МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ
И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Методические указания к лабораторным работам

по дисциплине

СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Уровень основной образовательной программы: **бакалавриат**

Направление подготовки: **09.03.04 «Программная инженерия»**

Форма обучения: **очная**

Факультет систем управления (ФСУ)

Кафедра автоматизации обработки информации (АОИ)

Курс 4 Семестр 7

Разработчики:
профессор каф. АОИ
_____Н.В. Замятин

Томск 2016

СОДЕРЖАНИЕ

1. Введение……..2
2. Лабораторная работа 1. Классификация знаний……....4
3. Лабораторная работа 2. Выявление знаний в системах искусственного интеллекта Нечеткие системы……6
4. Лабораторная работы 3. Построение моделей в экспертных системах (Пролог)…….…….9
5. Лабораторная работы 4. Продукции в системах искусственного интеллекта…….20.
6. Лабораторная работы 5. Фреймовые модели в системах искусственного интеллекта…….22
7. Лабораторная работа. 6. Нейронные сети в системах искусственного интеллекта……23
8. Лабораторная работа 7. Работа с редакторами онтологий……....27
9. Лабораторная работы 8. Построение экспертных систем различных .предметных областей (CLIPS)……..……32
10. Библиографический список……..41

1. ВВЕДЕНИЕ

Цель изучения дисциплины

Задачей искусственного интеллекта как научного направления является воссоздание с помощью компьютера разумных рассуждений и действий.

Из всего многообразия научных и технических исследований, определяемых искусственным интеллектом, в учебной дисциплине «Системы искусственного интеллекта» выбраны напрвления, связанные с проблемами представления знаний и вывода на знаниях, а также принципами построения систем искусственного интеллекта в том числе в виде экспертных систем.

Цель изучения дисциплины - ознакомление студентов с методами и моделями представления и обработки знаний, системами искусственного интеллекта (экспертными системами).

Место дисциплины в системе освоения профессиональной образовательной программы

Дисциплина «Системы искусственного интеллекта» связана с такими дисциплинами как общая математика, алгебра, вероятность и статистика, специальные главы математики, логика, информатика, алгоритмизация и программирование.

Требования к уровню освоения содержания курса

По окончанию изучения дисциплины «Системы искусственного интеллекта» студент должен:

иметь представление

- о знаниях, методах их получения, хранения и обработки;
- об искусственном интеллекте как научном направлении и о решаемых здесь задачах;
- о возможностях технологии интеллектуальных систем и путях применения данных технологий при разработке экспертных систем;

знать

- основные модели и методы искусственного интеллекта;
- принципы построения и методы разработки экспертных систем;

Разделы дисциплины

- Понятие об искусственного интеллекта……..……15
- Инженерия знаний
- Модели представления знаний
- Нейронные сети
- Нечеткое представления знаний
- Методы вывода и поиска решений в системах искусственного интеллекта

- Языки программирования систем искусственного интеллекта
- Инструментальные средства разработки систем искусственного интеллекта
- Разработка и проектирование систем искусственного интеллекта
- Архитектура систем искусственного интеллекта
- Экспертные системы

2. Лабораторная работа №1. Классификация знаний. Исследование предметной области/

Цель работы. Изучить заданную предметную область и построить модель знаний в виде графа.

Методические указания. Для построения модели представления знаний в виде графа необходимо выполнить следующие шаги:

1) Определить целевые действия задачи (являющиеся решениями).

2) Определить промежуточные действия или цепочку действий, между начальным состоянием и конечным (между тем, что имеется, и целевым действием).

3) Опередить условия для каждого действия, при котором его целесообразно и возможно выполнить. Определить порядок выполнения действий.

4) Добавить конкретные факты , исходя из поставленной задачи.

5) Преобразовать полученный порядок действий и соответствующие им факты, условия и действия.

6) Для проверки правильности построения записать цепочки, явно проследив связи между ними.Этот набор шагов предполагает движение при построениимодели от результата к начальному состоянию, но возможно и движение от начального состояния к результату (шаги 1 и 2).

7) Присвоить обозначения фактам Ф, правилам П, действиям Д.

8) Построить граф предметной области. (пример рис.1)

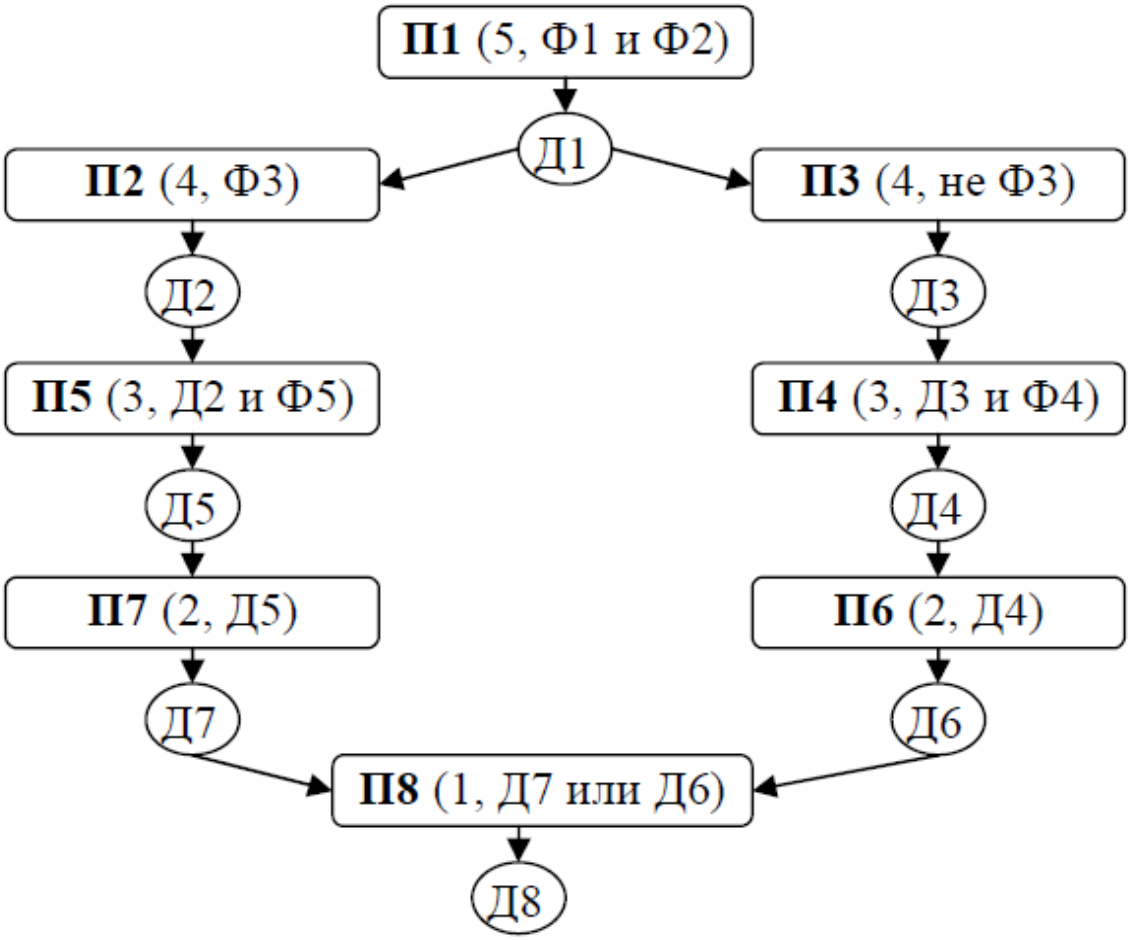


Рис. 1 – Пример графа модели знаний

Варианты заданий

1. Построить модель представления знаний в предметной области «Железная дорога» (продажа билетов).

2. Построить модель представления знаний в предметной области «Торговый центр» (организация).

3. Построить модель представления знаний в предметной области «Автозаправка» (обслуживание клиентов).

4. Построить модель представления знаний в предметной области «Компьютерные сети» (организация).

5. Построить модель представления знаний в предметной области «Университет» (учебный процесс).

6. Построить модель представления знаний в предметной области «Компьютерная безопасность» (средства и способы ее обеспечения).

7. Построить модель представления знаний в предметной области «Компьютерная безопасность» (угрозы).

8. Построить модель представления знаний в предметной области «Интернет-кафе» (организация и обслуживание).

9. Построить модель представления знаний в предметной области «Разработка информационных систем» (ведение информационного проекта).

10. Построить модель представления знаний в предметной области «Туристическое агентство» (работа с клиентами).

11. Построить модель представления знаний в предметной области «Кухня» (приготовление пищи).

12. Построить модель представления знаний в предметной области «Больница» (прием больных).

13. Построить модель представления знаний в предметной области «Кинопрокат» (ассортимент и работа с клиентами).

14. Построить модель представления знаний в предметной области «Прокат автомобилей» (ассортимент и работа с клиентами).

15. Построить модель представления знаний в предметной области «Операционные системы» (функционирование).

16. Построить модель представления знаний в предметной области «Информационные системы» (виды и функционирование).

17. Построить модель представления знаний в предметной области «Предприятие» (структура и функционирование).

Контрольные вопросы

1. Что такое факт?

2.Дайте определение данным?

3. Что такое знания?

4. Что такое поле знаний?

5. Кто такой инженер - когнитолог?

Содержание отчета

- цель работы
- краткие теоретические сведения
- описание предметной области
- граф предметной области
- Ответы на вопросы

3. Лабораторная работа № 2. Выявление знаний в системах искуссвенного интеллекта. Нечеткая логика. Формирование функций принадлежности в программной среде Fuzzy Logic Toolbox

Цель работы: ознакомится со способами и средствами описания нечётких множеств и продукций в системе нечёткого вывода в интерактивном режиме использования графических средств пакета **Fuzzy Logic Toolbox**.

2.1. Общие сведения о пакете Fuzzy Logic Toolbox

Для рассмотрения результатов разработки и функционирования систем нечёткой логики будем использовать графические средства пакета **Fuzzy Logic Toolbox**. Эти же средства используются и при разработке систем нечёткого вывода как графический объектно-ориентированный язык автоматического программирования.

В состав этих средств входят:

— редактор систем нечёткого вывода **FIS Editor** (FIS);

— редактор функций принадлежности систем нечёткого вывода **Membership Function Editor** (MFE);

— редактор правил систем нечёткого вывода **Rule Editor**;

— программа просмотра правил системы нечёткого вывода **Rule Viewer**;

— программа просмотра поверхности нечёткого вывода **Sur-face Viewer**.

Для описания нечётких высказываний используются нечёткие лингвистические переменные (ЛП).

ЛП — это именованная переменная, которая принимает свои значения из множества лингвистических термов, т.е. символьных величин. Для нечёткой ЛП терм-множество задаётся как нечёткое множество. Этот процесс называется фаззификацией. Фаззификация является одной из проблемных задач описания нечёткого вывода и отражает индивидуальные эмпирические знания автора. Нечёткие высказывания в условной части нечёткой продукции могут быть составными, соединёнными связками “И” и/или “ИЛИ”. Эти связки при исчислении высказываний реализуются логическими или арифметическими операциями пересечения или объединения, соответственно.

При получении результата по каждому правилу необходимо дать оценку степени его истинности. Эта оценка зависит от степени истинности высказываний условной части правила, степени истинности отношения, положенного в основу правила, между исходными утверждениями (посылкой) и заключением, т.е. сте-пени истинности импликации, и степени истинности высказыва-ния относительно значения из терм-множества возможных ре-зультатов, приведенного в правиле. Получение оценки степени истинности заключения, полученного по правилу, называют активизацией.
В случае необходимости получения чёткого количественно-го значения результата оно может быть получено на основании функции принадлежности терм-множества результата различными способами по алгоритмам, названым по именам их авторов (Мамдани, Сугено, Цукамото и т.д.), что определяет тип системы нечёткого вывода. Эта операция называется дефаззификацией.

