МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

К а ф е д р а «Вычислительная техника»

СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА (ЭКСПЕРТНЫЕ СИСТЕМЫ)

Методические указания к лабораторным работам

Самара
Самарский государственный технический университет
2014

Печатается по решению редакционно-издательского совета СамГТУ

УДК 004.8(076)
ББК 32.813я73
С 40

С 40 **Системы искусственного интеллекта (экспертные системы):** метод. указ. к лаб. работам / Сост. *В.А. Лукиных, А.А. Цыганов*. – Самара: Самар. гос. техн. ун-т, 2014. – 38 с.

Приведена краткая теория, пользуясь которой студенты могут получить необходимые знания и навыки в области систем искусственного интеллекта и теории алгоритмов. Для каждой работы приведены указания по ее выполнению. Определен порядок выполнения работы, содержание отчета и дан список контрольных вопросов, позволяющих закрепить изучаемый материал.

Предназначены для бакалавров направления подготовки 230100 «Информатика и вычислительная техника» по профилю «Вычислительная техника». Они могут быть использованы при исследовании реальных вычислительных систем соответствующих классов.

УДК 004.8(076)
ББК 32.813я73
С 40

Рецензент канд. техн. наук, доцент *И.В. Воронцов*

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Язык Prolog

Prolog – это декларативный язык программирования общего назначения. Он используется в системах обработки естественных языков, исследованиях искусственного интеллекта, экспертных системах, онтологиях и других предметных областях, для которых естественно использование логической парадигмы [5].

Prolog был создан в 1972 с целью сочетать использование логики с представлением знаний. С тех пор у него появился ряд диалектов, расширяющих основу языка различными возможностями.

Этот язык является одним из старейших, но все еще одним из наиболее популярных языков логического программирования, хотя и значительно менее популярен, чем основные императивные языки. Он используется в системах обработки естественных языков, исследованиях искусственного интеллекта, экспертных системах, онтологиях и других предметных областях, для которых естественно использование логической парадигмы.

Prolog был создан под влиянием более раннего языка Planner и позаимствовал из него следующие идеи:

- обратный логический вывод (вызов процедур по шаблону, исходя из целей);
- построение структура управляющей логики в виде вычислений с откатами;
- принцип «отрицание как неудача»;
- использование разных имен для разных сущностей и т.д.

Главной парадигмой, реализованной в языке Prolog, является логическое программирование [4]. Как и для большинства старых языков, более поздние реализации, например, Visual Prolog, добавляют в язык более поздние парадигмы, например, объектно-ориентированное или управляемое событиями программирование, иногда даже с элементами императивного стиля.

Пролог (Prolog) – язык логического программирования, основанный на логике дизъюнктов Хорна, представляющей собой подмножество логики предикатов первого порядка.

Будучи декларативным языком программирования, Prolog воспринимает в качестве программы некоторое описание задачи и сам производит поиск решения, пользуясь механизмом бэктрекинга и унификацией.

Пролог относится к так называемым декларативным языкам, требующим от автора умения составить формальное описание ситуации. Поэтому программа на Прологе не является таковой в традиционном понимании, так как не содержит управляющих конструкций типа if … then, while … do; нет даже оператора присваивания. В Прологе задействованы другие механизмы. Задача описывается в терминах фактов и правил, а поиск решения Пролог берет на себя посредством встроенного механизма логического вывода.

Перечень возможных синтаксических конструкций языка Prolog невелик, и в этом смысле он прост для изучения. С другой стороны, декларативный стиль программирования оказывается столь непривычным и новым для опытных программистов, что вызывает шок и в ряде случаев оказывается тормозом.

Главной парадигмой, реализованной в языке Prolog, является логическое программирование. Prolog использует один тип данных, терм, который бывает нескольких видов:

- атом – имя без особого смысла, используемое для построения составных термов;
- числа и строки такие же, как и в других языках;
- переменная обозначается именем, начинающимся с прописной буквы, и используется как символ-заполнитель для любого другого терма;
- составной терм состоит из атома-функтора, за которым следует несколько аргументов, которые в свою очередь является атомами.

Программы, написанные на чистом языке Prolog, описывают отношения между обрабатываемыми сущностями при помощи клауз Хорна. Клауза – это формула вида Голова :- Тело., которая читается следующим образом: «чтобы доказать/решить Голову, следует доказать/решить Тело». Тело клаузы состоит из нескольких предикатов (целей клаузы), скомбинированных с помощью конъюнкции и дизъюнкции. Клаузы с пустым телом называются фактами и эквивалентны клаузам вида Голова :- true. (true – не атом, как в других языках, а встроенный предикат).

Другой важной частью Prolog являются предикаты. Унарные предикаты выражают свойства их аргументов, тогда как предикаты с несколькими аргументами выражают отношения между ними. Ряд встроенных предикатов языка выполняет ту же роль, что и функции в других языках.

Предикаты с несколькими аргументами могут действовать в нескольких направлениях в зависимости от того, какие из аргументов уже связаны, а какие – нет.

Наконец, для того чтобы быть языком общего назначения, Prolog должен предоставлять ряд сервисных функций, например, процедур ввода/вывода. Они реализованы как предикаты без специального логического смысла, которые всегда оцениваются как истинные и выполняют свои сервисные функции как побочный эффект оценивания.

Структура программы на языке Prolog отличается от структуры программы, написанной на процедурном языке. Пролог-программа является собранием правил и фактов. Решение задачи достигается интерпретацией этих правил и фактов. При этом пользователю не требуется обеспечивать детальную последовательность инструкций, чтобы указать, каким образом осуществляется управление ходом вычислений на пути к результату. Вместо этого он только определяет возможные решения задачи и обеспечивает программу фактами и правилами, которые позволяют ей отыскать требуемое решение.

Во всех других отношениях Prolog не отличается от традиционных языков программирования. Как и в случае программы написанной на любом другом языке, Пролог-программа предназначена для решения отдельной задачи.

Целью выполнения программы на Prolog является оценивание одного предиката. Имея этот предикат и набор правил и фактов, заданных в программе, Prolog пытается найти значения переменных, при которых целевой предикат принимает значение истинности.

Атом записывается со строчной буквы или заключается в кавычки, когда требуется запись с прописной буквы. Например, 'Atom'.

Переменные, записывающиеся с прописной буквы, отличаются от переменных в процедурных языках программирования, они не связаны с конкретной ячейкой памяти, а скорее ближе к математическому пониманию переменной. Например, X is 2 + 2.

