francisco",678974)

Другие географически значимые факты – реки, озера и горы:

river("rio grande",3033,["colorado","new mexico","texas"]) lake("ontario",19684,["new york"]) mountain("alaska","ak","browne tower",4429)

В этих фактах используются списки, если имеется переменное число вхождений сущностей, присоединяющихся к объекту, который описывает факт, например штаты, границы которых проходят по реке или озеру.

Имеется и факт, используемый для установления самых высоких и самых низких географических точек в каждом штате: highlow("wisconsin","wi","timms hill",595,"lake michigan",177)

Граница является предикатом, который задает для каждого штата список граничащих с ним штатов:

border("alabama","al",["tennessee","georgia","florida", "mississippi"])

Наконец, имеются факты о важнейших дорогах. Благодаря этому отношению номер маршрута ассоциируется со списком штатов, через которые проходит данная дорога.

road("15",["montana","idaho","utah","arizona","nevada", "california"])

Пользователь делает запрос, который программа преобразует в список символов. Первый большой модуль осуществляет операцию фильтрования, устраняя из этого списка посторонние (лишние) для данного вопроса лексемы. Для этого система использует предикат filter. Он также удаляет лишние грамматические элементы, такие, как запятые, точки и кавычки.

Отфильтрованный список слов обрабатывается программой, которая называется синтаксическим анализатором. Именно она выполняет просмотр и поиск шаблонов. Программа синтаксического анализа должна установить основные связи, составляющие сущность вопроса. Она должна также определить другие виды структурной информации в вопросе, которые позволяют использовать основные связи для сбора данных с целью получения ответа на вопрос. В результате этого процесса получается шаблон с реализованными (конкретизированными) аргументами. Шаблон представляет собой единичный сложный объект данных, который передается на следующую стадию обработки.

Фактически шаблон является нормализированной формой первоначального запроса. Структура была воспринята правильно, а

лишняя или случайная информация игнорировалась. После преобразования вопроса в эту форму синтаксическим анализатором система использует функцию под названием eval для получения доступа к базе данных и выдаче ответов.

Для каждой связи, которая представляется важной для получения ответа на вопрос, существует низкоуровневая функция доступа, которая заново комбинирует информацию в базе данных, чтобы программа могла быть написана так, как будто эта связь присутствует в явной форме. Например, если нам нужно узнать названия рек длиной более 1000 км, в базе данных необходимо иметь подобные факты:

length-of-river(columbia,453). length-of-river(hudson,257).

Для экономии низкоуровневые функции доступа написаны так, что для программы данные как будто существуют в такой форме в момент, когда программа фактически обращается к ним, хотя на самом деле они хранятся в другой, более компактной форме. Единственные явные данные по рекам, которыми располагает GEOBASE, хранятся в форме river("columbia",1953,["washington","oregon"]).

Всякий раз при вызове функции eval она генерирует один возможный ответ. Шаблон, ассоциированный с начальным запросом, используется для управления процессом. Эта функция применяется в предложении findall для того, чтобы все ответы могли быть собраны вместе в случае, если вопрос допускает несколько вариантов ответов. Остальные части предиката удаляют избыточные данные на выходе, затем компонуют его и распечатывают выходные данные.

На рис. 2.1 показана блок-схема с основными информационными потоками в системе.

