Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
Амурский государственный университет
(ГОУВПО «АмГУ»)

УТВЕРЖДАЮ
Зав. кафедрой ИиУС
_____ А.В. Бушманов
«___» _____ 2007 г.

Учебно-методический комплекс

по дисциплине «Системы искусственного интеллекта»

для студентов специальности

230102 – Автоматизированные системы обработки информации и управления;

Составитель: доцент кафедры ИУС Акилова И.М.

Факультет математики и информатики

Кафедра информационных и управляющих систем

2007 г.

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного
университета

И.М.Акилова

Системы искусственного интеллекта для специальности 230102 «Автоматизированные системы обработки информации и управления»: учебно-методический комплекс дисциплины. / Акилова И.М. – Благовещенск. Изд-во Амурского гос. ун-та, 2007. 201 с.

Пособие содержит рабочую программу, курс лекций, методические рекомендации по проведению и выполнению лабораторных и практических работ. Составлено в соответствии с требованиями государственного образовательного стандарта.

1. ВЫПИСКА ИЗ ГОСУДАРСТВЕННОГО ОБРАЗОВАТЕЛЬНОГО СТАНДАРТА ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

Направление подготовки дипломированного специалиста

654600 – Информатика и вычислительная техника

Специальность

220200 (230102) – Автоматизированные системы обработки информации и управления.

Квалификация – *инженер*

Индекс	*Наименование дисциплин и их основные разделы*	Всего часов
СД.07	**Системы искусственного интеллекта**	104
	Искусственный интеллект как научное направление, представление знаний, рассуждений и задач; эпистомологическая полнота представления знаний и эвристические эффективные стратегии поиска решения задач; модели представления знаний: алгоритмические, логические, сетевые и продукционные модели; сценарии; экспертные системы: классификация и структура; инструментальные средства проектирования, разработки и отладки; этапы разработки; примеры реализации.	

2. РАБОЧАЯ ПРОГРАММА

По дисциплине «Системы искусственного интеллекта»

Для специальности 230102 – «Автоматизированные системы обработки информации и управления»

Курс 4 семестр 7

Лекции 30 (час.) Экзамен - семестр

Практические (семинарские) занятия 15 (час.) Зачет 7 семестр

Лабораторные занятия 15 (час.)

Самостоятельная работа 44 (час.)

Всего часов 104 час.

Составитель: доцент Акилова Ирина Михайловна

Факультет математики и информатики

Кафедра Информационных и управляющих систем

2006

1. ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ, ЕЕ МЕСТО В УЧЕБНОМ ПРОЦЕССЕ

1.1. Цели и задачи дисциплины

Цель дисциплины - получение теоретических знаний и практического опыта по использованию методов искусственного интеллекта в решении задач АСОИиУ.

Дисциплина связана с предшествующими ей дисциплинами: "Информационная технология", "Организация вычислительных систем". "Алгоритмические языки и программирование", "Технология программирования", "Организация баз данных".

В результате изучения курса студенты:

- Должны знать основные понятия технологии искусственного интеллекта, основные средства проектирования и разработки программ современными интеллектуальными методами, модели представления и получения знаний. Владеть языками логического и обьектно-ориентированного программирования для решения задач АСОИиУ.
- Должны уметь применять необходимые методы искусственного интеллекта при разработке различных задач АСОИиУ. Уметь представлять знания различными моделями и выбирать наиболее эффективные. Практически использовать ПРОЛОГ, обьектно-ориентированные и алгоритмические языки для разработки интеллектуальных задач.

2. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

2.1. Федеральный компонент

Программа курса «Системы искусственного интеллекта» составлена в соответствие с требованиями государственного образовательного стандарта специализации – Автоматизированные системы обработки информации и управления, специализации 230102, блок специальных дисциплин СД.07.

2.2. Наименование тем, их содержание, объем в лекционных часах

ТЕМАТИЧЕСКИЙ ПЛАН ЛЕКЦИОННЫХ ЗАНЯТИЙ

№ темы	Наименование темы	Кол-во часов
1	История развития искусственного интеллекта	4
2	Представление знаний в интеллектуальных системах	6
3	Стратегии получения знаний	6
4	Методы работы со знаниями	4
5	Методы извлечения знаний	6
6	Методология структурирования знаний	4
Итого		30

1. ИСТОРИЯ РАЗВИТИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА - 4 часа.

Этапы развития искусственного интеллекта. Японский проект компьютеров пятого поколения. Современные направления развития искусственного интеллекта.

2. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ - 4 часа.

Данные и знания. Особенности знаний. Модели представления знаний. Компоненты продукционных систем. Классификация ядер продукции.

3. СТРАТЕГИИ ПОЛУЧЕНИЯ ЗНАНИЙ – 6 часов.

Психологический, лингвистический и гносеологический аспекты извлечения знаний.

4. МЕТОДЫ РАБОТЫ СО ЗНАНИЯМИ - 6 часов.

Основные понятия. Системы приобретения знаний от экспертов. Формализация качественных знаний.

5. МЕТОДЫ ИЗВЛЕЧЕНИЯ ЗНАНИЙ – 6 часов.

Классификация методов извлечения знаний. Коммуникативные методы извлечения знаний. Текстологические методы извлечения знаний.

6. МЕТОДОЛОГИЯ СТРУКТУРИРОВАНИЯ ЗНАНИЙ – 4 часа.

Языки семиотического моделирования. Стадии структурирования. Методы

структурирования. Психосемантика и методы многомерного шкалирования.

2.3. Практические занятия, их содержание и объем в часах.

ТЕМАТИЧЕСКИЙ ПЛАН ПРАКТИЧЕСКИХ ЗАНЯТИЙ

№ темы	Наименование темы	Кол-во часов
1	Представление знаний с помощью логики предикатов. Выводы в естественной системе. Операции со знаниями при помощи принципа резолюции.	6
2	Продукционные модели представления знаний. Основные системы продукций.	2
3	Представление знания о заданной предметной области в виде фрейм-сети. Теория фреймов. Модели представления знаний фреймами.	3
4	Представление знания о заданной предметной области в виде семантических сетей. Процедурное представление семантических сетей.	4
	Всего	15

2.4. Лабораторные занятия, их наименование и объем в часах.

ТЕМАТИЧЕСКИЙ ПЛАН ПРАКТИЧЕСКИХ ЗАНЯТИЙ

№ темы	Наименование темы	Кол-во часов
1	Изучение базовых возможностей программной среды Turbo Prolog.	4
2	Использование рекурсии для логического программирования	2
3	Операции со списками.	2
4	Работа с файлами	3
5	Работа с внутренней и внешней базами данных	4
	Всего	15

2.5. Самостоятельная работа студентов – 44 часа

1. Работа с окнами на Прологе.
2. Графические возможности языка Пролог:

1. Программа строит на экране два равносторонних треугольника и окрашивает треугольники, а также пространство между ними в разные цвета.

2. На экране строите окружность, затем внутри нее строятся три радиуса под углом 120 градусов, составляющие трехлучевую звезду, которая совершает поворот вокруг своего центра против часовой стрелки на 240 градусов.

3. На экране строится окружность, затем внутри нее строится радиус, который совершает один оборот против часовой стрелки.

4. Программа выводит на экран в окно, несколько строк текста, выделяет верхнюю строку прямоугольником другого цвета и позволяет перемещать прямоугольник вверх и вниз с помощью клавиш перемещения курсора вверх-вниз, а также клавиш <Home> и < End> . После выбора нужной строки следует нажать клавишу <Enter>. В ответ на нажатия любых других клавиш выводится сообщение об ошибке.

5. Программа строит и закрашивает два квадрата, расположенные один внутри другого, в разные цвета. Сначала маленький квадрат (размером 100*100 единиц экрана) закрашивается красным цветом, а затем большой квадрат (размером 300*300 единиц экрана) - голубым цветом.

6. Программа выводит на экран в окно несколько строк текста, выделяет верхнюю строку прямоугольником другого цвета. После выбора нужной строки следует нажать клавишу <Enter>. После завершения выбора в первом меню предлагается второе меню. После выбора во втором меню программа сообщает о сделанном выборе. Первое и второе окна для меню различаются цветом, размером, количеством выводимых строк.

7. Программа строит ромбы заданных цветов и размеров в заданных местах экрана с заданным углом наклона к вертикальной оси. Цвет, размер сторон ромба, угол между сторонами, координаты места расположения ромба, угол наклона осей ромба по отношению к вертикали задаются пользователем с пульта дисплея в режиме диалога.

8. Программа строит ромбы заданных цветов и размеров в заданных местах экрана с заданным углом наклона к вертикальной оси. Цвет, размер сторон ромба, угол между сторонами, координаты места расположения ромба, угол наклона осей ромба по отношению к вертикали задаются пользователем с клавиатуры. Для выбора цвета на экран выводится разноцветное меню.

9. Перемещение квадрата с (без) оставлением(я) следа. Программа строит в центре экрана квадрат размером 50*50 единиц экрана, а затем перемещает его на 10 единиц экрана при каждом нажатии на соответствующие клавиши перемещения курсора: вправо, влево, вверх, вниз, в середину экрана - клавиша <Home>. Размеры квадрата изменяются на 10 единиц при каждом нажатии клавиш: > - размеры увеличиваются, < - размеры уменьшаются. При нулевых или отрицательных размерах сторон квадрата вместо квадрата ставится точка. При перемещении квадрата след остается или нет в зависимости от того, какая клавиша нажата: <PgUp> - след не остается (уже построенное изображение стирается), <PgDn> - оставляется след. Шаг перемещения и изменения сторон квадрата можно изменять в пределах 1 - 9, нажимая клавиши 1 - 9, при нажатии клавиши 0 устанавливается шаг 10. Цвет изменяется циклически нажатием клавиши <Esc> . После белого цвета квадрат становится невидимым, т.к. его цвет совпадает с цветом фона. Чтобы сделать квадрат снова видимым, надо еще раз изменить цвет, нажав клавишу <Esc>, и сделать квадрат зеленым. Программа заканчивает работу по нажатию на клавишу <End>.

Замечание. Работу выполнять группой из двух студентов.

10. Построение двумерного цветного графического изображения. В центре экрана появляется направляющая линия, которую можно поворачивать против часовой стрелки на 10 градусов нажатием клавиши перемещения курсора вверх или на 10 градусов по часовой стрелке нажатием клавиши перемещения курсора вниз, а также перемещать направляющую линию вперед нажатием клавиши перемещения курсора вправо и назад

нажатием клавиши перемещения курсора влево, в середину экрана - клавиша <Home>. При перемещении направляющей линии за ней остается след, если была нажата клавиша < PgDn>. След не остается, если была нажата клавиша <PgUp>. Направляющая линия при этом становится зеленой. При движении по построенному ранее изображению оно стирается. При нажатии на клавишу <Enter> направляющая линия изменяет цвет и становится частью вычерчиваемой линии, а направляющая линия перемещается в конец вычерчиваемой линии. Этот режим удобен при построении прямых линий большой длины. Если перо поднято (PgUp) или цвет следа совпадает с цветом фона, то при нажатии на клавишу <Enter> направляющая линия перемещается вперед на расстояние одной своей длины. Размеры направляющей линии могут быть изменены нажатием клавиш: > - размеры увеличиваются на величину шага, < - размеры уменьшаются на величину шага. Шаг перемещения направляющей линии, шаг изменения ее длины, угол ее поворота (в градусах) можно изменять в пределах от 1 до 9; при нажатии клавиши 0 устанавливается шаг 10 (10 градусов для поворота направляющей, 10 единиц экрана для перемещения направляющей и вычерчивания следа). По умолчанию устанавливается шаг = 10. Иногда при повороте или удлинении направляющей линии на экране остаются неудаленные "лишние" точки. Этого не происходит при работе с шагом = 1. Цвет оставляемого следа (вычерчиваемой линии) совпадает с цветом точки на конце направляющей линии. Цвет оставляемого следа и направляющей линии изменяется циклически нажатием на клавишу <Esc>, при этом цвет следа и направляющей линии различаются (цвет направляющей: красный - желтый - красный - зеленый; цвет следа: зеленый - красный - желтый). После белого цвета вычерчиваемая линия становится невидимой, т.к. ее цвет совпадает с цветом фона. Чтобы сделать вычерчиваемую линию снова видимой, надо еще раз изменить цвет (на зеленый), нажав клавишу <Esc>, или опустить перо клавишей <PgDn>. Поднять перо, чтобы при перемещении направляющей линии не оставлять за ней след, можно клавишей

<PgUp>. Цвет фона изменяется циклически нажатием клавиши <Ins>. Для отображения в текущей позиции текста следует нажать клавишу "t" и ввести текст. Текст будет отображен на расстоянии двух направляющих линий от текущей точки. Цвет текста совпадает с цветом оставляемого следа, если след невидим, то текст будет зеленым. Построенное изображение может быть записано в файл, имя которого вводится с клавиатуры, при нажатии на клавишу "s". Изображение может быть считано из файла при нажатии на клавишу "l". При нажатии на клавишу "h" на экран выводится подсказка. Программа заканчивает работу при нажатии на клавишу <End>.

Замечание. Работу выполнять группой из двух студентов.

11. Вращение трехмерного проволочного куба. Параметры задаются. Программа строит на экране трехмерный цветной проволочный куб и выполняет его поворот на 360 градусов вокруг начала координат [0,0,0]. Скорости вращения куба вокруг осей X,Y,Z и скорость изменения размеров куба задаются с клавиатуры. Скорости вращения могут изменяться от 0 до 360 и обязательно должны быть числами, кратными 360. Скорость изменения размеров куба может изменяться в пределах от -10 до +10. При увеличении размеров куба до 300 единиц экрана дальнейшее их увеличение прекращается. При уменьшении размеров куба размеры уменьшаются до нуля, куб превращается в точку, а затем опять начинает расти до максимально допустимых размеров (-300 единиц экрана). В качестве примера попробуйте задать скорость вращения вокруг оси X = 1, вокруг Y =1, вокруг Z=0, скорость изменения размеров куба = -1.

Замечание. Работу выполнять группой из двух студентов.

12. Построение трехмерного цветного графического изображения. В центре экрана появляется направляющая фигура (в виде стилизованного самолета), направление движения которой можно изменять, поворачивая ее вправо - нажатием клавиши перемещения курсора вправо, влево - нажатием клавиши перемещения курсора влево, вверх - нажатием клавиши перемещения курсора вверх, вниз - нажатием клавиши перемещения курсора вниз, а

также поворачивая фигуру вокруг ее продольной оси вправо нажатием клавиши ">" и влево нажатием клавиши "<". Направляющую фигуру можно перемещать: назад нажатием клавиши <F1> и вперед нажатием клавиши <F2>. При перемещении направляющей фигуры за ней остается след, если была нажата клавиша <PgDn>. Цвет линий, параллельной оси направляющей фигуры, при этом совпадает с цветом оставляемого следа. След не остается, если была нажата клавиша <PgUp>. Линия, параллельная оси направляющей фигуры, при этом становится, невидимой. При движении по построенному ранее изображению последнее стирается. Направляющую фигуру можно переместить в центр экрана нажатием клавиши <Home>, при этом фигура поворачивается в исходное положение вдоль оси X. При нажатии на клавишу <F3> осевая линия направляющей фигуры изменяет цвет и становится частью вычерчиваемой линии, а направляющая фигура перемещается в конец вычерчиваемой линии. Этот режим удобен при построении прямых линий большой длины. Размеры направляющей фигуры могут быть изменены нажатием клавиш: F9 - размеры уменьшаются на величину шага, F10 - размеры увеличиваются на величину шага. Шаг перемещения направляющей фигуры за одно нажатие на соответствующую клавишу, шаг изменения размеров направляющей фигуры, угол ее поворота (в градусах) можно изменять в пределах от 1 до 9; при нажатии клавиши 0 устанавливается шаг 10 (10 градусов для поворота направляющей, 10 единиц экрана для перемещения направляющей и вычерчивания следа). В начале работы программы по умолчанию устанавливается шаг = 10. При поворотах и изменении размеров направляющей фигуры могут иногда оставаться "лишние" точки. Это происходит из-за низкой разрешающей способности терминала при работе в цветном графическом режиме (всего 320*200 адресуемых точек экрана), хотя система *Пролог* допускает работу с цветным графическим экраном 1024*768 точек. Чтобы при перемещениях направляющей фигуры "лишних" точек на экране не оставалось, можно, уменьшая размеры направляющей фигуры, превратить ее в точку красного

цвета. Направление движения точки легко можно вычислить, зная шаг изменения угла поворота (например, по умолчанию в начале работы программы устанавливается шаг угла поворота = 10 градусов при каждом нажатии на соответствующие клавиши). Цвет вычерчиваемой линии (оставляемого следа) изменяется циклически нажатием клавиши <F4>: зеленый - красный - белый – цвет фона - зеленый и т.д. После белого цвета вычерчиваемая линия становится невидимой, т.к. ее цвет совпадает с цветом фона. Цвет вычерчиваемой линии (оставляемого следа) совпадает с цветом биссектрисы угла направляющей фигуры. Если вычерчиваемая линия невидима, то биссектриса тоже невидима. Чтобы вычерчиваемую линию снова сделать видимой, надо еще раз изменить цвет (на зеленый), нажав клавишу <F4>, или опустить перо клавишей <PgDn>. Цвет оставляемого следа при этом станет зеленым. Поднять перо, чтобы при перемещении направляющей линии не оставлять за ней след, можно клавишей <PgUp>. Цвет фона изменяется циклически нажатием клавиши <F5>. Построенное изображение может быть записано в файл, имя которого вводится с клавиатуры при нажатии на клавишу "F6". Изображение может быть считано из файла при нажатии на клавишу "F7". Для отображения текста в текущей позиции следует нажать клавишу <F8> и набрать на клавиатуре текст. В конце ввода текста нажать клавишу <Enter>. Текст отобразите на расстоянии одной длины от начала направляемой фигуры. Если перо поднято, то текст будет зеленого цвета. При нажатии на. клавишу "h" на экран выводится подсказка. Программа заканчивает работу при нажатии на клавишу <End>.

Замечание. Работу выполнять группой из двух студентов.

2.6.Вопросы к зачету

I часть:

1. Основные конструкции языка Пролог.
2. Переменные и константы языка Пролог.

3. Стандартные предикаты языка Пролог.
4. Факты и правила на Прологе.
5. Унификация термов.
6. Отсечение.
7. Использование рекурсии.
8. Работа с окнами.
9. Основные предикаты при работе с файлами.
10. Основные предикаты при работе с внутренней базой данных.
11. Основные предикаты при работе с внешней базой данных.
12. Предикаты для работы с цепочками.
13. Алгоритм создания случайных чисел.
14. Как найти в списке подсписок заданного размера и остаток?
15. Как вставить подсписок в список?
16. Алгоритм удаления элементов из списка
17. Алгоритм замены вхождения числа в списке на другое число.
18. Алгоритм сложения двух списков.
19. Инициализация графического режима.
20. Основные графические объекты на Прологе.

II часть:

1. Основные определения в логике предикатов.
2. Формулы первого и второго порядков в логике предикатов.
3. Правила вывода в логике предикатов.
4. Интерпретация формул.
5. Преобразование правильно построенных формул в предложения.
6. Сущность принципа резолюций.
7. Основные понятия продукционных правил.
8. Классификация ядер продукции.
9. Гипотетический силлогизм.
10. Основные понятия семантических сетей.
11. Основные типы объектов в семантических сетях.

12. Классификации семантических сетей.
13. Канонические графы.
14. Теория фреймов.
15. Структура данных фрейма.
16. Свойства фреймов.

III часть:

1. Этапы развития искусственного интеллекта.
2. Японский проект компьютеров пятого поколения.
3. Современные направления развития искусственного интеллекта.
4. Данные и знания.
5. Особенности знаний.
6. Модели представления знаний.
7. Компоненты продукционных систем.
8. Психологический аспект извлечения знаний
9. Лингвистический аспект извлечения знаний
10. Гносеологический аспект извлечения знаний.
11. Основные понятия методов работы со знаниями.
12. Системы приобретения знаний от экспертов.
13. Формализация качественных знаний.
14. Классификация методов извлечения знаний.
15. Коммуникативные методы извлечения знаний.
16. Текстологические методы извлечения знаний.
17. Языки семиотического моделирования.
18. Стадии структурирования.
19. Методы структурирования.
20. Психосемантика и методы многомерного шкалирования.

2.7. Виды контроля

Для проверки эффективности преподавания дисциплины проводится контроль знаний студентов.

При этом используются следующие виды контроля:

- текущий контроль за аудиторной и самостоятельной работой обучаемых осуществляется во время проведения аудиторных занятий посредством устного опроса;
- промежуточный контроль осуществляется два раза в семестр в виде анализа итоговых отчетов;
- итоговый контроль в виде зачета осуществляется после успешного прохождения студентами текущего и промежуточного контроля и сдачи отчета по самостоятельной работе и устного экзамена при ответах экзаменуемого на два вопроса в билете и дополнительные вопросы по желанию экзаменатора.

2.8. Требования к знаниям студентов, предъявляемые на зачете

Для получения зачета студент должен посещать занятия, студентом должны быть выполнены все лабораторные работы, самостоятельное задание и отчет к нему, а также знание теоретического материала в объеме лабораторного курса.

«Зачтено» - студент не имеет задолженностей по семестровым отчетным работам. Хорошо владеет теоретическим материалом.

«Незачтено» - студент не отчитался по семестровым отчетным работам, не знает теоретический материал.

3. Рекомендуемая литература

Основная

1. Девятков В.В. Системы искусственного интеллекта. – М. Издательство МГТУ имени Н.Э.Баумана, 2001 г.
2. А. Тэйс, П.Грибомонт, Ж. Луис и др. Логический подход к искусственному интеллекту. От классической логики к логическому программированию. М.: Мир,1998 г.

3. Лорьер Ж.-Л. Системы искусственного интеллекта: Пер. с фр. М.: Мир, 1991 г.
4. Поспелов Г.С. Искусственный интеллект – прикладные системы, 1985 г.
5. Искусственный интеллект. Справочник в трех томах. 1990 г.
6. Стерлинг Л., Шапиро З. Искусство программирования на языке Пролог. – М. Мир,1990 г.

Дополнительная

1. Осуга С., Обработка знаний. – М., Мир, 1989 г.
2. Уэно Х.,Исидзука М. Представление и использование знаний. – М., Мир 1989 г.
3. Янсон А. Турбо – Пролог в сжатом изложении. – М., Мир, 1995 г.

Методическое обеспечение

1. Акилова И.М. Практикум. Логическое программирование. Благовещенск, 2002 г.
2. Акилова И.М Практикум. Программирование на языке Турбо – Пролог. г. Благовещенск 2002 г.

4. Необходимое техническое и программное обеспечение

Лекции проводятся в стандартной аудитории, оснащенной в соответствии с требованиями преподавания теоретических дисциплин.

Для проведения лабораторных работ необходим компьютерный класс на 12-14 посадочных мест пользователей. В классе должен быть установлен язык программирования Пролог.

5. Учебно-методическая (технологическая) карта дисциплины

Номер недели	Номер темы	изучаемые на лекцииВопросы,	Занятия		и методические пособияИспользуемые наглядные	Самостоятельная работа студентов		Форма контроля
			Практические	Лабораторные		Содержание	Часы	
1	2	3	4	5	6	7	8	9

1	1	1	1	1	1-осн.	Изучение литературы по языку программирования Пролог	16	
2	1	2-3			1-осн.			
3	2	1-2	1	1	1-осн.			злр.
4	2	3			1-осн.			
5	2	4-5	1	2	1-осн.			злр.
6	3	1			1-осн.			
7	3	2	2	3	1-осн.			злр.
8	3	3			1-осн.			
9	4	1-2	3	3-4	1-осн.	Выполнение задания для самостоятельной работы		злр.
10	4	3	3-4		1-осн.			
11	5	1		4	1-осн.		14	злр.
12	5	2	4		1-осн.			
13	5	3		5	1-осн.			
14	6	1-2	4		1-осн.	сб.	4	
15	6	3-4		5	1-осн.			злр.

Условные обозначения:

осн. – основная литература

злр - защита лабораторной работы

сб. - собеседование по результатам самостоятельной работы студентов

3. ГРАФИК САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

Содержание	Объем в часах	Сроки и форма контроля
Изучение графических возможностей логического языка программирования Пролог.	10 час.	Собеседование
Составление графической программы на Прологе	34 час.	Собеседование по отчету

4. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПРОВЕДЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

По выбранной теме студенты выполняют практическую работу.

Практическая работа включает следующие разделы:

1. Обоснование актуальности выбранной тематики и описание целей выполнения работы.

2. Составление и отладка программы.

3. Составление отчета.

5. ПЕРЕЧЕНЬ УЧЕБНИКОВ, УЧЕБНЫХ ПОСОБИЙ

5.1. Перечень основной литературы

1. Девятков В.В. Системы искусственного интеллекта. – М. Издательство МГТУ имени Н.Э.Баумана, 2001 г.
2. А. Тэйс, П.Грибомонт, Ж. Луис и др. Логический подход к искусственному интеллекту. От классической логики к логическому программированию. М.: Мир,1998 г.
3. Лорьер Ж.-Л. Системы искусственного интеллекта: Пер. с фр. М.: Мир, 1991 г.
4. Поспелов Г.С. Искусственный интеллект – прикладные системы, 1985 г.
5. Искусственный интеллект. Справочник в трех томах. 1990 г.
6. Стерлинг Л., Шапиро З. Искусство программирования на языке Пролог. – М. Мир,1990 г.

5.2. Перечень дополнительной литературы

1. Осуга С., Обработка знаний. – М., Мир, 1989 г.
2. Уэно Х.,Исидзука М. Представление и использование знаний. – М., Мир 1989 г.
3. Янсон А. Турбо – Пролог в сжатом изложении. – М., Мир, 1995 г.

5.3. Методическое обеспечение

1. Акилова И.М. Практикум. Логическое программирование. Благовещенск, 2002 г.
2. Акилова И.М Практикум. Программирование на языке Турбо – Пролог. г. Благовещенск 2002 г.

6. КРАТКИЙ КОНСПЕКТ ЛЕКЦИЙ

Лекция № 1 **Тема:** ***История развития искусственного интеллекта.***

Под искусственным интеллектом понимают научное направление, в рамках которого ставятся и решаются задачи аппаратного и программного моделирования тех видов человеческой деятельности, которые традиционно считаются интеллектуальными.

Свойство человека решать любые творческие задачи, обучаться, объяснять решение, представлять в памяти знания об окружающем мире выделяет его из всего живого мира. Являясь универсальной биомашиной по обработке информации он все-таки сталкивается с определенными трудностями по восприятию и анализу больших потоков информации, особенно в последнее время.

Этапы развития искусственного интеллекта

1. Кибернетики того времени пытались построить машины, моделирующие человеческий мозг. Появление первых электронных вычислительных машин, способных выполнять вычисления больших чисел десятки тысяч раз в секунду придавало очень большой оптимизм в исследованиях по искусственному интеллекту.

Розенблатом был создан самоорганизуюшийся автомат - PERCEPTRON, который считался грубой моделью сетчатки глаза. Его можно было научить распознавать образы, хотя их количество ограничивалось. Энтузиазм в отношении разработок подобных систем основывался на кибернетических представлениях Норберта Винера и Уоррена Маккалока об абстрактных нейронных сетях. Считалось, что можно взять сильно связанную систему модельных нейронов, которой раньше ничего не было известно и обучить ее тому, что задумал создатель методом поощрений и наказаний. При этом тот факт, что мозг человека содержит 10 в 10 степени нейронов тактично обходился.

Причиной появления представлений об абстрактных нейронных сетях являлись достижения в психологии и нейрофизиологии того времени. Ученые уже имели представления о строении нейронов и их взаимосвязи. Развитие

электроники привело к попыткам реализовать схемное представление нейрона и нейронных сетей.