В чистом Прологе предложения ограничиваются Дизъюнктами Хорна [2], записываемыми в следующем виде: Вывод :- Условие. Это читается как: Заголовок ИСТИНА, если Тело ИСТИНА. Тело правила содержит ссылки на предикаты, которые называются целями правила.

Предложения с пустым Телом называются Фактами. Например: Кот(Иван). Это эквивалентно правилу: Кот(Иван) :- ИСТИНА.

Пролог реализован практически для всех известных операционных систем и платформ. В число операционных систем входят OS для мэйнфреймов, всё семейство Unix, Windows, OS для мобильных платформ.

Многие современные реализации языка имеют внутреннее расширение за счет ООП-архитектуры. Кроме проприетарных решений, существуют свободные реализации Пролога.

Пролог критикуется в первую очередь за свою недостаточную гибкость, отчего решения на обычных языках программирования (типа C++, Java) в сочетании с базами данных оказываются более технологичными, чем аналогичные решения на Прологе. Негибкость заключается в трудности изучения языка, более высоких требованиях

к квалификации программиста на Прологе, трудности отладки программы, неразвитости технологии программирования, плохой контролируемости промежуточных результатов.

Игра «Жизнь»

Игра «Жизнь» проходит на размеченной на клетки поверхности. Каждая клетка на этой поверхности может находиться в разных состояниях: быть «живой», то есть содержать там особь определенного вида, или быть «мёртвой», то есть пустой. Клетка имеет восемь соседей – окружающих ее клеток. Распределение живых клеток в начале игры называется первым поколением. Каждое следующее поколение рассчитывается на основе предыдущего. Игра прекращается, если на поле не останется ни одной живой клетки, если при очередном шаге ни одна из клеток не меняет своего состояния или если конфигурация на очередном шаге в точности повторит себя же на одном из более ранних шагов. Эти простые правила приводят к огромному разнообразию форм, которые могут возникнуть в игре [3].

Рассмотрим поверхность тора и представим себе прямоугольник. Дискретизация по координатам x и y – на 64 части. Таким образом, вся поверхность делится на 64*64=4096 ячейки и дискретна. Каждый прямоугольник окружает двумя слоями соседей (их 24). Предполагается, что каждый прямоугольник может быть пустым или использован одним из видов. Дискретизация по времени Δt предполагает, что за Δt возможны следующие изменения:

1. Если пустой прямоугольник может получить любой из видов.
2. Если вид в прямоугольнике может завершить существование.

Для того чтобы можно было наблюдать процессы на поверхности, нужно иметь на экране содержимое матрицы 64*64 в текущий момент t, а в памяти формировать следующее состояние матрицы для момента времени $t + \Delta t$. При наступлении данного момента эта матрица выводится на экран, а в памяти формируется ее следующее состояние. Модель должна выполнить дискретизацию во

времени и в пространстве. Состояние каждой ячейки на экране следует обозначать цветом. Моделирование должно осуществляться в пошаговом режиме и в режиме постоянного осуществления шагов по таймеру с возможностью остановки в любой момент. Правила на эволюции по поверхности можно представить в виде двух таблиц.

Таблица создания вида представлена в табл. 1.

Таблица 1

	0			1			2			3			4			5			6			7			8		
	>	=	<	>	=	<	>	=	<	>	=	<	>	=	<	>	=	<	>	=	<	>	=	<	>	=	<
1	*a*	*b*	*c*	*z*	*z*	*z*																					
2																											
3																											
4																											
5																											
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
n																											

a, *b*, *c*, *z* – целые числа.

Если сверху вниз в строчке все условия верны, то вид появится в пустом прямоугольнике.

Таблица завершения существования вида представлена в табл. 2.

Таблица 2

	0			1			2			3			4			5			6			7			8		
	>	=	<	>	=	<	>	=	<	>	=	<	>	=	<	>	=	<	>	=	<	>	=	<	>	=	<
1	*a*	*b*	*c*	*z*	*z*	*z*																					
2																											
3																											
4																											
5																											
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
n																											

Если прямоугольник занят и в строчке вида все условия верны, то вид завершает существование. В процессе наблюдения все виды должны сохраниться и не должно быть периодов в эволюции.

Экспертная система CLIPS

CLIPS – это программная среда для разработки экспертных систем. *CLIPS* является продукционной системой [6]. Основная идея состоит в представлении знаний в виде такой формы:

```
Правило1:
   ЕСЛИ
     (выполняются условия1)
   ТОГДА
     (выполнить действия1)
 Правило2:
   ЕСЛИ
     (выполняются условия2)
   ТОГДА
     (выполнить действия2)
 ...
```

Информация, на основании которой экспертная система делает логический вывод, называется фактами. В *CLIPS* есть два вида фактов: упорядоченные и шаблонные. Шаблонные факты имеют шаблон, задаваемый конструкцией *deftemplate*. Упорядоченные – не имеют явной конструкции *deftemplate*, однако она подразумевается. Шаблонный факт напоминает структуру в языке С или запись в языке *Pascal*. Поля называются слотами и объявляются конструкцией *slot*. Например, следующий шаблон объявляет шаблон с именем *cars* и полями: *model*, *color* и *number*.

```
(deftemplate cars
  (slot model)
  (slot color)
  (slot number)
)
```

Факты размещаются в рабочей памяти. Новые факты помещаются в рабочую память командой *assert*. Например, команда

```
(assert (cars))
```

добавит в рабочую память упорядоченный факт *cars*.

Следующая команда поместит шаблонный факт с тремя атрибутами.

```
(assert
  (cars
    (model "Audi")
    (color Black)
    (number "WY 2576")
  )
)
```

CLIPS не допускает помещения в рабочую память фактов с одинаковыми значениями слотов, хотя при необходимости это можно разрешить соответствующей настройкой.

Знания предметной области представляются в CLIPS в виде правил, которые имеют следующую структуру:

```
(условия)       {синонимы: антецеденты в логике,
                 левая часть - LHS в терминах CLIPS}
=>
(действия)      {синонимы: консеквенты в логике,
                 правая часть - RHS в терминах CLIPS}
```

Левая часть правила – это условие его срабатывания, а правая часть – это те действия, которые должны выполниться в случае выполнения условий. Знак => – это специальный символ, разделяющий LHS и RHS.