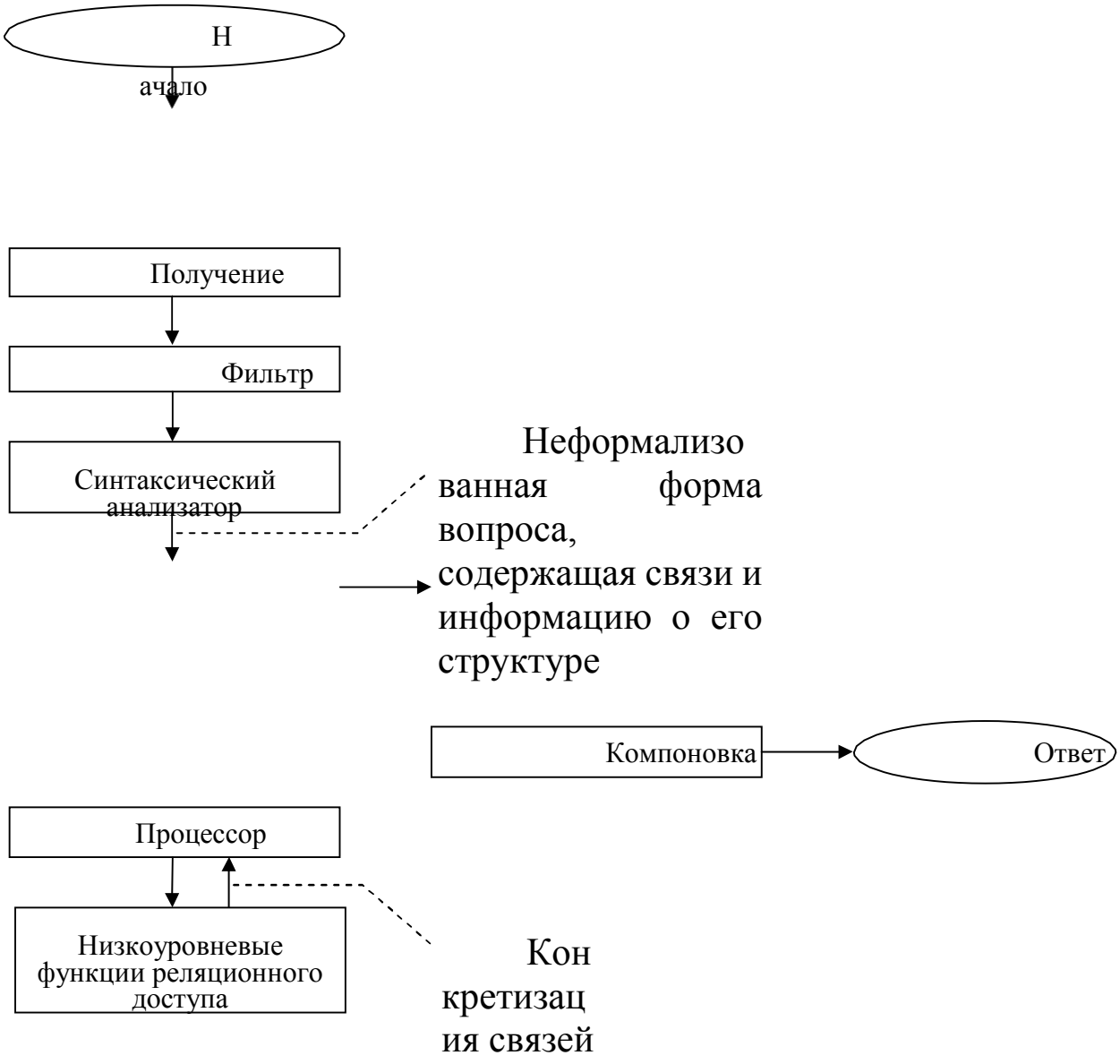

Рис. 2.1- Информационные потоки в программе GEOBASE

В схематической форме код для высокоуровневого функционирования программы будет выглядеть следующим образом:

loop:-

write("Query: "),readln(STR),STR >< "", scan(STR,LIST), % Возвращает список символов filter(LIST,LIST1), % Удаляет знаки пунктуации и

посторонние символы pars(LIST1,E,Q), % Обрабатывает запрос findall(A,eval(Q,A),L),

unik(L,L1), %

Удаляет повторяющиеся объекты (например, Найти реки штатов, граничащих с Техасом)

write_list(0,L1), write_unit(E),listlen(L1,N), write_solutions(N), !, loop.

loop.Ядро программы содержится в подпрограммах pars и eval.

Синтаксический анализатор pars имеет три аргумента:

pars(Data, E, Q)

Первый аргумент – входная переменная, содержащая отфильтрованный запрос. Заполненный шаблон, получаемый в результате синтаксического анализа, связывается с выходной переменной Q, являющейся третьим аргументом. Второй аргумент не столь важен, и мы не будем его рассматривать.

Пример 1. Запрос: Какие имеются города? (What cities do you have?)

К тому времени, когда запрос поступает на синтаксический анализ, в нем уничтожены все лишние слова, и осталось лишь слово "cities" (города). Это первый аргумент при вызове программы анализа pars. По завершении выполнения программы выходным переменным будут присвоены следующие значения:

E = "city"

Q = q_e("city")

Шаблон, который был определен, несложен. Он показывает тип вопроса, в частности вопроса, где речь идет о сущностях (объектах) без их уточнения. Аргумент означает, что задан вопрос о сущности (объекте) "city".

Пример 2. Запрос: Какова длина Гудзона? (How long is the Hudson?)

Данные, поступающие в подпрограмму анализа (pars), имеют форму списка важных слов, а именно:

["long", "hudson"]

По завершении работы программа анализа возвращает такие значения выходных переменных:

E = "length"

Q = q_eaec("lengh","of","river","hudson").