Но особого развития этого направления в искусственном интеллекте в те времена не произошло, все таки и техника была еще примитивна, и исследования переключились на исследование других методов имитации разумного поведения при решении задач. К тому же Минский и Пэйперт провели теоретическую работу и показали что возможности перцепторна ограниченны, а экспериментальные результаты оказались недостаточно хорошими.

Исследования нейронных сетей практически прекратилось, но не совсем. В настоящее время теория нейронных сетей получила новое развитие.

2. На новые рубежи в исследовании искусственного интеллекта указали Аллен Ньюэлл и Герберт Саймон из университета Карнеги-Меллона (США). Центральным для их подхода явилось представление об эвристическом поиске. Они считали что мышление человека основано на сочетании простых задач манипулирования символами - сравнение, поиск, модификация и т.п., а следовательно эти задачи может выполнить и компьютер.

Они достигли реальных практических результатов т.к. уровень развития компьютеров того времени позволял это сделать. В конце 50-х годов появились первые эвристические программы: ЛОГИК - ТЕОРЕТИК, предназначенный для доказательства теорем в исчислении высказываний и ОБЩИЙ РЕШАТЕЛЬ ЗАДАЧ. То что их программы действительно моделируют человеческое мышление они доказывали, сравнивая рассуждения при доказательстве теоремы думающего вслух человека. В начале 70-х они даже предложили общую методику составления программ, моделирующих мышление и создали систему GPS. Эта система была универсальной в том отношении, что "не было конкретного указания, к какой предметной области относится данная задача". Но эта универсальность относилась только к ограниченной области математических головоломок: Ханойская башня, проблема миссионеров и

туземцев и др., которые с точки зрения людей проблемами не являются. Основой системы являлся "поиск в глубину" и процесс постепенного разбиения задачи на подзадачи, пока не будет получена легко решаемая подзадача.

3. Большое значение на дальнейшее развитие ИИ оказало появление метода резолюций предложенный Робинсоном, который является исчерпывающим методом доказательства в логике предикатов. Именно в это время появляется язык программирования нового типа - PROLOG, (программирование логики), который является одним из универсальных языков программирования задач искусственного интеллекта.

Постепенно интерес к поиску эффективных и универсальных эвристик переместился в сторону разработок интеллектуальных задач с узкой предметной областью. В 70-х годах начали появляться первые экспертные системы.

Экспертные системы стали первым реальным практическим применением искусственного интеллекта при решении интеллектуальных задач, решение которых с позиций алгоритмического программирования не имеет смысла. Они ознаменовали собой третий этап развития искусственного интеллекта.

Первые системы: MYSIN и DENDRAL - давали результаты сравнимые с экспертами среднего класса. MYSIN была родоначальником целой серии медико-диагностических машин, используемых в клинической практике. Предназначенная для диагностики инфекционных заболеваний, она приводила правдоподобные заключения даже если не все исходные данные были верны. Имитируя рассуждения эксперта, она оперировала с "неточными" знаниями и могла объяснить процесс своих "рассуждений". Она выполняла работу, которой люди обучаются годами, и хотя обладала ограниченными возможностями по сравнению даже с плохим врачом по объему разносторонних знаний, работа ее оставляла большое впечатление. По сути дела она стала эталоном для создания большинства экспертных систем.

Созданная позже система PROSPECTOR, работающая в области геологии, открыла ранее неизвестные запасы молибдена. Система PUFF – созданная по подобию MYSIN и диагностирующая дыхательные заболевания, до сих пор используется в Тихоокеанском центре вблизи Сан-Франциско.

Основное внимание в ИИ переместилось с моделирования человеческого мышления на разработку машинно-ориентированных методов решения задач.

Целью ИИ стала разработка программ, способных решать "человеческие задачи", с которыми до недавнего времени мог справиться только человек.Экспертные системы явились причиной необузданного оптимизма в практических разработках интеллектуальных программ 70-х годов. Но основная трудность с которой начали сталкиваться при разработке экспертных систем - это получение знаний. Именно обширность и качество базы знаний и определяют успех экспертной системы. Получение знаний от экспертов оказалось очень не легкой задачей. Появилась даже отдельная профессия - инженер по знаниям (аналитик знаний) который формализует суждения человека-эксперта в один из методов представления знаний искусственного интеллекта.

Новое направление в развитии искусственного интеллекта стали связывать с методами получения знаний самой интеллектуальной системой. Дуг Ленат из Стэндфордского университета создал машинную обучающуюся систему EURISKO, которая автоматически улучшает и расширяет свой запас эвристических правил. Эта система выигрывала три года подряд в учебной военной игре (несмотря на то что правила игры каждый раз менялись, чтобы помешать ей это сделать), она произвела переворот в области создания СБИС (сверхбольших интегральных схем), изобретя трехмерный узел типа И/ИЛИ. Появление программ автоматической индукции подобным EURISKO, является важнейшим этапом в развитии искусственного интеллекта в последние десять лет. До сих пор перенесение умения специалиста-человека в машинную программу есть утомительная и долгая процедура. Таким образом искусственный интеллект 80-х перенес внимание в область проблем машинного

обучения.

Искусственный интеллект, двигаясь в своем развитии по спирали замкнул круг и поставил перед исследователями задачу, которую пытались решить первые исследователи искусственного интеллекта - машинное обучение и накопление знаний, решение которых в настоящее время опирается на более мощную технику и на более широкие потребности общества нежели это было раньше.

Говоря об истории искусственного интеллекта, нельзя не вспомнить и то, какой вклад внесла Япония в его развитие. Благодаря ей исследования в этой области получили большую поддержку во многих ведущих странах мира. Речь идет об японском проекте компьютеров 5-го поколения, о котором Япония объявила в 1978 году.

Министерство внешней торговли и промышленности Японии поручило электротехнической лаборатории выработать проект вычислительных систем 90-х годов. Предполагалось, что проект не должен конкурировать с направлением, выбранным фирмой IBM. Формально к проекту приступили в 1982 году. Проект машин 5-го поколения охватывает весьма широкую область. Его задачей является создание и программного обеспечения и аппаратная реализация архитектуры.

В проекте было выделено 7 направлений исследований:

Вычислительная система 5-го будет ориентированна на обработку знаний, обладать возможностями логического вывода и развитым интеллектуальным человеко-машинным интерфейсом.

На первом этапе создается персональная последовательная машина вывода. В дальнейшем появляется высокопроизводительная параллельная машина. Быстродействие будет измеряться числом логических выводов в секунду.

Намечаемая производительность: 1 ГЛВС

1 ЛВС = 100 - 100000 миллионов машинных операций в секунду.

Был создан план работ: начальный промежуточный и заключительный.

1. Создание высокопроизводительной персональной ПРОЛОГ - машины
2. Создание машины реляционной базы данных
3. Проведение фундаментальных исследований.

В 1985 году должны появится первые результаты.

В настоящее время проект не реализован, хотя его завершение должно быть в начале 90-х годах. Но его главное значение в том, что интерес к искусственному интеллекту возрос необычайно сильно. Почти во всех ведущих странах мира были созданы исследовательские группы и выделены большие средства для научных исследований.

Наиболее очевидный факт, который проявился благодаря игре в машины пятого поколения, состоит в том, что для игры не хватает людей. В этой области до сих пор ощущается острый недостаток работников и на них очень большой спрос.

Поэтому в настоящее время возникают задачи подготовки следующего поколения исследователей, разработка специального курса и привлечений финансирующих результатов.

В настоящее время исследования искусственного интеллекта ведутся в двух направлениях: программно - прогматическое и бионическое.

Первое занимается созданием программ на компьютерах Фон-Нейманского типа, с помощью которых можно решать те задачи, решение которых до этого считалось исключительно прерогативой человека.

Программно – прогматическое направление

1. Информация о мышлении и языке:
2. Интеллектуальные программы:
3. Распознающие и узнающие программы
4. Прочие программы
5. Модели поведения
6. Программы доказательства теорем
7. Эвристические программы
8. Работа со знаниями:

9. Интеллектуальное программирование

10. Автоматический синтез программ

11. Инструментальные системы

12. Интеллектуальные программные системы

13. Интеллектуальные информационные системы

14. Интеллектуальные системы проектирования и научных исследований

15. Обучающие системы

Бионическое направление

- информация о морфологической (нейрофизиологической) структуре;
- нейробионической подход;
- информация о целостных структурах организма;
- структурно-эвристический;
- информация о функциональных механизмах организма;
- гомеостатический.

Интересуется проблемами искусственного воспроизведения тех структур и процессов, которые характерны для живого человеческого мозга и которые лежат в основе процесса решения задач человеком. Это направление имеет четко выраженный фундаментальный характер, и его интенсивное развитие невозможно без одновременного глубокого изучения мозга нейрофизиологическими, морфологическими и психологическими методами.

Буквально за последние несколько лет интерес к ИИ возрос по ряду объективных причин. Самой важной из них является информационная революция.

Развитие средств вычислительной техники за последнее пятилетие создало предпосылки для реализации многих исследовательских разработок, которые не имели большого практического применения из-за недостаточно развитой вычислительной техники.

В ведущих странах мира произошел значительный перенос информации в память компьютеров и ее обработка. Общество уже требует от ЭВМ не только данных для решения задач, но и того как их решать.

Поэтому применение методов ИИ, наделение программ интеллектуальными функциями поведения является закономерным этапом развития информационного общества. Решение проблемы создает новые проблемы и так далее.

В настоящее время областью массового применения средств ИИ стали системы:

- Обработки ЕЯ грамматик
- Распознавание речи и изображения
- Системы машинного перевода
- Экспертные системы.

Последние являются целым отдельным направлением в рамках исследований по ИИ. Именно в них получено очень большое число практических разработок.

Исследования в ЭС еще называют инженерией знаний. В задачу этого направления входят исследование и разработка программ, использующих знания и процедуры вывода для решения задач, являющихся трудными для людей - экспертов.

Можно дать определение ЭС: это программные комплекса, накапливающие опыт специалистов в некоторой предметной области в целях тиражирования для консультации менее подготовленных пользователей.

Огромный интерес к ЭС вызван тремя причинами:

1. Они ориентированны на решение широкого круга слабо формализованных задач, которые считались недоступными для компьютеров.
2. Специалисты, не знающие программирования могут самостоятельно разрабатывать ЭС при помощи оболочек ЭС.
3. ЭС при решении практических задач достигают результатов, не уступающих, а иногда и превосходящих возможности людей-экспертов.

Лекции № 2,3 **Тема: Представление знаний в интеллектуальных системах.**

В настоящее время в исследованиях по искусственному интеллекту (ИИ) выделились шесть направлений:

1. Представление знаний.
2. Манипулирование знаниями.
3. Общение.
4. Восприятие.
5. Обучение.
6. Поведение.

В рамках направления "Представление знаний" решаются задачи, связанные с формализацией и представлением знаний в памяти интеллектуальной системы (ИС). Для этого разрабатываются специальные модели представления знаний и языки для описания знаний, выделяются различные типы знаний. Изучаются источники, из которых ИС может черпать знания, и создаются процедуры и приемы, с помощью которых возможно приобретение знаний для ИС. Проблема представления знаний для ИС чрезвычайно актуальна, т.к. ИС - это система, функционирование которой опирается на знания о проблемной области, которые хранятся в ее памяти.

Данные и знания. Основные определения.

Информация, с которой имеют дело ЭВМ, разделяется на *процедурную* и *декларативную*. Процедурная информация овеществлена в *программах*, которые выполняются в процессе решения задач, декларативная информация - в *данных*, с которыми эти программы работают. Стандартной формой представления информации в ЭВМ является *машинное слово*, состоящее из определенного для данного типа ЭВМ числа двоичных разрядов - *битов*. Машинное слово для представления данных и машинное слово для представления команд, образующих программу, могут иметь одинаковое или разное число разрядов. В последнее время для представления данных и команд используются одинаковые по числу разрядов машинные слова.

Содержимое памяти образует *информационную базу*. По мере развития исследований в области ИС возникла концепция знаний, которые объединили в себе многие черты процедурной и декларативной информации. *База знаний* - необходимая составляющая программного комплекса ИИ. Машины,

реализующие алгоритмы ИИ, называются *машинами, основанными на знаниях*, а подраздел теории ИИ, связанный с построением экспертных систем, - *инженерией знаний.*

Особенности знаний:

21. *Внутренняя интерпретируемость*. Каждая информационная единица должна иметь уникальное имя, по которому ИС находит ее, а также отвечает на запросы, в которых это имя упомянуто. Когда данные, хранящиеся в памяти, были лишены имен, то отсутствовала возможность их идентификации системой. Данные могла идентифицировать лишь программа, извлекающая их из памяти по указанию программиста, написавшего программу. Что скрывается за тем или иным двоичным кодом машинного слова, системе было неизвестно.

Таблица 1

Фамилия	Год рождения	Специальность	Стаж, число лет
Попов	1965	Слесарь	5
Сидоров	1946	Токарь	20
Иванов	1925	Токарь	30
Петров	1937	Сантехник	25

Если, например, в память ЭВМ нужно было записать сведения о сотрудниках учреждения, представленные в табл. 1, то без внутренней интерпретации в память ЭВМ была бы занесена совокупность из четырех машинных слов, соответствующих строкам этой таблицы. При этом информация о том, какими группами двоичных разрядов в этих машинных словах закодированы сведения о специалистах, у системы отсутствуют. Они известны лишь программисту, который использует данные табл. 1 для решения возникающих у него задач. Система не в состоянии ответить на вопросы типа "Что тебе известно о Петрове?" или "Есть ли среди специалистов сантехник?".

При переходе к знаниям в память ЭВМ вводится информация о некоторой *протоструктуре информационных единиц*. В рассматриваемом примере она представляет собой специальное машинное слово, в котором

указано, в каких разрядах хранятся сведения о фамилиях, годах рождения, специальностях и стажах. При этом должны быть заданы специальные словари, в которых перечислены имеющиеся в памяти системы фамилии, года рождения, специальности и продолжительности стажа. Все эти *атрибуты* могут играть роль имен для тех машинных слов, которые соответствуют строкам таблицы. По ним можно осуществлять поиск нужной информации. Каждая строка таблицы будет экземпляром протоструктуры. В настоящее время СУБД обеспечивают реализацию внутренней интерпретируемости всех информационных единиц, хранящихся в базе данных.

22. *Структурированность*. Информационные единицы должны обладать гибкой структурой. Для них должен выполняться "принцип матрешки", т.е. рекурсивная вложимость одних информационных единиц в другие. Каждая информационная единица может быть включена в состав любой другой, и из каждой информационной единицы можно выделить некоторые составляющие ее информационные единицы. Другими словами, должна существовать возможность произвольного установления между отдельными информационными единицами отношений типа "часть - целое", "род - вид" или "элемент - класс".
23. *Связность*. В информационной базе между информационными единицами должна быть предусмотрена возможность установления связей различного типа. Прежде всего эти связи могут характеризовать отношения между информационными единицами. Семантика отношений может носить декларативный или процедурный характер. Например, две или более информационные единицы могут быть связаны отношением "одновременно", две информационные единицы - отношением "причина - следствие" или отношением "быть рядом". Приведенные отношения характеризуют декларативные знания. Если между двумя информационными единицами установлено отношение "аргумент - функция", то оно характеризует процедурное знание, связанное с вычислением определенных функций. Далее будем различать

отношения структуризации, функциональные отношения, каузальные отношения и *семантические отношения*. С помощью первых задаются иерархии информационных единиц, вторые несут процедурную информацию, позволяющую находить (вычислять) одни информационные единицы через другие, третьи задают причинно - следственные связи, четвертые соответствуют всем остальным отношениям.

Между информационными единицами могут устанавливаться и иные связи, например, определяющие порядок выбора информационных единиц из памяти или указывающие на то, что две информационные единицы несовместимы друг с другом в одном описании.

Перечисленные три особенности знаний позволяют ввести общую модель представления знаний, которую можно назвать *семантической сетью*, представляющей собой иерархическую сеть, в вершинах которой находятся информационные единицы. Эти единицы снабжены индивидуальными именами. Дуги семантической сети соответствуют различным связям между информационными единицами. При этом иерархические связи определяются отношениями структуризации, а неиерархические связи - отношениями иных типов.

24. *Семантическая метрика*. На множестве информационных единиц в некоторых случаях полезно задавать отношение, характеризующее ситуационную близость информационных единиц, т.е. силу ассоциативной связи между информационными единицами. Его можно было бы назвать *отношением релевантности* для информационных единиц. Такое отношение дает возможность выделять в информационной базе некоторые типовые ситуации (например, "покупка", "регулирование движения на перекрестке"). Отношение релевантности при работе с информационными единицами позволяет находить знания, близкие к уже найденным.

25. *Активность*. С момента появления ЭВМ и разделения используемых в ней информационных единиц на данные и команды создалась ситуация, при которой данные пассивны, а команды активны. Все процессы, протекающие в ЭВМ, инициируются командами, а данные используются этими командами лишь в случае необходимости. Для ИС эта ситуация не приемлема. Как и у человека, в ИС актуализации тех или иных действий способствуют знания, имеющиеся в системе. Таким образом, выполнение программ в ИС должно инициироваться текущим состоянием информационной базы. Появление в базе фактов или описаний событий, установление связей может стать источником активности системы.

Перечисленные пять особенностей информационных единиц определяют ту грань, за которой данные превращаются в знания, а базы данных перерастают в *базы знаний* (БЗ). Совокупность средств, обеспечивающих работу с знаниями, образует *систему управления базой знаний* (СУБЗ). В настоящее время не существует баз знаний, в которых в полной мере были бы реализованы внутренняя интерпретируемость, структуризация, связность, введена семантическая мера и обеспечена активность знаний.

Модели представления знаний. Неформальные (семантические) модели.

Существуют два типа методов представления знаний (ПЗ):

1. Формальные модели ПЗ;
2. Неформальные (семантические, реляционные) модели ПЗ.

Очевидно, все методы представления знаний, которые рассмотрены выше, включая продукции (это система правил, на которых основана продукционная модель представления знаний), относятся к неформальным моделям. В отличие от формальных моделей, в основе которых лежит строгая математическая теория, неформальные модели такой теории не придерживаются. Каждая неформальная модель годится только для конкретной предметной области и поэтому не обладает универсальностью, которая присуща моделям формальным. Логический вывод - основная операция в СИИ - в формальных системах строг и корректен, поскольку подчинен жестким

аксиоматическим правилам. Вывод в неформальных системах во многом определяется самим исследователем, который и отвечает за его корректность.

Каждому из методов ПЗ соответствует свой способ описания знаний.

1. *Логические модели.* В основе моделей такого типа лежит *формальная система*, задаваемая четверкой вида: $M = <T, P, A, B>$. Множество T есть *множество базовых элементов* различной природы, например слов из некоторого ограниченного словаря, деталей детского конструктора, входящих в состав некоторого набора и т.п. Важно, что для множества T существует некоторый способ определения принадлежности или не принадлежности произвольного элемента к этому множеству. Процедура такой проверки может быть любой, но за конечное число шагов она должна давать положительный или отрицательный ответ на вопрос, является ли x элементом множества T. Обозначим эту процедуру П(T). Множество P есть множество *синтаксических правил*. С их помощью из элементов T образуют *синтаксически правильные совокупности*. Например, из слов ограниченного словаря строятся синтаксически правильные фразы, из деталей детского конструктора с помощью гаек и болтов собираются новые конструкции. Декларируется существование процедуры П(P), с помощью которой за конечное число шагов можно получить ответ на вопрос, является ли совокупность X синтаксически правильной. В множестве синтаксически правильных совокупностей выделяется некоторое подмножество A. Элементы A называются *аксиомами*. Как и для других составляющих формальной системы, должна существовать процедура П(A), с помощью которой для любой синтаксически правильной совокупности можно получить ответ на вопрос о принадлежности ее к множеству A. Множество B есть множество *правил вывода*. Применяя их к элементам A, можно получать новые синтаксически правильные совокупности, к которым снова можно применять правила из B. Так формируется *множество выводимых* в данной формальной системе *совокупностей*. Если имеется процедура

П(*B*), с помощью которой можно определить для любой синтаксически правильной совокупности, является ли она выводимой, то соответствующая формальная система называется *разрешимой*. Это показывает, что именно правило вывода является наиболее сложной составляющей формальной системы. Для знаний, входящих в базу знаний, можно считать, что множество *A* образуют все информационные единицы, которые введены в базу знаний извне, а с помощью правил вывода из них выводятся новые *производные знания*. Другими словами формальная система представляет собой генератор порождения новых знаний, образующих множество *выводимых* в данной системе *знаний*. Это свойство логических моделей делает их притягательными для использования в базах знаний. Оно позволяет хранить в базе лишь те знания, которые образуют множество *A*, а все остальные знания получать из них по правилам вывода.

2. *Сетевые модели*. В основе моделей этого типа лежит конструкция, названная ранее семантической сетью. Сетевые модели формально можно задать в виде $H = <I, C_1, C_2, ..., C_n, \Gamma>$. Здесь *I* есть множество информационных единиц; $C_1, C_2, ..., C_n$ - множество типов связей между информационными единицами. Отображение Γ задает между информационными единицами, входящими в *I*, связи из заданного набора типов связей. В зависимости от типов связей, используемых в модели, различают *классифицирующие сети, функциональные сети* и *сценарии*. В классифицирующих сетях используются отношения структуризации. Такие сети позволяют в базах знаний вводить разные иерархические отношения между информационными единицами. Функциональные сети характеризуются наличием функциональных отношений. Их часто называют *вычислительными моделями*, т.к. они позволяют описывать процедуры "вычислений" одних информационных единиц через другие. В сценариях используются каузальные отношения, а также отношения типов "средство - результат", "орудие - действие" и т.п. Если в сетевой модели

допускаются связи различного типа, то ее обычно называют семантической сетью.

3. *Продукционные модели*. В моделях этого типа используются некоторые элементы логических и сетевых моделей. Из логических моделей заимствована идея правил вывода, которые здесь называются *продукциями*, а из сетевых моделей - описание знаний в виде семантической сети. В результате применения правил вывода к фрагментам сетевого описания происходит трансформация семантической сети за счет смены ее фрагментов, наращивания сети и исключения из нее ненужных фрагментов. Таким образом, в продукционных моделях процедурная информация явно выделена и описывается иными средствами, чем декларативная информация. Вместо логического вывода, характерного для логических моделей, в продукционных моделях появляется *вывод на знаниях*.

4. *Фреймовые модели*. В отличие от моделей других типов во фреймовых моделях фиксируется жесткая структура информационных единиц, которая называется *протофреймом*. В общем виде она выглядит следующим образом:

(Имя фрейма:

Имя слота 1(значение слота 1)

Имя слота 2(значение слота 2)

. . . . . .

Имя слота К (значение слота К)).

Значением *слота* может быть практически что угодно (числа или математические соотношения, тексты на естественном языке или программы, правила вывода или ссылки на другие слоты данного фрейма или других фреймов). В качестве значения слота может выступать набор слотов более низкого уровня, что позволяет во фреймовых представлениях реализовать "принцип матрешки".

При конкретизации фрейма ему и слотам присваиваются конкретные имена и происходит заполнение слотов. Таким образом, из протофреймов

получаются *фреймы - экземпляры.* Переход от исходного протофрейма к фрейму - экземпляру может быть многошаговым, за счет постепенного уточнения значений слотов. Связи между фреймами задаются значениями специального слота с именем "Связь". Часть специалистов по ИС считает, что нет необходимости специально выделять фреймовые модели в представлении знаний, т.к. в них объединены все основные особенности моделей остальных типов.

Формальные модели представления знаний.

Система ИИ в определенном смысле моделирует интеллектуальную деятельность человека и, в частности, - логику его рассуждений. В грубо упрощенной форме наши логические построения при этом сводятся к следующей схеме: из одной или нескольких посылок (которые считаются истинными) следует сделать "логически верное" заключение (вывод, следствие). Очевидно, для этого необходимо, чтобы и посылки, и заключение были представлены на понятном языке, адекватно отражающем предметную область, в которой проводится вывод. В обычной жизни это наш естественный язык общения, в математике, например, это язык определенных формул и т.п. Наличие же языка предполагает, во - первых, наличие алфавита (словаря), отображающего в символьной форме весь набор базовых понятий (элементов), с которыми придется иметь дело и, во - вторых, набор синтаксических правил, на основе которых, пользуясь алфавитом, можно построить определенные выражения.

Логические выражения, построенные в данном языке, могут быть истинными или ложными. Некоторые из этих выражений, являющиеся всегда истинными. Объявляются *аксиомами* (или *постулатами*). Они составляют ту базовую систему посылок, исходя из которой и пользуясь определенными правилами вывода, можно получить заключения в виде новых выражений, также являющихся истинными.

Если перечисленные условия выполняются, то говорят, что система удовлетворяет требованиям *формальной теории.* Ее так и называют

формальной системой (ФС). Система, построенная на основе формальной теории, называется также *аксиоматической системой.* Формальная теория должна, таким образом, удовлетворять следующему определению:

всякая формальная теория $F = (A, V, W, R)$, определяющая некоторую аксиоматическую систему, характеризуется:

наличием алфавита (словаря), A,

множеством синтаксических правил, V,

множеством аксиом, лежащих в основе теории, W,

множеством правил вывода, R.

Исчисление высказываний (ИВ) и исчисление предикатов (ИП) являются классическими примерами аксиоматических систем. Эти ФС хорошо исследованы и имеют прекрасно разработанные модели логического вывода - главной метапроцедуры в интеллектуальных системах. Поэтому все, что может и гарантирует каждая из этих систем, гарантируется и для прикладных ФС как моделей конкретных предметных областей. В частности, это гарантии непротиворечивости вывода, алгоритмической разрешимости (для исчисления высказываний) и полуразрешимости (для исчислений предикатов первого порядка).

ФС имеют и недостатки, которые заставляют искать иные формы представления. Главный недостаток - это "закрытость" ФС, их негибкость. Модификация и расширение здесь всегда связаны с перестройкой всей ФС, что для практических систем сложно и трудоемко. В них очень сложно учитывать происходящие изменения. Поэтому ФС как модели представления знаний используются в тех предметных областях, которые хорошо локализуются и мало зависят от внешних факторов.

1. *Продукционные системы*

Продукции наряду с фреймами являются наиболее популярными средствами представления знаний в ИИ. Продукции, с одной стороны, близки к логическим моделям, что позволяет организовывать на них эффективные процедуры вывода, а с другой стороны, более наглядно отражают знания, чем

классические логические модели. В них отсутствуют жесткие ограничения, характерные для логических исчислений, что дает возможность изменять интерпретацию элементов продукции.

Компоненты продукционных систем

В общем виде под продукцией понимается выражение следующего вида: (i); Q;P;A⇒B;N. Здесь i - имя продукции, с помощью которого данная продукция выделяется из всего множества продукций. В качестве имени может выступать некоторая лексема, отражающая суть данной продукции (например, "покупка книги "), или порядковый номер продукций в их множестве, хранящимся в памяти системы.

Элемент Q характеризует сферу применения продукции. Такие сферы легко выделяются в когнитивных структурах человека. Наши знания как бы "разложены по полочкам". На одной полочке хранятся знания о том, как надо готовить пищу, на другой как добраться до работы, и т.п. Разделение знаний на отдельные сферы позволяет экономить время на поиск нужных знаний. Такое же разделение на сферы в базе знаний ИИ целесообразно и при использовании для представления знаний продукционных моделей.