Правила объявляются с помощью команды defrule. Пример правила:

```
(defrule search-black-audi
  (cars (model "Audi") (color Black))
  =>
  (printout t "Есть черный Audi!" crlf)
 )
```

Данное правило активируется тогда, когда в рабочей памяти появится факт с атрибутами (model "Audi") и (color Black).

Активация правила не означает его выполнение. Активация правила – это помещение правила в рабочий список правил или agenda в CLIPS.

Чтобы активированные правила выполнились, нужно выполнить команду (run).

Лабораторная работа № 1
ЛОГИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ ЗНАНИЙ

Цель работы – знакомство с работой в среде Visual Prolog и создание простых проектов с представлением знаний в виде генеалогического древа.

ЗАДАНИЕ

1. Освоить основные операции в среде Visual Prolog.
2. Выполнить индивидуальное задание согласно варианту (табл. 3).

Варианты задания

Таблица 3

Бригада №	Задание
1	В генеалогическом древе найти всех тестей
2	В генеалогическом древе найти всех братьев
3	В генеалогическом древе найти всех сестер
4	В генеалогическом древе найти всех двоюродных братьев
5	В генеалогическом древе найти всех двоюродных сестер
6	В генеалогическом древе найти всех дедушек
7	В генеалогическом древе найти всех бабушек
8	В генеалогическом древе найти всех племянников
9	В генеалогическом древе найти всех племянниц
10	В генеалогическом древе найти всех прабабушек
11	В генеалогическом древе найти всех прадедушек
12	В генеалогическом древе найти всех детей в неполной семье

МЕТОДИКА ВЫПОЛНЕНИЯ

1. Запустите Visual Prolog, создайте папку для проекта.
2. Создайте проект.
3. Наберите в среде Visual Prolog текст примера.
4. Выполните проект и убедитесь, что он правильно работает.

5. Модифицируйте проект согласно индивидуальному заданию.
6. Выполните и отладьте проект.

Пример

```
domains
n=string          % определяется n как строчка

predicates
мужчина(n)        % предикат мужчина с одним аргументом
                  % как строчка
женщина(n)        % предикат женщина с одним аргументом
                  % как строчка
родитель(n,n)     % предикат родитель с двумя аргументами
                  % как строчками
теща(n)           % предикат теща с аргументом-строкой

clauses
мужчина("Иван").          % Факты, которые описывают
мужчина("Коля").          % генеалогическое древо
мужчина("Петя").          %
мужчина("Юра").           %
женщина("Марья").         %
женщина("Аня").           %
женщина("Вера").          %
родитель("Иван","Вера"). %
родитель("Марья","Вера").%
родитель("Коля","Петя"). %
родитель("Аня","Петя").  %
родитель("Вера","Юра").  %
родитель("Петя","Юра").  %
теща(X):-
женщина(X),родитель(X,Y),женщина(Y),родитель(Y,_).
% X,Y,Z - переменные
% _ - аргумент, приобретающий любое значение (например,
      в предикате родитель (Z,_) это второй аргумент)

goal
теща(X).
```

СОДЕРЖАНИЕ ОТЧЕТА

Отчет готовится в печатном виде один на бригаду. В отчет включить задание, листинги проекта с комментариями и результаты тестовых выполнений программы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое предикат?
2. Как объявляются и используются переменные?
3. Как пишутся факты и предложения в разделе clauses?

Лабораторная работа № 2
ДИНАМИЧЕСКАЯ СЕТЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Цель работы – работа в среде Visual Prolog и решение задач в ориентированных сетях.

ЗАДАНИЕ

1. Освоить решение задач в ориентированных сетях на основе Visual Prolog.
2. Выполнить индивидуальное задание согласно варианту (табл. 4).

Варианты задания

Таблица 4

Бригада №	Задание
1	На ориентированной сети дорог найти самый короткий маршрут
2	На ориентированной сети дорог найти самый экономичный маршрут
3	На ориентированной сети дорог найти самый короткий маршрут на асфальтовых дорогах
4	На ориентированной сети дорог найти самый экономичный маршрут, не используя грунтовые дороги
5	На ориентированной сети дорог найти самый экономичный маршрут на асфальтовых дорогах
6	На ориентированной сети дорог найти самый короткий маршрут, не используя грунтовые дороги
7	На ориентированной сети дорог найти самый короткий маршрут, не используя магистрали

Продолжение табл. 4

Бригада №	Задание
8	На ориентированной сети дорог найти самый экономичный маршрут, не используя магистрали
9	На ориентированной сети дорог найти самый малый по времени маршрут на асфальтовых дорогах
10	На ориентированной сети дорог найти самый малый по времени маршрут, не используя грунтовые дороги
11	На ориентированной сети дорог найти самый малый по времени маршрут, не используя магистрали
12	На ориентированной сети дорог найти самый малый по времени маршрут

МЕТОДИКА ВЫПОЛНЕНИЯ

1. Запустите Visual Prolog, создайте папку для проекта.
2. Создайте проект.
3. Наберите текст примера.
4. Выполните проект и проверьте, что он правильно работает.
5. Модифицируйте текст в соответствии с заданием.
6. Убедитесь в том, что разработанная программа работает правильно.