2.1.3. Редактор функций принадлежности (MFE)

Редактор функций принадлежности в графическом режиме обеспечивает задание и изменение функции принадлежности любых термов ЛП СНВ.

Для фаззификации лингвистической переменной СНВ следует выделить ее изображение – именованный прямоугольник в левой верхней части окна редактора (см. рис. 1).

В окне редактора выводятся графики функций принадлежности для всех значений выделенной ЛП (по умолчанию для трёх значений).

Для описания функции принадлежности каждого значения ЛП используются три поля: **Name**, **Type** и **Params**. Описываемая функция выделяется щелчком по её графику. В поле **Name** устанавливается значение ЛП. В поле **Type**, выбором элемента меню, устанавливается имя нужной функции принадлежности (одной из 11-ти встроенных). В поле ввода **Params** указываются необходимые параметры функции принадлежности, которые определяют положение ее модальных значений на числовой шкале, диапазон изменения которой указывается в полях ввода **Range** и **Display range**.

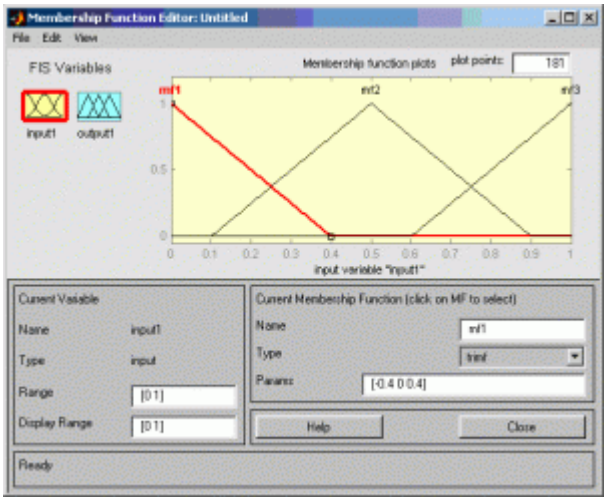


Рис. 1. Окно редактора Membership Function Editor

Эти операции выполняются над всеми значениями из терм-множеств лингвистических переменных СНВ.

Добавление нового значения ЛП со встроенной функцией принадлежности производится по команде основного меню **Edit > Add MF**.

Удаление ненужного значения ЛП производится нажатием клавиши **Delete**, после выделения графика функции принадлежности этого значения.

Задание. Для полученного варианта и разработанного графа ПрО сформировать лингвистические переменные и получить для них функции принадлежности.

Контрольные вопросы

1 Понятие о нечетких множествах?

2. Зачем нужна функция принадлежности?

3. В чем суть лингвистической переменной?

4. Как выполняется нечеткий вывод?

5. Дайте определение нечеткой импликации?

Содержание отчета

- цель работы
- краткие теоретические сведения
- описание предметной области
- структура нечеткой системы
- Листинг программы
- Ответы на вопросы

4. Практическое занятия 3. Построение моделей в системах искусственного интеллекта (декларативный язык ПРОЛОГ)

1 Цель лабораторной работы

Изучить среду визуальной разработки Visual Prolog. Создать проект и запустить его на выполнение.

2 Запуск визуальной среды разработки приложений Visual Prolog.

Для того, чтобы запустить Visual Prolog, необходимо выполнить следующие

действия: Пуск | Программы | Visual Prolog 5.2 | Visual Prolog 32.При этом открывается основное окно, которое называется окном **Task** (рис. 1).

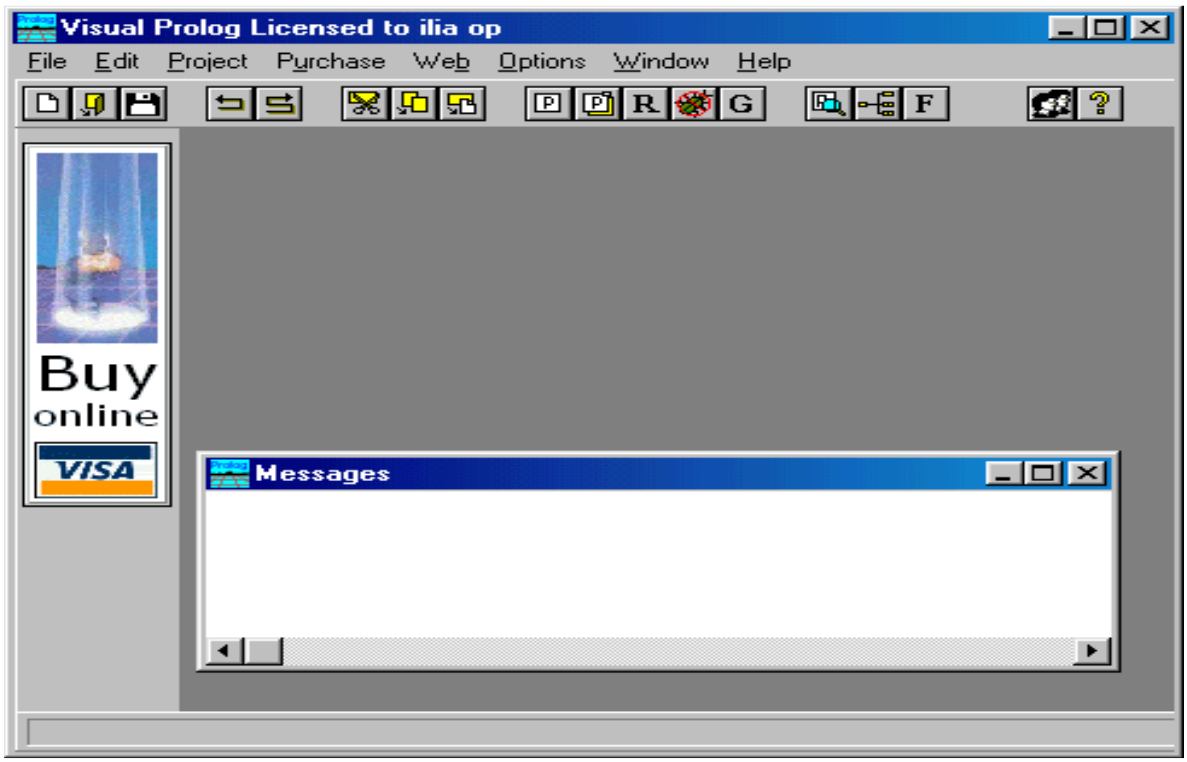


Рис. 1 Окно Task Обычно в нем доступны меню **File, Edit, Project, ptions, Help и Window**, но при fктивизации некоторых других окон в меню могут появиться дополнительные пункты.

Часто используемые команды меню могут быть выполнены и при помощи кнопок на панели инструментов (рис. 2).

Рис. 2. Панель инструментов. Каждая из пиктограмм (табл. 1) на панели инструментов выполняет ту же функцию, что и соответствующая команда меню. Таблица 1. Команды меню и кнопки панели инструментов Кнопка панели инструментов Команда меню Кнопка панели инструментов

Команда меню File | New Edit | Copy File | Open Edit | Paste File | Save Project (Compile file) Edit | Undo Project | Build Edit | edo roject | Run Edit | Cut Project | Debug

Project | Test Goal Options Fon |t Temporary |Project | Browse Help | Local Help

Project | Tree

В нижней части окна **Task,** расположена строка подсказки. Ош разделена на две части (рис.3).

Рис. 3. Строка подсказки

Левое поле используется для отображения контекстно-зависимой информации, например, подсказок для командных кнопок на панели инструментов или информации о текущем элементе управления в редакторе диалоговых окон и т. д.

Крайнее правое поле используется построителем программ (make facility) для отображения состояний генерации/компиляции/компоновки текущего ресурса.

3 Создание проекта.

Для создания проекта требуется определить некоторые (не предопределенные) опции компилятора Visual Prolog. Для этого выполните следующие действия:

1. Запустите среду визуальной разработки Visual Prolog. При первом запуске VDE () проект не будет загружен, и вы увидите окно, показанное на рис. 4. Также вас проинформируют, что по умолчанию создан инициализационный файл для Visual Prolog VDE.

2. Создайте новый проект.

Выберите команду **Project | New Project,** активизируется диалоговое окно **Application Expert.**

3. Определите базовый каталог и имя проекта.

Имя в поле **Project Name** следует определить как "Test". Щелкните мышью внутри поля **Name of .VPR File.** Также установите флажок **Mulltiprogrammer Mode** и щелкните мышью внутри поля **Name of.PRJ File**. Вы увидите, что появится имя файла проекта **Test.prj (**рис. 4).

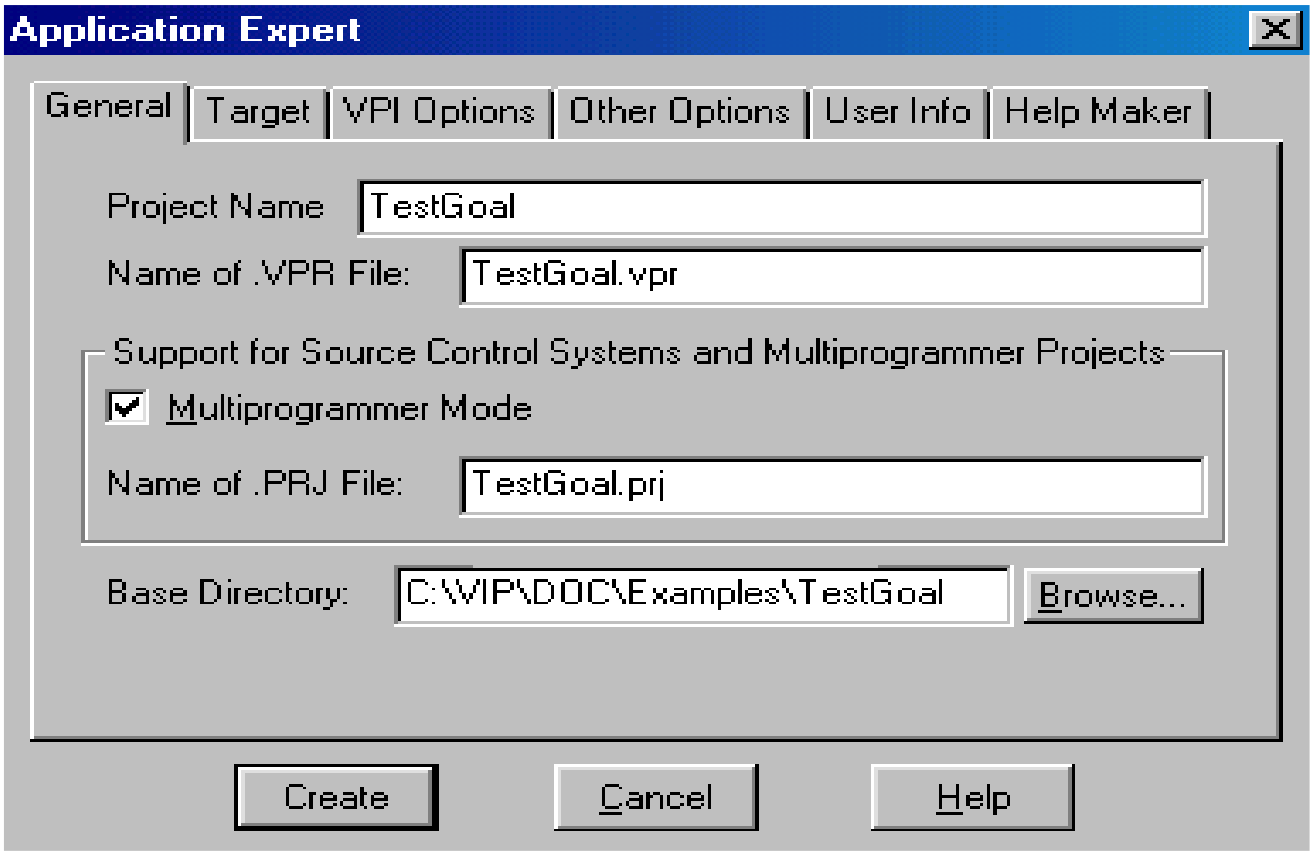


Рис. 4. Общие установки диалогового окна **Application Expert**

Определите цель проекта. На вкладке **Target** рекомендуется выбрать параметры, отмеченные на рис. 5.