Основным элементом продукции является ее ядро: A⇒ B. Интерпретация ядра продукции может быть различной и зависит от того, что стоит слева и справа от знака секвенции ⇒. Обычное прочтение ядра продукции выглядит так: ЕСЛИ A, ТО B, более сложные конструкции ядра допускают в правой части альтернативный выбор, например, ЕСЛИ A, ТО B1, ИНАЧЕ B2. Секвенция может истолковываться в обычном логическом смысле как знак логического следования B из истинного A (если A не является истинным выражением, то о B ничего сказать нельзя). Возможны и другие интерпретации ядра продукции, например A описывает некоторое условие, необходимое для того, чтобы можно было совершить действие B.

Элемент P есть условие применимости ядра продукции. Обычно P представляет собой логическое выражение (как правило предикат). Когда P

принимает значение "истина", ядро продукции активизируется. Если P "ложно", то ядро продукции не может быть использовано.

Элемент N описывает постусловия продукции. Они актуализируются только в том случае, если ядро продукции реализовалось. Постусловия продукции описывают действия и процедуры, которые необходимо выполнить после реализации B. Выполнение N может происходить сразу после реализации ядра продукции.

Если в памяти системы хранится некоторый набор продукций, то они образуют систему продукций. В системе продукций должны быть заданы специальные процедуры управления продукциями, с помощью которых происходит актуализация продукций и выбор для выполнения той или иной продукции из числа актуализированных. В ряде систем ИИ используют комбинации сетевых и продукционных моделей представления знаний. В таких моделях декларативные знания описываются в сетевом компоненте модели, а процедурные знания - в продукционном. В этом случае говорят о работе продукционной системы над семантической сетью.

Классификация ядер продукции.

Ядра продукции можно классифицировать по различным основаниям. Прежде всего все ядра делятся на два больших типа: детерминированные и недетерминированные. В детерминированных ядрах при актуализации ядра и при выполнимости A правая часть ядра выполняется обязательно; в недетерминированных ядрах B может выполняться и не выполняться. Таким образом, секвенция $\Rightarrow$ в детерминированных ядрах реализуется с необходимостью, а в недетерминированных - с возможностью. Интерпретация ядра в этом случае может, например, выглядеть так: ЕСЛИ A, ТО ВОЗМОЖНО B.

Возможность может определяться некоторыми оценками реализации ядра. Например, если задана вероятность выполнения B при актуализации A, то продукция может быть такой: ЕСЛИ A, ТО С ВЕРОЯТНОСТЬЮ P РЕАЛИЗОВАТЬ B. Оценка реализации ядра может быть лингвистической,

связанной с понятием терм - множества лингвистической переменной, например: ЕСЛИ А, ТО С БОЛЬШЕЙ ДОЛЕЙ УВЕРЕННОСТИ В. Возможны иные способы реализации ядра.

Детерминированные продукции могут быть однозначными и альтернативными. Во втором случае в правой части ядра указываются альтернативные возможности выбора, которые оцениваются специальными весами выбора. В качестве таких весов могут использоваться вероятностные оценки, лингвистические оценки, экспертные оценки и т.п.

Особым типом являются прогнозирующие продукции, в которых описываются последствия, ожидаемые при актуализации А, например: ЕСЛИ А, ТО С ВЕРОЯТНОСТЬЮ Р МОЖНО ОЖИДАТЬ В.

Представление простых фактов

Представление - это действие, делающее некоторое понятие воспринимаемым посредством фигуры, записи, языка или формализма. Теория знаний изучает связи между субъектом (изучающим) и объектом. Знание (в объективном смысле) - то, что известно (то, что знаем после изучения).

Представление знаний – формализация истинных убеждений посредством фигур, записей или языков. Нас особенно интересуют формализации, воспринимаемые (распознаваемые) ЭВМ. Возникает вопрос о представлении знаний в памяти ЭВМ, т.е. о создании языков и формализмов представления знаний. Они преобразуют наглядное представление (созданное посредством речи, изображением, естественным языком, вроде английского или немецкого, формальным языком, вроде алгебры или логики, рассуждениями и т.д.) в пригодное для ввода и обработки в ЭВМ. Результат формализации должен быть множеством инструкций, составляющих часть языка программирования.

Представлению знаний присущ пассивный аспект: книга, таблица, заполненная информацией память. В ИИ подчеркивается активный аспект представления: знать должно стать активной операцией, позволяющей не только запоминать, но и извлекать воспринятые (приобретенные, усвоенные)

знания для рассуждений на их основе. Следовательно, истоки представления знаний - в науке о познании, а его конечная цель - программные средства информатики. Во многих случаях подлежащие представлению знания относятся к довольно ограниченной области, например:

- описание состояния человека
- описание ситуации в игре (например, расположение фигур в шахматах)
- описание размещения персонала предприятия
- описание пейзажа

Для характеристики некой области говорят об "области рассуждений" или "области экспертизы". Численная формализация таких описаний в общем мало эффективна. Напротив, использование символического языка, такого, как язык математической логики, позволяет формулировать описания в форме, одновременно близкой и к обычному языку, и к языку программирования. Впрочем, математическая логика позволяет рассуждать, базируясь на приобретенных знаниях: логические выводы действительно являются активными операциями получения новых знаний из усвоенных.

В силу всех этих причин математическая логика лежит в основе различных представлений в ИИ. Данный раздел посвящен представлению простых фактов с помощью логики предикатов. Логическое представление служит также отправной точкой для других представлений (таких как "сетевые" и "объективные"), используемых в ИИ.

Синтаксис логики предикатов.

Язык логики предикатов задается синтакисом. Для представления знаний базисные синтаксические категории языка изображаются такими символами, которые несут достаточно четкую информацию и дают довольно ясную картину об области рассуждений (экспертизы).

Алфавит языка предикатов первого порядка состоит из:

1) разделители: запятая, открывающая и закрывающая скобки;
2) константы, обозначаемые строчными буквами или соединением таких букв, например: a, st, друг;

3) переменные, обозначаемые прописными буквами, например: X, ST, АДРЕС;
4) предикаты, обозначаемые прописными буквами, например P, Q, БОЛЬШЕ;
5) функции, устанавливающие зависимость и отображающие значения одной предметной области в значения другой (или той же), n – местные функции могут служить аргументами предиката. Функции будем обозначать строчными буквами: f,g,t;
6) логические функции:
   - ¬ (отрицание или дополнение). Высказывание "¬A" читается "не A". Оно истинно (И), если высказывание A – ложно (Л);
   - ∧ (конъюнкция). Высказывание "A∧B" читается "A и B". Оно истинно в том случае, когда истинно как A, так и B;
   - ∨ (дизъюнкция). Высказывание "A∨B" читается "A или B". Оно истинно, если истинно хотя бы одно из высказываний;
   - → (импликация). Высказывание "A→B" читается "если A, то B". Оно ложно в том и только в том случае, если A истинно, а B ложно;
   - ↔ (эквивалентность). Высказывание "A↔B" читается "A тогда и только тогда, когда B". Оно истинно в тогда и только тогда, когда A и B имеют одно и тоже истинностное значение;
   - ∃ (квантор существования). Высказывание "∃A" читается "существует A";
   - ∀ (квантор общности). Высказывание "∀A" читается "для любого A".

Пропозициональной формой, или *формулой алгебры логики*, называют всякое высказывание составленное из некоторых исходных высказываний посредством логических операций, т. е. если F и G - пропозициональные формы, то ¬F, (F∨G), (F∧G), (F→G), (F↔G) – пропозициональные формы.

Определение формулы – основного объекта логики предикатов включает понятие "терм".

Терм – выражение, включающее константы, переменные или n – местные функции $f(t_1,t_2,\ldots,t_n)$, где $t_1,t_2,\ldots,t_n$ – термы.

Например: f(X,Y), f(b,вес(Z)) – термы

P(X,голубой), вес(P(b)) – не термы, т.к. P – предикат

Атом или *элементарная (атомная) формула* – это выражение, включающее константы, переменные, функции и предикаты. Таким образом, если P – n – местный предикат, а $t_1,t_2,\ldots,t_n$ – термы, то $P(t_1,t_2,\ldots,t_n)$ – атом.

Например: P(X,голубой), ВХОД(стол, X, под(окно)) – являются атомами. f(X,Y) – не атом.

Формула или правильно построенная формула (ППФ) определяется следующим образом: всякий атом есть ППФ;

Если F и G – ППФ, а X - переменная, тогда $\neg H$, $(G \vee H)$, $(G \wedge H)$, $(\exists X)G$, $(\forall X)H$ – ППФ.

Выражение “первого порядка” во фразе “исчисление предикатов первого порядка” связано с определением ППФ, в которых запрещается квантифицировать символы предикатов и функций.

Например:

$(\forall P)P(a)$
$(\forall f)(\forall X)P(f(X),b)$ - не являются ППФ логики предикатов первого порядка.

На практике ППФ используется для представления знаний. Не всегда легко представить знания, выраженные на естественном языке, с помощью ППФ. Например, выражение ”если два объекта равны, то они имеют одинаковые свойства” можно представить так;

$$\forall P \forall X \forall Y(PABHO(X,Y) \rightarrow (P(X) \leftrightarrow P(Y))).$$

Но это выражение не является формулой первого порядка, так как квантифицируется предикат.

Правилом вывода называют процедуру, которая из одной или нескольких ППФ производит другие ППФ. Задавая фиксированное множество правил вывода можно рассматривать следующее семейство проблем: исходя из

выбранного множества ППФ применением некоторого числа раз правил вывода можно получить заранее заданную ППФ.

Исходные ППФ называют *аксиомами*, а ППФ, полученные из правил вывода, называют *теоремами*. Цель применения правил вывода, ведущих от аксиом к теореме, называют *доказательством теоремы.*

Интерпретация формул.

Формула имеет определенный смысл, т. е. обозначает некоторое высказывание, если существует какая-либо интерпретация. Интерпретировать формулу - это значит связать с ней определенное непустое множество D, т. е. конкретизировать предметную область, называемую также *областью интерпретации,* и указать:

• для каждой константы в формуле - конкретный элемент из D;

• для каждой n-местной функциональной буквы в формуле - конкретную n-местную функцию на D;

• для каждой n-местной предикатной буквы в формуле - конкретное отношение между n элементами из D.

Примеры применения логики для представления знаний.

Проиллюстрируем синтаксис логики предикатов, сопоставляя нескольким русским фразам их перевод на язык логического формализма.

- По русски: Сергей посылает книгу Марине,

Логически: Посылка(Сергей, Марина, Книга).

-По русски: Каждый человек прогуливается,

Логически: $\forall$ x (Человек(x) $\rightarrow$ Прогуливается(x)).

-По русски: Некоторые люди прогуливаются,

Логически: $\exists$ x (Человек(x) $\wedge$ Прогуливается(x)).

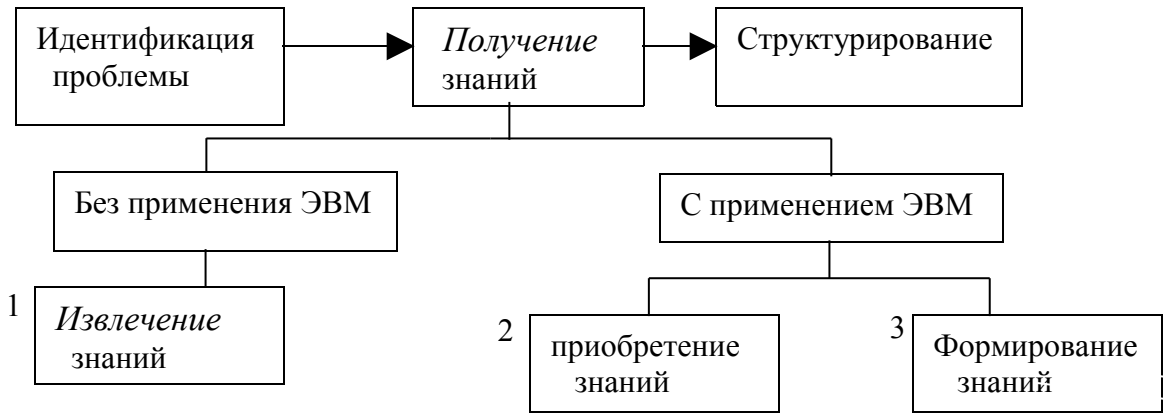
(Сравнивая два последних примера, видим, что замена прилагательного "каждый" на "некоторые" влечет при переводе не только замену квантора $\forall$ на $\exists$, но и замену связки $\rightarrow$ на $\wedge$. Это иллюстрирует тот факт, что перевод фразы естественного языка на логический, вообще говоря, не является трафаретной операцией.)

По русски: Ни один человек не прогуливается,

Логически: □ (x (Человек(x) □ Прогуливается(x))).

Лекция № 4 **Тема:** ***Стратегии получения знаний***.

Выделяют три стратегии проведения стадии получения знаний при разработке экспертных систем.



На современном этапе разработки экспертных систем в нашей стране, третья стратегия является наиболее актуальной, поскольку промышленных систем приобретения и формирования знаний на отечественном рынке программных средств практически нет.

Извлечение знаний – это процедура взаимодействия эксперта с источником знаний, в результате которой становятся явными процесс рассуждений специалистов при принятии решения и структура их представлений о предметной области.

Разработчикам экспертных систем при самостоятельной разработке методов извлечения приходится сталкиваться со следующими трудностями:

- организационные неувязки;
- неудачный способ извлечения, не совпадающий со структурой знаний в данной области;
- неадекватная модель (язык) для представления знаний;
- неумение наладить контакт с экспертом;
- терминологический разнобой;

- отсутствие целостной системы знаний в результате извлечения только «фрагментов»;

- упрощение и уплощение «картины мира» эксперта.

Процесс извлечения знаний – это длительная и трудоёмкая процедура, в которой инженеру по знаниям необходимо воссоздать модель предметной области, которой пользуются эксперты для принятия решения. Может ли эксперт сам извлечь из себя знания? Это нежелательно по следующим причинам:

1) большая часть знаний эксперта – это результат многочисленных наслоений. И часть зная что из $A \Rightarrow B$, эксперт не даёт себе отчёта, что цепочка его рассуждений была длиннее: например $A \Rightarrow Q \Rightarrow R \Rightarrow B$;

2) так как мышление диалогично, то диалог инженера по знаниям и эксперта – наиболее естественная форма «раскручивания» лабиринта памяти эксперта, в которой хранятся знания, частью носящие невербальный характер, т. е. выраженные не в форме слов, а в форме наглядных образов. И в процессе объяснения инженеру по знаниям эксперт на эти ассоциативные образы надевает чёткие словесные образы, т. е. вербализует знания;

3) эксперту гораздо труднее создать модель предметной области в следствии той глубины и необозримости информации, которой он обладает. Многочисленные причинно-следственные связи реальной предметной области образуют сложную систему, из которой выделить «скелет» или главную структуру иногда доступнее аналитику, владеющему к тому же системной методологией. Любая модель – это упрощение, а упрощать легче с меньшим знанием деталей.

Для того чтобы разобраться в природе извлечения знаний выделяют три основные аспекта этой процедуры:

1) психологический;
2) лингвистический;
3) гносеологический.

Психологический **аспект**.

Он является главным, поскольку определяет успешность и эффективность взаимодействия инженера по знаниям (аналитика) с основным источником знаний – экспертом – профессионалом. Психологический аспект выделяется ещё и потому, что извлечение знаний происходит чаще всего в процессе непосредственного общения разработчиков системы.

Выделено четыре основных уровня общения:

1) уровень манипулирования, когда один субъект рассматривает другого как средство или помеху по отношению к проекту своей деятельности;

2) уровень «рефлексивной игры», когда субъект в проекте своей деятельности учитывает «контрпроект» другого субъекта, но не признаёт за ним самоценность и стремится к «выигрышу», к реализации своего проекта;

3) уровень правого общения, когда субъекты признают право на существование проектов деятельности друг друга и пытаются согласовать хотя бы внешне;

4) уровень нравственного общения, когда субъекты внутренне принимают общий проект взаимной деятельности.

Четвертый уровень общения может характеризовать степень профессионализма инженера по знаниям.

Известно, что потери информации при разговорном общении велики:

Хранится в памяти	Приобрело слов.форму	Высказано	Выслушано	Понято	Осталось в памяти
100%	90%	80%	70%	60%	24%
эксперт	о б щ е н и е				инженер по знаниям

В связи с этим рассмотрим проблему увеличения информативности общения аналитика и эксперта за счёт использования психологических знаний.

Предполагается такая структурная модель общения при извлечении знаний:

- участники общения (партнёры);
- средства общения (процедура);
- предмет общения (знания).

В соответствии с этой структурой выделяют три «слоя» психологических проблем, возникающих при извлечении знаний:

1) Контактный слой.

На любой коллективный процесс влияет атмосфера, возникающая в группе участников. Важно, чтобы в коллективе разработчиков складывались кооперативные, а не конкурентные отношения. Т.к. прогнозировать совместимость общения со 100% гарантией невозможно, выделяют ряд черт личности, характера и других особенностей участников общения, оказывающих влияние на эффективность процедуры. Знание этих психологических закономерностей составляет часть психологической культуры, которой должен обладать инженер по знаниям для успешного проведения стадии извлечения знаний.

а) под личностью понимается устойчивая система психологических черт, характеризующая индивидуальность человека;

б) темперамент (наиболее контактны сангвиники и холерики);

в) когнитивный стиль человека, под которым понимается совокупность критериев предпочтения при решении задач и познании мира, специфическая для каждого человека. Когнитивный стиль определяет способ достижения результата. Это способ познания, который позволяет людям с разными способностями добиваться одинаковых результатов в деятельности. Инженеру по знаниям полезно изучить и прогнозировать свой когнитивный стиль, а также стиль эксперта. Особенно важны такие характеристики когнитивного стиля, как:

1) полезависимость – поленезависимость;

Поленезависимость отражает способность человека концентрировать внимание лишь на тех аспектах проблемы, которые нужны для решения конкретной задачи, и умение отбрасывать всё лишнее.

2) импульсивность – рефлективность;

Под импульсивностью понимается быстрое принятие решения (часто без его достаточного обоснования), а под рефлективностью – склонность к рассудительности.

3) ригидность – гибкость;

Ригидность – гибкость характеризует способность человека к изменению установок и точек зрения в соответствии с изменяющейся ситуацией. Ригидные люди не склонны менять свои представления и структуру восприятия, гибкие легко приспосабливаются к новой обстановке.

4) когнитивная эквивалентность;

Когнитивная эквивалентность характеризует способность человека к различению понятий и к разбиению их на классы и подклассы. Чем уже диапазоны когнитивной эквивалентности, тем более тонкую классификацию способен провести индивид, тем больше признаков понятий он может выделить.

2) Процедурный слой.

Проблемы процедурного слоя касаются проведения самой процедуры извлечения знаний.

Общие закономерности проведения процедуры:

а) ситуация общения;

б) возраст (в специальной литературе техники проведения интервью (что очень близко к работе аналитика) указывается, что желательный возраст интервьютора от 25 до 50 лет;

в) использование наглядного материала;

Независимо от метода извлечения знаний, выбранного в конкретной ситуации, его реализация возможна разными способами. Например, можно

учитывать широко известную классификацию людей, занимающихся интеллектуальной деятельностью, на художественный и мыслительный тип.

Важно, что определив тип эксперта, инженер по знаниям может плодотворно использовать любой из методов извлечения, зная, что люди художественного типа легче воспринимают зрительную информацию в форме рисунков, графиков, диаграмм, т. к. эта информация воспринимается через первую сигнальную систему. Напротив, эксперты мыслительного типа лучше понимают язык формул и текстовую информацию. При этом учитывается факт, что большую часть информации человек получает от зрения.

г) число Ингве-Миллера;

Извлечение знаний – это профессиональный разговор, и на его успешность влияет длина фраз, которую произносит инженер по знаниям. Этот факт был установлен американским учёным-лингвистом Ингве и психологом Миллером при проведении исследования о причинах низкой усвояемости команд. Оказалось, человек лучше всего воспринимает предложения глубиной (или длиной) 7 ± 2 слова. Это число можно считать мерой «разговорности» речи.

д) невербальная компонента общения;

Т.к. извлечение знаний происходит чаще всего в процессе общения, то большая часть информации поступает к инженеру по знаниям в форме предложений на естественном языке. Но внешняя речь эксперта есть воспроизведение его внутренней речи (мышления). При этом для передачи этой внутренней речи эксперт использует и невербальные средства, такие как: интонация, мимика, жесты. Опытный инженер по знаниям старается записывать по возможности в протоколы эту дополнительную информацию. Невербальная компонента общения важна и для проблем контактного слоя при установлении контакта, когда по отдельным жестам и выражению лица

эксперта инженер по знаниям может установить границу возможной «дружественности» общения.

е) протоколирование результатов.

Существует три способа протоколирования результатов:

- запись на бумагу непосредственно в ходе беседы (недостатки - мешает беседе, трудно записать всё);
- магнитофонная запись (недостаток – может сковывать эксперта);
- запоминание с последующей записью сразу после беседы (недостаток – аналитик должен быть с блестящей памятью).

3) Когнитивный слой.

Каким образом аналитик может убедиться, что построенное им поле знаний соответствует модели мира предметной области, которой пользуется эксперт.

Несколько советов инженеру по знаниям с позиции когнитивной психологии:

а) не навязывать эту модель представления, которая эму (аналитику) более понятна и естественна;

б) пытаться выявить различные формы семантической репрезентации, учитывая, что они могут существенно отличатся у разных экспертов;

в) использовать различные методы работы с экспертами, исходя из того, что, метод должен подходить к эксперту как «ключ к замку»;

г) чётко осознавать цель процедуры извлечения или её главную стратегию, которая может быть определена как выявление основных понятий предметной области и связывающих их отношений;

д) чаще рисовать схемы, отображающие изложения рассуждений эксперта. Это связано с образной репрезентации информации в памяти человека. (Использовать вербально-визуальную игру «урок рисования»);

- эксперт излагает свои рассуждения;
- аналитик рисует схему или картинку, соответствующую его пониманию рассказа эксперта;

- аналитик воспроизводит рассуждения эксперта, глядя на его рисунок;
- эксперт поправляет аналитика (вербально);
- аналитик исправляет рисунок.

е) учитывать, что на эффективность процедуры извлечения нечёткой информации влияет форма вопроса. Инженер по знаниям должен понимать, что той или иной формой вопроса он может облегчить (или усложнить) сравнение объектов. Особая проблема для аналитика – это уяснить, какова же должна быть форма вопроса, т.е. фактически речь идёт о конструировании интервью или методики опроса эксперта.

Лекция №5 **Лингвистический аспект**

Выделяют три слоя важных для инженерии знаний лингвистических проблем:

1. Общий код.
2. Понятийная структура.
3. Словарь пользователя.

1) Проблема «общего кода».

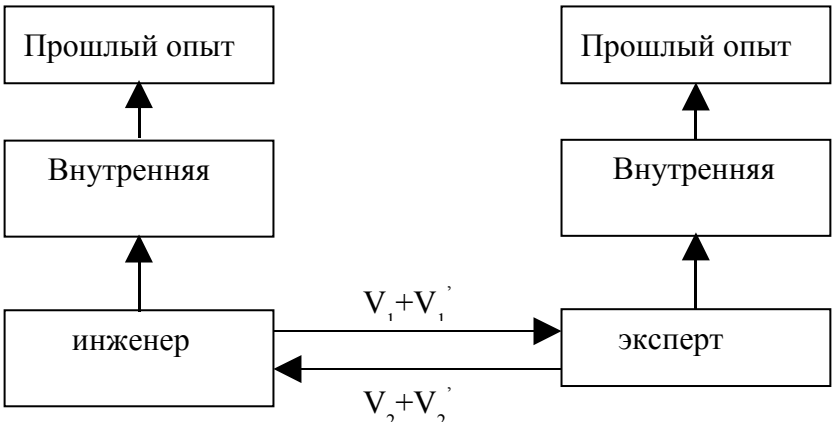


Схема лингвистического общения.

V_1, V_2 – языки, на которых говорят аналитик и эксперт.

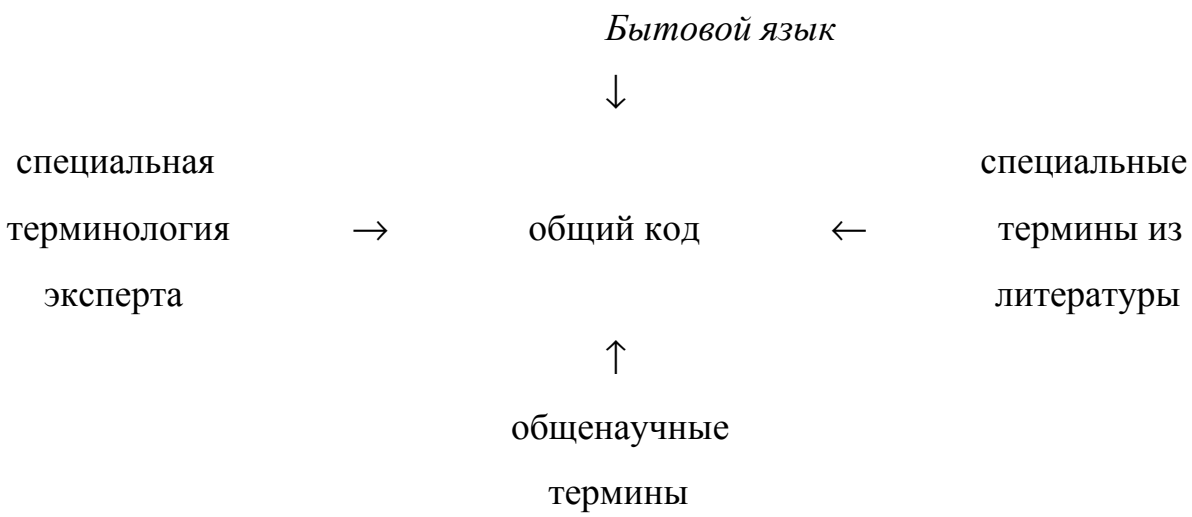
V_1', V_2' - невербальные компоненты общения.

Язык аналитика V_1 состоит из трёх компонент:

- терминов предметной области, которые он узнал из специальной литературы в период подготовки;
- общенаучной терминологии из его «теоретического багажа»;
- бытового разговорного языка, которым пользуется аналитик.

Язык эксперта V_2 состоит из:

- специальной терминологии, принятой в предметной области;
- общенаучной терминологии;
- бытового языка;
- неологизмов (профессиональный жаргон).



Выработка общего кода начинается с выписывания инженером по знаниям всех терминов, употребляемых экспертом, и уточнения их смысла. Затем следует группирование терминов и выбор синонимов. Разработка общего кода заканчивается составлением словаря терминов предметной области с предварительной группировкой их по смыслу, т.е. по понятийной близости.

2) Понятийная структура.

Лингвистическая работа инженера по знаниям на данном слое проблем заключается в построении связных объектов (фрагментов) с помощью «сшивания» терминов.

Например: «был в кино».

Фактически эта работа является подготовкой к этапу концептуализации, где это «шитьё» приобретает некоторый законченный вид. При тщательной работе аналитика и эксперта в понятийных структурах начинает проглядывать иерархия понятий, которая хорошо согласуется с теорией универсального предметного кода, согласно которой при мышлении используются не языковые конструкции, а их коды в форме некоторых абстракций, что в общем согласуется с результатами когнитивной психологии.

Иерархия абстракций – это глобальная схема, которая может быть положена в основу концептуального анализа структуры знаний любой предметной области. Лингвистический эквивалент иерархии - иерархия понятий, которую необходимо построить в понятийной структуре, формируемой инженером по знаниям.

Следы ощущений, восприятий, представлений	взгляд на историю болезни больного X
↓	↓
образная память	образ больного X
↓	↓
инварианты предметов	мужчина 30-40 лет с болезнью Y
↓	↓
обобщения	болезнь Y
↓	↓
понятия	потология

На стадии концептуализации эта иерархическая структура превращается в пирамиду знаний.