Пример

```
domains
n=integer     % n - тип - целое
m=n*          % m - тип - список целых
l=string      % l - тип - строчка символов
d=d(n,n,l,n) % d - дорога к пункту, длина, покрытие,
              %     ограничение скорости
k=d*          % k - список дорог

predicates
gr(n,k)                 % n - пункт, от которого идет
                        %     список дорог
путь(n,n,m,n,n,l)       % путь от A до B, маршрут, длина,
```

```
                        % расход горючего, марка машины
марш(n,n,m,n,m,m,n,l)% путь от A до B, маршрут, длина,
                        % обязательные пункты,
                        % необязательные пункты, расход
                        % горючего, марка машины
нетэлемент(n,m)         % целое n не содержится в списке m
элемент(n,k)            % пункт n содержится в списке дорог,
                        % и есть дорога к пункту n
элемент1(d,k)           % дорога d есть в списке дорог k
element(n,m)            % целое n есть в списке целых m
есть(m,m)               % список целых m в 1 аргументе
                        % должен находиться в 2 аргументе
нет(m,m)                % ни один из первого списка не
                        % должен находиться во втором списке
min(n,m)                % 1 аргумент - это самый минимальный
                        % целый в списке 2 аргумента
menelement (n,m,n)      % 1 аргумент должен быть в списке,
                        % но не меньше 3 аргумента
авто(l,n,n,n)           % марка, расход на 100 км грунта,
                        % асфальта, магистрали
расход(l,l,n)           % марка, покрытие, расход на 100 км

clauses
%следующие строки описывают сеть дорог
gr(1,[d(2,3,"A",90),d(3,2,"A",90),d(5,4,"A",90),
      d(8,5,"A",90)]).
gr(2,[d(3,11,"A",90),d(5,9,"A",90),d(6,10,"G",30)]).
gr(3,[d(4,4,"A",90),d(7,3,"G",30),d(6,2,"M",110) ]).
gr(4,[d(7,6,"A",90),d(10,7,"M",110)]).
gr(5,[d(6,8,"M",110),d(8,6,"A",90),d(9,7,"A",90)]).
gr(6,[d(7,5,"M",110),d(9,10,"A",90)]).
gr(7,[d(10,8,"A",90),d(9,9,"A",90)]).
gr(8,[d(9,2,"G",30),d(11,3,"M",110)]).
gr(9,[d(10,5,"M",110),d(11,4,"A",90)]).
gr(10,[d(11,6,"M",110)]).
gr(11,[]).

авто("Audi",12,10,6). % Audi на 100 км расходует на
                      % грунте 12л, на асфальте 10л,
                      % на магистрали 6л
авто("BMW",15,12,8).  % BMW на 100 км расходует на
                      % грунте 15л, на асфальте 12л,
                      % на магистрали 8л
% следующая строка описывает расход на грунте
расход(Ав,Пок,Литр):-Пок="G", авто(Ав, Литр,_,_).
```

```
% следующая строка описывает расход на асфальте
расход(Ав,Пок,Литр):-Пок="A", авто(Ав,_,Литр,_).
% следующая строка описывает расход на магистрали
расход(Ав,Пок,Литр):-Пок="M", авто(Ав,_,_,Литр).

element(X,[X|_]).    % факт, если целое в голове списка
element(X,[_|L]):- element(X,L).  % если X находится
                                  % в хвосте списка
есть([],_).          % факт, что пустой список будет в
                     % любом списке
% если X есть в списке L и L1 будет в списке L
есть([X|L1],L):- element(X,L),есть(L1,L).

нет([],_).           % факт, что в пустом списке ничего
                     % нет из любого списка
% если X нет в L и L1 не попадет в L
нет([X|L1],L):- нетэлемент(X,L),нет(L1,L).
% X, если голова списка дорог Y идет в Z и Z=X
элемент(X,[Y|_]):-Y=d(Z,_,_,_),Z=X.
% если X нет в голове списка дорог, то X можно найти в L
элемент(X,[_|L]):-элемент(X,L).

элемент1(X,[X,_]).   % факт, что дорога X находится в
                     % голове списка дорог
% если дорога X не в голове списка, то X в L
элемент1(X,[_|L]):- элемент1(X,L)

нетэлемент(_,[]).    % факт, что любое целое не может
                     % быть в пустом списке
% если X не совпадает с головой и его нет в хвосте
нетэлемент(X,[Y|L]):- X<>Y,нетэлемент(X,L).
% X в голове и X не меньше Z
menelement(X,[X|_],Z):-X>=Z.
% искать X не меньше Z в списке L
menelement(X,[_|L],Z):-menelement(X,L,Z).
% найти такое Z, что все значения в списке целых меньше Z
min(Z,L):- menelement(Z,L,Z),
           findall(X,menelement(X,L,Z),C),C=L.

путь(S,Z,L,D,B,A):- gr(S,L1),элемент(Z,L1),L=[S,Z],
        элемент1(d(Z,V,Пок,_),L1),расход(A,Пок,Литр),
        D=V,B=V*Литр.
% если есть дорога от S до Z, то L=[S,Z], а параметры
% дороги позволяют найти длину дороги и расход горючего
путь(S,Z,L,D,B,A):- gr(S,L1),элемент(Y,L1),L=[S,Y],
```

```
        элемент1(d(Y,V,Пок,_),L1),расход(A,Пок,Литр),
        путь(Y,Z,L2,D1,B1,A),L=[S,Y|L2],D=D1+V,
        B=B1+V*Литр.
% ищется дорога от S до Y, затем ищется хвост пути
марш(S,F,L,D,Z,R,B,A):- путь(S,F,L,D,B,A),есть(R,L),
                        нет(Z,L).

goal
% из всех путей выбираются маршруты по параметрам
findall(B,марш(2,5,L,_,[],[], B, "Audi"), C,
        min(Z,C),марш(2,5,L,D,[],[], B, "Audi")).
```

СОДЕРЖАНИЕ ОТЧЕТА

Отчет готовится в печатном виде один на бригаду. В отчет включить задание, листинги проекта с комментариями и результаты тестовых прогонов.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие предикаты использовались в лабораторной работе и каково их назначение?
2. Каково назначение использованных переменных?
3. Как пишутся факты и предложения в clauses?
4. Что такое индуктивные определения?
5. Какие встроенные предикаты вы знаете?

Лабораторная работа № 3
РЕШЕНИЕ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ VISUAL PROLOG

Цель работы – работа в среде Visual Prolog и создание сложных проектов.

ЗАДАНИЕ

1. Освоить основные операции в среде Visual Prolog (табл. 5).
2. Разработать domains.
3. Разработать predicates.

4. Разработать clauses.
5. Разработать goal.

Варианты задания

Таблица 5

Бригада №	Задание
1	В генеалогическом древе найти всех предков с набором профессий
2	В генеалогическом древе найти всех предков с набором болезней
3	В генеалогическом древе найти всех предков с набором хобби
4	В генеалогическом древе найти всех предков с набором посещенных стран
5	В генеалогическом древе найти всех предков с набором посещенных городов
6	В генеалогическом древе найти всех предков с набором игр
7	В генеалогическом древе найти всех предков с набором нескольких сыновей
8	В генеалогическом древе найти всех предков с набором нескольких дочерей
9	В генеалогическом древе найти всех предков с набором нескольких братьев
10	В генеалогическом древе найти всех предков с набором нескольких сестер
11	В генеалогическом древе найти всех предков с набором нескольких внуков
12	В генеалогическом древе найти всех предков с набором нескольких внучек