Выбранный шаблон определяет вопрос как вопрос о сущности ("length"), об ассоциации ("of") и о другой сущности ("river"), а также константу ("hudson"), с которой связаны вышеуказанные сущности и

ассоциации. Шаблон является нормализованной, внутренней формой данного вопроса. Мы могли бы задать этот вопрос и другими способами, но в любом случае результат, выдаваемый программой pars, был бы тем же. Например:

Как велик Гудзон? (How big is Hudson?)

Какова длина Гудзона? (What is the length of Hudson?)

Пример 3. Запрос: Какие реки не протекают через Техас? (What rivers do not run through Texas?)

В этом случае ввод для программы анализа pars будет несколько более сложным. Могут быть проигнорированы только два слова. Аргумент ввода имеет вид:

["rivers","not","run","through","texac"]

Выход будет выглядеть следующим образом:

E = "river"

Q = q_not("river",q_eaec("river","in","state","texas"))

Шаблон, выбираемый системой, является вложенной формой, включающей в себя более простой шаблон. Этот простой шаблон иллюстрирует основную используемую связь, а шаблон, в состав которого он входит, показывает, как использовать эту связь для генерации ответов. Очевидно, мы получим список рек, и каждая река посредством подзапроса будет проверена на то, протекает ли она действительно в штате Техас. Отрицательный ответ окажется ответом на первоначальный запрос.

Пример 4. Запрос: Какой самый большой город в штате Вашингтон? (What is the largest city in Washington?)

Входной аргумент:

["largest","city","in","washington"]

Программа анализа pars установит, что речь идет о городах, но будет рассматривать форму нормализированного запроса:

E = "city"

Q = q_max("city",q_eaec("city","in","state","washington"))

Здесь также выбираются вложенные шаблоны. Внутренний шаблон определяет базовую связь, а внешний выражает способ использования этой связи для генерации ответов.

Имеется два предиката, служащие низкоуровневыми функциями доступа к базе данных. Обработка на более высоких уровнях происходит с помощью этих функций и никогда не затрагивает базу данных непосредственно.

Система обладает знаниями о некоторых низкоуровневых сущностях (объектах): река, город, штат, дорога, озеро и т.п. Функция доступа к сущности (ent) берет тип объекта в качестве своего первого параметра и возвращает во втором параметре одно значение, которое эта сущность может принять. Например, если тип объекта город (city), то последовательные вызовы ent дадут следующую информацию:

ent(city,albuquerque) ent(city,arlington) ent(city,austin)

Это не факты, а правила, и именно они позволяют программе работать таким образом, как будто она имеет отдельные списки типов

объектов, которые важны для ее функционирования и прежде всего для того, чтобы все данные имели одинаковую форму.

Правила сущностей выглядят следующим образом:

ent(city,NAME):-

city(_,_,NAME,_). ent(state,NAME):-

state(NAME,_,_,_,_,_,_,_,_,_). и т.д.

Аналогично тому, как ent может сгенерировать все экземпляры определенного типа объекта, имеется еще одна схожая функция, db, которая также может сгенерировать все экземпляры определенного типа связи. Связь состоит из двух объектов, соединенных другим словом, определяющим эту связь. Например: штат с рекой

db(state,with,river,STATE,RIVER):- river(RIVER,_,LIST), member(STATE,LIST).

Дорога в штате

db(road,in,state,ROAD,STATE):- road(ROAD,LIST),

member(STATE,LIST).

Для каждой связи, известной системе, существует отдельное правило (подобное приведенному). Когда функция доступа к данным о дороге в штате применяется последовательно, она возвращает такую информацию:

db(road,in,state,1,new york) db(road,in,state,17,new york) db(road,in,state,80,new york)

Информация о дорогах в определенных штатах выбирается следующим образом:

```
db(road,in,state,ROAD,STATE):- road(ROAD,LIST),
member(STATE,LIST).
```

Каждый раз, когда возвращается предикат road (дорога), выдается номер дороги и список штатов, через которые она проходит. Предикат member затем проверяет, входит ли название штата из запроса db в данный список. Если это так, правило выполняется.

После того как стандартная форма запроса, представленная шаблоном, установлена программой анализа pars, вызывается подпрограмма eval. Она имеет два аргумента, первый из которых является заполненным шаблоном, второй – ответом. Например, предположим, что задан вопрос:

Какова длина Гудзона? (How long is the Hudson?)