Аналитик вынужден всё время помнить о трудности передачи образов и представлений в вербальной форме. Полезными тут оказываются свойства многозначности слов естественного языка.

Например: слово «острый».

Такое свойство «переноса модальности» называется синестезией и расширяет выразительные способности языка.

3) Словарь пользователя.

Лингвистические результаты, отнесённые к слоям общего кода и понятийной структуры, направлены на создание адекватной базы знаний.

Для разработки пользовательского интерфейса необходима дополнительная доработка словаря общего кода с поправкой на доступность системы.

Важность хорошего пользовательского интерфейса можно продемонстрировать на примере: абсолютно лысого пользователя спрашивают о цвете волос.

Характерными лингвистическими неудачами подстерегающего начинающего инженера по знаниям являются:

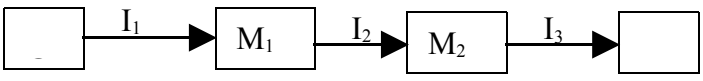
1. разговор на разных языках (из за слабой подготовки инженера по знаниям);
2. соотношение не с тем контекстом и неадекватная интерпретация терминов (из-за отсутствия обратной связи, т.е. слишком независимой работы инженера по знаниям);
3. отсутствие отличий между «общим кадром» и языком пользователя (не учтены различия в уровне знаний эксперта и пользователя).

гносеологический аспект

Гносеология – это раздел философии, связанный с теорией познания, или теорией отражения действительности в сознании человека.

Инженерия знаний, как наука, дважды гносеологична – сначала действительность (0) отражается в сознании эксперта (M_1), а затем деятельность и опыт эксперта интерпретируются сознанием инженера по знаниям (M_2), что служит уже основой для построения третьей интерпретации (Pz) – поля знаний экспертной системой.

Схема:



Выделяют 5 форм знаний:

Z_1 – знания в памяти человека;

Z_2 – материализованные знания (статьи, учебники, монографии и т.д.);

Z_3 – поле знаний (полуформализованное описание Z_1 и Z_2);

Z_4 – знания на языках представления знаний (формализация Z_3);

Z_5 – база знаний в ЭВМ.

Если описать процессы I_2 и I_3 в этой терминологии, то мы имеем дело с превращением экспертного знания Z_1 и теоретического опыта Z_2 в поле знаний Z_3, которое есть материализация модели мира M_2 инженера по знаниям.

В процессе извлечения знаний аналитика в основном интересует компонент Z_1, связанный с индивидуальными неканоническими знаниями экспертов, поскольку предметные области именно с таким типом знаний считаются наиболее восприимчивыми к внедрению экспертных систем. Эти области обычно называют эмпирическими, т.к. в них накоплен большой объём отдельных эмпирических фактов и наблюдений, в то время как их теоретическое обобщение – вопрос будущего.

Если считать, что инженер по знаниям извлекает только фрагмент Z_1', т.е. часть из системы знаний эксперта Z_1, то его задача, во-первых, стараться, чтобы структура Z_1' соответствовала Z_1, и, во-вторых, чтобы Z_1' как можно более полно содержал Z_1. При этом инженеру по знаниям может помочь системная методология, позволяющая использовать известные принципы логики научных исследований. Эта методология заставляет его за частным всегда увидеть общее, т.е. строить цепочки: факт → обобщённый факт → эмпирический закон → теоретический закон.

Не всегда инженер по знаниям дойдёт до последнего звена этой цепочки, но уже само стремление к движению бывает плодотворным. Такой подход полностью согласуется со структурой научного знания, которое имеет два уровня:

1. эмпирический (наблюдения, явления);
2. теоретический (законы, абстракции, обобщения).

1. Основными методологическими критериями научности, позволяющими считать научными и само новое знание и способ и способ его получения, является:

а) внутренняя согласованность и непротиворечивость;

Этот критерий в эмпирических областях просто не работает – в них факты часто не согласуются друг с другом, определения противоречивы. Аналитику, знающему особенности эмпирического знания – его модальность, противоречивость и неполноту, приходится сглаживать эти «шероховатости» эмпирики.

Модальность знания означает возможность его существования в различных категориях, т.е. в конструкциях существования и долженствования. Т.о. часть закономерностей возможна, другая обязательна и т.д. Кроме того, приходится различать такие оттенки модальности, как:

- Э знает, что… ;
- Э думает, что… ;
- Э хочет, чтобы… ;
- Э считает, что… .

Возможная противоречивость эмпирического знания – естественное следствие из основных законов диалектики, и противоречия эти не всегда должны разрешаться в поле знаний, а напротив, именно противоречия служат чаще всего отправной точкой в рассуждениях экспертов.

Неполнота знания связана с невозможностью полного описания предметной области. Задача аналитика эту неполноту ограничить определёнными рамками полноты, т.е. сузить границы предметной области, либо ввести ряд ограничений и допущений, упрощая проблему.

б) системность;

Системно-структурный подход к познанию ориентирует аналитика на рассмотрение любой предметной области с позицией закономерности системного целого и взаимодействия составляющих его частей.

в) объективность;

Лучше говорить о глубине понимания, чем об объективности знания. Понимание – это сотворчество, процесс истолкования объекта с точки зрения субъекта. Все усилия аналитик должен сосредоточить на понимании проблемы.

г) историзм;

Этот критерий связан с развитием. Познание настоящего – это познание породившего его прошлого. Инженер по знаниям всегда должен рассматривать процессы с учётом временных изменений – как связь с прошлым, так и связь с будущим.

2. Методологическая структура познания может быть представлена как последовательность этапов:

1) Описание и обобщение фактов.
2) Установление связей и закономерностей
3) Построение идеализированной модели.
4) Объяснение и предсказание моделей

Рассмотрим эти этапы с позиции деятельности инженера по знаниям.

1) Описание и обобщение фактов.

Это «сухой остаток» бесед аналитика с экспертом. Тщательность и полнота ведения протоколов во время процедуры извлечения и пунктуальная «домашняя работа» над ними – вот залог продуктивного первого этапа познания.

2) Установление связей и закономерностей.

Реконструируя рассуждения эксперта, инженер по знаниям может опираться на две наиболее популярных теории мышления – логическую и ассоциативную.

Традиционная логика формирует критерии, которые гарантируют точность, валидность, непротиворечивость общих понятий, суждений и выводов. Инженер по знаниям использует операции традиционной логики и выделяет их в схеме рассуждений эксперта.

Это следующие операции: определение, сравнение и различие, анализ, абстрагирование, обобщение, классификация, категоризация, образование суждений, умозаключение.

Теория ассоциаций представляет мышление как цепочку идей, связанных общими понятиями. Основными операциями такого мышления являются:

- ассоциации, приобретённые на основе различных связей;
- припоминание прошлого опыта;
- пробы и ошибки со случайными успехами;
- привычные реакции.

Наряду с двумя этими теориями большой интерес для инженерии знаний может представлять гештальтпсихология. Под гештальтом понимается принцип целостности восприятия – как основа мышления. Гештальтпсихологи стараются во всём выделить некий целостный образ или структуру, как базис для понимания процессов и явлений окружающего мира. Эта теория близка теории фреймов и направлена на постижение глубинного знания, которое характеризуется стабильностью и симметрией. При этом важен так называемый «центр ситуации», относительно которого развивается знание о предметной области.

Для инженера по знаниям это означает, что выявляя различные фрагменты знаний, он не должен забывать о главном, которое влияет на остальные компоненты и связывает их в некоторую структурную единицу.

В гештальт-теории существует закон «стремления к хорошему гештальту», согласно которому структуры сознания стремятся к гармонии, связности, простоте.

3) Построение идеализированной модели.

Для построения модели, отражающей представление субъекта о предметной области, необходим специализированный язык, с помощью которого можно описывать и конструировать те идеализированные модели

мира, которые возникают в процессе мышления. Язык этот создаётся постепенно с помощью категориального аппарата, принятого в соответствующей предметной области, а также формально-знаковых средств математики и логики. Инженеру по знаниям необходимо овладение такими специфическими гносеологическими приёмами, как идеализация, огрубление, абстрагирование, которые позволяют адекватно отображать в моделях реальную картину мира. Эти приёмы доводят приёмы и признаки объектов до пределов, позволяющих воспроизводить законы действительности в чистом виде. Инженер по знаниям, который стремится познать предметную область, должен быть готов постоянно изменять свои уже утвердившиеся способы восприятия и оценки мира, и даже отказаться от них. При этом тщательнее всего следует проверять правильность суждений, которые кажутся самыми очевидными.

4) Объяснение и предсказание моделей.

Этот завершающий этап структуры познания является одновременно и частным критерием истинности полученного знания. Если выявленная система знаний эксперта полна и объективна, то на её основании можно делать прогнозы и объяснять любые явления из данной предметной области.

Наиболее часто встречающиеся неудачи, связанные с гносеологическими проблемами инженерии знаний:

1) Обрывность, фрагментарность знаний (из-за нарушения принципа системности или ошибок в выборе фокуса внимания).
2) Противоречивость знаний (из-за естественной противоречивости природы и общества, из-за неполноты извлечённых знаний, и из-за некомпетентности эксперта).
3) Ошибочная классификация (из-за неправильного определения числа классов или неточного описания класса).
4) Ошибочная атрибуция, т.е. приписывание объекту свойств, которыми он не владеет (из-за неглубокой проработки знаний о предметной области).

5) Ошибочный уровень обобщения (из-за чрезмерной детализации или обобщённости классов объектов).
6) Ошибочная модель (из-за неадекватных когнитивных структур).

Лекция № 6 **Тема:** ***Методы работы со знаниями.***

Основные понятия

Приобретением знаний называется выявление знаний из источников и преобразование их в нужную форму, а также перенос в базу знаний ИС. Источниками знаний могут быть книги, архивные документы, содержимое других баз знаний и т. п., т. е. некоторые *объективизированные знания,* переведенные в форму, которая делает их доступными для потребителя. Другим типом знаний являются *экспертные знания,* которые имеются у специалистов, но не зафиксированы во внешних по отношению к нему хранилищах. Экспертные знания являются *субъективными.* Еще одним видом субъективных знаний являются *эмпирические знания.* Такие знания могут добываться ИС путем наблюдения за окружающей средой (если у ИС есть средства наблюдения).

Ввод в базу знаний объективизированных знаний не представляет особой проблемы, выявление и ввод субъективных и особенно экспертных знаний достаточно трудны. Чтобы разработать методологию приобретения субъективных знаний, получаемых от эксперта, надо четко различать две формы репрезентации знаний. Одна форма связана с тем, как и в каких моделях хранятся эти знания у человека-эксперта. При этом эксперт не всегда осознает полностью, как репрезентированы у него знания. Другая форма связана с тем, как инженер по знаниям, проектирующий ИС, собирается их описывать и представлять. От степени согласованности этих двух форм репрезентации между собой зависит эффективность работы инженера по знаниям.

Кроме понятий репрезентируются и отношения между ними. Как правило, отношения между понятиями определяются процедурным способом, а отношения между составляющими понятий (определяющими структуру понятия) – декларативным способом. Наличие двух видов описаний заставляет

в моделях представления знаний одновременно иметь оба компонента, например, семантическую сеть и продукционную систему, как это представлено в когнитивной модели.

При приобретении знаний важную роль играют так называемое *поле знаний,* в котором содержатся основные понятия, используемые при описании предметной области, и свойства всех отношений, используемых для установления связей между понятиями. Поле знаний связано с концептуальной моделью проблемной области, в которой еще не учтены ограничения, которые неизбежно возникают при формальном представлении знаний в базе знаний. Переход от описания некоторой области в поле знаний к описанию в базе знаний аналогичен переходу от концептуальной модели базы данных к ее логической схеме, когда уже зафиксирована система управления базой данных. Важно отметить, что переход непосредственно к формальным представлениям в базе знаний без этапа концептуального описания в поле знаний приводит к многочисленным ошибкам, что замедляет процесс формирования базы знаний интеллектуальных систем.

Системы приобретения знаний от экспертов

1. Проблемы, возникающие при извлечении экспертных знаний, некоторые психологи связывают с так называемой когнитивной защитой. Была развита теория человеческого познания, основанная на понятии "персональных конструктов", которые человек создает и пытается приспособить к реалиям мира. Данная теория (теория персональных конструктов) использовалась для создания системы извлечения экспертных знаний и показала свою способность успешно преодолевать когнитивную защиту (нежелание экспертов достичь четкого и осознанного ими истолкования основных понятий, отношений между понятиями и приемов решения задач в интересующей инженера по знаниям проблемной области).
2. В системе KRITON для приобретения знаний используются два источника: эксперт с его знаниями, полученными на практике (эти

знания, как правило, неполны, отрывочны, плохо структурированы); книжные знания, документы, описания инструкции (эти знания хорошо структурированы и фиксированы традиционными средствами). Для извлечения знаний из первого источника в KRITON применена техника интервью, использующая стратегии репертуарной решетки и разбиения на ступени. При этом применяется прием переключения стратегий: если при предъявлении тройки семантически связанных понятий эксперт не в состоянии назвать признак, отличающий два из них от третьего, система запускает стратегию разбиения на ступени и предпринимает попытку выяснения таксономической структуры этих понятий с целью выявления признаков, их различающих.

Для выявления процедурных знаний эксперта в KRITON применен метод протокольного анализа. Он осуществляется в пять шагов. На первом шаге протокол делится на сегменты на основании пауз, которые делает эксперт в процессе записи. Второй шаг – семантический анализ сегментов, формирование высказываний для каждого сегмента. На третьем шаге из текста выделяются операторы и аргументы. Далее делается попытка поиска по образцу в базе знаний для обнаружения переменных в высказываниях (переменная вставляется в высказывание, если соответствующая ссылка в тексте не обнаружена). На последнем шаге утверждения упорядочиваются в соответствии с их появлением в протоколе.

Анализ текста используется в KRITON для выявления хорошо структурированных знаний из книг, документов, описаний, инструкций.

3. Метод выявления модели предметной области. Первая фаза – формирование инженером по знаниям грубой модели предметной области путем определения предикатов и сортов их возможных аргументов и сообщения системе фактов об области, выражаемых этими предикатами. Система выявляет свойства предикатов и устанавливает отношения между ними, структурируя таким образом предметную область. На второй фазе с помощью метазнаний (общих структур), отражающих особенности

человеческого мышления, осуществляется проверка соответствия фактов предикатам, индуктивный вывод правил из фактов, вывод правил из других правил.

Основные этапы реализации системы приобретения знаний:

1. Интервью для определения актуальной области, в которой происходит процесс решения интересующей проблемы, и расчленение ее на автономные области.
2. Автоматизированное интервью для выявления и формирования декларативной модели предметной области.
3. Протокольный анализ к выявленным на предыдущем этапе понятиям и отношениям предметной области для пополнения модели процедурными знаниями (этапы 2 и 3 можно использовать попеременно до тех пор, пока модель не достигнет нужной полноты).
4. Протокольный анализ для пополнения декларативных знаний модели.
5. Проверка полноты модели. Обычно протокольный анализ выявляет пустоты в модели. Имеется в виду случай, когда понятия, использованные в "мыслях вслух", недостаточно описаны. В этом случае интервью и протокольный анализ повторяются.

Формализация качественных знаний

При формализации качественных знаний может быть использована теория нечетких множеств, особенно те ее аспекты, которые связаны с *лингвистической неопределенностью,* наиболее часто возникающей при работе с экспертами на естественном языке. Под лингвистической неопределенностью подразумевается не полиморфизм слов естественного языка, который может быть преодолен на уровне понимания смысла высказываний в рамках байесовской модели, а качественные оценки естественного языка для длины, времени, интенсивности, для целей логического вывода, принятия решений, планирования.

Лингвистическая неопределенность в системах представления знаний задается с помощью *лингвистических моделей* основанных на теории

лингвистических переменных и теории приближенных рассуждении. Эти теории опираются на понятие *нечеткого множества,* систему операций над нечеткими множествами и методы построения *функций принадлежности.*

Одним из основных понятий, используемых в лингвистических моделях, является понятие *лингвистической переменной.* Значениями лингвистических переменных являются не числа, а слова или предложения некоторого искусственного либо естественного языка. Например, числовая переменная "возраст" принимает дискретные значения между нулем и сотней, а целое число является значением переменной. Лингвистическая переменная "возраст" может принимать значения: молодой, старый, довольно старый, очень молодой и т. д. Эти термы – лингвистические значения переменной. На это множество (как и на числа) также налагаются ограничения. Множество допустимых значений лингвистической переменной называется *терм-множеством.*

При вводе в ЭВМ информации о лингвистических переменных и терм-множестве ее необходимо представить в форме, пригодной для работы на ЭВМ. Лингвистическая переменная задается набором из пяти компонентов: L, L(a), U, R, O, где L - имя лингвистической переменной; L(a) - ее терм-множество; U - область, на которой определены значения лингвистической переменной; R описывает операции по порождению производных значений лингвистической переменной на основе тех значений, которые входят в терм-множество. С помощью правил из O можно расширить число значений лингвистической переменной, т. е. расширить ее терм-множество.

При получении от экспертов информации о виде функций принадлежности необходимо учитывать характер измерений (первичные и производные измерения) и тип шкалы, на которую проецируются измерения и на которой будут определяться функции принадлежности. На этой шкале задается вид допустимых операторов и операций, т. е. некоторая алгебра для функций принадлежности. Кроме того, следует различать характеристики, которые можно измерять непосредственно и характеристики, которые являются

качественными и требуют попарного сравнения объектов, обладающих этими характеристиками, чтобы определить их отношение к исследуемому понятию.

Можно выделить две группы методов построения функций принадлежности: прямые и косвенные. В прямых методах эксперт непосредственно задает правила определения значений функции принадлежности. Эти значения согласуются с его предпочтениями на множестве объектов. К прямым методам относится непосредственное задание функции принадлежности таблицей, формулой или примером. В косвенных методах значения функции принадлежности выбираются таким образом, чтобы удовлетворялись заранее сформулированные условия. Экспертная информация является только исходной для дальнейшей обработки. Дополнительные условия могут налагаться как на вид получаемой информации, так и на процедуру обработки. Примерами дополнительных условий могут служить следующие: функция принадлежности должна отражать близость к заранее выделенному эталону, объекты множества являются точками в параметрическом пространстве; результатом процедуры обработки должна быть функция принадлежности, удовлетворяющая условиям интервальной шкалы; при попарном сравнении объектов, если один объект оценивается в k раз сильнее, чем другой то второй объект оценивается в $1/k$ раз сильнее, чем первый объект и т. д.

Как правило, прямые методы используются для описания понятий, которые характеризуются измеримыми признаками (высотой, ростом, массой, объемом). В этом случае удобно непосредственное задание функции принадлежности. Так как люди часто искажают оценки, например сдвигают их в направлении концов оценочной шкалы, то прямые измерения, основанные на непосредственном определении значений функции принадлежности, могут быть использованы только в том случае, когда такие искажения незначительны или маловероятны. Косвенные методы более трудоемки, чем прямые, но обладают стойкостью к искажениям в ответе. Результатом применения косвенных методов является интервальная шкала, для которой "условие

безоговорочного экстремума": при определении степени принадлежности множество исследуемых объектов должно содержать по крайней мере два объекта, численные представления которых на интервале [0; 1] - 0 и 1 соответственно.

Функции принадлежности могут отражать мнение как некоторой группы экспертов, так и одного уникального эксперта. Комбинируя возможные два метода построения функций принадлежности с двумя типами экспертов (коллективным и уникальным), можно получить четыре типа экспертизы.

Пример формализации качественных знаний

При анализе ситуации эксперт рассуждает в семантическом пространстве (пространстве шкал), в котором ситуации соответствует оцененный образ. Семантическое пространство аналогично субъективному пространству ощущений, в котором формируется внутренний образ внешних сигналов и возникают субъективные связи между свойствами (признаками, параметрами). В зависимости от индивидуального восприятия одно и то же значение признака может быть оценено по-разному. Однако для конкретного индивидуума оцененная ситуация является инвариантом относительно определенного класса ситуаций. Следовательно, при отождествлении реальных значений признаков с семантическим образом существенной является форма нечеткого отображения пространства признаков в семантическое пространство.

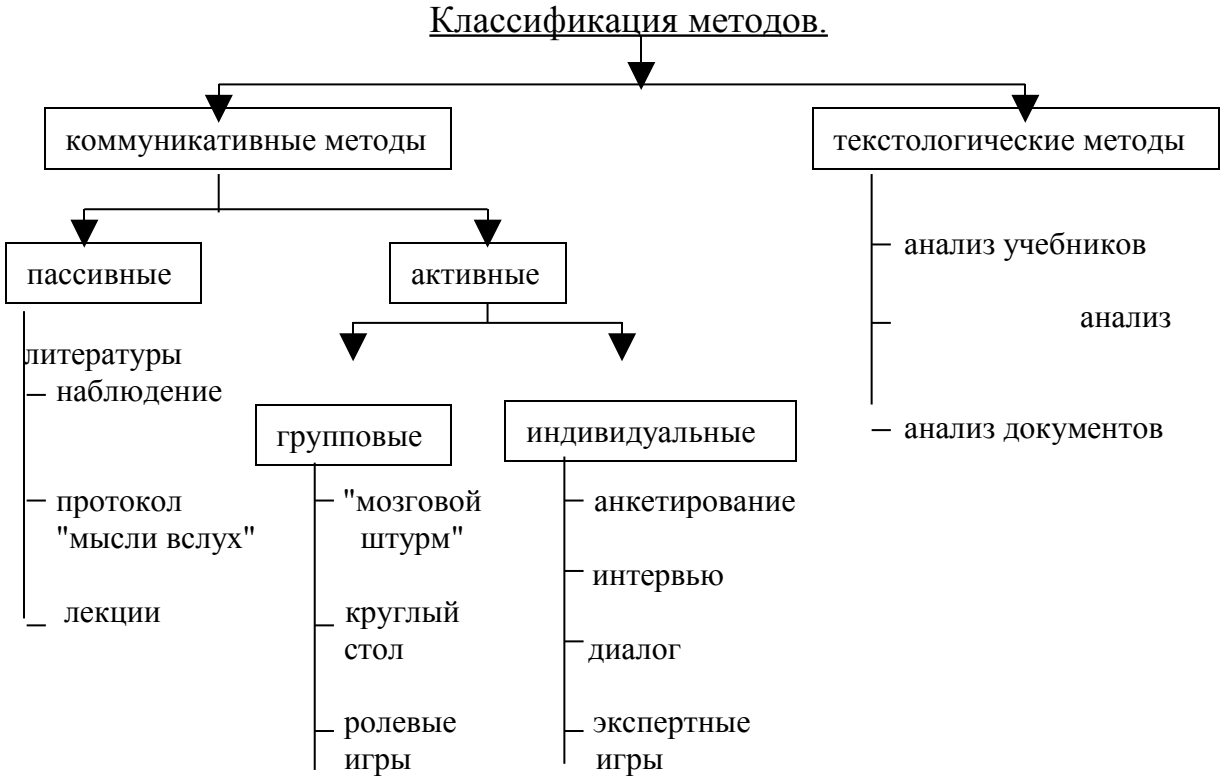
Отображение любой ситуации на единичный интервал происходит таким образом, что точка интервала характеризует степень проявления некоторого свойства (0 соответствует отсутствию свойства, 1 – интересующему нас максимальному проявлению свойства). При построений функции принадлежности используется *модель измерений,* которая определяется двумя параметрами: типом шкалы принадлежности, на которую отображается информация от эксперта и типом измерения (прямой или косвенный). Шкала называется фундаментальной, если она допускает прямое взаимодействие множества U и того нечеткого свойства, которое нас интересует. Такая шкала

дает возможность прямого измерения субъективного восприятия нечетких множеств на U со свойствами понятия.

Процесс формализации знаний, полученных у эксперта, состоит из следующих шагов:

1. выбор метода измерения нечеткости,
2. получение исходных данных посредством опроса эксперта,
3. реализация алгоритма построения функции принадлежности.

Лекция № 7,8 **Тема:** ***Методы извлечения знаний.***



На выбор метода влияют три фактора:

- личные особенности инженера по знаниям;
- личные особенности эксперта;

- характеристика предметной области.

Классификация людей по психологическим характеристикам делит всех на три типа:

1) мыслители (познавательный тип) - ориентированы на интеллектуальную работу, учёбу,теоретические обобщения и обладают такими характеристиками когнитивного стиля, как поленезависимость и рефлексивность.

2) собеседники (эмоционально-коммуникативный тип) - это общительные, открытые люди, готовые к сотрудничеству.

3) практики (практический тип) - предпочитают действие разговорам, хорошо реализуют замыслы других, направлены на результативность работы.

Для характеристики предметных областей можно предложить следующую классификацию:

- хорошо документированные
- средне документированные
- слабо документированные.

Если представить знания $Z_{ПО}$ предметной области, то $Z_{ПО} = Z_1 \cup Z_2$.

Предметные области можно разделить по критерию структурированности знаний. Под структурированностью будем понимать степень теоретического осмысления и выявленности основных закономерностей и принципов, действующих в данной предметной области.

По степени структурированности знаний предметные области могут быть:

- хорошо структурированными - с чёткой аксиоматизацией, широким применением математического аппарата, устоявшейся терминологией;
- средне структурированными - с определившейся терминологией, развивающейся теорией, явными взаимосвязями между явлениями;
- слабо структурированными - с размытыми определениями, богатой эмпирикой, скрытыми взаимосвязями, с большим количеством "белых пятен".

Пассивные методы

I) Наблюдения.

Есть две основные разновидности проведения наблюдений:

- наблюдение за реальным процессом;
- наблюдение за имитацией процесса.

Обычно используют обе разновидности.

Сеансы наблюдений могут потребовать от инженера по знаниям:

1) овладение техникой стенографии для фиксации действий эксперта в реальном масштабе времени;

2) ознакомление с методиками хронометрирования для чёткого структурирования производственного процесса по времени;

3) развитие навыков "чтения по глазам", т.е. наблюдательности к жестам, мимике и др. невербальным компонентам общения;

4) серьёзного предварительного знакомства с предметной областью т.к. из-за отсутствия "обратной связи" иногда многое непонятно в действиях эксперта.

Протоколы наблюдений после сеансов в ходе домашней работы тщательно расшифровываются, а затем обсуждаются с экспертами.

II) Анализ протоколов "мыслей вслух" отличаются от наблюдений тем, что эксперта просят не просто прокомментировать свои действия и решения, но и объяснить, как это решение было найдено.

Во время рассуждений эксперта все его слова протоколируются инженером по знаниям, при этом полезно отмечать даже паузы и междометия. Иногда этот метод называют "вербальные отчёты".

Основной трудностью является принципиальная сложность для среднего человека объяснить, что он думает. Автор теории фреймов М. Минский считает, что "только как исключение, а не правило человек может объяснить то, что он знает".

III) Лекции.

Активные индивидуальные методы.

I) Анкетирование:

Инженер по знаниям заранее составляет опросник или анкету, размножает её и использует для опроса нескольких экспертов. Сама процедура может производиться двумя способами:

1) Аналитик вслух задаёт вопросы и сам заполняет анкету по ответам эксперта.

2) Эксперт самостоятельно заполняет анкету после предварительного инструктирования.

Выбор способа зависит от конкретных условий.

Рекомендации при составлении анкет:

1) Анкета не должна быть монотонной и однообразной.

2) Должна быть приспособлена к языку экспертов.

3) Следует учитывать, что вопросы влияют друг на друга, поэтому последовательность вопросов должна быть строго продумана.

4) Желательно стремиться к оптимальной избыточности.