Пример

```
domains
n=string
m=n*

predicates
человек(n,n,m)
родитель(n,n)
предок(n,n)
полиглотпредок(n,n,m)
```

```
есть(m,m)
элемент(n,m)
полиглоты(n,m,m)

clauses
человек("Дима","м",["русский","немецкий"]).
человек("Федя","м",["русский","английский"]).
человек("Рома","м",["русский","французский"]).
человек("Иван","м",["русский","немецкий"]).
родитель("Дима","Федя").
родитель("Рома","Иван").
родитель("Федя","Рома").
предок(X,Y):-родитель(X,Y).
предок(X,Y):-родитель(X,Z),предок(Z,Y).
элемент(X,[X|_]).
элемент(X,[_|L]):-элемент(X,L).
есть([],_).
есть([X|L1],L):-элемент(X,L),есть(L1,L).
полиглотпредок(X,Y,L1):-
предок(X,Y),человек(X,_,L),есть(L1,L).
полиглоты(Y,L1,C):-findall(X,полиглотпредок(X,Y,L1),C).

goal
полиглоты("Иван",["русский"],C).
```

МЕТОДИКА ВЫПОЛНЕНИЯ

1. Запустите Visual Prolog, создайте папку для проекта и сам проект.
3. Создайте разделы domains, predicates, clauses, goal.
4. Выполните проект.

СОДЕРЖАНИЕ ОТЧЕТА

Отчет готовится в печатном виде один на бригаду. В отчет включить задание, листинги проекта с комментариями и результаты тестовых прогонов.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие предикаты используются в данной работе?
2. Какое назначение имеют созданные вами переменные?
3. Как описываются факты и предложения в блоке clauses?

Лабораторная работа № 4
РЕШЕНИЕ ЗАДАЧ В НЕОРИЕНТИРОВАННЫХ СЕТЯХ

Цель работы – работа в среде Visual Prolog и решение задач в неориентированных сетях.

ЗАДАНИЕ

1. Освоить решения задач в неориентированных сетях на основе Visual Prolog.
2. Выполнить индивидуальное задание и на неориентированной сети дорог найти маршрут согласно варианту (табл. 6).

Варианты задания

Таблица 6

Бригада №	Задание
1	Самый короткий маршрут
2	Самый экономичный маршрут
3	Самый короткий маршрут на асфальтовых дорогах
4	Самый экономичный маршрут без использования грунтовых дорог
5	Самый экономичный маршрут на асфальтовых дорогах
6	Самый короткий маршрут без использования грунтовых дорог
7	Самый короткий маршрут без использования магистралей
8	Самый экономичный маршрут без использования магистралей
9	Самый малый по времени маршрут на асфальтовых дорогах
10	Самый малый по времени маршрут без использования грунтовых дорог
11	Самый малый по времени маршрут без использования магистралей
12	Самый малый по времени маршрут

МЕТОДИКА ВЫПОЛНЕНИЯ

1. Запустите Visual Prolog, создайте папку для проекта.
2. Создайте проект.

3. Наберите текст примера и убедитесь, что он правильно работает.
4. Модифицируйте текст в соответствии с заданием.
5. Выполните и отладьте проект.

Пример

```
domains
n=integer          % n – тип – целое
m=n*               % m – тип – список целых
l=string           % l – тип – строчка символов
d=d(n,n,l,n)       % d – дорога к пункту, длина, покрытие,
                   % ограничение скорости
k=d*               % список дорог

predicates
gr(n,k)                % n – пункт, от которого идет список
                       % дорог
путь(n,n,m,n,n,l)      % путь от A до B, маршрут, длина,
                       % расход горючего, марка машины
путь1(n,m,n,n,m,n,n,l)
марш(n,n,m,n,m,m,n,l)% путь от A до B, маршрут, длина,
                       % обязательные пункты,
                       % необязательные пункты, расход
                       % горючего, марка машины
нетэлемент( n,m)       % целое n не содержится в списке m
элемент(n,k)           % пункт n содержится в списке дорог,
                       % и есть дорога к пункту n
элемент1(d,k)          % дорога d идет в списке дорог k
element(n,m)           % целое n есть в списке целых m
есть( m,m )            % список целых m в 1 аргументе
                       % должен находиться во 2 аргументе
нет(m,m)               % ни один из первого списка не
                       % должен находиться во втором списке
min(n,m)               % 1 аргумент - это самый минимальный
                       % целый в списке 2 аргумента
menelement (n,m,n)     % 1 аргумент должен быть в списке,
                       % но не меньше 3 аргумента
авто(l,n,n,n)          % марка, расход на 100 км грунта,
                       % асфальта, магистрали
расход(l,l,n)          % марка, покрытие, расход на 100 км

clauses
%следующие строки описывают сеть дорог
gr(1,[d(2,3,"A",90),d(3,2,"A",90),d(5,4,"A",90),
```

```
      d(8,5,"A",90)]).
gr(2,[d(3,11,"A",90),d(5,9,"A",90),d(6,10,"G",30),
      d(1,3,"A",90)] ).
gr(3,[d(4,4,"A",90),d(7,3,"G",30),d(1,2,"A",90),
      d(2,11,"A",90),d(6,2,"M",110) ]).
gr(4,[d(7,6,"A",90),d(10,7,"M",110),d(3,4,"A",90)]).
gr(5,[d(6,8,"M",110),d(8,6,"A",90),d(9,7,"A",90),
      d(1,4,"A",90),d(2,9,"A",90)]).
gr(6,[d(3,2,"M",110),d(7,5,"M",110),d(2,10,"G",30),
      d(5,8,"M",110),d(9,10,"A",90)] ).
gr(7,[d(10,8,"A",90),d(3,3,"G",30),d(4,6,"A",90),
      d(6,5,"M",110),d(9,9,"A",90)] ).
gr(8,[d(9,2,"G",30),d(11,3,"M",110),d(1,5,"A",90),
      d(5,6,"A",90)]).
gr(9,[d(6,10,"A",90),d(7,9,"A",90), d(10,5,"M",110),
      d(5,7,"A",90),d(8,2,"G",30),d(11,4,"A",90)]).
gr(10,[d(4,7,"M",110),d(7,8,"A",90),d(9,5,"M",110),
       d(11,6,"M",110)]).
gr(11,[d(9,4,"A",90), d(10,6,"M",110), d(8,3,"M",110)]).

авто("Audi",12,10,6). % Audi на 100 км расходует на
                      % грунте 12л, на асфальте 10л,
                      % на магистрали 6л
авто("BMW",15,12,8).  % BMW на 100 км расходует на
                      % грунте 15л, на асфальте 12л,
                      % на магистрали 8л

% следующая строка описывает расход на грунте
расход(Ав,Пок,Литр):-Пок="G", авто(Ав, Литр,_,_).
% следующая строка описывает расход на асфальте
расход(Ав,Пок,Литр):-Пок="A", авто(Ав,_,Литр,_).
% следующая строка описывает расход на магистрали
расход(Ав,Пок,Литр):-Пок="M", авто(Ав,_,_,Литр).

element(X,[X|_]).    % факт, если целое в голове списка
element(X,[_|L]):- element(X,L).  % если X находится
                                  % в хвосте списка
есть([],_).          % факт, что пустой список будет в
                     % любом списке
% если X есть в списке L и L1 будет в списке L
есть([X|L1],L):- element(X,L),есть(L1,L).

нет([],_).           % факт, что в пустом списке ничего
                     % нет из любого списка
% если X нет в L и L1 не попадет в L
```

```
нет([X|L1],L):- нетэлемент(X,L),нет(L1,L).

% X, если голова списка дорог Y идет в Z и Z=X
элемент(X,[Y|_]):-Y=d(Z,_,_,_),Z=X.

% если X нет в голове списка дорог, то X можно найти в L
элемент(X,[_|L]):-элемент(X,L).

элемент1(X,[X,_]).   % факт, что дорога X находится в
                     % голове списка дорог
% если дорога X не в голове списка, то X в L
элемент1(X,[_|L]):-элемент1(X,L)

нетэлемент(_,[]).    % факт, что любое целое не может
                     % быть в пустом списке
% если X не совпадает с головой и его нет в хвосте
нетэлемент(X,[Y|L]):- X<>Y,нетэлемент(X,L).
% X в голове и X не меньше Z
menelement(X,[X|_],Z):-X>=Z.
% искать X не меньше Z в списке L
menelement(X,[_|L],Z):-menelement(X,L,Z).
% найти такое Z, что все значения в списке целых меньше Z
min(Z,L):- menelement(Z,L,Z),
           findall(X,menelement(X,L,Z),C),C=L.

% пути в неориентированной сети
путь(S,Z,L,D,B,A):- путь1(S,[Z],0,0,L,D,B,A).
путь1(S,[S|L1],D1,B,[S|L1],D1,B,_).
путь1(S,[Y|L1],D1,B1,L,D,B,_):-gr(X,L2),
           элемент (Y,L2),элемент1(d(Y,V,Пок,_),L2),
           нетэлемент(X,L1),расход(A,Пок,Литр),D2=D1+V,
           B2=B1+V*Литр,путь1(S,[X,Y|L1],D2,B2,L,D,B,A).
марш(S,F,L,D,Z,R,B,A):- путь(S,F,L,D,B,A), есть(R,L),
                        нет(Z,L).

goal
% из всех путей выбираются маршруты по параметрам
findall(B,марш(3,1,L,_,[],[2],B,"Audi"),C,
        min (Z,C),марш(3,1,L,D,[],[2], Z, "Audi")).
```