Когда синтаксический анализ будет выполнен, вступит в действие подпрограмма eval: eval(q_eaec("length","of","river","hudson"),ANSWER)

Далее eval завершит свою работу, и аргумент ANSWER будет связан следующим образом: eval(q_eaec("length","of","river","hudson"),257)

В некоторых ситуациях на запрос может быть выдано несколько ответов, что обеспечивается главной программой путем вложения eval в предложение findall, которое собирает все ответы в список: findall(A,eval(Q,A),L), где L – список ответов.

Эта функция должна быть определена с помощью ряда правил. Для наглядности поясним их на тех же примерах, которые

использовались при демонстрации высокоуровневого функционирования подпрограммы анализа (pars).

Пример 1. Запрос: Какие имеются города? (What cities do you have?)

Программа анализа преобразует этот вопрос в стандартную форму, представленную заполненным шаблоном:

q_e("city")

Правило eval, запускаемое этим вводом, таково:

```
eval(q_e(E),ANS):-
  ent(E,ANS).
```

Иными словами, мы можем вычислить выражение, если найдем в базе данных что-нибудь наподобие

ent("city",ANS)

Эту задачу выполняет правило сущности (объекта), рассмотренное выше. При каждом его запуске из базы данных извлекается название одного города.

Пример 2. Запрос: Какова длина Гудзона? (How long is the Hudson?)

Ниже приводится правило eval, которое будет выполнено:

```
eval(q_eaec(E1,A,E2,C),ANS):-
  db(E1,A,E2,ANS,C).
```

db является функцией доступа для получения связей, поэтому в базе данных должно иметься что-нибудь наподобие

db(length,of,river,ANS,C)

Большинство запросов требует обработки в правилах eval.

Пример вложенного запроса:

Какие реки не протекают через Техас? (What rivers do not run through Texas?)

Программа анализа выдает стандартизированную форму запроса:

```
q_not("river",q_eaec("river","in","state","texas"))
```

В данном случае выполняется правило eval:

```
eval(q_not(E,TREE),ANS):-
findall(X,eval(TREE,X),L), ent(E,ANS),
not(member(ANS,L)).
```

Переменная TREE связывается с целым вопросом q_eaec. Затем вопрос подвергается оценке внутри предиката findall и выдается список всех рек в Техасе. Предикат ent генерирует название реки всякий раз, когда вызывается. Он используется в качестве пробного ответа. Последнее предложение оценивает ответ и, если река принадлежит к списку рек, протекающих через Техас, ее название игнорируется. В противном случае оно выдается как ответ на запрос.

Это пример так называемой стратегии «генерации и проверки», применяемой для получения ответов на запросы. Она не сложна, но высокой скорости выполнения запросов при большом количестве данных не обеспечивает. В нашем случае скорость вполне приемлема.

Во всех приведенных примерах заполненные шаблоны управляют доступом к базе данных неявно. Правила eval являются

особым случаем, в них учитываются знания по способам обработки определенных типов вопросов. Все правила eval принимают шаблоны как аргументы. Шаблоны представляют собой сложные объекты данных, имеющие свои собственные аргументы (на более низком уровне). Это именно те аргументы, которые в конечном итоге появляются в правой части правила eval, и они управляют выборкой данных. Глобальная архитектура правила eval позволяет определить, как из разрозненных "кусочков" в конце концов получается ответ.

3 Порядок выполнения работы

Изучить пример системы, использующей ЕЯИ (GEOBASE).

Модифицировать систему для работы с требуемой предметной областью согласно заданию.

Отладить программу.

Получить распечатки примеров работы программы.

Оформить отчет.

4 Содержание отчета

Вариант задания, включающий описание предметной области.

Исходный текст разработанной программы.

Распечатки экрана с результатами работы программы.

Выводы по работе.

5 Контрольные вопросы

1. Цель применения систем ЕЯИ.
2. Каков алгоритм работы программы GEOBASE?
3. Как происходит выборка информации из базы данных?
4. Для чего используется синтаксический анализатор?

6 Список использованных источников

1. Попов Э. В. Общение с ЭВМ на естественном языке. – М.: Наука, 1986.
2. Попов Э. В. Экспертные системы: Решение неформализованных задач в диалоге с ЭВМ. – М.: Наука, 1987.
3. Марселлус Д. Программирование экспертных систем на Турбо Прологе: пер. с англ. – М.: Финансы и статистика, 1994.
4. Системы искусственного интеллекта [Электронный ресурс]: Методические указания к выполнению лабораторных работ / Сост. Гудков П.А. - Пенза: Пензенский гос. ун-т, 2007. - 53 с. – Режим доступа: http://window.edu.ru/resource/709/59709