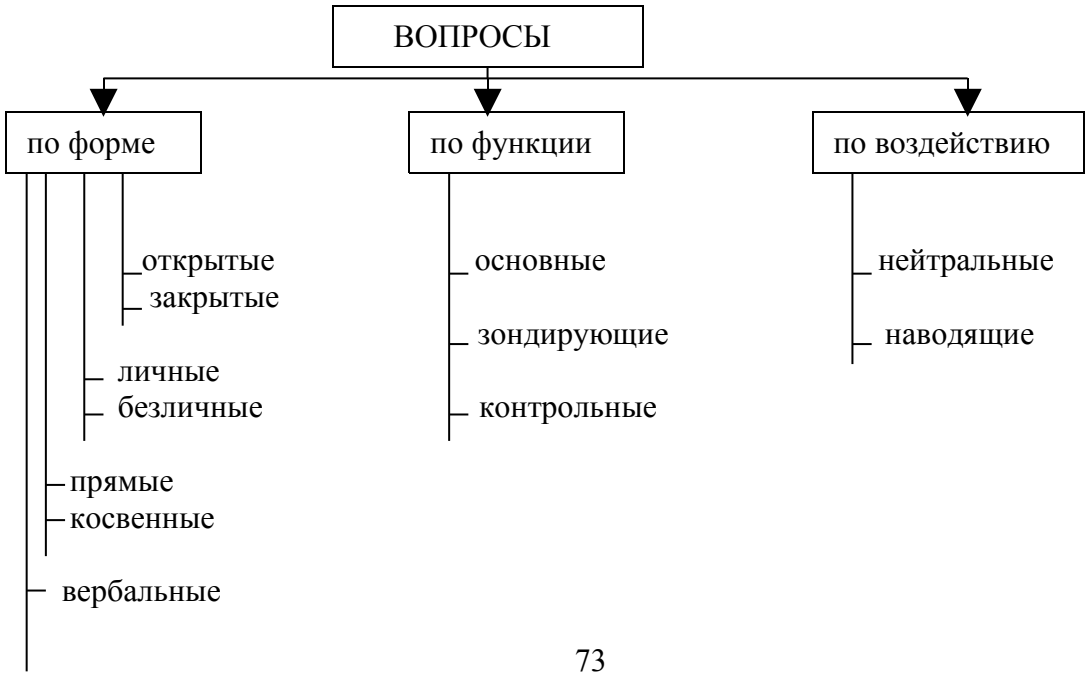
5) Анкета должна иметь "хорошие манеры".

II) Интервью.

Все вопросительные предложения можно разбить на два типа:

1) Вопросы с неопределённостью, относящиеся ко всему предложению;

2) Вопросы с неполной информацией.



– с использованием наглядного материала

Также полезно различать и включать в интервью следующие вопросы:

- контактные ("ломающие лёд" между аналитиком и экспертом);
- буферные (для разграничения отдельных тем интервью);
- оживляющие память эксперта (для реконструкции отдельных случаев из практики);
- "провоцирующие" (для получения спонтанных, неподготовленных ответов).

Три основные характеристики вопросов, которые влияют на качество интервью:

- язык вопроса (понятность, лаконичность, терминология);
- порядок вопросов (логическая последовательность и немонотонность);
- уместность вопросов (этика, вежливость).

III) Свободный диалог.

Свободный диалог - это метод извлечения знаний в форме беседы инженера по знаниям и эксперта, в котором нет жёсткого регламентированного плана и вопросника.

При диалоге необходимо предусмотреть следующие стадии:

1) Начало беседы (знакомство, создание у эксперта "образа" аналитика, объяснение целей и задач работы).

2) Диалог по извлечению знаний.

3) Заключительная стадия (благодарность эксперту, подведение итогов, договор о последующих встречах).

Схема подготовки к свободному диалогу.

– овладение общей культурой

– знакомство с системной методологией

ирования

— подготовка к ситуации общения

— знакомство с экспертом

— тестирование эксперта

– изучение когнитивной психологии

Активные групповые методы

Основное достоинство активных методов - это возможность одновременного "поглощения" знаний от нескольких экспертов, взаимодействие которых вносит в этот процесс элемент принципиальной новизны от наложения разных взглядов и позиций.

I) "Круглый стол".

Число участников дискуссии колеблется от 3 до 5-7. Существует специфика, связанная с поведением человека в группе:

1) от инженера по знаниям подготовка "круглого стола" потребует дополнительных усилий, как организационных, так и психологических;

2) большинство участников будет говорить под воздействием "эффекта фасада" совсем не то, что сказали бы в другой обстановке, т.е. желание произвести впечатление на других экспертов будет существенно "подсвечивать" их высказывания.

Перед началом дискуссии ведущему полезно:

- убедиться, что все правильно понимают задачу;
- установить регламент;
- чётко сформулировать тему.

II) "Мозговой штурм".

Впервые этот метод был использован в 1939 г. в США А. Осборном как способ получения новых идей в условиях запрещения критики. Основная идея штурма - это определение процедуры генерирования идей в замкнутой группе специалистов от процесса анализа и оценки высказанных идей. При

последующем анализе всего лишь 10-15% идей оказываются разумными, но среди них бывают весьма оригинальные. Оценивает результаты обычно группа экспертов, не участвовавшая в генерации.

Фиксация хода сеанса - традиционная.

III) Экспертные игры.

Понятие экспертной игры или игры с экспертом в целях извлечения знаний восходит к трём источникам - это понятие деловой игры, широко используемое при подготовке специалистов и моделировании; понятие диагностической игры и компьютерные игры.

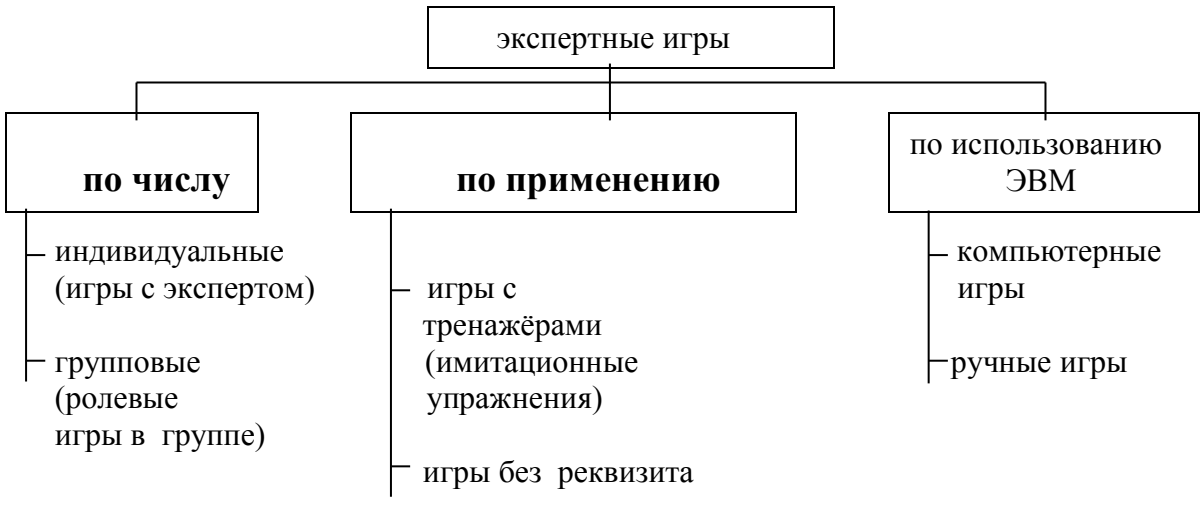
Под деловой игрой чаще всего понимают эксперимент, где участникам предлагается производственная ситуация, а они на основе своего жизненного опыта , своих общих и специальных знаний и представлений принимают решения. Решения анализируются, и вскрываются закономерности мышления участников эксперимента.

Существует три основных типа деловых игр:

- учебные;
- планово-производственные;
- исследовательские.

Диагностическая игра - это та же деловая игра, но применяемая конкретно для диагностики методов принятия решений в медицине.

Классификация экспертных игр



Игры с экспертом:

1) "Учитель и ученик" – инженер по знаниям берёт на себя роль ученика и на глазах у эксперта выполняет его работу, а эксперт поправляет ошибки "ученика".

2) "Двух врачей" – инженер по знаниям берёт на себя роль врача, который знает хорошо больного, а эксперт играет роль консультанта. Консультант задаёт вопросы и делает прогноз о целесообразности применения того или иного вида лечения.

3) Сначала эксперта просят написать обоснование для собственного прогноза. Накапливается несколько таких обоснований, а через некоторое время эксперту зачитывают только его обоснование и просят сделать прогноз. Эксперт дополняет обоснование, тем самым выявляются скрытые (для самого эксперта) пласты знаний.

4) "Фокусировка на контексте" - эксперт играет роль экспертной системы, а инженер по знаниям - роль пользователя. Разыгрывается ситуация консультации. Первые вопросы эксперту выявляют наиболее значимые понятия, самые важные аспекты проблемы.

Компьютерные игры:

1) Позиционные игры.

2) Динамические игры (связанные со скоростью реакции).

3) Зрелищные или диалоговые фильмы, где пользователь может влиять на сюжет.

4) Обучающие, в которых пользователь, играя, осваивает какие-то навыки или узнаёт что-то новое для себя.

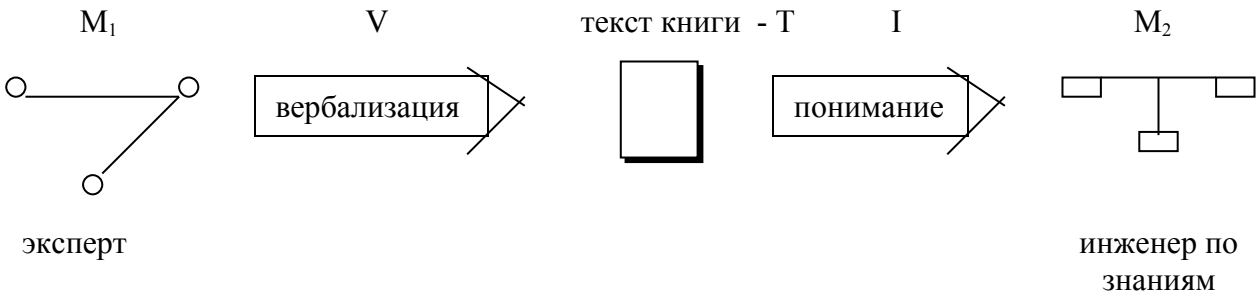
Текстологические методы

Текстологические методы извлечения знаний, используя основные положения текстологии, отличаются принципиально от её методологии; во - первых, характером и природой своих источников; во - вторых, жёсткой

прагматической направленностью извлечения конкретных профессиональных знаний.

Задачу извлечения знаний из текстов можно сформулировать как задачу понимания и выделения смысла текста.

Схема извлечения знаний из спец. текстов.



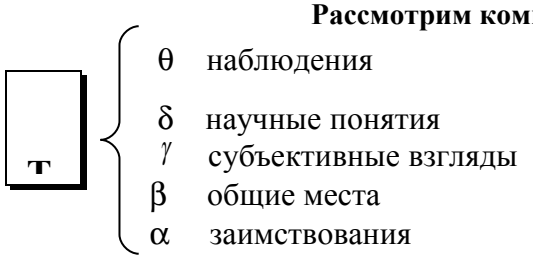
Выделим такие смысловые структуры:

M_1 – смысл, который пытался заложить автор;

M_2 – смысл, который постигает читатель, в данном случае инженер по знаниям в процессе интерпретации I. При этом T – это словесное одеяние M_1, т. е. Результат вербализации V.

Сложность заключается в принципиальной невозможности совпадения знаний, образующих M_1 и M_2, из – за того, что M_1 образуется за счет всей совокупности представлений, потребностей, интересов и опыта автора, лишь малая часть которых находит отражение в тексте T. Соответственно и M_2 образуется в процессе интерпретации текста T за счет привлечения всей совокупности научного и человеческого багажа читателя. Таким образом, два инженера по знаниям извлекут из одного T две различные модели M^i и M^j.

Рассмотрим компоненты научного текста



При извлечении знаний аналитику, интерпретирующему текст, приходится решать задачу декомпозиции этого текста на перечисленные выше компоненты для выделения истинно значимых для реализации базы знаний

фрагментов. Сложность интерпретации научных и специальных текстов заключается еще и в том, что любой текст приобретает смысл только в контексте, где под контекстом понимается окружение, в которое «погружен» текст.

Различают микро- и макроконтекст. Микроконтекст – это ближайшее окружение текста. Макроконтекст – это вся система знаний, связанная с предметной областью.

Центральное звено процедуры извлечения знаний – понимание, т.е. формирование «второго текста», а именно семантической структуры.

В нашей терминологии – это попытка воссоздания семантической структуры M_1 в процессе формирования модели M_2.

Основными моделями понимания текста являются:

1) Выдвижение предварительной гипотезы о смысле всего текста (предугадывание);
2) Определение значений непонятных слов;
3) Возникновение общей гипотезы о содержании текста;
4) Уточнение значений терминов и интерпретация отдельных фрагментов текста под влиянием общей гипотезы (от целого к частям);
5) Формирование некоторой смысловой структуры текста за счет установления внутренних связей между отдельными важными словами и фрагментами, а также за счет образования абстрактных понятий, обобщающих конкретные фрагменты знаний;
6) Корректировка общей гипотезы относительно содержащихся в тексте фрагментов знаний (от частей к целому);
7) Принятие основной гипотезы, т.е. формирование M_2.

Центральными моментами процесса I являются шаги 5 и 7.

При анализе текста важно выявление внутренних связей между отдельными элементами текста и понятиями. Традиционно выделяются два вида связей в тексте – эксплицитные (или явные связи), которые выражаются во внешнем дроблении текста, и имплицитные (скрытые связи).

На процесс понимания I и модель M_2 влияют следующие компоненты:

M_2 {
- φ — личный опыт аналитика
- ε — общенаучная эрудиция аналитика
- ω — предварительные знания инженера по знаниям о предметной области
- $(\alpha, \beta, \lambda, \theta)'$ — экстракт текста T
}

Практическая методика анализа текстов с целью извлечения и структурирования знаний:

1) Составление «базового» списка литературы для ознакомления с предметной областью и чтение по списку;
2) Выбор текста для извлечения знаний;
3) Первое знакомство с текстом (беглое прочтение). Для определения значения незнакомых слов – консультации со специалистами или привлечение справочной литературы;
4) Формирование первой гипотезы о макроструктуре текста;
5) Внимательное прочтение текста с выписыванием ключевых слов и выражений, т.е. компрессия текста;
6) Определение связей между ключевыми словами, разработка макроструктуры текста в форме графа или «сжатого» текста;
7) Формирование поля знаний на основе макроструктуры текста.

Лекция № 9 ***Тема: Методология структурирования знаний.***

Pz – поле знаний является некоторой семиотической моделью, которая может быть представлена как граф, рисунок, таблица, диаграмма, формула или текст в зависимости от вкуса инженера по знаниям и особенностей предметной области.

История: к XVII веку сложились два подхода к разработке универсальных языков: языки классификации и логико-конструктивные языки. К первому примыкают проекты, восходящие к идее Ф. Бэкона, - это языки Вилкинса и Далгарно. Второй подход связан с исследованиями в рамках поиска универсального метода познания, наиболее чётко высказанного Р. Декартом, а затем в проекте универсальной характеристики Г. Лейбница. Именно Лейбниц

наметил основные контуры учения о символах, которые в соответствии с его замыслами в XVIII веке развивал Г. Ламберт, который дал имя науке «семиотика».)

Языки семиотического моделирования как естественное развитие языков ситуационного управления называется первым приближением к языку инженерии знаний. Именно изменчивость и условность знаков делают семиотическую модель применимой к сложным сферам реальной человеческой деятельности. Поэтому главное на стадии концептуализации – сохранение естественной структуры поля знаний, а не выразительные возможности языка.

Традиционно семиотику подразделяют на синтаксис (отношения между знаками), семантику (отношения между знаками и реальностью), прагматику (отношения между знаками и их пользователями).

I.Синтаксис.

Обобщённая структура поля знаний может быть представлена:

$Rz = < X, Y, M >$, где (1)

X – структура исходных данных, которая подлежит интеграции и обработке средствами ЭС;

Y – структура поля знаний, которую выдаёт система;

M – операционная модель предметной области, на основании которой происходит превращение X в Y.

Форма и состав X и Y существенно влияют на модель M и в неявном виде всегда присутствуют в модели репрезентации предметной области в памяти эксперта. Операционная модель предметной области M складывается из двух составляющих:

$M = <Z,G, >$, где (2)

G – структурированное описание реальной предметной области с её объектами A и отношениями между ними R_A.

Z – знания эксперта о методах принятия решения в данной предметной области, в которую входят некоторые понятия, обобщающие объекты

отдельных классов по их специфическим признакам (или концепты) B, отношения между ними R_B, а также стратегии манипулирования или для нахождения решения S. G и часть Z образуют фактическую составляющую Rz.

Таким образом $G = <A, R_A>$ (3)

$Z = <B, R_B, S>$ (4)

Подставив 3 и 4 в 2, получим структуру операционной модели

$M = <A, R_A, B, R_B, S>$, (5)

где

A – (объекты) люди, машины, звёзды и т. д.;

R_A – начальник, подчинённый, часть-целое, причина-следствие, далеко-близко и т.д.;

B – (понятия) друг, враг, польза, вред, болезнь и т.д.;

R_B – косвенно-явно, способствует-неспособствует, подходит-неподходит;

S – концепции принятия решений S.

Компоненты M можно сгруппировать следующим образом:

$M = <(A,B), (R_A, R_B), S>$, (6)

где

(A,B) - *A* – потенциально возможные элементы предметной области;

(R_A, R_B) – R – множество связей.

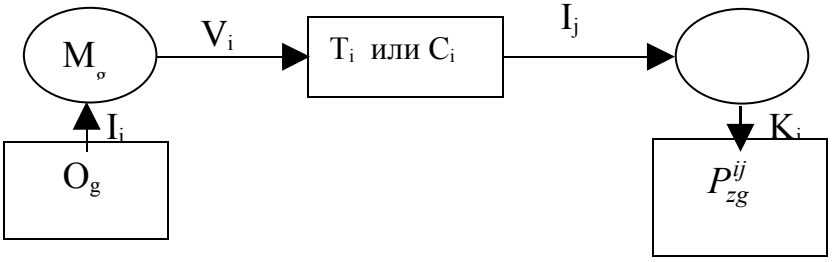
Динамику и активность полю знаний придаёт компонент S, который порождает действие над элементами A и R, осуществляющие преобразование условий X в результат Y.

В процессе работы ЭС происходит перебор структур A = {A,B} с возможным порождением новых элементов A’.

II. Семантика.

Семантику поля знаний можно рассматривать на двух уровнях. На первом уровне P_{Zg}^{i} есть семантическая модель знаний эксперта i о некоторой предметной области g. На втором уровне любое поле знаний является моделью некоторых знаний, и следовательно, можно говорить о смысле его как некоторого зеркала действительности.

Схема образования поля знаний. (схема отношений между реальной действительностью и полем знаний).



эксперт i инженер по знаниям j

1) I_i – это восприятие и интерпретация действительности O_g предметной области g i-м экспертом. В результате I_i в памяти эксперта образуется модель M_{gi} как семантическая репрезентация действительности и его личного опыта по работе с ней.
2) V_i – вербализация опыта i-го эксперта, когда он пытается объяснить свои рассуждения S_i и передать свои знания Z_i инженеру по знаниям. В результате образуется либо текст T_i, либо речевое соотношение C_i.
3) I_j – восприятие и интерпретация сообщений T_i или C_i j-м инженером по знаниям. В результате образуется модель мира M_{gj}.
4) K_j – кодирование и вербализация M_{gj} в форме поля знаний P_{Zg}^{ij}.

Пример: Два человека прибегают на вокзал за две минуты до отхода поезда. В кассы очередь. В автоматических кассах свободно, но не у того не у другого нет мелочи. Следующий поезд через 40 мин. Оба опаздывают на важную встречу.

Интерпретация 1-го: нельзя приходить на вокзал менее чем за 10 мин.

Интерпретация 2-го: надо всегда иметь мелочь в кармане.

Вербализация 1-го: опоздал, т.к. не расчитал время.

Вербализация 2-го: опоздал, т.к. на вокзале неразбериха.

III Прагграматика.

Под прагматикой будем понимать практические аспекты разработки и использования поля, т.е. как от черновиков извлечения знаний перейти к модели.

<u>стадии структурирования</u>

При разработке поля знаний существует три главных вопроса:

- что делать в данный момент;
- как реализовать то, что хочешь делать;
- почему именно это надо делать.

Существует последовательность стадий проведения концептуального анализа знаний, отвечающая на первый вопрос:

1. определение входных и выходных данных;
2. составление словаря терминов;
3. выявление объектов, понятий и их атрибутов;
4. выявление связей между понятиями;
5. выделение метапонятий и детализация понятий;
6. построение пирамиды знаний;
7. определение отношений;
8. определение стратегии принятия решений.

методы структурирования

1. Методы выявления объектов, понятий и их атрибутов.

Понятие или концепт – это обобщение предметов некоторого класса по их специфическим признакам.

На этом этапе определяются также интенсионалы (очерчивают понятие через взаимосвязь значимых признаков) и экстенсионалы (очерчивают понятия через перечисление конкретных экземпляров объекта).

Все методы выявления понятий делятся на:

1) традиционные, основанные на математическом аппарате распознавания образов и классификации;
2) нетрадиционные, основанные на методологии инженерии знаний.

2. Методы выявления связей между понятиями.

Основной упор в существующих моделях делается на понятия, а связи вводят весьма примитивные (в основном причинно-следственные).

В последнее время всё больше внимания уделяется взаимосвязанности структур знаний. Для этого было введено понятие сценария как некоторой структуры представления знаний. Основу сценария составляет КОП (концептуальная организация памяти) и мета-КОПы – некоторые обобщающие структуры.

Сценарии делятся на фрагменты и сцены. Связи между фрагментами - временные или пространственные, внутри фрагмента – самые различные: ситуативные, ассоциативные, функциональные и т.д.

Все методы выявления таких связей можно разделить на 2 группы:

1) формальные;
2) неформальные.

Неформальные методы выделения связей придумывает инженер по знаниям для того, чтобы вынудить эксперта указать явные и неявные связи между понятиями.

К неформальным методам относятся:

1) метод «сортировки карточек» в группы;
2) метод построения замкнутых кривых, когда эксперта просят обвести замкнутой кривой связанные друг с другом понятия.

После того, как определены связи между понятиями, все понятия как бы распадаются на группы.

3. Методы выделения мета-понятий и детализация понятий.

Процесс образования мета-понятий, т.е. интерпретации групп понятий, полученных на предыдущей стадии, как и обратная процедура-детализация (разукрупнение) понятий – не поддающиеся формализации операции. Это не всегда удаётся: например в системе АВТАНТЕСТ (в которой были заложены знания о тесте, ставшем классическим в психологической диагностике свойств личности) при образовании мета-понятий, полученных методами кластерного анализа, интерпретация заняла несколько месяцев и не может считаться

удовлетворительной. Это связано с тем, что формальные методы выделяют «искусственные» концепты.

4. Пирамида знаний.

Знания консультанта $И_3$

Знания врача $И_2$

Знания медсестры $И_1$

Стратегии принятия решений на различных уровнях будут существенно отличаться.

Методы построения пирамиды знаний обязательно включают использования наглядного материала

5. Методы определения отношений.

Существует более 200 базовых видов различных отношений, существующих между понятиями.

Помимо универсальных отношений (пространственных, временных, причинно-следственных) существуют ещё и специфические отношения, присущие той или иной предметной области.

<u>психосемантика и методы многомерного шкалирования</u>

Психосемантика наследует структуры сознания через реконструкцию индивидуальной системы знаний. Основная значимость моментов психосемантики состоит в том, что они позволяют выявлять те категориальные структуры сознания эксперта, которые могут не осознаваться ими самими.Основным методом экспериментальной психосемантики является метод реконструкции субъективных семантических пространств. Здесь психосемантика сплетается с лингвистической семантикой – с методологией выявления значений слов, с лексикографией и структурными исследованиями. Но лингвистические методы в основном направлены на анализ текстов, отчуждённых от субъекта, от его мотивов и замыслов. Психолингвистические методы обращаются непосредственно к испытуемому. Большинство из них

связано с различными формами субъективного шкалирования. В этом случае исследователь получает численно представленные стандартизированные данные, легко поддающиеся статистической обработке.

Под семантическим пространством понимается система признаков, описаний предметной области, определенным образом структурированная. В основе построения семантических процессов лежит семантическая процедура (факторный анализ, многомерное шкалирование, кластерный анализ) позволяющая группировать ряд отдельных признаков описания в более ёмкие категории – факторы. Говоря на языке поля знаний, это – построение концептов более высокого уровня абстракции. При геометрической интерпретации семантического пространства значение отдельного признака отображается как точка или вектор с заданными координатами внутри n-мерного пространства, координатами которого выступают выделенные факторы.

Построение семантического пространства включает переход к описанию предметной области на более высоком уровне абстракции, т.е. переход от языка, содержащий больший алфавит признаков описания, к более ёмкому языку концептуализации.

В зависимости от опыта и профессиональной компетентности испытуемых размерность пространства и расположение в ней первичных понятий может существенно варьироваться. Эта особенность семантических пространств может быть использована на стадии контроля в процессах обучения, при тестировании экспертов и пользователей.

На основании получаемых методами психосемантики моделей можно проводить контроль знаний. Контроль структуры знаний проводится на основе сопоставления семантических пространств хороших специалистов и новичков. Уровень знаний определяет степень согласованности семантических пространств.

Построение семантического пространства включает:

1) выбор и применение соответственного метода оценки семантического сходства. Этот шаг включает в себя эксперимент с испытуемым, которым

предлагается оценить общность предъявляемых стимульных признаков на некоторой шкале.

2) построение структуры семантического пространства на основе математического анализа полученной матрицы сходства. При этом происходит уменьшение числа исследуемых понятий за счёт обобщения и получения генерализованных осей.
3) идентификацию, интерпретацию выделенных факторных структур, кластеров, осей. На этом шаге необходимо найти смысловые эквиваленты, языковые «ярлыки» для выделенных структур. Часто к интерпретации привлекают группу экспертов.

7. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

ЛАБОРАТОРНАЯ РАБОТА № 1.
ИЗУЧЕНИЕ РАБОТЫ С ИНТЕГРИРОВАННОЙ ОБОЛОЧКОЙ СИСТЕМЫ ТУРБО ПРОЛОГ.

Общие сведения

1. Турбо-Пролог, версия 2.0

Система Турбо Пролог версия 2.0 может работать на ПЭВМ, совместимых с IBM PC XT/AT и PS/2, с ОЗУ минимум 384 Кбайт и двумя НГМД по 360 Кбайт. Рекомендуется иметь ОЗУ 512/640 Кбайт и НМД типа "Винчестер". В файле config.sys должно быть указано files=20 buffers=40.

Программа на Турбо Прологе состоит из следующих в определенном порядке секций и имеет следующую структуру [2]:

constants /* Секция объявления констант. Может отсутствовать */

domains /* Секция объявления нестандартных и/или составных типов данных. Может отсутствовать */

database - имя_ВБД /* Необязательная секция объявления предикатов для работы с внутренней базой данных (ВБД) */

predicates /* Секция объявления предикатов */

clauses /* Секция объявления правил и фактов */

goal /* Секция объявления внутренней цели. Может отсутствовать */

При составлении программы на Прологе необходимо соблюдать следующие ограничения:

- комментарии в программе могут располагаться в программе на любом месте. Комментарий начинается либо с символа % либо с последовательности символов /* и заканчиваться */;

- в программе может использоваться только один раз секция GOAL;

- все предикаты в CLAUSES с одинаковыми именами должны записываться подряд;

- большинство стандартных предикатов выполняют несколько функций в зависимости от состояния параметров, входящих в предикат. Известные параметры называют входными (INPUT – (i)), неизвестные – выходными (OUTPUT – (o)). Совокупность входных параметров определяет работу предиката. Эта совокупность называется проточным шаблоном.