СОДЕРЖАНИЕ ОТЧЕТА

Отчет готовится в печатном виде один на бригаду. В отчет включить задание, листинги проекта с комментариями и результаты тестовых прогонов.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие предикаты используются?
2. Как называются переменные?
3. Как пишутся факты и предложения в clauses?
4. Что такое индуктивные определения?
5. Что из себя представляют встроенные предикаты (find all)?

Лабораторная работа № 5
МОДЕЛЬ ЛАБИРИНТА В ИСКУССТВЕННОМ ИНТЕЛЛЕКТЕ

Цель работы – знакомство с работой в среде CLIPS и создание простых проектов.

ЗАДАНИЕ

1. Освоить основные операции в среде CLIPS.
2. Разработать поле знаний.
3. Разработать поле знаний в среде CLIPS.
4. Выполнить компиляцию в CLIPS.
5. Выполнить тестирование экспертной системы.

МЕТОДИКА ВЫПОЛНЕНИЯ

1. Создайте поле знаний в предметной области.
2. Запустите CLIPS, создайте текстовый файл с расширением .clp, сохраните его в папке, где находится CLIPS.
3. Наберите текст примера в созданном файле.
4. В режиме LOAD откройте файл с расширением *.clp и исследуйте имеющиеся ошибки.
5. Исправьте ошибки в файле *.clp.
6. Закончите работу с экспертной системой.