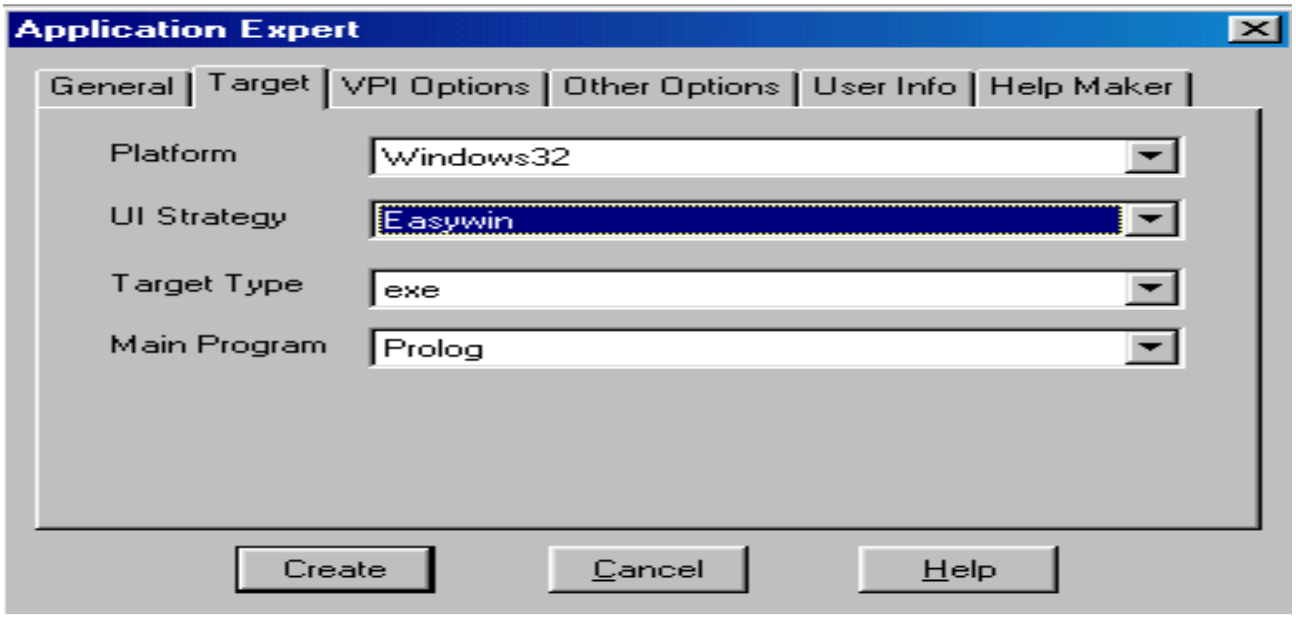


Рис. 5. Установки на вкладке Target диалогового окна **Application Expert**

Теперь нажмите кнопку **Create** для того, чтобы создать файлы проекта по умолчанию.

4. Установите требуемые опции компилятора для созданного проекта. Для активизации диалогового окна **Compiler Options** выберите команду **Options | Project | Compiler Options.** Откройте вкладку **Warnings.** Выполните следующие действия:

• установите переключатель Nondeterm**.** Это нужно для того, чтобы компилятор Visual Prolog принимал по умолчанию, что все определенные пользователем предикаты — недетерминированные (могут породить более одного решения);

• снимите флажки **Not Quoted Symbols, Strong Type Conversion Check** и **Check Type of Predicates.** Это будет подавлять некоторые возможные предупреждения компилятора;

•нажмите кнопку **ОК,** чтобы сохранить установки опций компилятора.

В результате этих действий диалоговое окно **Compiler Options** будет выглядеть, как показано на рис. 6.

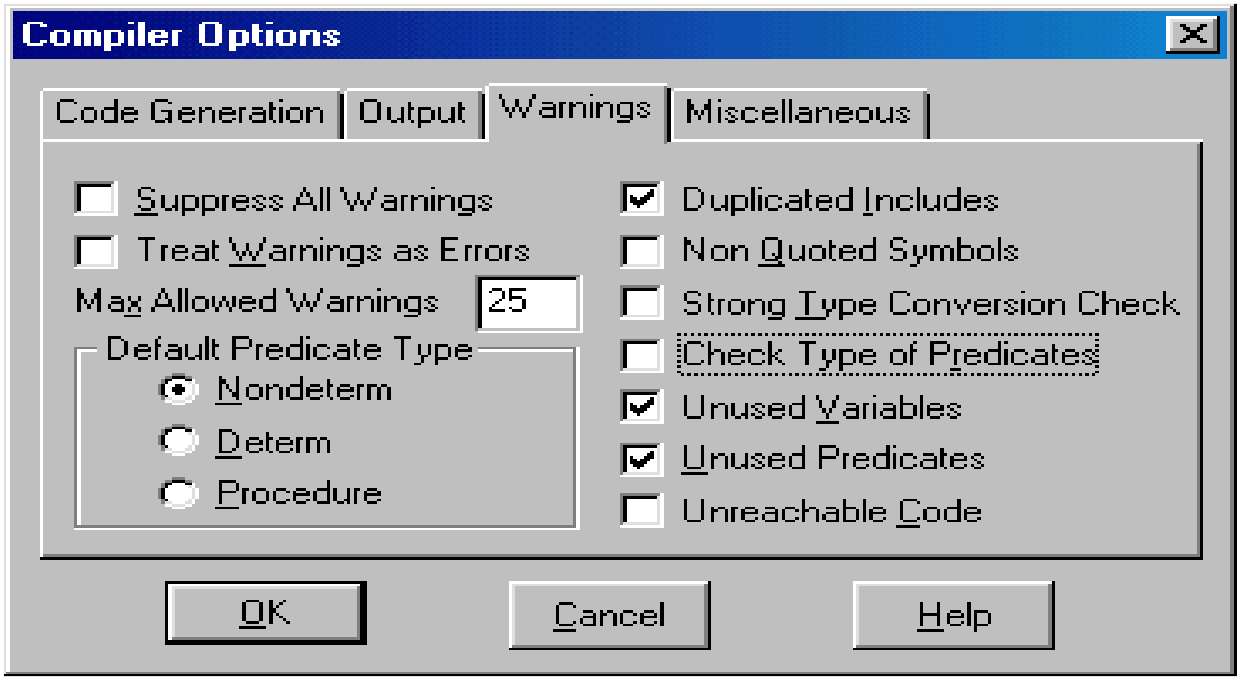


Рис. 6. Установки опций компилятора

4 Запуск и тестирование программы

Для проверки того, что ваша система настроена должным образом, следует выполнить следующие действия:

1. В окне проекта открыть файл **test.pro** (рис. 7)

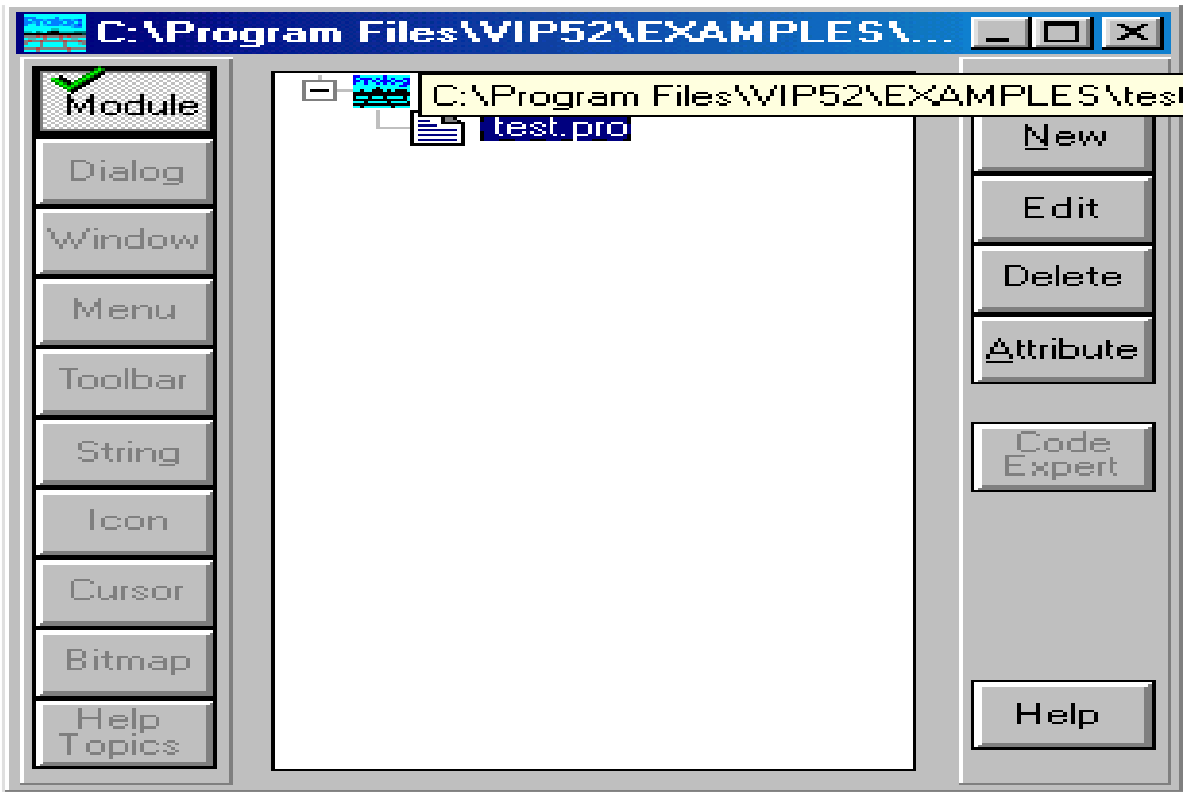


Рис. 7 Окно проекта.

2. В разделе GOAL наберите с клавиатуры write ("Hello world"), nl .

3. Нажать на панели инструментов кнопку (либо комбинацию клавиш <Ctrl>+<G>, либо активировать команду **Project | Test Goal**). В терминологии языка Пролог это называется *GOAL*, и этого достаточно для программы, чтобы она могла быть выполнена. Если ваша система установлена правильно, то экран монитора будет выглядеть, как показано на рисунке 8.