1.1. Интегрированная оболочка системы Турбо-Пролог предоставляет следующие возможности:

- создавать и редактировать тексты программ;

- выполнять и отлаживать программы;

- транслировать программы в объектные файлы;

- компоновать объектные файлы в исполняемые модули;

- получать справочную информацию, изменять размеры окон и их цвет;

- устанавливать параметры и конфигурацию системы.

При первоначальном входе в интегрированную среду Турбо-Пролога на экране монитора появляется главное меню (рис.1). В верхней строке находятся названия 6 основных режимов работы системы. Текущее положение в меню отмечено выделяющейся по цвету и яркости прямоугольной полоской. Перемещая эту полоску (курсор) с помощью клавиш с горизонтальными стрелками нажатием клавиши Enter можно выбрать необходимый режим. Это можно сделать также одновременным нажатием клавиши Alt и первой буквы

названия соответствующего меню, например, для выбора режима редактирования достаточно нажать Alt-E.

Для удобства работы для наиболее часто используемых операций в оболочке Турбо Пролога вместо выбора из меню (или подменю) можно использовать нажатие функциональных клавиш, либо определенного сочетания клавиш (Hot keys). Действие той или иной функциональной клавиши может быть различным в зависимости от того, в каком режиме находится система. Более полную подсказку можно получить нажатием клавиш Alt-H.

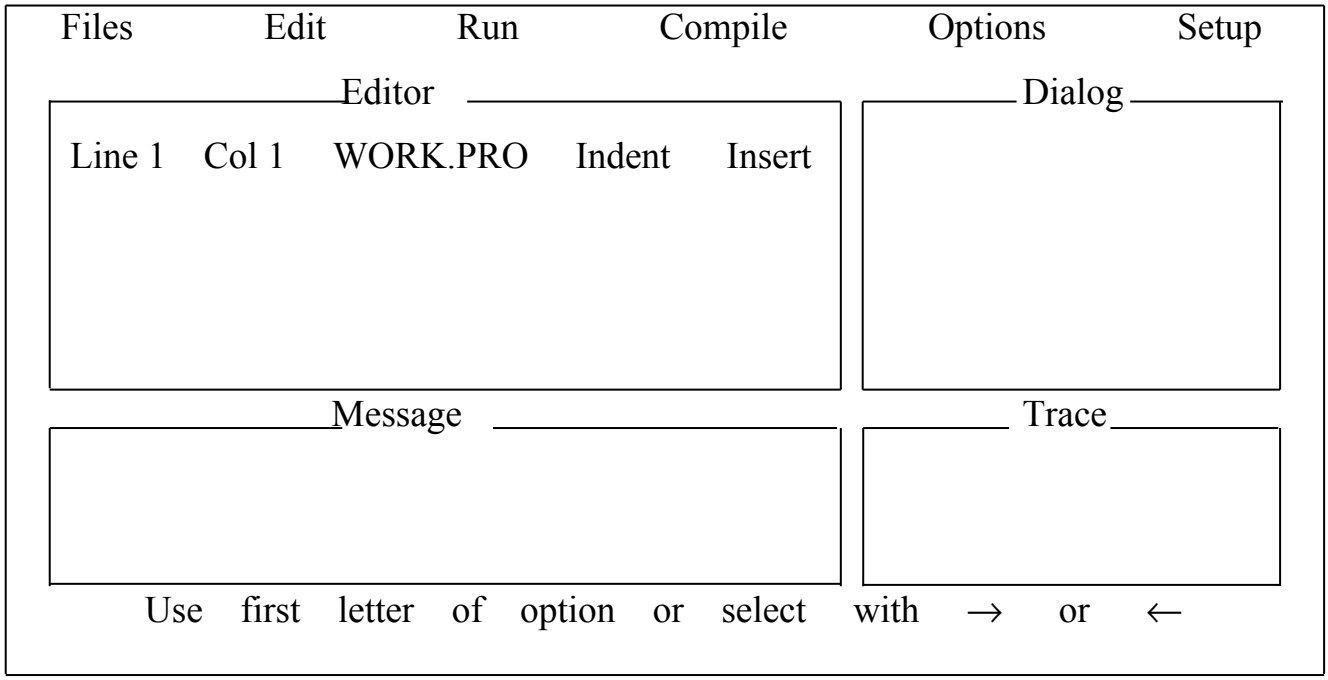


Рис. 1

Экран разделен на 4 окна:

- окно редактора текстов, в которое загружаются отлаживаемые или редактируемые программы;
- окно диалога является по умолчанию окном, если в программе не назначены другие, ввода и вывода из программы;
- окно для вывода сообщений системы;
- окно сообщений отладчика (трассировки).

Первоначальное разделение экрана может быть изменено в режиме Setup. Окно, в котором находится курсор, называется текущим окном.

Нижняя строка является строкой подсказки или иначе навигационной строкой. Навигационную строку вверху, показывающую местоположение курсора (Line - номер текущей строки, Col - позиция курсора в строке), название редактируемого файла и режимы редактирования, имеет также окно редактирования. Если имя редактируемого фала не задано, то по умолчанию он называется WORK.PRO. Если файл с таким именем находится в текущей директории, то он автоматически загружается в окно редактирования.

Все режимы главного меню, кроме Edit и Run, содержат дополнительные подменю. Выход из любого подменю осуществляется нажатием клавиши Esc.

1.2. Цикл разработки программы

Турбо Пролог поддерживает в широком смысле структурное программирование и сопровождение проекта. Цикл разработки программы (без постановочной части) можно представить следующей схемой:

1. С помощью встроенного редактора текстов исходный текст программы вводится в систему.

2. Периодически, по мере ввода новых предикатов, программа транслируется для выявления синтаксических ошибок, которые тут же устраняются.

3. С помощью встроенных средств трассировки производится отладка каждого нового предиката.

4. Периодически производится структуризация программы. Независимые части программы выделяются в отдельные модули.

5. Процесс повторяется до получения программы, удовлетворяющей внешним спецификациям, которые, в частности, сами могут быть написаны на Турбо Прологе.

2. Основные режимы работы

Основные режимы работы [2] заданы главным меню в верхней строке экрана. Такими режимами являются:

Files - работа с файлами и взаимодействие с MS-DOS

Edit - ввод и редактирование программы

Run - выполнение и отладка программы

Compile - трансляция и компоновка программы

Options - установка режимов компиляции

Setup - установка режимов работы оболочки

2.1. Работа с файлами и взаимодействие с MS-DOS

При выборе режима Files в верхней части экрана под словом Files появится подменю (рис. 2.).

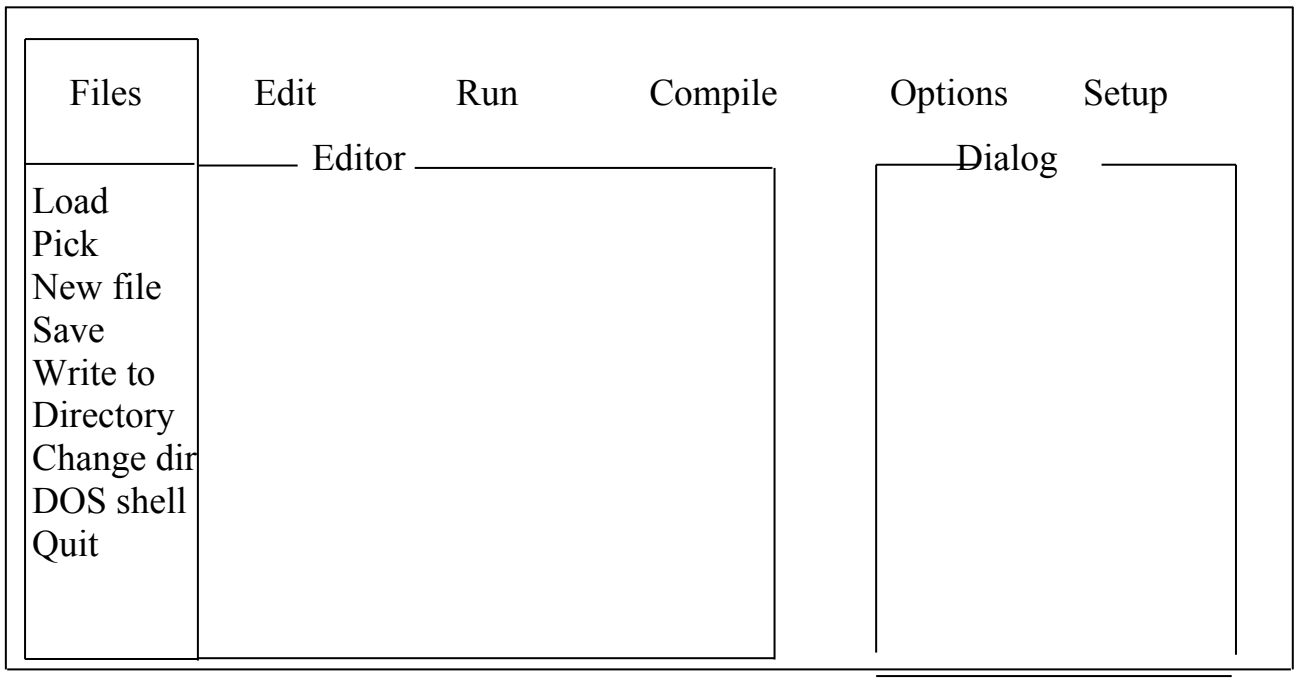


Рис. 2

Перемещение по подменю осуществляется также с помощью клавиш со стрелками, вводом первой (заглавной) буквы слова. Для быстрого выполнения наиболее частых операций имеются зарезервированные функциональные клавиши или одновременное нажатие нескольких клавиш (Hot keys).

Load - позволяет загрузить исходный текст программы на Прологе в окно редактора для его последующей модификации, выполнения или отладки.

Pick - позволяет работать из оболочки Турбо-Пролога одновременно с

8-ю файлами.

New file - очистка окна редактора текстов для ввода нового файла.

Save - сохранение редактируемой программы в файле на диске.

Write to - запись редактируемой программы в заданный файл на диске.

Directory - вывод содержимого указанной директории на экран.

Change dir - смена текущей директории. Равносильна команде ChDir в MS DOS.

OS shell - выполнение команды операционной системы. Выход из системы Турбо-Пролог в MS-DOS с возвратом по команде EXIT.

Quit - окончание работы. Выход из системы Турбо-Пролог в MS-DOS.

При работе с файлами приняты следующие соглашения об их типах:

.PRO - файл с исходным текстом программы

.BAK - файл сохранения после редактирования

.PRJ - файл с именами модулей, входящих в проект разрабатываемой системы

.EXE - исполняемый файл после компоновки

.OBJ - объектный файл после трансляции

.ARC - упакованный файл архивации

2.2. Ввод и редактирование программы

Встроенный редактор интегрированной оболочки Турбо-Пролога имеет набор команд, который является подмножеством команд широко известного редактора текстов WordStar. Кроме того, при редактировании можно пользоваться более короткими командами, пользуясь функциональными и управляющими клавишами.

Все команды редактирования можно переопределить, однако учитывая, что они совпадают с оболочками языков Турбо Паскаль, Турбо Си и Турбо Бейсик, это не рекомендуется делать.

Как и при работе с обычным текстовым редактором, ввод может происходить в режиме вставки текста или в режиме наложения его на уже существующий. Переход с одного режима в другой осуществляется нажатием клавиши Ins.

В случае слияния файлов, копирования блока из одного файла в другой и т.п. в системе Турбо Пролог может быть открыто дополнительное окно редактирования (Aux Editor) в нижнем правом углу экрана.

Команды редактора [2] условно делятся на команды перемещения, команды удаления текста, команды работы с блоками текста и вспомогательные. Выход из редактора в главное меню - F10 или по одновременному нажатию клавиши Alt и клавиши с первой буквой имени соответствующего режима.

Команды перемещения

Направление перемещения	Клавиши
Курсор вправо на один символ	Стрелка вправо или Ctrl-D
Курсор влево на один символ	Стрелка влево или Ctrl-S
Курсор вверх на одну строку	Стрелка вверх или Ctrl-E
Курсор вниз на одну строку	Стрелка вниз или Ctrl-X
Курсор вправо на одно слово	Ctrl-Стрелка вправо
Курсор влево на одно слово	Ctrl-Стрелка влево
Курсор в конец строки	End
Курсор в начало строки	Home
Курсор в начало окна	Ctrl-Home
Курсор в конец окна	Ctrl-End
Курсор вверх на один экран	PgUp или Ctrl-R
Курсор вниз на один экран	PgDn или Ctrl-C
Курсор в начало файла	Ctrl-PgUp или Ctrl-Q R
Курсор в конец файла	Ctrl-PgDn или Ctrl-Q C

Команды удаления текста

Название команды	Клавиши
Удаление символа над курсором	Del
Удаление символа слева от курсора	BackSpace или Ctrl-H
Удаление слова над курсором	Ctrl-T
Удаление строки над курсором	Ctrl-Y
Удаление начала строки до курсора	Ctrl-Q T

Удаление от курсора до конца строки	Ctrl-Q Y

Если при редактировании программы возникли какие-либо трудности, то нажатие клавиши F1 вызовет следующее меню системы подсказки:

Edit Help - [Помощь при редактировании] Заголовок меню

Show help file - вывод на экран файла справочника по всей системе. Нажатие F5 позволяет увеличить окно со справочником до размеров экрана. Повторное нажатие уменьшит его до прежних размеров. Выход из справочника по Esc.

Cursor movement - вывод на экран справочника по командам для перемещения курсора.

Insert & Delete - справочник по операциям вставки и удаления текста.

Block Functions - справочник по операциям над блоками текста.

WordStar-like - справочник по командам редактирования, аналогичным командам редактора текстов WordStar.

Miscellaneous - справочник по командам редактирования не входящим в какую-либо группу.

Global functions - справочник по командам, выполняющим операции над всем текстом.

Hot keys - справочник по клавишам для быстрого выполнения наиболее часто встречающихся операций. При этом выбранная с помощью клавиш со стрелками операция может быть тут же выполнена.

2.3. Выполнение и отладка программ

Режим Run обеспечивает компиляцию находящейся в окне редактора текстов программы и ее выполнение. Этот режим обеспечивает быструю интерактивную отладку программы, так как при обнаружении синтаксических и семантических ошибок имеется возможность их тут же исправить. Мощным инструментом отладки программ в системе являются встроенные средства трассировки.

Возможны следующие виды трассировки:

- пошаговая трассировка всей программы

- трассировка заданных предикатов

- трассировка в режиме оптимизации

- интерактивное включение/выключение трассировки

- интерактивный вывод результатов трассировки и выходных сообщений на печать и в дисковый файл.

Отметим, что трассировка является также важным средством изучения Турбо Пролога, так как, в случае непонимания работы какого-либо предиката, его работу можно посмотреть по шагам.

По умолчанию режим трассировки выключен (Off). Для установки/отключения режима трассировки следует выборать подменю "Директивы компилятора" меню Options (См. п. 2.5) и в нем выбрать вход Trace. На экране появится небольшое меню, содержащее три входа:

Trace - включить режим полной трассировки;

ShortTrace - включить режим сокращенной трассировки;

Off - выключить трассировку.

Если выбрать Trace (выход из всех подменю по Esc), то после запуска программы на выполнение (Run или Alt-R) в окне диалога появится подсказка Goal:. После ввода цели, в окне трассировки за словом CALL: появляется вызываемый предикат.

Нажатие клавиши F10 разрешает выполнение следующего шага программы. Одновременно в окне редактирования курсор показывает на выполняемый в текущий момент предикат.

После подсказки REDO: отображаются результаты вычисления (унификации).

При успешном поиске подстановке после слова RETURN: выводится ее результат. При неудаче выводится подсказка FAIL.

Трассировка в любой момент может быть прекращена нажатием клавиши Esc.

Нажатие Alt-T позволяет изменить режим трассировки. При этом на экране появляется следующее меню:

Status On	Запрет/разрешение отображения статуса

Trace window On Запрет/разрешение окна трассировки

Edit window On Запрет/разрешение окна редактирования

После нажатия Shift-F10 можно по желанию изменить размеры окон.

Внутри программы можно размещать передикат trace(on), который включает режим трассировки, или trace(off) - выключающий ее.

Нажатие клавиш Alt-P позволяет перенаправить вывод результатов трассировки на печать или в файл PROLOG.LOG на диске.

Режим ShortTrace отличается от Trace только тем, что компилятор производит оптимизацию программы (в режиме Trace оптимизацияне производится). Это особенно важно при отладке рекурсивных вызовов, особенно при "хвостовой рекурсии", когда рекурсивный вызов является последней подцелью в теле предиката. Этот вид рекурсивных вызовов оптимизируется Турбо Прологом и преобразуется в цикл, что экономит память при исполнении.

2.4. Трансляция и компоновка программ

Подменю режима Compile содержит следующие входы:

Memory - отладочная трансляция в память

OBJ file - трансляция в файл типа .OBJ

EXE file (auto link) - трансляция с созданием исполняемого (.EXE) файла с автоматической компоновкой модулей

Project (all modules) -трансляция всех входящих в проект модулей;

Link only - выполнение компоновки модулей в исполняемый файл.

2.5. Установка режимов компиляции

Исходный текст программы на Турбо Прологе перед выполнением проходит обязательную фазу компиляции и компоновки. Для определения параметров этих процессов служит режим Options.

Меню режима Options содержит следующие входы:

Link options (Опции компоновки)

Содержит подменю:

Map file ON/OFF - создавать или нет файл с картой компоновки.

Libraries - ввод имен пользовательских библиотек для компоновки.

Edit PRJ file (Редактирование файла с описанием проекта)

Запрашивается имя файла, в котором перечислены модули, входящие в проект. Этот файл затем используется компоновщиком.

Compiler Directive (Директивы компиляции)

Содержит подменю (названия входов даны в круглых скобках) позволяющие установить:

- распределение памяти (Memory allocation) под код, стек, тип и рекурсию. Размер этих областей устанавливается в параграфах (параграф равен 16 байтам);

- какие виды контроля выполнять во время исполнения программы (Run-time check): нажатие клавиши Breack, нарушение границ стека и целочисленное переполнение;

- допустимый уровень ошибок трансляции (Error level): ошибки недопустимы, по умолчанию (1), максимальный (2);

- выдачу предупреждения при наличии недетерминированных предикатов (Non-determ warning ON/OFF);

- предупреждение о наличии переменных, которые используются в предикате только один раз (Variable used once warning ON/OFF);

- уровень трассировки (Trace): полная, сокращенная и трассировка выключена;

- включение вывода диагностики по результатам трансляции (Diagnostics ON/OFF).

2.6. Получение справочной информации, изменение размеров окон и их цвета

Подменю режима SetUp содержит следующие входы:

Colors - изменение цвета окон при наличии цветного монитора. На экран выводится таблица цветовой гаммы, из которой для каждого окна выбирается цвет фона и цвет изображения;

Window size - изменение размеров окон. При выборе этой опции в нижней строке экрана появляется подсказка, как с помощью клавиш со стрелками изменить размер или местоположение окна.

Directories - установка директорий по умолчанию.

Miscellaneous - разные опции связанные с настройкой на адаптер видеомонитора;

Load SYS file - загрузка файла с параметрами интегрированной среды;

Save SYS file сохранение параметров настройка среды интегрированной среды в файле типа *.SYS.

3. Стандартные предикаты

3.1. Общесистемные стандартные предикаты

В этом разделе приведены предикаты [2], позволяющие использовать возможности предоставляемые операционной системой MS DOS.

- date(Год,Месяц,День) (integer, integer, integer): прототип (o,o,o) - считывает системную дату, прототип: (i,i,i) - установить дату.

Например:

date(J,M,T) - результат: J=2001,M=11,T=01

date(2001,02,26) - системная дата устанавливается на 26.02.2001.

time(Час, Минуты, Секунды, Сотые_секунды) (integer, integer, integer, integer): прототип (o,o,o,o) - связывает соответствующие параметры с текущим временем, прототип (i,i,i,i) - устанавливает системное время.

Например:

time(S,M,Sek,_) - результат: S=15,M=35,Sek=22

time(17,05,0,0) - часы будут поставлены на 17:05:00.

- system(Строка_с_командой_DOS) (string): прототип (i) - заданная строка должна быть допустимой командой DOS. Разрешаются встроенные и внешние команды DOS, такие как файлы с расширением ".BAT". Заданная команда выполняется. Предикат system не выполняется, если команда некорректна с точки зрения DOS. Если "Строка_с_командой_DOS" пустая (""), то COMMAND.COM активируется в интерактивном режиме.

Например:

system("dir A:") - выводится каталог накопителя A.

- comline(Строка) (string): прототип (o) - читает параметры, которые заданы в команде вызова скомпилированной программы на Турбо – Прологе.

Например:

TEST abc (вызов в MS - DOS) - если в программе с именем TEST стоит команда comline(X), то строка abc будет связана с X.

- beep - производит звуковой сигнал

- bios(Номер_прерывания,Регистры_входные,Регистры_выходные) (integer, reg, reg): прототип (i,i,o) - обеспечивает формирование прерывания с заданным номером. Регистры получают значения, установленные параметром "Регистры_входные". После обработки прерывания содержимое регистров связывается с параметром "Регистры_выходные". Параметры "Регистры_входные" и "Регистры_выходные" должны принадлежать домену regdom, который определяется следующим образом:

regdom = reg(AX, BX, CX, DX, SI, DI, DS, ES), где все аргументы имеют тип integer.

Например: bios($21, reg(AX,0,0,0,0,0,0,0),reg(Nr,_,_,_,_,_,_,_)) - производится чтение с текущего накопителя. Номер накопителя связывается с Nr.

3.2. Предикаты преобразования типов

- char_int(Символ,Число) (char,integer): прототип (i,o) - связывает параметр "Число" с кодом ASCII параметра "Символ"; прототип (o,i) - связывает параметр "Символ" с символом, код которого определяется параметром "Число"; прототип (i,i) - выполняется успешно, если код, определяемый параметром "Число", является ASCII – кодом символа, определяемого параметром "Символ".

Например:

char_int('A',X) - переменная X принимает значение 65.

char_int(X,66) - переменная X принимает значение 'B'.

char_int(‘A’,65) - выполняется успешно.

- str_char(Строка,Символ) (string,char): прототип: (i,o) - заданная первым параметром строка, состоящая из единственного символа, преобразуется в символ. Символ связывается со вторым параметром; прототип (o,i) - преобразуется символ в строку, состоящую из единственного символа и связывает ее с заданной переменной; прототип: (i,i) - выполняется успешно, если параметры связаны с представлениями одного и того же символа.

Например:

str_char(“A”,X) - результат X=’A’.
str_char(X,’A’) - результат X=”A”.
str_char(“A”,‘A’) - выполняется успешно.

- str_int(Строка,Целое число) (string,integer): прототип:(i,o) - преобразует строку в целое число. Число связывается со вторым параметром; прототип (o,i) - связывает с первым параметром строку, представляющую собой запись целого числа, связанного со вторым параметром; прототип(i,i) - выполняется успешно, если связанная с первым параметром строка является представлением числа, связанного со вторым параметром.

Например:

str_int(“123”,X) - результат: X=123
str_int(X,456) - результат: X=”456”
str_int(“234”,234) - выполняется успешно.

- str_real(Строка,Действительное число) (string,real): прототип:(i,o) - связывает второй параметр с действительным числом, определяемым записью числа в строке, заданной первым параметром; прототип (o,i) - связывает с первым параметром строку, представляющую собой запись действительного числа, заданного вторым параметром; прототип:(i,i) - выполняется успешно, если связанная с первым параметром строка является представлением числа, связанного со вторым параметром.

Например:

str_real(“1.23”,X) - результат: X=1.23

str_real(X,0.56) - результат: X=”0.56”

str_real(“4.567”,4.567) - выполняется успешно.

- file_str(Имя_ файла_ DOS,Строка) (string,string): прототип (i,o) - читает строку из заданного файла и связывает ее с параметром "Строка". Максимально допустимый размер строки – 64 К. Признаком конца строки является символ Ctrl – Z (десятичный код ASCII=26).

Например: file_str(“B:TEXT1”,X) - символы из файла TEXT1 на накопителе B будут прочитаны и связаны с переменной X.

- field_str(Строка, Столбец, Длина, Строка_символов) (integer, integer, integer, string): прототип (i,i,i,i) - записывает строку, связанную с параметром "Строка_символов", в поле, определяемое длинной и номерами строки и столбца. Если строка длиннее, чем заданное поле, то записывается только начало строки. Если строка короче, то оставшиеся позиции поля заполняются пробелами; прототип (i,i,i,o) - строка, определяемая длиной и позицией, связывается с параметром "Строка_символов". Проследите, чтобы поле в текущем окне соответствовало параметрам.

Например:

field_str(15,5,5,"hollo") - строка "hollo" записывается в поле, начинающееся с позиции (15,5).

field_str(10,30,5,X) - строка длиной 5, начинающаяся с позиции (10,30), связывается с переменной X.

- str_len(Строка,Длина) (string,integer): прототип (i,o) - с параметром "Длина" связывается количество символов в заданной строке; прототип (i,i) - выполняется успешно, если строка имеет заданную длину.

Например:

str_len("hollo", X) - результат: X=5.

str_len("book", 4) - выполняется успешно.

- isname(Строка) (string) прототип (i) - выполняется успешно, если последовательность символов, связанная с параметром, представляет собой имя, допустимое в Турбо – Прологе.

Например:

isname ("abcd") - выполняется успешно.

- upper_lower(Строка1,Строка2) (string, string): прототип (i,i) - выполняется успешно, если с первым и вторым параметром связаны идентичные строки, представленные соответственно прописными и строчными буквами; прототип (i,o) - связывает со вторым параметром строку, полученную из строки, связанной с первым параметром, заменой прописных букв на строчные; прототип (o,i) - связывает с первым параметром строку, полученную из строки, связанной со вторым параметром, заменой строчных букв на прописные.

Например:

upper_lower("A","a") - выполняется успешно.

upper_lower("ZDF",X) - результат: X="zdf"

upper_lower(X,"house") - результат: X="HOUSE"

3.2. Арифметические операции

+	сложение
-	вычитание
*	умножение
/	деление
mod	абсолютная величина
div	целочисленное деление

3.3. Операторы отношений

Операторы отношений являются инфиксными операторами (т.е. должны находится между двумя сравниваемыми величинами). Свободные переменные в операторах отношений не допускаются. Для операторов отношений приняты следующие обозначения:

< меньше; > больше; = равно; <= меньше или равно; >= больше равно; <> не равно.

3.4. Математические функции

Функция	**Описание**
abs(X) /* (Var) (i) */	Возвращает абсолютное значение X
round(X) /* (Var) (i) */	Возвращает округленное целое значение X
sqrt(X) /* (Var) (i) */	Возвращает квадратный корень из X
trunc(X) /* (Var) (i) */	Возвращает целое значение X отбрасывая дробную часть
exp(X) /* (Var) (i) */	Возвращает значение e в степени X
log(X) /* (Var) (i) */	Возвращает десятичный логарифм X
ln(X) /* (Var) (i) */	Возвращает натуральный логарифм X
arctan(X) /* (Radians) (i) */	Возвращает арктангенс X
cos(X) /* (Radians) (i) */	Возвращает косинус X
sin(X) /* (Radians) (i) */	Возвращает синус X
tan(X) /* (Radians) (i) */	Возвращает тангенс X

Все тригонометрические функции требуют, чтобы аргумент X задавался в радианах.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 1

Тема: Создание отношений в Прологе.

Последовательность действий

1. Изучить пояснения к работе.