Пример

```
;Экспертная система на языке CLIPS
;Приведенная ниже экспертная система способна
;принимать решения пользователя или сотрудника
;технического отдела

;=======================================================
;Вспомогательные функции
;=======================================================
;Функция ask-question задает пользователю вопрос,
;полученный в переменной ?question, и получает от
;пользователя ответ, принадлежащий списку допустимых
;ответов, заданному в $?allowed-values

(deffunction ask-question (?question $?allowed-values)
   (printout t ?question)
   (bind ?answer (read))
   (if (lexemep ?answer)
       then (bind ?answer (lowcase ?answer)))
   (while (not (member ?answer ?allowed-values)) do
      (printout t ?question)
      (bind ?answer (read))
      (if (lexemep ?answer)
          then (bind ?answer (lowcase ?answer))))
   ?answer)

;-------------------------------------------------------
; Функция yes-or-no-p задает пользователю вопрос,
; полученный в переменной ?question, и получает от
; пользователя ответ yes (y) или по(n). В случае
; положительного ответа функция возвращает значение TRUE,
; иначе – FALSE

(deffunction yes-or-no-p (?question)
   (bind ?response (ask-question ?question yes no y n))
   (if (or (eq ?response yes) (eq ?response y))
       then TRUE
       else FALSE))

;=======================================================
;Вопросы
;=======================================================
(defrule determine_r1 ""
  (not (r1))
  (not (result ?))
```

```
  =>
  (assert(r1(yes-or-no-p "Компьютер включается?(y/n)?")))
)

(defrule determine_r2 ""
  (not (r2))
  (not (result ?))
  =>
  (assert(r2(yes-or-no-p "В офисе горит свет?(y/n)?")))
)

(defrule determine_r3 ""
  (not (r3))
  (not (result ?))
  =>
  (assert(r3(yes-or-no-p "Горит лампочка на
мониторе?(y/n)?")))

)
(defrule determine_r4 ""
  (not (r4))
  (not (result ?))
  =>
  (assert(r4(yes-or-no-p "Есть какая-либо картинка на
мониторе?(y/n)?")))
)
(defrule determine_r5 ""
  (not (r5))
  (not (result ?))
  =>
  (assert(r5(yes-or-no-p "Звучит диаг. сигнал спикера
(2дл 1 кор)?(y/n)?")))
)
(defrule determine_r6 ""
  (not (r6))
  (not (result ?))
  =>
  (assert(r6(yes-or-no-p "Звучит диаг. сигнал спикера
(1дл 1 кор)?(y/n)?")))
)
(defrule determine_r7 ""
  (not (r7))
  (not (result ?))
  =>
```

```
  (assert(r7(yes-or-no-p "Есть сообщение о неверном
загрузочном диске?(y/n)?")))

)
(defrule determine_r8 ""
  (not (r8))
  (not (result ?))
  =>
  (assert(r8(yes-or-no-p "В FDD или CDROM имеются
носители?(y/n)?")))
)
(defrule determine_r9 ""
  (not (r9))
  (not (result ?))
  =>
  (assert(r9(yes-or-no-p "Операционная система
загружается?(y/n)?")))
)
(defrule determine_r10 ""
  (not (r10))
  (not (result ?))
  =>
  (assert(r10(yes-or-no-p "Есть сообщение 'NTLDR if
missing'?(y/n)?")))
)
(defrule determine_r11 ""
  (not (r11))
  (not (result ?))
  =>
  (assert(r11(yes-or-no-p "Есть сообщение о неверной
активации Windows?(y/n)?")))
)
(defrule determine_r12 ""
  (not (r12))
  (not (result ?))
  =>
  (assert(r12(yes-or-no-p "Не подходит пароль?(y/n)?")))
)
(defrule determine_r13 ""
  (not (r13))
  (not (result ?))
  =>
  (assert(r13(yes-or-no-p "Ошибка сетевой проверки
пользователя?(y/n)?")))
)
```

```
(defrule determine_r14 ""
  (not (r14))
  (not (result ?))
  =>
  (assert(r14(yes-or-no-p "Чувствуется запах палёного
текстолита или горелой проводки?(y/n)?")))
)
(defrule determine_r15 ""
  (not (r15))
  (not (result ?))
  =>
  (assert(r15(yes-or-no-p "Внутренние хосты
пингуются?(y/n)?")))
)
(defrule determine_r16 ""
  (not (r16))
  (not (result ?))
  =>
  (assert(r16(yes-or-no-p "Внешние хосты
пингуются?(y/n)?")))
)
(defrule determine_r17 ""
  (not (r17))
  (not (result ?))
  =>
  (assert(r17(yes-or-no-p "На основном свиче горят
лампочки?(y/n)?")))
)
(defrule determine_r18 ""
  (not (r18))
  (not (result ?))
  =>
  (assert(r18(yes-or-no-p "Сеть работает
нормально?(y/n)?")))
)
(defrule determine_r19 ""
  (not (r19))
  (not (result ?))
  =>
  (assert(r19(yes-or-no-p "Сайты открываются?(y/n)?")))
)
(defrule determine_r20 ""
  (not (r20))
  (not (result ?))
  =>
```

```
  (assert(r20(yes-or-no-p "Работает ли ICQ?(y/n)?")))
)
;=========================================================
;И ответы на них
;=========================================================
(defrule determine_ish1 ""
   (r1 FALSE)
   (r2 TRUE)
   (r3 FALSE)
   =>
  (assert (result "Неисправен пилот. Необходимо заменить
его."))
)
(defrule determine_ish2 ""
   (r1 TRUE)
   (r2 TRUE)
   (r3 TRUE)
   (r4 FALSE)
   (r5 TRUE)
   =>
  (assert (result "Неисправна видеокарта"))
)
(defrule determine_ish3 ""
   (r1 FALSE)
   (r3 TRUE)
   (r6 TRUE)

   =>
  (assert (result "Неисправен модуль памяти"))
)
(defrule determine_ish4 ""
   (r4 TRUE)
   (r7 FALSE)
   (r8 FALSE)

   =>
  (assert (result "Неисправен HDD"))
)
(defrule determine_ish5 ""
   (r4 TRUE)
   (r7 TRUE)
   (r9 FALSE)
   (r10 TRUE)
   =>
  (assert (result "Неисправен загрузчик"))
```

```
)
(defrule determine_ish6 ""
   (r4 TRUE)
   (r7 FALSE)
   (r9 FALSE)
   (r10 FALSE)
   =>
  (assert (result "Неисправна файловая система"))
)
(defrule determine_ish7 ""
   (r9 TRUE)
   (r11 TRUE)
   =>
  (assert (result "Неверная или просроченная активация
Windows"))
)
(defrule determine_ish8 ""
   (r9 TRUE)
   (r11 FALSE)
   (r12 FALSE)
   (r13 FALSE)
   =>
  (assert (result "Ошибка профиля пользователя.
Восстановите Backup"))
)
(defrule determine_ish9 ""
   (r1 TRUE)
   (r5 TRUE)
   (r6 FALSE)
   (r9 FALSE)
   =>
  (assert (result "Ошибки системы SMART"))
)
(defrule determine_ish10 ""
   (r1 FALSE)
   (r2 TRUE)
   (r3 TRUE)
   (r14 TRUE)
   =>
  (assert (result "Неисправен блок питания. Срочно
замените его."))
)
(defrule determine_ish11 ""
   (r1 TRUE)
   (r9 TRUE)
```

```
   (r14 TRUE)
   =>
  (assert (result "Вероятно, проблемы с материнской
платой"))
)
(defrule determine_ish12 ""
   (r1 TRUE)
   (r5 FALSE)
   (r6 FALSE)
   (r7 FALSE)
   (r9 FALSE)
   (r8 TRUE)
   =>
  (assert (result "Дискета или компакт-диск мешают
загрузке"))
)
(defrule determine_ish13 ""
   (r18 FALSE)
   (r15 TRUE)
   (r16 FALSE)
   (r17 TRUE)
   =>
  (assert (result "Проблемы у провайдера. Немного
подождите..."))
)
(defrule determine_ish14 ""
   (r18 FALSE)
   (r15 TRUE)
   (r16 FALSE)
   (r17 FALSE)
   =>
  (assert (result "Неисправен основной Switch"))
)
(defrule determine_ish15 ""
   (r15 FALSE)
   (r16 FALSE)
   =>
  (assert (result "Неверная настройка или физическое
повреждение сетевого кабеля или оборудования"))
)
(defrule determine_ish16 ""
   (r16 TRUE)
   (r19 FALSE)
   (r20 FALSE)
   =>
```

```
  (assert (result "Проблемы на Proxy-сервере"))
)
(defrule determine_ish17 ""
   (r16 TRUE)
   (r19 FALSE)
   (r20 TRUE)
   =>
  (assert (result "Проблемы на DNS-сервере"))
)
(defrule determine_ish18 ""
   (r15 FALSE)
   (r16 TRUE)
   =>
  (assert (result "Ошибка настройки сетевого
оборудования"))
)
(defrule determine_ish19 ""
   (r15 TRUE)
   (r13 TRUE)
   =>
  (assert (result "Недоступен контроллер домена"))
)

; Правило print-repair выводит на экран диагностическое
; сообщение по устранению найденной неисправности.
(defrule print-repair ""
  (declare (salience 10))
  (result ?item)
  =>
  (printout t crlf crlf)
  (printout t "Найденная неисправность:")
  (printout t crlf crlf)
  (format t " %s%n%n%n" ?item)
```