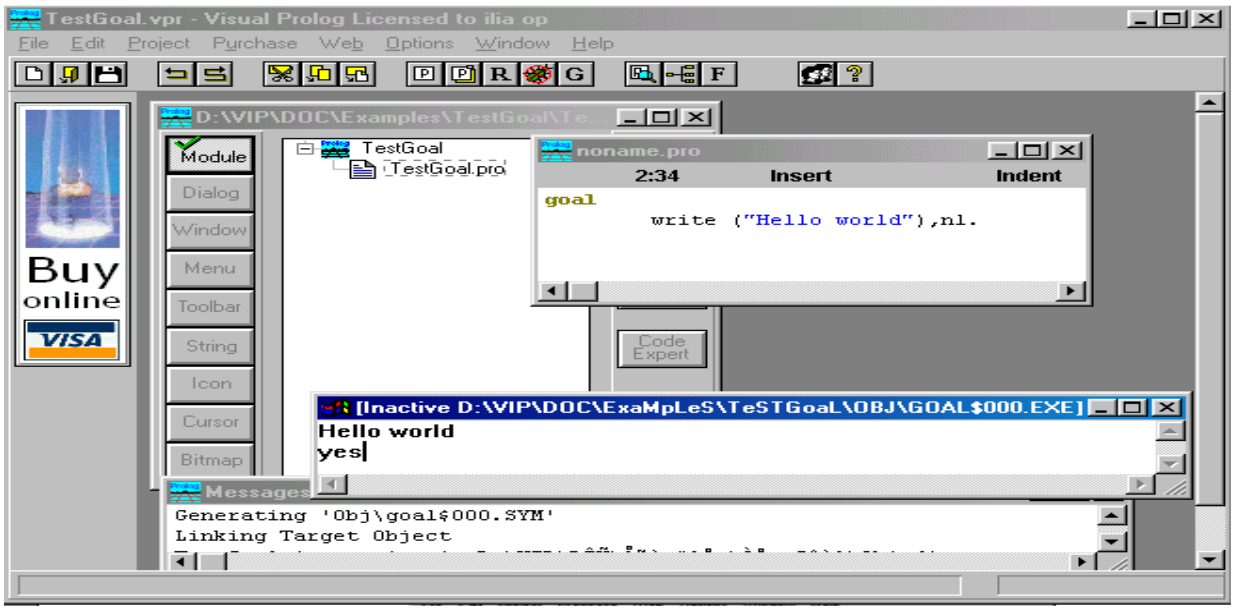


Рис. 8. Тестовая программа “Hello world”.

Результат выполнения программы будет расположен вверху в отдельном окне**,** которое необходимо закрыть перед тем, как тестировать другую GOAL.

5 Свойства утилиты Test Goal

Утилита среды визуальной разработки интерпретирует GOAL как специальную программу, которая компилируется, компонуется, генерируется в исполняемый файл и Test Goal запускает его на выполнение. Эта утилита внутренне расширяет заданный код GOAL, чтобы сгенерированная программа находила все возможные решения и показывала значения всех используемых переменных. Утилита Test Goal компилирует этот код с использованием опций компилятора, заданных для открытого проекта (рекомендуемые опции компилятора для TestGoal-проекта определили ранее).

6 Обработка ошибок

Если вы допустили ошибки в программе и пытаетесь скомпилировать ее, то среда визуальной разработки отобразит окно Errors (Warnings), которое будет содержать список обнаруженных ошибок.

Дважды щелкнув на одной из этих ошибок, вы попадете на место ошибки в исходном тексте. Можно воспользоваться клавишей <F1> для вывода на экран интерактивной справочной системы Visual Prolog. Когда окно помощи откроется, щелкните по кнопке Search, наберите номер ошибки, и на экране появится соответствующее окно помощи с более полной информацией о ней.

Подробному рассмотрению основных функций интегрированной среды визуальной разработки VDE Visual Prolog посвящена следующая глава.

7 Команды построения

7.1 Команда Project / Compile Module

Эта команда (ей соответствует комбинация клавиш <Ctrl>+<F9>) делает попытку компилировать модуль, содержащий редактируемый в данный момент файл. Выполнение команды зависит от следующих свойств файла:

- если файл имеет расширение pro и является модулем текущего проекта, то VDE пытается компилировать этот файл;

- если файл не является модулем текущего проекта и его расширение — pro, pre, inc, con или dom, то VDE пытается найти модуль проекта, который включает этот файл, и откомпилировать первый найденный модуль;
- во всех остальных случаях VDE пытается компилировать модуль, выбранный в окне проекта. VDE не может компилировать файл, который не является частью открытого проекта. Вместо этого файла VDE будет компилировать модуль, выбранный в окне проекта.

Если в VDE не открыт ни один проект, то никакие файлы компилироваться не будут. Команда меню Project | Compile Module заблокирована; комбинация клавиш <Ctrl>+<F9> не работает. Единственно возможное действие — это запустить утилиту Test Goal.

7.2 Команда Project / Build

Если со времени последнего построения проекта были изменены какие-либо ресурсы, то эксперты кода могут обновить некоторые секции в исходных файлах перед построением.

Эта команда (ей соответствует комбинация клавиш <Alt>+<F9>) строит проект, проверяя метки времени всех исходных файлов в проекте, поэтому если исходные файлы (или файлы, которые в них включены) являются более новыми, чем зависимые **OBJ**-файлы, то соответствующие модули проекта будут перекомпилированы.

Команда **Build** также строит файлы ресурсов и файл интерактивной справки (если необходимо). Затем проект компонуется для генерации целевого модуля (исполняемая программа или DLL).

7.3 Команда Project / Rebuild All

Эта команда (ей соответствует комбинация клавиш <Ctrl>+<Alt>+<F9>) выполняет то же действие, что и **Project | Build,** причем все файлы будут повторно сгенерированы или откомпилированы и скомпонованы независимо от их меток времени.

7.4 Команда Project I Stop Building

Эта команда (ей соответствует комбинация клавиш <Alt>+<F10>) используется для остановки компиляции/компоновки.

7.5 Команда Project I Run

Если необходимо, то эта команда (ей соответствует клавиша <F9>) выполнит действие **Project | Build** и затем запустит сгенерированный исполняемый файл.

7.6 Команда Project I Link Only

Эта команда (ей соответствует комбинация клавиш <Shift>+<F9>) используется для выполнения компоновки. В этом случае построитель программ вызывает компоновщика и не проверяет, нужно ли повторно компилировать какие-либо модули проекта (или даже впервые компилировать).

7.7 Команда Project / Test Goal

Эта команда (ей соответствует комбинация клавиш <Ctrl>+<G>) используется для тестирования простых целей (Goals). Программа компилируется и компонуется в специальном режиме, и затем запускается соответствующий исполняемый файл.

Утилита Test Goal ищет все решения для определенной в программе цели. Для каждого решения Test Goal отображает значения всех переменных из секции GOAL и число решений. Эта особенность — удобный способ проверить локальные предикаты в модуле.

Например, следующая цель:

GOAL X = 2; X = 1, Y = X + 1

приведет к такому результату (рис. 9):

Рис. 9 Вывод режима тестирования цели.

7.8 Команда Resource I Build Resource Only

Когда окно проекта активизировано, в меню Project появляется команда Resource. При выборе этого пункта (или нажатии комбинации клавиш <Alt>+<F8>) генерируются выбранные файлы с расширениями rc и res и необходимые файлы констант.

7.8.1 Пример

Загрузите программу в среду визуальной разработки Visual Prolog и запустите ее утилитой Test Goal.

```
predicates
likes(symbol,symbol)
clauses
likes(ellen,tennis).
likes(john,football).
likes(tom,baseball).
likes(eric,swimming).
likes(mark,tennis) .
likes(bill,Activity):-likes (tom, Activity) .
goal
likes(bill, baseball).
```

Утилита Test Goal ответит в окне приложения: yes (да)

Попробуйте также следующий запрос в GOAL-разделе: likes(bill,

tennis). Утилита Test Goal ответит: no (нет).

8 Команды отладки

8.1 Команда Project | Debug

Запускает процесс отладки. Отладчик также можно запустить сочетанием клавиш CTRL+SHIFT+F9.

При помощи диалога View можно открывать дополнительные информационные окна, которые отображают различные состояния среды и переменных в режиме отладки:

View → Call Stack (Открывает информационное окно стека вызова)

View → Local Variables (Открывает информационное окно локальных переменных)

Для выполнения шагов отладки используются следующие команды:

Run → Trace Intro [F7]

Run → Step Over [F8]

Run → Run to Cursor [F4]

Рис. 10. Окно отладчика с открытым листингом ch02e01.pro

Задание. Подготовить программу на языке ПРОЛОГ для полученного варианта**.**. Запустить программу в среде Visual Prolog в режиме отладки.

Контрольные вопросы

1. В чем суть логической модели знаний?
2. Какие области имеет программа на языке пролог?
3. Как выполняется вывод в ПРОЛОГ?
4. Чем процедурное программирование отличается от декларативного?
5. Rакую роль выполняет fail?

Содержание отчета

- цель работы
- краткие теоретические сведения
- описание предметной области
- структура программы

- Листинг программы
- Ответы на вопросы

5. Лабораторная работа № 4. Продукции в системах искусственного интеллекта.

Цель работы – изучение механизма вывода в продукционных системах.

Методические рекомендации

Продукционная система состоит из трех основных компонентов. Первый из них – это набор правил, используемый как база знаний, иногда его еще называют *базой правил*. Второй компонент – это *база фактов* или *рабочая память* – память для временного хранения, в которой хранятся предпосылки, касающиеся конкретных задач предметной области, и результаты выводов, получаемые на их основании. Третий компонент реализует *механизм логического вывода*, обрабатывающий правила в соответствии с содержанием рабочей памяти; другое название этого компонента – машина логического вывода.

Пусть *база правил* в продукционной системе имеет содержимое:

если *F* и *B* то *Z*; если *C* и *D* то *F*; если *A* то *D*;

рабочая память: *A, B, H, C*.

Рассмотрим, каким образом "работают" правила. Система построена так, что один раз выбранное правило из базы правил выполняться будет только один раз. Оно как бы «выгорает». Первым выгорает правило «если *A* то *D*», так как *A* уже имеется в базе данных. В качестве следствия этого правила получается логический вывод о наличии ситуации *D*, которая заносится в рабочую область. Это вызывает выгорание правила «если *C* и *D* то *F*», и, как следствие, выводится ситуация *F* и она заносится в базу данных. Это, в свою очередь, вызывает выгорание правила «если *F* и *B* то *Z*» с занесением *Z* в базу данных. Такой способ называется прямым выводом. Графически вывод можно изобразить в виде И/ИЛИ дерева (рис. 2).

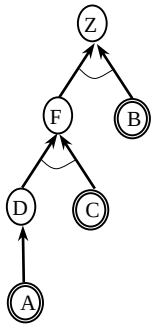


Рис. 2.

Существует и другой способ вывода, называемый *обратным выводом*. При его использовании система начинает работу с формулировки того, что требуется доказать, например, предполагая, что ситуация *Z* присутствует, и выполняет только те правила, которые имеют отношение к доказательству предположения.

Если на каждом этапе логического вывода существует множество применяемых правил, то это множество носит название *конфликтного набора*, а выбор одного из них называется разрешением конфликта. Чтобы повысить эффективность продукционной системы, необходимо решить проблему управления последовательностью применения правил или управления выводом.

Задание

База правил и рабочая память в продукционной системе имеет содержимое, заданное в вариантах. Проиллюстрировать графически механизм прямого и обратного логического вывода факта *A*. Обратите внимание на изменение содержимого рабочей памяти в процессе вывода. Проведите упорядочение правил вывода. Рассмотрите возможные конфликты при прямом и обратном выводе. Варианты из 1 лабораторной работы.