2. Загрузить Турбо Пролог. На экране появится главное меню и окна, изображенные на рис. 1.

3. Нажимая клавишу с левой или правой горизонтальной стрелкой последовательно пройти по главному меню. Обратите внимание на появление подменю по мере перехода от одного режима к другому. Войти в подменю Files. С помощью клавиш с вертикальными стрелками переместить

прямоугольную полоску на слово Directory. Нажать клавишу Enter (/вв/). На экран будет выведено содержание текущей директории.

4. Сделать текущим окно редактирования. Вернуться в главное меню нажатием клавиши F10.

5. Войдите в режим редактирования (Alt-E), изучите действие клавиши F1 (меню подсказок) и введите следующие определения:

```
domains                    /вв/
  fakt = symbol            /вв/
predicates                 /вв/
  male(fakt)               /вв/
  female(fakt)             /вв/
  mother(fakt,fakt)        /вв/
  father(fakt,fakt)        /вв/
  wife(fakt,fakt)          /вв/
clauses                          /вв/
  male("саша").                  /вв/
  female("марина").              /вв/
  mother("наташа","саша").       /вв/
  father("петя","коля").         /вв/
  wife("наташа","петя").         /вв/
  ……
```

6. Введите в секцию clauses 5-7 фактов для предикатов male, female, mother, father, wife.

7. Пользуясь средствами Турбо Пролога постройте предикаты для выражения следующих связей между объектами:

```
son(X,Y)              /вв/
daughter(X,Y)         /вв/
brother(X,Y)          /вв/
grandmother(X,Y       /вв/
grandfather (X,Y)     /вв/
```

cusins(X,Y) /вв/

uncle(X,Y) /вв/

aunt(X,Y) /вв/

Например:

parent(X,Y):- father(X,Y);mother(X,Y). /вв/

husbend(X,Y):-wife(Y,X). /вв/

8. После ввода каждого нового предиката, программу следует откомпилировать, для чего нажать клавишу F9. Если при компиляции будут обнаружены синтаксические ошибки, то их следует исправить и добиться безошибочной компиляции (см. п. 2.2).

9. Если компиляция прошла успешно, перейти в режим исполнения: Run в главном меню или нажатие клавиш Alt-R.

10. В режиме исполнения каждый новый введенный вами предикат проверьте задавая в окне диалога после подсказки "Goal:" внешнюю цель с этим предикатом.

Например:

son(X,Y) /вв/

после ввода соответствующего предиката.

11. Включить режим трассировки и просмотреть выполнение предиката по шагам (см. п. 2.3), нажимая клавишу F10 для выполнения каждого следующего шага. В окне трассировки будет показано связывание переменных и результат вычисления предиката.

12. Используя меню Setup, изменить размеры окон редактирования и трассировки.

13. По окончанию работы выйдите из системы выбрав Quit в меню Files или нажав клавиши Alt-X.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 2

Тема: Использование рекурсии.

Приведенная ниже база данных «путешествие» содержит факты, каждый из которых имеет по четыре аргумента. Каждый факт устанавливает, что можно совершить путешествие на транспортных средствах некоторой компании (аргумент 1) из одного города (аргумент 2 – пункт отправления) в другой город (аргумент 2 – пункт назначения) и при этом воспользоваться некоторым видом транспорта (аргумент 4).

% компания отправл. прибытие вид трансп.
путешествие(круизавиа, благовещенск, москва, самолет).
путешествие(автосервис, благовещенск, белогорск, автобус).
путешествие(ж/дсервис, благовещенск, красноярск, поезд).
путешествие(круизавиа, москва, сочи, самолет).
путешествие(авиалинии, красноярск, брест, самолет).
путешествие(ж/дсервис, тында, владивосток, поезд).

1. Правило «можно_путешествовать» устанавливает косвенную связь между городами, которая будет соблюдаться в том случае, если возможно путешествие из одного города в другой через третий – промежуточный – город. Декларативная трактовка данного правила может быть такой:

Будет возможным совершить путешествие между городами «Город А» и «Город Б», если можно добраться из города «Город А» в промежуточный пункт «Город Б» и можно добраться из города «Город Б» в «Город В».

или

путешествие из города «Город А» в «Город В» будет возможным, если либо 1) существует прямая транспортная связь между этими городами, либо 2) можно совершить путешествие из города «Город А» в некоторый промежуточный пункт «Город Б»Б а затем добраться из города «Город Б» в «Город В».

2. Правило «конкурент» гласит, что любые две транспортные компании будут конкурентами, если обе они обслуживают один и тот же маршрут. Декларативная трактовка данного правила может быть такой:

«Компания 1» будет конкурентом для компании «Компания 2», если существуют два города «Город А» и «Город Б», такие, что «Компания 1» обеспечивает перевозки между городами «Город А» и «Город Б», и «Компания 2» обеспечивает перевозки между городами «Город А» и «Город Б».

Последовательность действий

Изучить пояснения к работе.

Воспользуйтесь редактором для создания файла для работы с приведенными фактами и правилами.

Добавьте в программу 10 - 15 фактов для предиката "путешествие" и промоделируйте отношения, включая все необходимые факты и правила. Протестируйте Вашу программу, чтобы проверить, выполняет ли она то, что было задумано.

Составьте запросы, позволяющие:

a) получить сведения обо всех парах компаний - конкурентов;
b) получить сведения о возможности совершить путешествие из одного города в другой с выводом всех городов через которые прошел маршрут путешествия от исходного пункта к месту назначения.

Получить один ответ на запрос в следующем виде, например:

M=благовещенск --- самолет → москва --- поезд → смоленск

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 3

Тема: Работа со списками.

1. Создать случайным образом список состоящий из K нулей и единиц.
2. Написать предикат PL(L+,N-) – истинный тогда и только тогда, когда N – предпоследний элемент списка L, имеющего не менее двух элементов.

3 3. Определите возведение в целую степень через умножение и деление.

4 4. Вставить подсписок в определенное место списка.

5 5. Удалить все заданные элементы из списка.

6 6. Сложить два списка.

7. Напишите предикат subst(+V, +X, +Y, -L) – истинный тогда и только тогда, когда список L получается после взаимной замены X на Y, т.е. X → Y, Y → X.

8. Напишите предикат, который определяет, является ли данное натуральное число простым.

9. Напишите предикат p(+N, +K, -L) – истинный тогда и только тогда, когда L – список всех последовательностей (списков) длины K из чисел 1, 2, …, N.

10. Напишите предикат p(+N, -L) – истинный тогда и только тогда, когда список L содержит все последовательности (списки) из N нулей и единиц, в которых никакая цифра не повторяется три раза подряд (нет куска вида XXX).

ЛАБОРАТОРНАЯ РАБОТА № 2.

РАБОТА С ВНУТРЕННЕЙ И ВНЕШНЕЙ БАЗАМИ ДАННЫХ СИСТЕМЫ ТУРБО ПРОЛОГ

1. Концепция работы с файлами

Турбо Пролог поддерживает работы с файлами, а также работу с внутренней и внешней базами данных. Одновременно в системе можно производить операции с двумя файлами (устройствами): одним входным и одним выходным. Файлы объявляются в секции domains следующим образом:

```
domains
 file = name _file
```

Например:

```
domains
file = textfile    или   file = my_file
```

В системе имеются предопределенные файлы с именами: keyboard (клавиатура), printer (устройство печати), com1 (коммуникационный порт), screen (экран), stdin (стандартное устройство ввода), stdout (стандартное устройство вывода), stderr (стандартное устройство для сообщений об

ошибках). По умолчанию stdin и stdout открыты и назначены соответственно на keyboad и screen.

Набор предикатов для ввода ориентирован на простые типы данных и включает в себя:

1) предикаты для работы с клавиатуры:

- inkey(Символ) (char): прототип (o) - читает символ со стандартного устройства ввода, если он доступен, иначе inkey считается невычисленным (fail).

Например:

```
clauses
  ready(X):-inkey(Y),X=Y,!.
```

При трассировке предикатов с inkey рекомендуется с помощью клавиш Alt-T установить Edit window в режим Off.

- keypressed - проверяет была ли нажата какая-либо клавиша на клавиатуре, если не была, то fail.

2) предикаты ввода данных:

- readln(Строка) (string), (o) - читает строку с текущего устройства ввода и связывает ее с заданной переменной.

Например:

```
readln(S)
```

- readterm(Область,Терм) (Name,Variable), (i,o) - читает объект, который был записан предикатом write. С помощью readterm осуществляется доступ к фактам в файле.

Например:

```
domains
  person=p(name,surname,height)
clauses
  readterm(person,p(N,S,H))
```

Файл, открытый при помощи предикатов openread и readdevice, содержит:

```
p("Seep","Maier",195)
```

Соответственно: N = Seep S = Maier H = 195.

- readint(Целочисленная_переменная) (integer), (o) - читает целое число с текущего устройства вывода и связывает его с заданной переменной.

- readreal(Переменная_вещественного_типа) (real), (o) - читает действительное число с текущего устройства ввода и связывает его с заданной переменной.

- readchar(Символьная_Переменная) (char), (o) - читает символ с текущего устройства ввода и связывает его с заданной переменной. В отличие от inkey устанавливает режим ожидания ввода.

- file_str(ИмяФайлаDOS,Строка) (string,string), (i,o) - читает строку из заданного файла и связывает ее с параметром "Строка". Максимально допустимый размер строки - 64 К. Признаком конца строки является символ Ctrl -Z (десятичный код ASCII = 26).

Например: file_str("B:TEXT1",X)

3) предикаты вывода данных:

- write(A1,A2,A3,…) (A1,A2,A3,… - константы или переменные), (i) - записывает заданные значения на текущее устройство вывода. Наряду с константами и переменными может использоваться также и обратный слэш. Он встречается в следующих комбинациях:

\n – выдается пробел;

\t – происходит переход к следующей позиции табуляции;

\номер – код ASCII выдаваемого символа.

Например: write(“введенное имя”,name) – на экране, который обычно является текущим устройством вывода, появляется текст: “введенное имя максим”, если переменная name связана со строкой “максим”

- nl - выводит пустую строку

- writef(Формат, A1,A2,A3,…) (i) – предикат форматного вывода данных.

В строке формат после знака процента используются следующие ключи:

% g числа с плавающей точкой в наиболее компактном формате (по умолчанию).

% e числа с плавающей точкой в экспоненциальной форме.

% f числа с плавающей точкой в формате с фиксированной точкой.

Например: writef(″%5.2″,X) – результат 2.80, если X=2.8.

При работе с дисковым файлом его сначала следует открыть с указанием типа выполняемых затем действий: чтения, записи, добавления текста в конец файла или модификации содержимого файла. Это делается с помощью следующих предикатов:

- openread(Символическое_имя_файла, Имя_файла_в_DOS) (file, string), прототип (i,i) – открывает файл для чтения.

Например: openread(td,″C:text.dat″) – файл DOS text.dat на устройстве C будет открыт для чтения под именем td.

- openwrite (Символическое_имя_файла, Имя_файла_в_DOS) (file, string), прототип (i,i) – открывает файл для записи.

Например: openwrite(users,″B:us.dat″) – файл DOS us.dat на устройстве B будет открыт для записи под именем users.

- openappend(Символическое_имя_файла, Имя_файла_в_DOS) (file, string), прототип (i,i) – открывает файл для дополнения.

Например: openappend(persons,″B:person.txt″) – файл DOS person.txt на устройстве B будет открыт для дополнения под именем persons.

- openmodify(Символическое_имя_файла, Имя_файла_в_DOS) (file, string), прототип (i,i) – открывает файл для модификации (как для чтения, так и для записи).

Например: openmodify(addr,″A:ADDRESSES″) – файл DOS ADDRESSES на устройстве A будет открыт для чтения или записи под именем addr.

Если в предикате openwrite задано имя файла уже существующего на диске, то после выполнения openwrite содержимое этого файла будет стерто. Проверить наличие на диске указанного файла, особенно если файл открывается не для записи, можно до выполнения операции его открытия с помощью предиката

- existfile(Имя_файла_в_DOS) (string), прототип (i) – выполняется успешно, если заданный файл присутствует в текущем каталоге, и завершается неудачно в противном случае.

Одновременно может быть открыто несколько файлов, но только два их них могут являться текущими файлами (устройствами) ввода и вывода, которые объявляются предикатами:

- readdevice(Имя_файла) (symbol): прототип (i) – присваивает текущему устройству вводимое заданное символическое имя файла; протопит (o) – связывает с параметром ″Имя_файла″ символическое имя файла текущего устройства ввода.

Например:

readdevice(adr) – последующие команды чтения осуществляют чтение из файла *adr.*

readdevice(X) – результат X=keyboard.

- writedevice(Символическое_имя_файла) (symbol): прототип (i) - ставит в соответствие текущему устройству вывода заданное символическое имя файла; протопит (o) – связывает с параметром имя текущего устройства вывода.

Например:

writedevice(addresses) – последующие команды записи могут использовать символическое имя файла *addresses;*

writedevice(X) – результат X= *addresses.*

По умолчанию файлы считаются текстовыми, однако установив режим работы с файлом можно работать с ним как с двоичным, однако при этом следует использовать только посимвольный ввод.

- filemode(Символическое_имя_файла, Тип Файла) (file,integer): прототип (i,i) режим: 0 – текстовый файл, 1 –двоичный файл – устанавливает тип заданного файла; прототип (i,o) – читает тип заданного файла и связывает его с параметром ″Тип файла″.

Например:

filemode(users,0) – тип файла *users* устанавливается как текстовый;

filemode(users,X) – результат X=0, если файл *users* – текстовый.

По завершению работы с файлом его следует закрыть:

- closefile(Символическое_имя_файла) (file): прототип (i) – имя файла не должно быть заключено в кавычки. Выполняется успешно, даже если файл перед этим не был открыт.

Например:

```
domains
        file = textfile
goal
openread(textfile, "goo.txt"), readdevice(textfile),
readln(Str),write(Str),closefile(textfile).
```

Файлы на диске из программы могут быть переименованы или удалены:

- renamefile(Старое_DOS_имя, Новое_DOS_имя) (string,string), (i,i)
- deletefile(Имя_файла_в_DOS) (string),(i).

К вспомогательным предикатам относятся:

- filepos(Символическое_имя_файла, Позиция_в_файле, Режим) (file,real,integer) - устанавливает указатель данного файла на заданную позицию. Режим 0 = относительно начала файла;

1 = относительно текущего положения указателя;

2 = относительно конца файла.

Например:

filepos(abc,10,0) – устанавливает указатель в файле *abc* на десятом байте.

- eof(Символическое_имя_файла) (file),(i) – выполняется успешно, если указатель текущей позиции файла указывает на конец файла, и завершается неудачно в противном случае.

- flush(Символическое_имя_файла) (file), (i) – содержимое внутреннего файлового буфера пересылается в заданный файл.

- disk(DosPath) /* (string): прототип (i) – устанавливает путь и накопитель; (o) – связывает с параметром текущий накопитель и путь.

Например:

disk(″C:\Prolog″) – на накопителе С будет установлен путь \Prolog.

disk(X) – результат X = C:\Prolog\Bin\qwer.

2. Внутренняя база данных

Турбо Пролог поддерживает работу с внутренней (ВБД) и внешней (дисковой - ДБД) базами данных, а также работу с дополнительной оперативной памятью (EMS ОЗУ).

ВБД состоит из фактов, которые добавляются/удаляются и существуют в ОЗУ только во время работы программы. Однако, их можно сохранить на диске. Для работы с ВБД Турбо Пролог использует следующие встроенные предикаты [1]:

- assert(факт) – заносит факт в базу данных перед другими фактами. В результате данный факт будет добавлен в начало базы данных. Факт должен быть термом, принадлежащим домену *dbasedom*.

- asserta(факт) - добавляет факты в ВБД перед существующими фактами.

- assertz(факт) - добавляет факты в ВБД после существующих фактов

- retract(факт) - удаление существующего факта из ВБД

- retractall(факт) - удаление существующих фактов из ВБД

- consult(имя_файла) – записывает в базу данных текстовый файл, который может быть создан в результате выполнения предиката *save*. Этот файл содержит факты, которые должны быть описаны в разделе *database*. Выполнение предиката *consult* не будет успешным, если в файле имеются синтаксические ошибки.

- save(имя_файла) - сохранение ВБД в файле или на жесткий диск.

Эти предикаты могут иметь один или два аргумента. Второй аргумент является необязательным и обозначает имя ВБД. Факты в ВБД хранятся в виде таблицы, которую легко модифицировать. В ВБД можно добавлять факты, но не правила. В фактах ВБД не может быть свободных переменных. Если ВБД не дано имя, то ей будет присвоено имя dbasedom.dba.

Пример программы с использованием внутренней базы данных:

```
domains
database
    base(integer,string,integer)
 predicates
    write_base(integer)
    ch_base(integer)
 clauses
     write_base(N):-base(N,Name,Group),
                    write("Номер ",N),nl,
                    write("Фамилия ",Name),nl,
                    write("Группа ",Group),nl,
                    write("----------------"),nl.
      ch_base(N):-write("Фамилия:"),readln(S),
                  write("Группа"),readint(S1),
                  retract(base(N,_,_)),
                  assert(base(N,S,S1)).
   goal
       consult("dd.dba"),
       write("Изменить запись.Номер-"),
       readint(N),
       write_base(N),
       ch_base(N),
       save("dd.dba").
```

3. Внешняя база данных

Внешняя база данных создается в случае, если объем данных больше объема свободной части ОЗУ или предполагается значительное расширение БД.

ДБД может быть расположена:

Встроенный атрибут	Местонахождение внешней БД
In_file	в файле на диске
In_memory	в оперативной памяти
In_ems	в EMS-памяти (расширение ОЗУ до 4 Мб.)

ДБД состоит из двух компонент: элементов данных (термы Турбо Пролога) и цепочек (chain), в которых хранятся термы. В цепочке может храниться неограниченное количество термов. ВДБД может быть любое число цепочек. Цепочка выбирается по имени. Имя цепочки – это просто строка символов. Для быстрого поиска данных цепочка может быть индексирована методом B+дерева [1].

Имена предикатов, работающих с ДБД, построены следующим образом: db_ <тип объекта данных><операция>.

Например: db_term_delete.

Если имя объекта не указано подразумевается вся база данных. Объявление имени ДБД:

domains

db_selector = имя_базы1, имя_базы2, ...

Ссылка на ДБД осуществляется по ее имени.

3. 1. Предикаты для работы со всей ДБД.

- db_create(имя_базы, имя_файла, расположение), (i,i,i) – создание новой ДБД:

Например:

db_create(db_name, "F.TXT", in_file)

db_create(db_mark, "marks", in_memory)

- db_open(имя_базы, имя_файла, расположение), (i,i,i) – открытие ранее созданной базы данных.

- db_copy(Имя_базы, имя_файла, Новое_расположение), (i,i,i) – перемещение бызы данных в другое место (например, из файла в ОЗУ или наоборот.

- db_close(Имя_базы), (i) – в конце работы ДБД должна быть закрыта.

- db_delete(Имя_базы), (i) – ДБД можно удалить.

3.2. Предикаты для работы с цепочками

Цепочки [3] ДБД аналогичны внутренней базе данных. Цепочки, в некотором смысле, можно отожествить с записями в обычной реляционной

СУБД. Они хранят данные в форме термов Пролога (т.е. в виде сложных объектов: списков, структур; или значений простых типов данных). ДБД может хранить любые допустимые в Турбо – Прологе типы термов.

Цепочки термов запоминаются последовательно и одновременно доступна только одна цепочка. Каждая цепочка имеет собственное имя. Имя цепочки нигде специально не объявляется, цепочка создается, когда ее имя впервые встретилось в предикатах записи фактов в ДБД. Поэтому в именах цепочек не следует допускать опечаток, иначе можно создать ненужную цепочку и таким образом "потерять данные".

Следующие предиката служат для работы с цепочками в ДБД:

- chain_inserta(X,W,S,T,_) – вставляет терм в начало базы;
- chain_insertz(X,W,S,T,_) – вставляет терм в конец базы;
- db_chains(X,C) – создает ссылку на цепь;
- chain_first(X,C,R) – позиционирует указатель на первый элемент цепи;
- ref_term(X,S,R,TERM) – возвращает терм на который указывает указатель;
- chain_next(X,R,NEXT) – возвращает указатель на следующий терм,

где X – селектор базы описывается в DOMAINS как DB_SELECTOR = mydba

Y – имя базы по стандарту ДОС в кавычках

Z – место расположения базы (в памяти или в виде файла)

W – описатель цепи

S – описатель структуры файла базы данных

T – терм

_ – необъязательная переменная возвращающая код ошибки если она есть

C – ссылка на описатель цепи

R – положение указателя в цепи

TERM – переменная конкретизированная термом на котором находится указатель

NEXT – переменная конкретизированная указателем на следующий терм

dbman – описатель структуры файла базы данных

4. Индексация цепочек методом B+дерева.

При добавлении термов с использованием для индексации B+дерева с каждым ссылочным числом связан ключ. Так как это число ссылается на уникальную запись (вход) БД, нахождение правильного ключа быстро приводит к искомой записи данных. B+дерево (индексная цепочка) создается с помощью предиката:

- bt_create(Имя_ДБД, Имя_B+дерева, Селектор_B+дерева, Длина_Ключа, Длина_Узла), так как информация B+дерева хранится в том же файле ДБД, что и индексируемая цепочка, то необходимо указывать "Имя_ДБД" базы, в которой находится эта цепочка. "Имя_B+дерева" - это имя, которое ему дает пользователь; оно является строкой. Третий аргумент является выходным, он относится к специальному встроенному простому типу bt_selector и используется в ряде других предикатов для идентификации созданного B+дерева. "Длина ключа" - длина самого длинного ключа по которому осуществляется поиск записи. Однако не следует злоупотреблять этим аргументом, так как хранение слишком длинных ключей в большой ДБД может потребовать значительного объема дисковой памяти.

B+дерево разбито на отдельные страницы или узлы. Каждый узел B+дерева является либо последним, либо порождает два нижележащих узла. Каждый узел содержит группу ключей из заданного диапазона. Так как никакие два узла не содержат одинаковые значения ключей, то можно быстро проверить не находится ли искомый ключ внутри диапазона конкретного узла. Если да, то быстро находится ссылочное число, если ключ меньше, то поиск ведется по левой ветви B+дерева, если больше - по правой. Аргумент "Длина_Узла" в bt_create определяет сколько ключей запоминается в каждом узле B+дерева. Для баз данных среднего размера рекомендуется "Длина_Узла" = 4.

Как и цепочки ДБД, B+деревья могут быть закрыты и снова открыты с помощью предикатов: bt_open() и bt_close().

Чтобы B+дерево было правильно сохранено, оно должно закрываться до закрытия соответствующего файла базы данных. Как и с ДБД B+дерево может быть модифицировано удалением или добавлением ключей. При индексировании ключи автоматически сохраняются системой. Для модификации используются два предиката: key_insert() и key_delete()

Для поиска по заданному ссылочному номеру следует использовать предикат key_search(), если заданный ключ не найден, возникает ситуация fail. Если существует 2 одинаковых ключа, то возвращается первый найденный. Чтобы найти следующий (предыдущий) нужно применить, соответственно, предикат key_next() или key_prev(). Полный список предикатов для работы с внитренней и внешней базами данных можно получить в процессе работы нажав клавишу F1. Они находятся в файле PROLOG.HLP.

Примеры программ для работы с ДБД:

1)
```
domains
    db_selector = mmm
predicates
    count_dba(integer)
    count(REF,INTEGER,INTEGER)
clauses
    count_dba(N):- chain_first(mmm,name,REF),count(REF,1,N).
    count(REF,N,N):-ref_term(mmm,string,REF,S),write(S).
    count(REF,N,N2):-chain_next(mmm,REF,NEXT),!,N1=N+1,
    count(NEXT,N1,N2).
goal
    db_create(mmm, "aaaa.ile", in_file),
    db_statistics(mmm,_,_,_,_),
    write("\nБаза имен.\n"),
    write("Введите номер не больше 8"),
    readint(N),
    chain_inserta(mmm,name,string,"ДМИТРИЙ",_),
```

```
        chain_insertz(mmm,name,string,"АЛЕКСЕЙ",_),
        chain_insertz(mmm,name,string,"МАКСИМ",_),
        count_dba(N),
        db_close(mmm).
```

2)
```
    domains
        db_selector = mydba
        dbman = person(firstname,age,city)
        firstname,city = srting
        age = integer
    predicates
        rd(ref)
    clauses
        rd(REF):-ref_term(mydba,dbman,REF,TERM),write(TERM),nl,fail.
        rd(REF):-chain_next(mydba,REF,NEXT),!,rd(NEXT).
        rd(_).
    goal
        clearwindow,
        db_create(mydba,"dd.bin",in_file),
        db_close(mydba),
        db_open(mydba,"dd.bin",in_file),
chain_inserta(mydba,mychain,dbman,person
                ("Унру Артур Яковлевич",21,"Комсомольск-на-Амуре"),_),
chain_insertz(mydba,mychain,dbman,person
                ("Смирнов Николай Семенович",20,"Благовещенск"),_),
        db_chains(mydba,CHAIN),
        chain_first(mydba,CHAIN,REF),
        rd(REF),readchar(_),
        db_delete("dd.bin",in_file).
```

Последовательность действий

1. Изучить пояснения к работе.

2. Загрузить Турбо Пролог.
3. Войти в режим редактирования (Alt-E) и ввести одну из предложенных в примере программ.
4. Включить режим трассировки и просмотреть выполнение предикатов по шагам (см. п. 2.3 предыдущей работы).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 1

Тема: Работа с базами данных.

1. Разработать программу с использованием файлов, ВБД, ДБД по темам:

1.1. Самолеты: наименование типа, фамилия конструктора, год выпуска, количество кресел, грузоподъемность (т).

1.2. Расчет движения: наименование воздушной линии, тип самолета, количество рейсов, налет (тыс. км.), пассажирооборот.

1.3. Перевозки: тип самолета, номер борта, количество рейсов, налет в часах, налет (тыс. км.).

1.4. Расписание: номер рейса, наименование рейса, тип самолета, стоимость билета, протяженность линии.

1.5. Сооружения аэропорта: наименование, площадь, этажность, год сооружения, стоимость (млн. руб.).

1.6. Ремонт аэродромных сооружений: наименование, шифр, вид ремонта, стоимость ремонта, наименование подрядчика.

1.7. Кассы авиабилетов: номер кассы, Ф.И.О. кассира, количество проданных билетов, суммарная выручка, дата продажи.

1.8. Характеристики персональных компьютеров: тип процессора, тактовая частота, емкость ОП (Мбайт), емкость ЖМД (Мбайт), тип монитора.

1.9. Города: наименование, количество жителей, площадь (кв. км.), год основания, количество ВУЗов).

1.10. Московские мосты: наименование, высота, ширина, количество опор, протяженность.

1.11. Линии московского метро: наименование, район линии, год пуска, протяженность (км.), количество поездов.

1.12. Легковые автомобили: марка, цвет, стоимость, изготовитель, максимальная скорость.

1.13. Продажа программных продуктов: наименование, фирма – изготовитель, стоимость (тыс. руб.), объем (Мбайт), количество на складе.

1.14. Абонентская плата за телефон: Ф.И.О. абонента, телефон, год установки, количество абонентов, плата за телефон.

1.15. Детские сады: наименование детского сада, номер сада, количество детей, район города, плата за месяц.

1.16. Сотрудники: Ф.И.О., табельный номер, дата рождения, оклад (тыс. руб.), стаж работы.

1.17. Ведомость зарплаты за текущий месяц: Ф.И.О., номер отдела, Табельный номер, количество рабочих часов, размер зарплаты.

1.18. Музеи: наименование, назначение, адрес, время работы, стоимость билета.