СОДЕРЖАНИЕ ОТЧЕТА

Отчет готовится в печатном виде один на студента. В отчет включить задание, листинги проекта с комментариями и результаты тестовых запусков программы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие функции используются с диалогом в экспертной системе?
2. Как используются приоритеты к правилам?

3. Какие значения переменных используются в правилах?
4. Как используются вероятности в экспертной системе?

Лабораторная работа № 6
РЕШЕНИЕ ИГРОВЫХ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ МОДЕЛИ ЛАБИРИНТА

Цель работы – сканирование пространства игры. Нахождение оптимального решения.

ЗАДАНИЕ

Освоить линейную и семантическую резолюцию (табл. 7).

Варианты задания

Таблица 7

Бригада №	Задание	
1	Семантическая резолюция	Секвенции вариант 1
2	Линейная резолюция	Секвенции вариант 2
3	Семантическая резолюция	Секвенции вариант 3
4	Линейная резолюция	Секвенции вариант 4
5	Семантическая резолюция	Секвенции вариант 5
6	Линейная резолюция	Секвенции вариант 6
7	Семантическая резолюция	Секвенции вариант 7
8	Линейная резолюция	Секвенции вариант 8
9	Семантическая резолюция	Секвенции вариант 9
10	Линейная резолюция	Секвенции вариант 10
11	Семантическая резолюция	Секвенции вариант 11
12	Линейная резолюция	Секвенции вариант 12

МЕТОДИКА ВЫПОЛНЕНИЯ

1. Найти набор дизъюнкций для секвенций.
2. Выполнить резолюцию для набора *S*.

СОДЕРЖАНИЕ ОТЧЕТА

Отчет готовится в рукописном или печатном виде один на бригаду. В отчет включаются задание, получение *S* и получение пустого дизъюнкта.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие известны резолюции?
2. Опишите последовательные этапы получения *S*.

Лабораторная работа № 7
ИСКУССТВЕННАЯ ЖИЗНЬ

Цель работы – эмерджентное обучение при решении задачи методами искусственного интеллекта.

ЗАДАНИЕ

1. Разработать модель программирования эволюции.
2. Разработать правила появления и завершения существования.
3. Разработать исходное значение.
4. Выполнить эволюцию видов с условиями согласно варианту в таблице 8.

МЕТОДИКА ВЫПОЛНЕНИЯ

1. Разработать модель эволюций.
2. Заполнить первую и вторую таблицы в соответствии с заданием.
3. Разработать приложение, моделирующее эволюцию видов.
4. Изучить эволюцию видов.

Варианты задания

Таблица 8

Бригада №	Задание
1	Эволюция двух видов
2	Эволюция трех видов
3	Эволюция четырех видов
4	Эволюция пяти видов
5	Эволюция шести видов
6	Эволюция семи видов
7	Эволюция восьми видов
8	Эволюция девяти видов
9	Эволюция десяти видов
10	Эволюция одиннадцати видов
11	Эволюция двенадцати видов
12	Эволюция тринадцати видов

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какая программная система использована для модели?
2. Какое минимальное значение может быть у Δt?
3. Что представляет собой дискретизация в пространстве до минимума прямоугольников?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Ильясов, Б.Г.* Интеллектуальные системы управления. Теория и практика: учебное пособие для вузов / Б.Г. Ильясов. – Радиотехника, 2009. – 392 с.

2. *Евгенев, Г.Б.* Интеллектуальные системы проектирования / Г.Б. Евгенев. – М.: МГТУ, 2009. – 334 с.

3. *Гаврилова, Т.А.* Базы знаний интеллектуальных систем / Т.А. Гаврилова, Хорошевский В.Ф. – СПБ: Питер, 2010. – 382 с.

4. *Люгер, Джордж Ф.* Искусственный интеллект / Джордж Ф. Люгер. – Москва-СПБ-Киев: Вильямс, 2005. – 863 с.

5. *Братко, И.* Алгоритмы искусственного интеллекта на языке Пролог / И. Братко. – Москва-СПБ-Киев: Вильямс, 2004. – 638 с.

6. *Частиков, А.П.* Разработка экспертных систем. Среда CLIPS / А.П. Частиков, Т.А. Гаврилова, Д.Л. Белов. – СПБ: БХВ, 2003. – 606 с.

СОДЕРЖАНИЕ

Учебное издание

Составители: *ЛУКИНЫХ Валерий Анатольевич*
ЦЫГАНОВ Александр Анатольевич

Системы искусственного интеллекта (экспертные системы)

Редактор *Ю.А. Петропольская*
Верстка *И.О. Миняева*
Выпускающий редактор *Н.В. Беганова*

Подписано в печать 14.10.14
Формат 60×84 1/16. Бумага офсетная
Усл. п. л. 2,18. Уч.-изд. л. 2,15
Тираж 50 экз. Рег. № 165/14

Федеральное государственное бюджетное
образовательное учреждение
высшего профессионального образования
«Самарский государственный технический университет»
443100, г. Самара, ул. Молодогвардейская, 244. Главный корпус

Отпечатано в типографии
Самарского государственного технического университета
443100, г. Самара, ул. Молодогвардейская, 244. Корпус № 8