Контрольные вопросы

1. В чем суть продукционной модели знаний?
2. Что такое антецендент?
3. Какой вывод эффективнее в продукционной системе?
4. Как выполняется вывод в продукционной системе знаний?
5. Как устраняются конфликты в продукционной модели знаний?

Содержание отчета

- цель работы
- краткие теоретические сведения

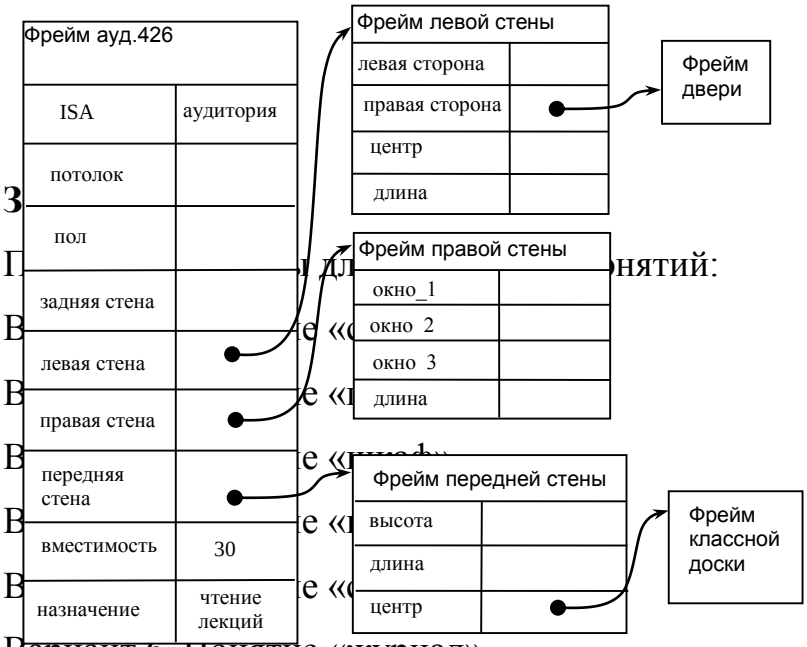
- описание предметной области
- структура продукционной модели
- Листинг программы
- Ответы на вопросы

6. Лабораторная работа № 6. Фреймовые модели представления знаний

Цель занятия – изучение представления статических знаний на основе фреймов.

Методические рекомендации

Фреймовая модель, или модель представления знаний, основанная на фреймовой теории М.Минского, представляет собой систематизированную в виде единой теории психологическую модель памяти человека и его сознания. Важным моментом в этой теории является понятие фрейма – структуры данных для представления некоторого концептуального объекта. Информация, относящаяся к этому фрейму, содержится в слотах. Все фреймы взаимосвязаны и образуют единую сеть фреймов. Однако четкого определения связи между фреймами и слотами может и не быть. Рассмотрим пример фреймовой системы, описывающей аудиторию 426.



З

П нятий:

В е «

В е «

В е «

В е «

В е «

Вариант 6. Понятие «журнал».

Вариант 7. Понятие «книга».

Вариант 8. Понятие «ребенок».

Вариант 9. Понятие «трактор».

Вариант 10. Понятие «посуда».

Контрольные вопросы

1. В чем суть фреймовой модели?
2. Каким образом выполняется наследование в фреймах?
3. Как реализуются фреймы в программной среде CLIPS ?
4. Что такое фрейм прототип?
5. В чем назначение фрейма демон?

Содержание отчета

- цель работы
- краткие теоретические сведения
- описание предметной области
- структура фреймовой сети
- Листинг программы
- Ответы на вопросы

7. Лабораторная работа № 6. Нейронные сети в системах искусственного интеллекта. «Аппроксимация функций нейронной сетью»

Цель работы: В среде Матлаб необходимо построить и обучить нейронную сеть для аппроксимации таблично заданной функции.

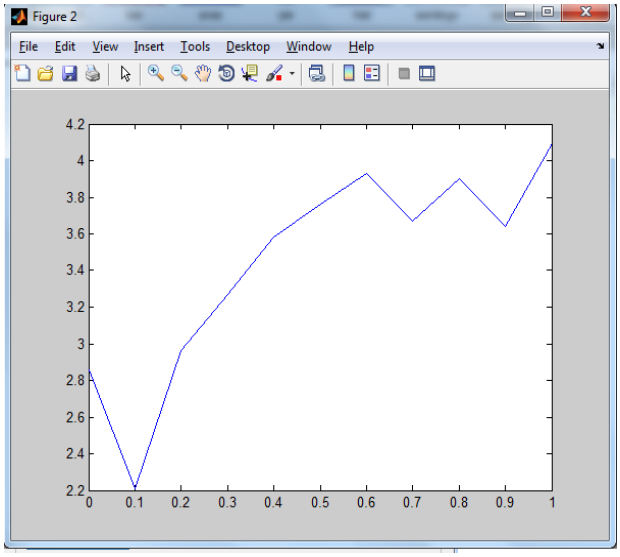
Ход работы Создаем таблицу экспериментальных данных:

yi – задано вариантом;

xi – вычисляется по следующей формуле:

$xi = a + h\,i$, $i = 0,1, ,10$, $h = (b - a)/10$ наотрезке $[a,b]$.

x_i											
$f(x_i)$											



а)

Рисунок 1 – график исходной функции

1. Создание и обучение нейронной сети:

```
x=[00.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1];
y=[2.86 2.21 2.96 3.27 3.58 3.76 3.93 3.67 3.90 3.64 4.09];
net=newff([0 3],[10,1],{'tansig','purelin'},'trainbfg');
net.trainParam.epochs=300;
net.trainParam.show=50;
net.trainParam.goal=1.37e-2;
[net,tr]=train(net,x,y);
an=sim(net,x);
plot(x,y,'+r',x,an,'-g'); hold on;
xx=[0.1850.86];
v=sim(net,xx)
plot(xx,v,'ob','MarkerSize',5,'LineWidth',2)
```

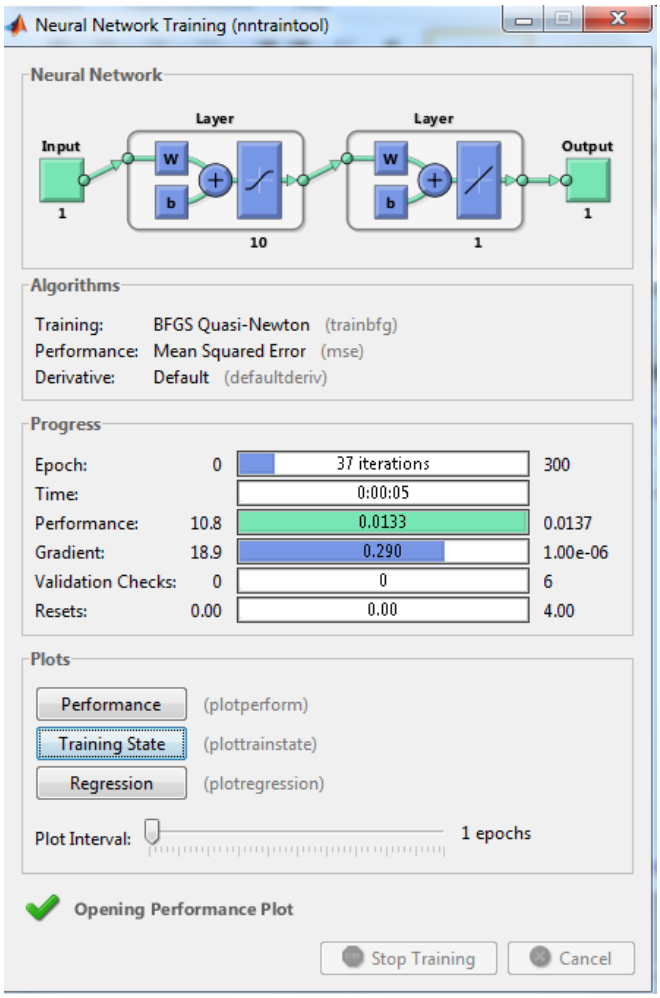


Рисунок 2 – Обучение сети

В процессе обучения сети получился график зависимости характеристики точности обучения сети от количества эпох (циклов), и вычисление среднеквадратичной ошибки сети составляет 0,013305 за 37 циклов:

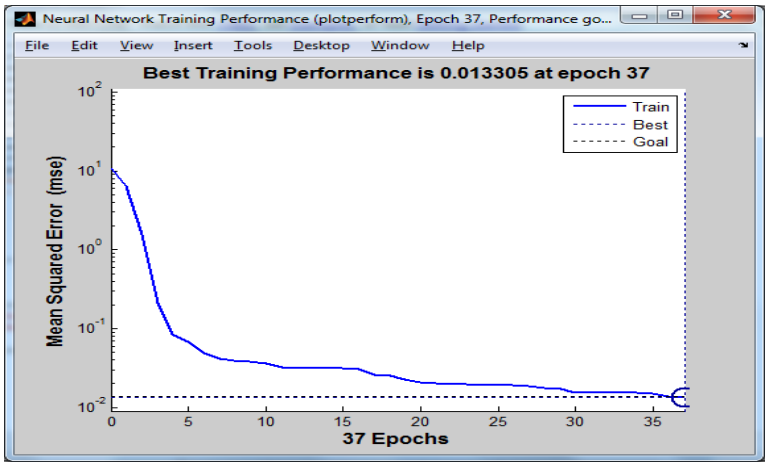


Рисунок 3 - Характеристика точности обучения в зависимости от количества эпох обучения

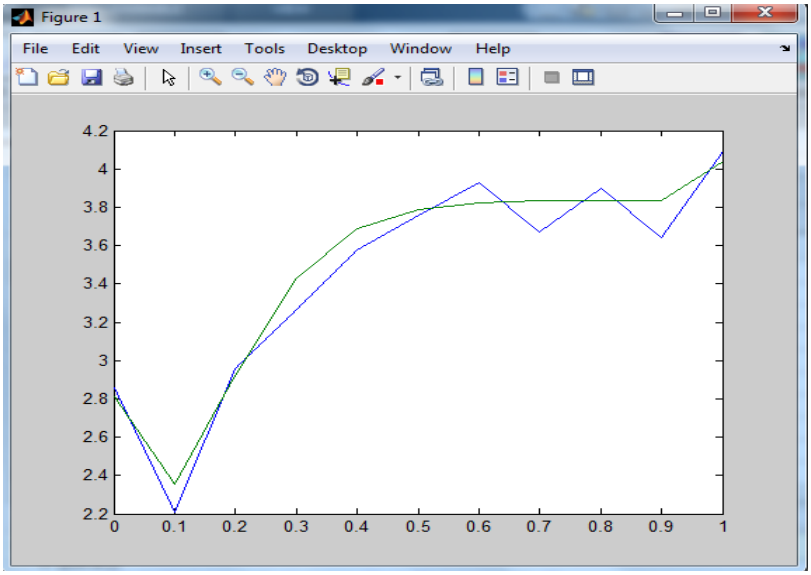


Рисунок 4 – Сравнение графиков исходной функции и аппроксимации

Аппроксимируем входящий набор точек методом МНК:
Результат для набора а) представлен ниже:

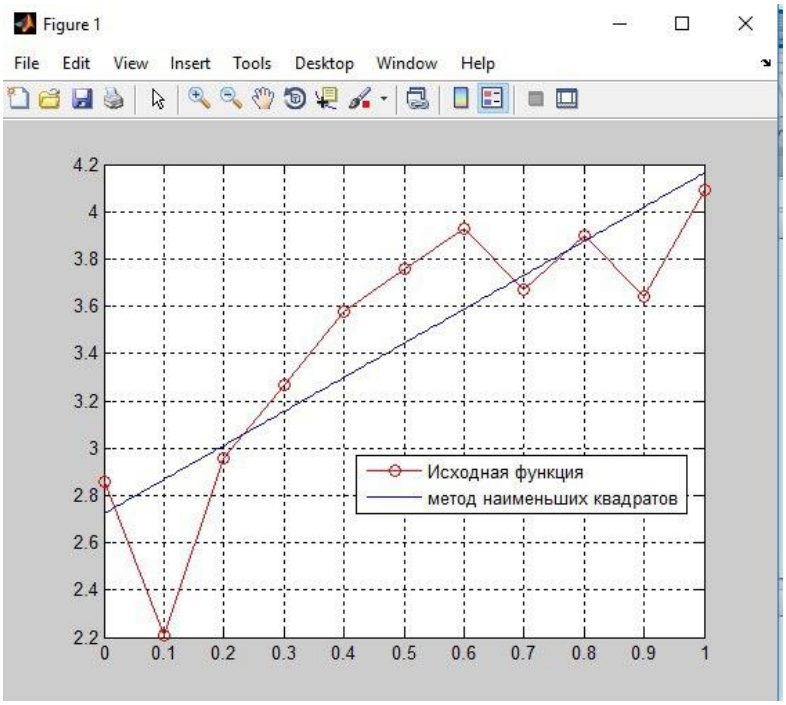


Рисунок 5 – Аппроксимация входящего набора точек методом МНК

Аналогичные действия проделываем для другого набора точек:

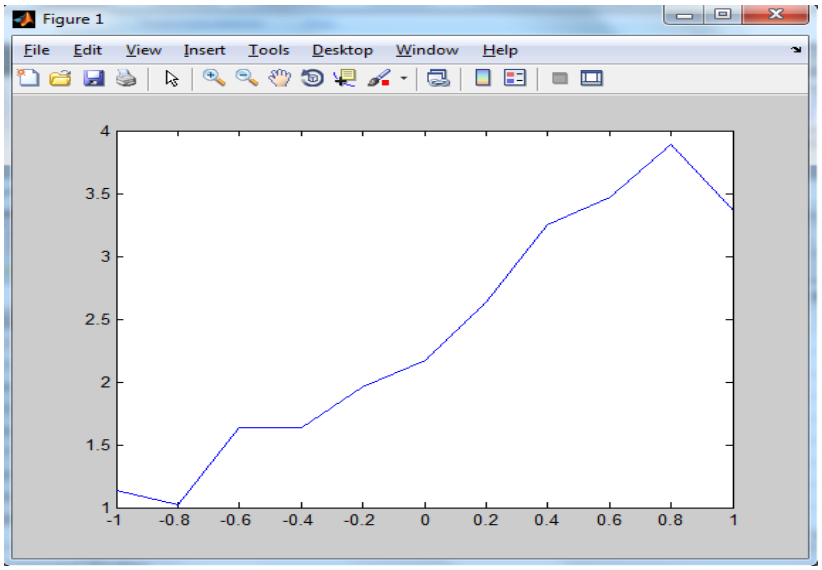
б) 

Рисунок 6 – график исходной функции

установлено по полученным результатам аппроксимации заданного набора значений функции, нейронная сеть намного лучше аппроксимирует исходное значении я функции, чем МНК.

Контрольные вопросы

1. Каким образом знаний хранятся в нейронной сети?
2. Какой метод обучения используется в нейронной сети?
3. Что такое синапс?
4. Как выполняется обучение в нейронной сети?
5. В чем назначение аксона?

Содержание отчета

- цель работы
- краткие теоретические сведения
- описание предметной области
- структура нейронной сети
- Листинг программы
- Ответы на вопросы

8. Лабораторная работа № 7. Работа с редакторами онтологий

Цель. Научиться создавать онтологии по заданной предметной оласти.

Задание.Создание онтологии в программе Protege

Онтологией называют схему, состоящую из классов связанных между собой посредством различных отношений и правил. Это своеобразная форма представления некоторой области знаний в формальном виде. В настоящее время онтологии широко используются в программировании, обучении, различного рода исследованиях.

Создание простой онтологии

Допустим необходимо разработать онтологию, выполняющую задачу классификации. Для примера возьмем простейшую классификацию попугаев.

В рамках предметной области можно выделим несколько основных классов, а именно:

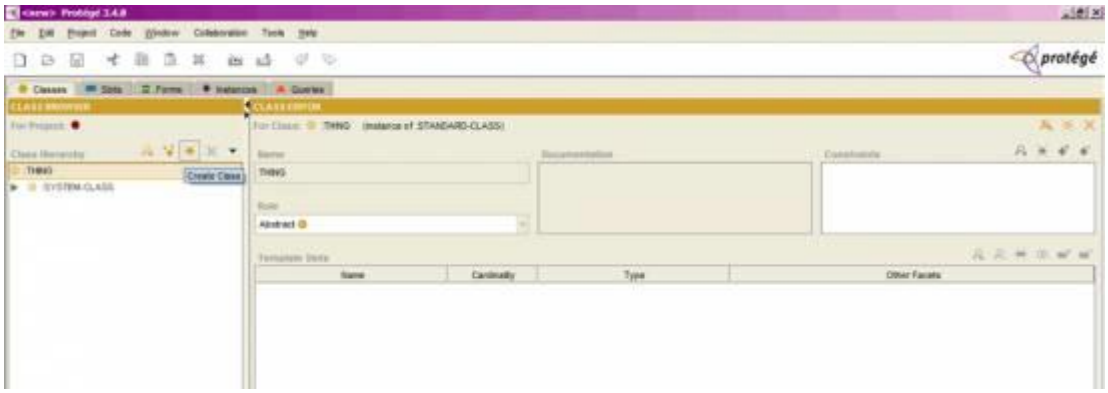
- Попугай – главный класс, содержащий 3 классса-наследника – мелкие попугаи, крупные попугаи и средние попугаи.
- Владелец – класс, содержащий информацию о человеке-владельце.
- Регион – класс, содержащий информацию о месте обитания попугая.

Далее можно переходить к созданию проекта.

Первым шагом запускаем программу Protege и создаем новый проект. В окне настроек выбираем Protege Files.

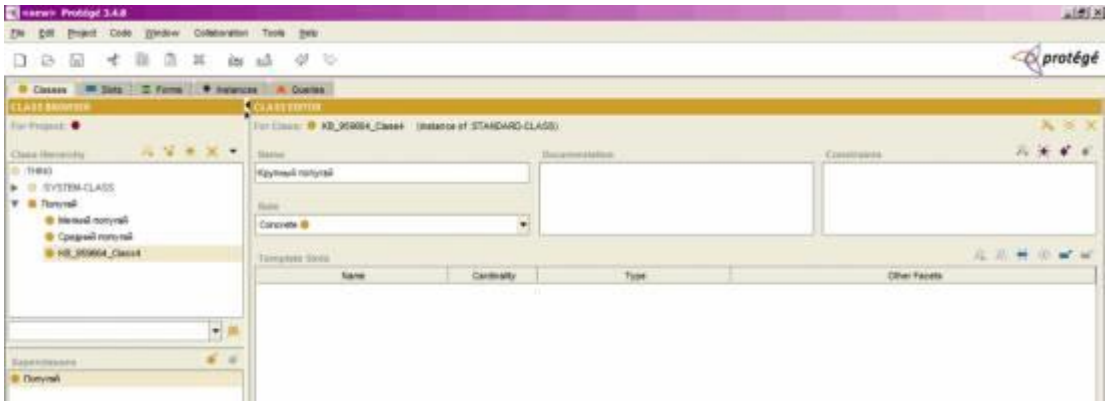
На экране появляется рабочее окно, в котором нам и предстоит работать.

Первым делом при создании онтологии необходимо создать классы. Все спроектированные нами классы будут отображаться в окне Class Browser. Для создания нового можно щелкнуть на иконку «Create Class» или правой кнопкой мыши на поле браузера классов с указанием действия создания класса.

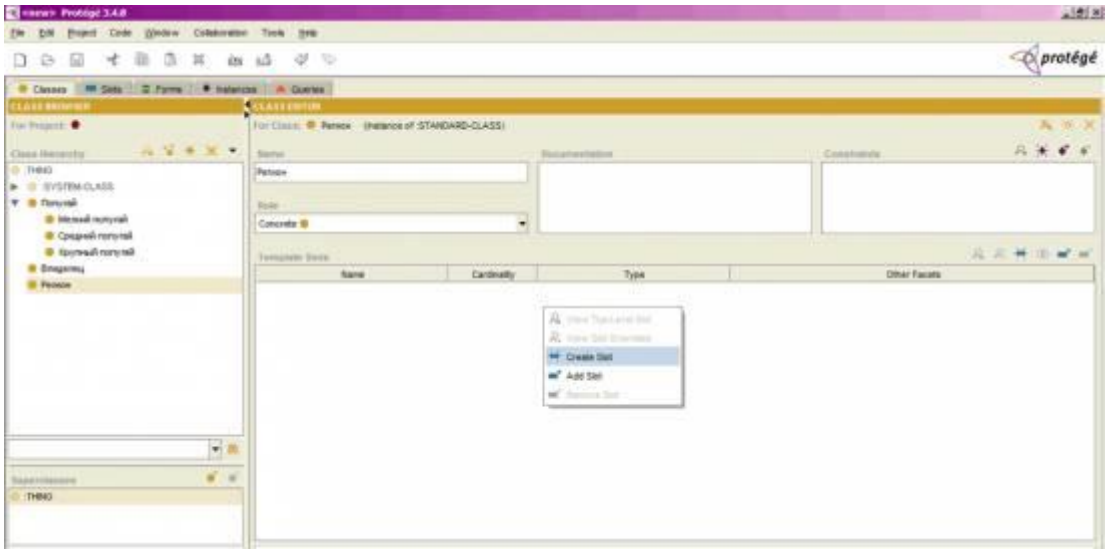


После создания класса, можно переименовать его в поле Name.

Для создания подкласса щелкаем по классу-родителю, нажимаем правую кнопку мыши и выбираем команду "Create Subclass”. При желании класс можно сделать абстрактным, выбрав соответствующий пункт в выпадающем списке Role.

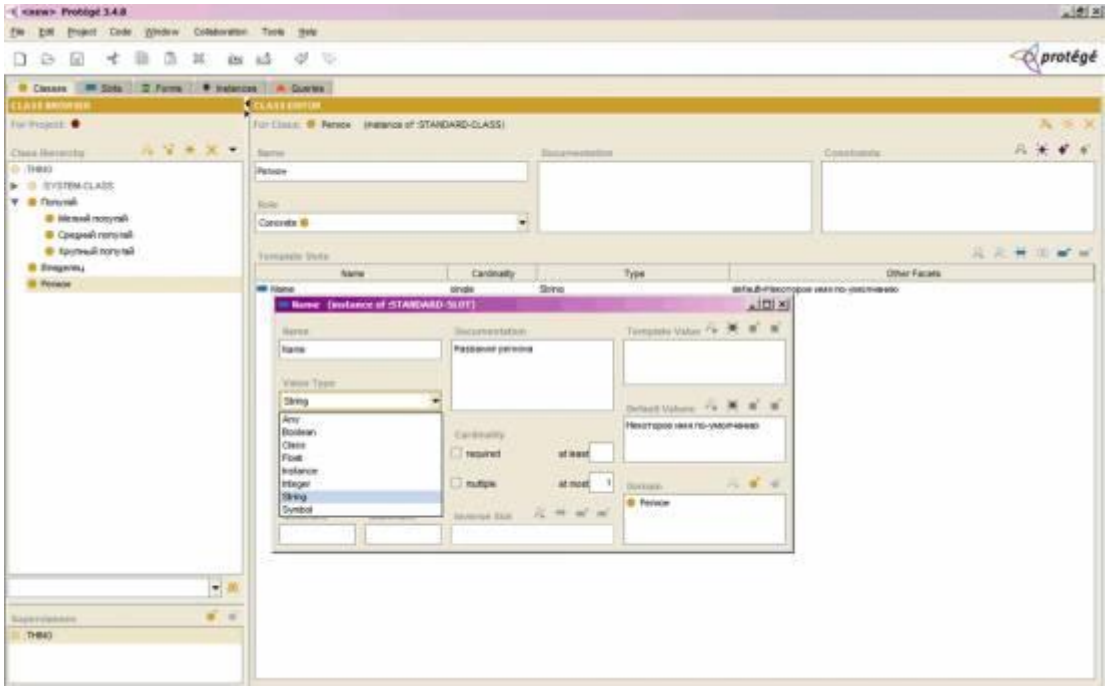


После создания классов необходимо прописать в них поля – свойства. К примеру, у класса Регион будет свойство «Имя», которое будет содержать название региона, в котором проживают попугаи. Для добавления свойства в класс необходимо щелкнуть правой кнопкой мыши в окне «Template Slots» и указать команду «Create Slot».



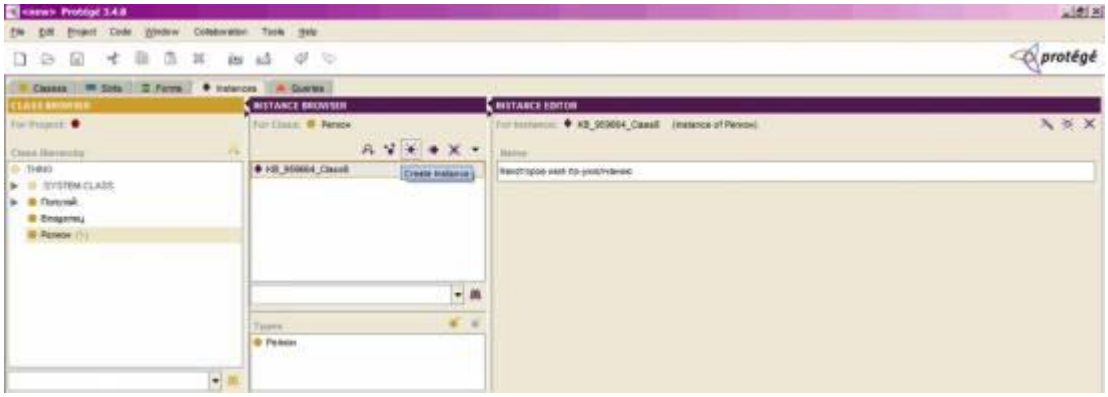
При создании слота ему можно задать название, тип, значение по-умолчанию, временное значение, описание и т.п. Стоит отметить, что в качестве типа слота может выступать объект другого класса. Таким способом в программе Protégé устанавливается взаимосвязь между 2 классами.

Если ранее какой-либо слот, например, «имя» уже создавался, то его можно просто добавить в класс (при условии, что он подходит), нажав на кнопку в виде прямоугольника с плюсом в правом верхнем углу окна Template Slots.

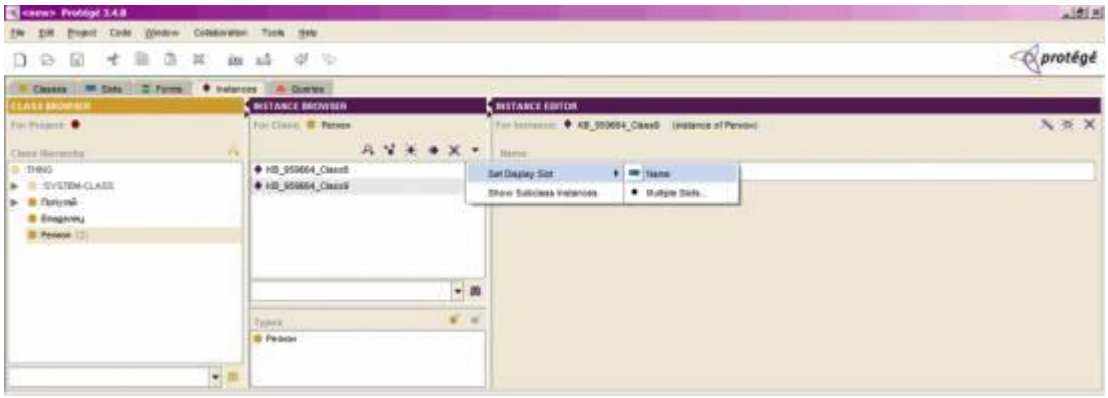


После создания слотов и классов можно приступать к созданию экземпляров или Instances. Для этого сверху щелкните на одноименную вкладку и посередине увидите окно под названием Instance Browser.

Для того, чтобы создать экземпляр какого-либо класса, щелкните в окне Class Browser на нужный класс, а затем в окне Instance Browser нажмите на иконку добавления сущности Create Instance. После нажатия в окне Instance Browser появится строчка с вновь созданной сущностью, а справа в окне Instance Editor поля, соответствующие слотам класса, который необходимо заполнить.



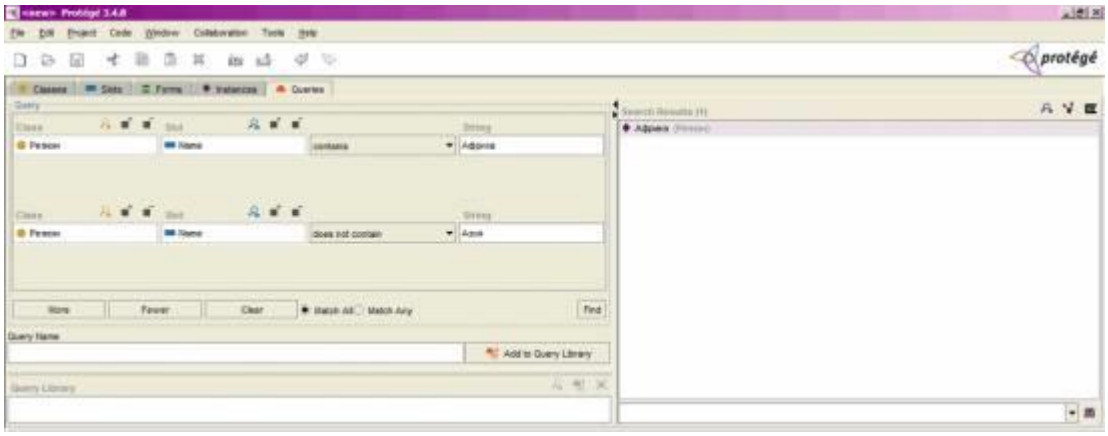
Чтобы в окне Instance Browser отображать объекты по какому-либо признаку, щелкните на треугольник справа на панели и выберите свойство, которое отображать в браузере. В данном примере укажем свойство «Имя».



После создания сущностей можно приступать к формированию запросов. Формы для них находятся во вкладке запросы.

Чтобы создать необходимый запрос, нужно выбрать класс, в котором будет производиться поиск, свойство, по которому будет производится поиск, а также указать признак. Под признаком может пониматься как строчка, так и условие is, is not, contains, begins with и так далее.

Если запрос необходимо сделать составным, то есть содержащим 2 и более условий, в окне следует нажать кнопку more и ввести данные в соответствующие поля.



Созданный запрос можно сохранить и оставить в библиотеке, введя внизу в поле имя и нажав кнопку «Add to Query Library».

Задание. Создать онтологии для полученного варианта. (Лабораторная работа № 1).

Контрольные вопросы

1. Что такое онтология для искусственного интеллекта?
2. Какие уровни онтологий известны?
3. Какие программные среды для построения онтологий известны?
4. Что такое дескриптивные логики ?
5. В чем назначение OWL?

Содержание отчета

- цель работы
- краткие теоретические сведения
- описание предметной области
- структура онтологий
- Листинг программы
- Ответы на вопросы

9. Лабораторная работа № 8. Построение экспертных систем различных предметных областей в программной среде CLIPS.

Цель: ознакомиться с особенностями языка CLIPS, получить практические навыки разработки советующих систем, основанных на использовании модели представления знаний.

Общие сведения

CLIPS располагает тремя механизмами представления знаний:

процедурным, эвристическим и объектно-ориентированным.

Процедурный механизм позволяет пользователю при помощивстроенных в язык функций разрабатывать или конструировать новыефункции, выполняющие некоторые действия или возвращающие какие-либозначения. В этом смысле CLIPS напоминает такие известные языкипрограммирования, как C, C++ или Pascal. Так, для созданияпользовательских функций используется конструктор deffunction, имеющийследующий синтаксис:

(deffunction имя_функции
[необязательный комментарий]
(список формальных параметров)
(действие_1)
(действие_2)
.....
(действие_N))

Например, определим функцию om(x,y), которая возвращает целуючасть частного от деления переменной y на переменную x:

(deffunction om
(?x ?y)
(div ?y ?x))

В CLIPS имя переменной начинается ссимвола “ ? “, и что для вызова функции (в данном случае – встроеннойфункции деления нацело div) используется префиксная нотация, а также нато, что вся конструкция представляет собой

список, состоящий из четырехполей. Этим CLIPS похож не только на C, но и на LISP.

Эвристический механизм представления знаний в CLIPS реализуетсяпри помощи правил в форме

ЕСЛИ условие_1 и ... и условие_N выполняются,ТО

ВЫПОЛНИТЬ действие_1 и ... и действие_N.

Список условий называется левой частью правила (Left-Hand Side илиLHS). Список действий называется правой частью правила (Right-Hand Sideили RHS). Возможность применить конкретное правило определяется тем,выполняются ли условия, которые сформулированы в его левой части.

Выполнение или невыполнение условий определяется в момент ихсопоставления с так называемыми фактами, которые образуют ни что иное,как базу данных.

В CLIPS такая база данных может представлять некоторуюпредметную область, исходное или текущее состояние какой-либо проблемы,может моделировать в пространстве или во времени поведение какой-либосистемы или любой сущности, которую можно описать посредствоммножества записей в виде списков.

Для создания базы данныхиспользуется конструктор deffacts. Его синтаксис:

(deffacts имя_базы_данных

[необязательный комментарий]

(факт_1)

(факт_2)

.....

(факт_N))

Каждый факт в базе данных представляет собой запись в виде списка.Список может содержать одно или несколько полей, принимающихсимвольные либо числовые значения. Список также может быть пустым.Если каждое условие в левой части правила находит себя среди фактов

– происходит активизация правила и выполняются ВСЕ действия,

записанные в его правой части. В противном случае правило неактивизируется.

Работа правила очень напоминает условный оператор if-then,присутствующий во многих процедурных языках программирования. Интерпретатор CLIPS при старте программы, содержащей множество фактов и правил,интерпретатор CLIPS запускает машину логического вывода, котораявыясняет, какие из правил можно активизировать. Это выполняетсяциклически, причем каждый цикл состоит из трех шагов:

· сопоставление фактов и правил;

· выбор правила, подлежащего активизации;

· выполнение действий, предписанных правилом.

Таким образом, правила, взаимодействующие с базой данных в видефактов, вносят в нее функциональность и образуют вместе с ней базу знаний.Для создания правила используется конструктор defrule, который имеетследующий синтаксис:

(defrule имя_правила
[необязательный комментарий]
[необязательное объявление]
(условие_1)
(условие_2)
.....
(условие_M)
=>
(действие_1)
(действие_2)
.....
(действие_N))

Левая часть правила отделяется от правойкомбинацией символов “=>”, а количество условий и действий в правиле вобщем случае не совпадает. Для пояснения вышесказанного рассмотримпример.

Пример. Предметная область, участники некоторой конференции, приехавших из разных городов. Все участники обычно проходят регистрацию. Пусть эта процедура представляет собой ввод сведений об участниках в базу данных, вкоторой на каждого участника выделяется одна запись (факт), состоящая из списка с тремя полями. Пусть первое поле имеет символьное значение rep сокращение от representative (представитель). В общем случае это значение может быть любым, а поле может отсутствовать. Во втором поле списка хранится фамилия участника, а в третьем – город, из которого участникприбыл. Содержимое фактов базы данных с именем rep может быть,например, таким:

```
(deffacts rep
(rep Alejnov Odessa)
(rep Ladak Odessa)
(rep Slobodjanjuk Lvov)
(rep Klitka Lvov)
(rep Bojko Kiev)
(rep Pustovit Odessa)
(rep Spokojnij Odessa)
(rep Shamis Odessa)
(rep Lobovko Kiev)
(rep Zadorozhna Lvov)
(rep Javorskij Lvov))
```

Используя любой текстовый редактор, создадим и сохраним базуданных в виде текстового ASCII-файла с именем, повторяющим имя базыданных (то есть rep). Это позволяет легко редактировать данные, независимоот каких-либо других программных модулей, добавляя новых участников илиудаляя выбывших.

После окончания конференции организаторы подводят итоги. Пусть требуется определитьколичество представителей от каждого города. Для каждого города задаем счетчик и последовательнопросматриваем списки в записях файла rep.

Если в записи третье поле списка имеет значение Kiev, то содержимое соответствующего счетчика увеличиваем на единицу. Для других городов – аналогично. Программа на языке CLIPS, реализующая указанный алгоритм, может быть, например, такой:

```
(defglobal ?*odessa* = 0)
(defglobal ?*kiev* = 0)
(defglobal ?*lvov* = 0)
(defrule start
(initial-fact)=>(printout t crlf «REPRESENTATIVES» crlf)
(defrule odessa
(rep ? Odessa)=>(bind ?*odessa* (+ ?*odessa* 1)))
(defrule kiev
(rep ? Kiev)
=>
(bind ?*kiev* (+ ?*kiev* 1)))
(defrule lvov
(rep ? Lvov)
=>
(bind ?*lvov* (+ ?*lvov* 1)))
(defrule result
(declare (salience -1))
(initial-fact)
=>
(printout t «from Odessa: « ?*odessa* crlf)
(printout t «from Kiev: « ?*kiev* crlf)
(printout t «from Lvov: « ?*lvov* crlf))
```

В первых трех строках программы при помощи конструктора defglobal объявляются три глобальные переменные: ?*odessa*, ?*kiev* и ?*lvov*. Эти

переменные являются счетчиками. В CLIPS переменная может быть илокальной – но тогда она связывается только с тем правилом, в которомобъявляется.

Далее следует правило с именем start, левая часть которого представляет собой запись (initial-fact). Так обозначается системный начальный факт, который создается в рабочей памяти интерпретатора CLIPS по команде (reset) до запуска программы на выполнение. В

В CLIPS-программах распространенными правилами являются такие, которые добавляют факты в базу данных, либо, наоборот, удаляют их. При старте программы в базе данных нет фактов, удовлетворяющих хотя бы одному правилу. В этом случае программа ничего не выполнит. Для того чтобы начать вычисления и используется системный начальный факт, который, независимо от фактов в базе данных, активизирует некоторое правило, добавляющее такие факты, которые, в свою очередь, активизируют правила, условия которых не выполнялись в начальный момент.

В данной программе (initial-fact) запускает правило start, которое активизируется независимо от фактов в файле rep и присутствует в программе только с одной целью – вывести заголовок. Для этого в его правой части вызывается встроенная функция printout с ключом t, выводящая на стандартное устройство вывода (монитор) заголовок, заключенный в кавычки. Комбинация символов crlf является аналогом endl в C++ и служит для перевода курсора на следующую строку.

Следующие три правила с именами odessa, kiev и lvov можно назвать ядром программы. В них производится подсчет количества участников – соответственно, из Одессы, Киева и Львова.

Рассмотрим правило lvov. Оно активизируется в том случае, когда в базе данных находится факт (rep ? Lvov). Символ “ ? “ во втором поле этого списка означает символ универсальной подстановки и заменяет собой любую фамилию. Отсюда следует, что правило lvov активизируется столько раз, сколько раз факт (rep ? Lvov) присутствует в базе данных. При этом столько

же раз выполнятся действия, содержащиеся в правой части правила. Встроенная функция bind – аналог оператора присваивания. Следовательно, содержимое переменной ?*lvov* увеличивается на единицу и результат сохраняется в этой же переменной.

Аналогично работают правила odessa и kiev.Действия, которые выполняются в последнем правиле программы, отражены в его названии. Правая часть правила особых комментариев не требует, в то время как левая часть заслуживает подробного рассмотрения. BCLIPS существует несколько стратегий очередности выполнения правил, а сами правила могут иметь приоритет, который задается встроенной функцией declare с параметром salience (особенность). Этот параметр можетпринимать целочисленные значения от –10000 до +10000. По умолчанию длявсех правил величина salience равна нулю. Если в правиле result не указать приоритет, оно будет конфликтовать с правилом start за очередность выполнения, так как у этих правил одинаковая левая часть. Для устраненияконфликта в правиле result приоритет указан явно и со знаком минус, в связис чем это правило выполнится последним.

Используя любой текстовый редактор, наберите и сохраните текст программы в ASCII-файле со стандартным для CLIPS-программ расширением .clp и с именем represent. Командой clips вызовем интерпретатор CLIPS, командой (load имя_файла) загрузим в интерпретатор файлы rep и represent.clp, командами (reset) и (run) запустим программу represent.clp на выполнение.

```
CLIPS (V6.21 06/15/03)
CLIPS> (load rep)
.....
TRUE
CLIPS> (load represent.clp)
.....
TRUE
CLIPS> (reset)
```

```
CLIPS> (run)
REPRESENTATIVES
from Odessa: 5
from Kiev: 2
from Lvov: 4
CLIPS>
```

Сообщение интерпретатора TRUE означает, что в файле нет синтаксических ошибок и команда загрузки выполнена корректно. Многоточием представлены другие сообщения интерпретатора, которые в данном случае опущены.

Как следует из описанных действий, в интерпретаторе CLIPS находятся два файла. Первый, с именем rep, является базой данных. Второй, с именем represent.clp, содержит сведения (правила) о том, как эти данные могут быть использованы. Таким образом, вместе файлы образуют базу знаний, которая содержит, по крайней мере, два знания. Первое – общий состав участников конференции. Его можно посмотреть, не выходя из интерпретатора по команде (facts). Второе знание – количество участников от каждого города. В рассмотренном примере база знаний состоит из двух программных модулей. Однако ничто не мешает использовать одну программу, сохраненную в одном файле.

Задание. Необходимо, изучив команды CLIPS и приведенный пример, разработать экспертную систему по заданной предметной области Практическое задание 1.

Контрольные вопросы

1. Какая модель знаний используется в CLIPS?
2. Зачем нужна объяснительная подсистема?
3. Что хранится в базе данных?

4. Как выполняется вывод советов в экспертной системе?

5. В чем закдючается назначение базы знаний?

Содержание отчета

- цель работы
- краткие теоретические сведения
- описание предметной области
- структура экспертной системы
- Листинг программы
- Ответы на вопросы

10. **Библиографический список**

1. Болотова Л. С. истемы искусственного интеллекта: модели и технологии, основанные на знаниях: учебник для вузов / Л. С. Болотова ; Министерство образования и науки Российской Федерации, Российский государственный университет инновационных технологий и предпринимательства, Государственный научно-исследовательский институт информационных технологий и телекоммуникаций "Информатика". - М. : Финансы и статистика, 2012. - 664 с (15 экз. в библиотеке ТУСУР)

2. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер.с польск.И.Д.Рудинского. ЭБС “Лань” [Электронный ресурс] / Д. Рутковская, М. Пилиньский, Л. Рутковский. — Электрон. дан. — М. : Горячая линия-Телеком, 2013. — 384 с. – Режим доступа http://e.lanbook.com/book/

3. Замятин Н. В. Нечеткая логика и нейронные сети: учебное пособие / Н. В. Замятин ; рец.: И. А. Ходашинский, С. Н. Ливенцов ; Министерство образования и науки Российской Федерации, Томский государственный университет систем управления и радиоэлектроники (Томск). - Томск: Эль Контент, 2014. - 146 с. (10 экз. в библиотеке ТУСУР)

4. Павлов С. Н. П Системы искусственного интеллекта : учеб. пособие. В 2-х частях. /С. Н. Павлов. — Томск: Эль Контент, 2011. —176 с. (10 экз. в библиотеке ТУСУР)

5. Цуканова, Н.И. Теория и практика логического программирования на языке Visual Prolog 7.Учебное пособие для вузов. [Электронный ресурс] : Учебные пособия / Н.И. Цуканова, Т.А. Дмитриева. — Электрон. дан. — М. : Горячая линия-Телеком, 2013. — 232 с. — Режим доступа: http://e.lanbook.com/book/11847.

6. Абдикеев Н. М. Проектирование интеллектуальных систем в экономике[Электронный ресурс]- Режим доступа www.dgybga.ru/.../proektirovanie-**intellektual**nih-**sistem**-v-ekonomike-uchebnik-abdigeev...

7. Козлов, А.Н. Интеллектуальные информационные системы: учебник /А.Н. Козлов; Мин-во с-х. РФ, ФГБОУ ВПО Пермская ГСХА. – Пермь: Изд-во ФГБОУ ВПО Пермская ГСХА, 2013.– 278 с. [Электронный ресурс] - Режим доступа. www.metodichka.x-pdf.ru/.../322646-1-an**kozlov**-intellektualnie-informacionnie-**sistem**

8. Интеллектуальные информационные системы и технологии : учебное пособие / Ю.Ю. Громов, О.Г. Иванова, В.В. Алексеев и др. – Тамбов : Изд-во ФГБОУ ВПО «ТГТУ», 2013. – 244 с. [Электронный ресурс]- Режим доступа.

9. Смагин, А. А. Интеллектуальные информационные системы : учебное пособие / А. А. Смагин, С. В. Липатова, А. С. Мельниченко. – Ульяновск : УлГУ, 2010. – 136 с. [Электронный ресурс]- Режим доступа.

10. Гущин А.Н., Радченко И.А. Экспертные системы: учебное пособие / А.Н. Гущин, И.А. Радченко; Балт. гос. техн. ун-т. — СПб., 2007 — с. [Электронный ресурс]- Режим доступа bezogr.ru/uchebnoe-posobie-sankt-peterburg-2007-gushin-a-n-radchenko-i-a.html

11. Новиков Ф. А. Искусственный интеллект: представление знаний и методы поиска решений: Учеб. пособие. – СПб.: Изд-во Политехн. ун-та, 2010. – 240 с.

[Электронный ресурс]- Режим доступа window.edu.ru/resource/677/76677