1.19. Экскурсии: наименование, страна, стоимость, продолжительность, транспорт.

1.20. Кинофильмы: наименование кинотеатра, стоимость билета, время сеансов, адрес мест, количество.

1.21. Книга – почтой: наименование, Ф.И.О. автора, номер по каталогу, издательство, стоимость книги.

1.22. Квартиры: адрес, площадь (кв. м.), сторона света, стоимость (1 кв. м.), этаж, количество комнат.

1.23. Склад товаров: номер магазина, наименование товара, артикул товара, цена единицы товара, количество товара.

1.24. Телевизоры на складе магазина: наименование, фирма – изготовитель, стоимость, размер экрана, количество на складе.

1.25. Холодильники на складе магазина: наименование, фирма – изготовитель, стоимость, емкость камеры, количество на складе.

2. Пользуясь средствами Турбо Пролога добавить предикаты для поиска фактов в построенной внутренней базе данных.

3.Каждый новый введенный вами предикат проверить задавая в окне диалога после подсказки Goal: внешнюю цель с этим предикатом.

ЛАБОРАТОРНАЯ РАБОТА № 3.

УНИВЕРСАЛЬНЫЙ ГРАФИЧЕСКИЙ ИНТЕРФЕЙС В ЯЗЫКЕ *ТУРБО ПРОЛОГ*.

Общие сведения

1. Режимы работы монитора

1.1. Определения

Текстовый режим [4] определяет вывод на экран, который разделен на ячейки (обычно 80 колонок на 25 строк), и может отображать только символы из кодового набора ПЭВМ. Каждая ячейка содержит атрибут и символ. Атрибут говорит о том, как выводится на экран символ (его цвет, мигание, интенсивность свечения).

Текущая позиция на экране отмечается мерцающим прямоугольником - курсором. Позиция курсора X,Y (координаты точки) задаются номером колонки и номером строки символа или точки в зависимости от режима. Началом координат является левый верхний угол экрана (1,1), при этом X увеличивается слева направо, а Y – сверху вниз.

Окно – прямоугольная область экрана (или весь экран), возможно окруженная рамкой и выделенная другим цветом или оттенком. Операции над окном производятся как над целым экраном. На экране одновременно могут находиться несколько окон, которые могут перекрывать друг друга. Окно, в котором находится курсор, называется текущим.

Для кодирования цвета окон в предикатах Турбо – Пролога приняты следующие соглашения [2]:

Цвет фона	Код	Цвет изображения	Код
черный	0	черный	0
голубой	16	голубой	1
зеленый	32	зеленый	2

сиреневый	48	сиреневый	3
красный	64	красный	4
фиолетовый	80	фиолетовый	5
коричневый	96	коричневый	6
белый	112	белый	7

Для получения яркого цвета к цвету изображения нужно добавить 8. Для получения атрибута окна нужно сложить цвет фона и изображения. Например, голубое изображение на белом фоне имеет атрибут: 112+1=113.

Графический режим [4] определяет построение изображения на экране из точек (пикселов). Размер экрана измеряется количеством точек по горизонтали (X) и по вертикали (Y) и зависит от типа видеоадаптера. Число точек по осям X,Y называется разрешением экрана. Каждая точка может быть включена или выключена, а на цветных мониторах иметь еще и цвет. Таким образом изображение на экране строится включением и окраской точек.

В графическом режиме нет курсора. Вместо него используется указатель текущей позиции экрана CP, который может быть перемещен в любое место экрана, где должен строиться элемент изображения.

При наличии видеоадаптера цветной графики (CGA, MCGA, VGA) можно установить цвет фона (экрана) и цвет изображения. Цвет фона и цвет изображения являются атрибутом окна и изображения. В процессе работы атрибут можно менять.

Турбо Пролог 2.0 поддерживает следующие видеоадаптеры [1]:

Название видеоадаптера	Драйвер
CGA - Color Graphics Adapter	CGA.BGI
EGA - Enhanced Graphics Adapter	EGAVGA.BGI
MCGA - Multi Color Graphics Adapter	
HGA - Hercules Graphics Adapter	HERC.BGI
VGA - Video Graphics Array	EGAVGA.BGI
IBM8514 - Super VGA	IBM8514.BGI
AT&T - 400-строковый, ПЭВМ Оливетти	ATT.BGI

Основными видеоадаптерами являются: CGA, EGA и VGA.

Графика CGA.

Графический адаптер CGA был первым цветным графическим адаптером на ПЭВМ фирмы IBM. CGA поддерживает один текстовый

режим (25*80) и два отдельных графических режима: с высоким (640*200) инизким (320*200) разрешением экрана. При низком разрешении палитра (набор цветов) одновременно выводимых на экран цветов состоит из 4 цветов, один из которых является цветом фона.

В режиме с высоким разрешением можно использовать только один цвет изображения на черном фоне.

Графика EGA.

EGA значительно расширяет графические возможности ПЭВМ. Как и CGA, видеоадаптер EGA может быть инициализирован для работы в двух режимах: низкого (640*200) и высокого (640*350) разрешения экрана. Оба эти режима позволяют выводить на экран одновременно палитру из 16 различных цветов.

Главным различием между этими режимами является количество предоставляемых буферных страниц. В режиме низкого разрешения в памяти платы EGA может быть сохранено четыре полных страницы графического экрана. Это позволяет пользовательской программе создавать графическое изображение в трех различных экранах во время вывода на экран четвертого. Эти три дополнительных экрана дают возможность строить изображения на невидимых экранах, затем быстро переходить от одного графического экрана к другому не ожидая, пока будет нарисовано новое изображение.

В режиме высокого разрешения EGA доступны две страницы: одна не-видимая и одна выведенная на экран.

Графика VGA.

Имеется совместимость с EGA и CGA. Максимальная палитра 256 цветов. Разрешение 640*480. Окна в графическом режиме. Как и окна текстового режима, окно в графическом режиме, именуемое в дальнейшем VP (viewport), используются для выбора конкретной части экрана для построения в нем изображения. По умолчанию VP после инициализации занимает весь экран. В отличие от окон текстового режима VP не буферизируются. Это

означает, что при перекрытии окон, информация в первом из них будет потеряна.

При записи предикатов в общем виде в строке комментарий указывается тип аргумента и вид передачи аргументов (текущий шаблон), т.е. какие аргументы являются входными (i), а какие выходными (o). Некоторые предикаты могут иметь несколько различных текущих шаблонов в зависимости от того, получают они связанные или несвязанные переменные.

1.2. Работа с окнами

Основным предикатом для создания окон является предикат для определения окна:

- makewindow(Ном_Окна, Атр_экрана, Атр_рамки, Заголовок, Строка, Столбец, Высота, Ширина,), где

Ном_Окна – логический номер, присваиваемый окну для ссылок в других предикатах. Номера 80-85 использовать не рекомендуется, так как они зарезервированы за пакетом TOOLBOX (см. раздел 3.);

Атр_экрана – атрибут цвета для внутренней области окна;

Атр_рамки – атрибут цвета для внешней области окна (рамки), 0 – если рамка отсутствует;

Заголовок – текст, помещаемый на верхней рамке окна;

Строка – номер строки верхнего левого угла окна;

Столбец – номер колонки верхнего левого угла окна;

Высота – высота окна в строках;

Ширина – ширина окна в колонках.

Например:

```
makewindow(1,7,135,"Пример окна",5,15,6,10)
```

Параметры должны соответствовать характеристикам аппаратуры, иначе выдается сообщение об ошибке. Созданное окно становится текущим окном. Экран может быть разбит на несколько окон.

Для перехода их одного окна в другое служит предикат

- shiftwindow(Ном_Окна): прототип (i) – активизирует окно с заданным номером; прототип (o) – связывает с параметром номер текущего окна.

Например:

shiftwindow(3) – активизируется окно с номером 3;

shiftwindow(X) – результат: X=1, если текущее окно имеет номер 1.

При переходе в другое окно координаты являются локальными т.е. отсчет идет от левого верхнего угла, которое имеет координаты (0,0).

- clearwindow – стирает содержимое текущего окна (очищает окно), при этом курсор перемещается в позицию (0,0);

- removewindow – удаляет текущее окно, если окна не существует, выдается ошибка времени выполнения.

Управление курсором осуществляется с помощью предиката

- cursor(Строка, Столбец): прототип (i,i) – передвигает курсор в текущем окне в позицию, заданную номерами строки и столбца. Номер строки может находится в диапазоне от 0 до 24, номер столбца – от 0 до 79. координаты левого верхнего угла экрана – (0,0); прототип (o,o) – связывает номера строки и столбца, определяющие позицию курсора, с соответствующими параметрами.

Например:

cursor(5,10) – курсор устанавливается на пятой строке в десятом столбце;

cursor(Z,S) – результат: Z=12, S=40, если курсор установлен на строке 12 в столбце 40.

- existwindow(Ном_Окна) – проверяет наличие окна с заданным логическим номером;

- setcolor(Окно_или_рамка) – позволяет пользователю интерактивно изменять цвет окна (0) или рамки (1). При этом появляется меню, аналогичное подменю Setup/Colors в интегрированной оболочке Турбо Пролога.

- attribute(Атрибут): прототип (i) – устанавливает значение атрибута, определяющее цвет фона текущего окна. Значение атрибута определяется параметром (см. табл.); прототип (o) – опрашивает атрибут текущего окна и связвает его с параметром.

Например:

attribute(88) – устанавливает розовый цвет текущего окна;

attribute(X) – если фон окна голубой, то значением переменной X будет число 24.

- scrol (Число_строк, Число_столбцов): прототип (i,i) – сдвигает содержимое текущего окна на заданное число строк и столбцов. Положительные значения аргументов задают скроллинг вниз и вправо, а отрицательные - вверх и влево.

Пример программы для работы с окнами:

```
predicates
        nondeterm repeat
goal
        makewindow(1, 1, 7, "one", 5, 0, 10, 20), write("ONE"),
        makewindow(8, 8, 7, "eight", 1, 10, 10, 20), write("EIGHT"),
        makewindow(9, 9, 7, "nine", 15, 20, 10, 20), write("NINE"),
        repeat,
        random(9,X), N=X+1, shiftwindow(9),
         shiftwindow(1),
         keypressed.
clauses
         repeat.
         repeat :- repeat.
```

2. Универсальный графический интерфейс (УГИ)

УГИ – это свыше 70 встроенных предикатов для работы с графикой. Эти предикаты позволяют работать как с отдельной точкой, так и с объектами типа линий, дуг, окружностей, многоугольников и др.

УГИ поддерживает различные виды штриховки линий, закраску поверхностей, различные виды шрифтов (фонты) и драйверы практически всех распространенных графических адаптеров ПЭВМ.

Полный список предикатов УГИ можно получить в процессе работы нажав клавишу F1. Они находятся в текстовом файле PROLOG.HLP. Чтобы применять УГИ необходимо иметь видеографический адаптер, поддерживаемый УГИ (например, нельзя применять на ЕС 1840).

2.1. Установка графического режима

Режим работы графического экрана устанавливается предикатом initgraph, который инициализирует графическую систему, загружая с диска соответствующий драйвер графического видеоустройства (см. файлы с расширением BGI) переводя его в графический режим работы. Предикат initgraph() имеет пять аргументов. Он связывает переменные NewDriver и NewMode с реальным драйвером и режимом работы, а также устанавливает значения по умолчанию всем графическим переменным (текущая позиция, цвет, палитра и т.д. в зависимости от видеоадаптера).

- initgraph(Graphdriver,Graphmode,NewDriver,NewMode,Pathtodriver) (integer,integer,integer,integer,string): прототип (i,i,o,o,i), где

Graphdriver – целое число, которое задает номер используемого драйвера, 0 означает, что система должна сама определить тип используемого в ПЭВМ видеоконтроллера и вернуть номер соответствующего ему драйвера (1 – CGA, 2 – MCGA, 3 – EGA, ..., 9 – VGA). См. также декларации в файле GRAPDECL.PRO;

Graphmode – графический режим. Символические константы для объявления режима определены в файле GRAPDECL.PRO.

Pathtodriver – путь к директории, где находится драйверы (*.BGI), пустая строка " " означает текущую директорию.

Например:

```
constants
bgi_path = " "
goal
InitGraph(G_Driver,G_Mode, _, _, bgi_Path)
```

После инициализации экрана он очищается и CP устанавливается равным координатам верхнего левого угла экрана (0,0). Иногда, если предполагается, что программа будет выполняться на различных ПЭВМ, более удобно перед выполнением предиката initgraph() выполнить предикат:

- detectgraph(Graphdriver,Graphmode) (integer,integer): прототип (o,o) – проверяет какой графический адаптер используется в системе и определяет режим его работы при наибольшем разрешении. Если графический адаптер не обнаружен, то возвращается – 2.

Например:

DetectGraph(G_Driver, G_Mode1),

Чтобы освободить память, занятую под графику, и вернуть экран в режим, используемый до initgraph, в конце работы нужно выполнить предикат: closegraph().

2.2. Перемещение по экрану

Так как характеристики видеотерминалов значительно отличаются, то с помощью следующих предикатов рекомендуется получить максимальные значения X и Y: getmaxX(X) и getmaxY(Y).

Для перемещения по плоскости изображения используются предикаты:

- getx(X) – получить текущую координату X;
- gety(Y) – получить текущую координату Y;
- moveto(X,Y) – переместить текущий указатель в точку X,Y;
- moverel(DeltaX,DeltaY) – переместить текущий указатель на DeltaX пикселов по горизонтали и на DeltaY пикселов по вертикали.

Предикат cleardevice не имеет аргументов. Он очищает графический экран перемещает указатель текущей позиции в точку (0,0).

2.3. Установка/изменение текущих значений

Предикаты установки значений:

- setlinestyle((Вид_Линии, Шаблон, Толщина) (integer,integer,integer): прототип (i,i,i) – устанавливает вид линии по умолчанию, используемый другими предикатами УГИ. Параметры: "Вид_Линии" – сплошная (0), из точек

(1), центрированная (2), пунктирная(3), задаваемая пользователем (4); "Шаблон" – 16-битовый шаблон задающий вид линии (только для "Вида_Линии"=4, иначе шаблон игнорируется): сплошная линия $FFFF, пунктирная $3333; "Толщина" – нормальная линия или утолщенная.

- setfillpattern(Список_шаблона, Цвет) (bgi_list,integer): прототип (i,i) – устанавливает определяемый пользователем шаблон – заполнитель. Шаблон в виде матрицы 8 на 8 бит кодируется в виде списка из 8 однобайтовых элементов, каждый из которых кодирует 8 бит. Если бит шаблона равен 1, то соответствующий ему пиксел будет выведен на экран.

- setfillstyle(Шаблон, Индекс_Цвета) (integer,integer): прототип (i,i) – устанавливает в качестве текущего заполнителя один из определенных в системе шаблонов, а текущий цвет заполнителя равным "Индекс_Цвета" в палитре. Константы для "Шаблона" приведены в файле GRAPDECL.PRO

- setpalette(Индекс, Реальный_Цвет) (integer,integer): прототип (i,i) – изменяет один цвет палитры, расположенный в ней под номером "Индекс", на цвет заданный параметром "Реальный_Цвет". Последний представляет собой зависящий от аппаратуры номер цвета для используемого драйвера. Индекс должен находится в диапазоне от 0 до <количество_цветов_в_текущей_палитре>. Для адаптера CGA можно изменять цвет только с индексом 0 (цвет фона). Файл GRAPDECL.PRO содержит определения констант для наиболее часто применяемых цветов.

- setallpalette(Список_Цветов) (bgi_ilist): прототип (i) – изменяет все цвета палитры в соответствии со "Списком_Цветов" (адаптеры EGA/VGA).

Текущий цвет фона и цвет изображения (из текущей палитры) можно получить/установить с помощью предикатов:

- getbkcolor(BkColor) (integer): прототип (o) – возвращает цвет фона;
- getcolor(Color) (integer): прототип (o) – возвращает цвет изображения;
- setbkcolor(Color) (integer): прототип (i) – установить цвет фона;
- setcolor(Color) (integer): прототип (i) – установить цвет изображения.

Следующие два предиката позволяют получить/изменить цвет заданной точки:

- getpixel(X,Y,Цвет) (integer,integer,integer): прототип (i,i,o);
- putpixel(X,Y,Цвет_Точки) (integer,integer,integer): прототип (i,i,i);

где (X,Y) - координаты точки (пиксела).

2.4. Построение графических объектов

В следующих предикатах углы отсчитываются против часовой стрелки, 0 градусов соответствует трем часам.

- arc(X,Y,StartAngle,EndAngle,R) (integer,integer,integer,integer,integer): прототип (i,i,i,i,i) – рисует текущим цветом дугу окружности с центром (X,Y) и радиусом R, начинающуюся с ″StartAngl″ и заканчивающуюся на ″EndAngl″;
- ellipse(X, Y, StAngle, EndAngle, Xradius, YRadius) (все аргументы целочисленные): прототип (i,i,i,i,i,i) – рисует эллипс;
- pieslice(X,Y,Нач_Угол, Кон_Угол, Радиус) (все аргументы целочисленные): прототип (i,i,i,i,i) – рисует и заполняет сектор круга от ″ Нач_Угол″ до ″Кон_Угол″ с центром в (X,Y) и заданным радиусом. Граница сектора выделяется текущим цветом изображения, а заполнение производится текущим шаблоном и цветом заполнителя;
- pieslicexy(X,Y,Нач_Угол, Кон_Угол, X_Радиус, Y_Радиус) (все аргументы целочисленные): прототип (i,i,i,i,i,i) – рисует и заполняет сектор эллипса с центром (X,Y), горизонтальным радиусом ″X_Радиус″, вертикальным – ″Y_Радиус″, от ″Нач_Угол ″до ″Кон_Угол″. Остальное как для pieslice;
- line(X0,Y0,X1,Y1) (все аргументы целочисленные): прототип (i,i,i,i) – рисует текущим цветом, стилем линии и толщиной прямую линию между двумя заданными точками: (X0,Y0) – начало; (X1,Y1) – конец. Предикат line не обновляет текущую позицию и всегда вычисляется;
- linerel(DeltaX,DeltaY) (integer,integer): прототип (i,i) – рисует линию от текущей позиции до точки приращения координат до которой равны (DeltaX, DeltaY). Текушая позиция изменяется на (DeltaX, DeltaY);

- lineto(X,Y) (integer,integer): прототип (i,i) – рисует линию от текущей позиции до точки с координатами (X,Y), после чего текущая позиция перемещается в (X,Y);

- rectangle(X, Y, X1, Y1) (все аргументы целочисленные): прототип (i,i,i,i) – рисует текущим цветом и стилем линии прямоугольник по координатам его верхнего левого (X,Y) и нижнего правого (X1,Y1) углов;

- drawpoly(PolyPointsList) (point_list): прототип (i) – рисует многоугольник, где (point_list) – список координат вершин X,Y.

Например: drawpoly([45,15, 85,45, 45,85, 15,45]).

- circle(X,Y,Radius) (integer,integer,integer): прототип (i,i,i) – рисует окружность.

При построении окружности учитывается отношение масштабов по горизонтали и по вертикали.

- getaspectratio(Xasp,Yasp) (integer,integer): прототип (i,i) – вычисляет коэффициенты искажения по горизонтали Xasp и по вертикали Yasp; прототип (o,o) – устанавливает новые коэффициенты;

- bar(Left,Top,Right,Bottom) (все аргументы целочисленные): прототип (i,i,i,i) – рисует заполненную текущим шаблоном прямоугольную полосу;

- bar3d(Left,Top,Right,Bottom,Depth,Topflag) (все аргументы целочисленные): прототип (i,i,i,i,i,i) – рисует заполненную текущим шаблоном трехмерную прямоугольную полосу;

- outtext(СтрокаТекста) (string): прототип (i) – выводит строку текста в окно вывода (viewport) используя текущий шрифт, размер, направление и установки выравнивания;

- outtextxy(X,Y, СтрокаТекста) (integer, integer, string): прототип (i,i,i) – выводит с заданной позиции (X,Y) строку текста в окно вывода используя текущий шрифт, размер, направление и установки выравнивания;

- getimage(Left,Top,Right,Bottom,BitMap) (все аргументы целочисленные): прототип (i,i,i,i,o) – сохраняет в памяти прямоугольную область экрана в переменной BitMap, которой потом уже нельзя манипулировать как другими

переменными – она используется в предикате putimage для вывода сохраненного изображения на экран, при этом координаты X,Y задают местоположение верхнего левого края области. Количество байтов требуемое для сохранения изображения может быть получено с помощью предиката imagesize.

- imagesize(Левый, Верхний, Правый, Нижний, Size) (integer,integer,integer,integer,string): прототип (i,i,i,i,o) возвращает в Size количесвто байтов, необходимых getimage для сохранения изображения. Сохраняемое изображение – это прямоугольник, задаваемый координатами "Левого Верхнего" и "Правого Нижнего" углов. Если для сохранения изображения требуется память больше 64К, то возвращается $FFFF.

- putimage(X,Y,Bitmap, BitOp) (integer,integer,string,integer): прототип (i,i,i,i) помещает изображение, сохраненное с помощью getimage, обратно на экран, при этом (X,Y) являются координатами верхнего левого угла изображения. Аргумент BitOp задает каким образом по цвету существующего пиксела и цвету пиксела выводимого изображения вычисляется цвет каждой результирующей точки на экране:

0 - копируются цвет изображения

1 - выполняется логическое OR (исключающее ИЛИ)

2 - выполняется логическое OR (ИЛИ)

3 - выполняется логическое AND (И)

4 - копируется инверсное изображение.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 1

Последовательность действий

1. Изучить пояснения к работе.

2. Войти в режим редактирования (Alt-E) и ввести определения предикатов для создания на экране трех непересекающихся окон (см. предикат makewindow).

3. Пользуясь средствами Турбо Пролога добавьте предикаты для

- перехода из одного окна в другое;

- очистки окна;
- редактирования текста в текущем окне;
- скроллинга текста в окне;
- удаления окна;
- изменения размеров окна;

- изменения цвета окна и рамки;

4. После ввода каждого нового предиката, программу следует откомпилировать для чего нажать клавиши ALT-F9. Если при компиляции будут обнаружены синтаксические ошибки, то их следует тут же исправить и добиться безошибочной компиляции (см. п. 2.2).

5. Если компиляция прошла успешно, перейти в режим исполнения: Run в главном меню или нажатие клавиш Alt-R.

6. Каждый новый введенный Вами предикат проверьте задавая в окне диалога после подсказки Goal: внешнюю цель с этим предикатом.

7. Включить режим трассировки и просмотреть выполнение предиката по шагам (см. п. 2.3).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ № 2

1. Инициализировать работу с графикой (предикат initgraph).
2. Построить простые графические объекты: круги, дуги,

8. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

ПРАКТИЧЕСКАЯ РАБОТА № 1

ПРЕДСТАВЛЕНИЕ ЗНАНИЙ И ПОЛУЧЕНИЕ ВЫВОДОВ С ПОМОЩЬЮ ЛОГИКИ ПРЕДИКАТОВ

Предикатом [2], или *логической функцией,* называется функция от любого числа аргументов, принимающая истинностные значения: истинно (1) и ложно (0). Аргументы принимают значения из произвольного, конечного или бесконечного множества D, называемого предметной областью. Предикат от *n* аргументов называют *n*-местным предикатом.

Предикат F(x), определенный на множестве D, задает определенное свойство элементам множества D и интерпретируется как высказывание “x обладает свойством F”, причем F принимает значение “истинно”, если это высказывание истинно, и значение “ложно”, если оно ложно. Предикат $F(x_1,x_2,\ldots,x_n)$ задает отношение между элементами $x_1,x_2,\ldots,x_n$ и интерпретирует как высказывание “ $x_1,x_2,\ldots,x_n$ находятся между собой в отношении F”.

Например: D – множество натуральных чисел. F(x) может означать, что x – четное или x – нечетное. G(x,y) - x > y или x делится на y.

Алфавит языка предикатов первого порядка состоит из:

1) разделители: запятая, открывающая и закрывающая скобки;

2) константы, обозначаемые строчными буквами или соединением таких букв, – например: a, st, друг;

3) переменные, обозначаемые прописными буквами, – например: X, ST, АДРЕС;

4) предикаты, обозначаемые прописными буквами, – например: P, Q, БОЛЬШЕ;

5) функции, устанавливающие зависимость и отображающие значения одной предметной области в значения другой (или той же), n-местные функции могут служить аргументами предиката. Функции будем обозначать строчными буквами: f,g,t;

6) логические функции:

$\neg$ (отрицание или дополнение). Высказывание “$\neg$A” читается “не A”. Оно истинно (И), если высказывание A – ложно (Л);

$\wedge$ (конъюнкция). Высказывание “A$\wedge$B” читается “A и B”. Оно истинно в том случае, когда истинно как A, так и B;

$\vee$ (дизъюнкция). Высказывание “A$\vee$B” читается “A или B”. Оно истинно, если истинно хотя бы одно из высказываний;

$\rightarrow$ (импликация). Высказывание “A$\rightarrow$B” читается “если A, то B”. Оно ложно в том и только в том случае, если A истинно, а B ложно;

$\leftrightarrow$ (эквивалентность). Высказывание “A$\leftrightarrow$B” читается “A тогда и только тогда, когда B”. Оно истинно в тогда и только тогда, когда A и B имеют одно и тоже истинностное значение;

$\exists$ (квантор существования). Высказывание “$\exists$A” читается “существует A”;

$\forall$ (квантор общности). Высказывание “$\forall$A” читается “для любого A”.

Пропозициональной формой [2], или *формулой алгебры логики*, называют всякое высказывание, составленное из некоторых исходных высказываний посредством логических операций, т. е. если F и G – пропозициональные формы, то $\neg$F, (F$\vee$G), (F$\wedge$G), (F$\rightarrow$G), (F$\leftrightarrow$G) – пропозициональные формы.

Пропозициональной функцией [5] называется выражение, содержащее переменную и превращающееся в истинное или ложное высказывание при подстановке вместо переменной имени предмета из определенной предметной области.

Пропозициональные функции делятся на одноместные, содержащие одну переменную, называемые свойствами (например, «x – композитор», «x-7=3»), и содержащие две и более переменных, называемые отношениями (например, «x-c=16», «объем куба x равен объему куба y).

Определение формулы – основного объекта логики предикатов – включает понятие “терм”.

Терм – выражение, включающее константы, переменные или n-местные функции $f(t_1,t_2,\ldots,t_n)$, где $t_1,t_2,\ldots,t_n$ – термы.

Например: f(X,Y), f(b,вес(Z)) – термы, P(X,голубой), вес(P(b)) – не термы, т.к. P – предикат.

Атом, или *элементарная (атомная) формула* – это выражение, включающее константы, переменные, функции и предикаты. Таким образом, если P – n-местный предикат, а $t_1,t_2,\ldots,t_n$ – термы, то $P(t_1,t_2,\ldots,t_n)$ – атом.

Например: P(X,голубой), ВХОД(стол, X, под(окно)) – являются атомами, f(X,Y) – не атом.

Формула или правильно построенная формула (ППФ) определяется следующим образом: всякий атом есть ППФ.

Если F и G – ППФ, а X – переменная, тогда $\neg$H, (G$\vee$H), (G$\wedge$H), ($\exists$X)G, ($\forall$X)H – ППФ.

Выражение “первого порядка” во фразе “исчисление предикатов первого порядка” связано с определением ППФ, в которых запрещается квантифицировать символы предикатов и функций.

Например:

$(\forall P)P(a)$
$(\forall f)(\forall X)P(f(X),b)$ – не являются ППФ логики предикатов первого порядка.

На практике ППФ используется для представления знаний. Не всегда легко представить знания, выраженные на естественном языке, с помощью ППФ. Например, выражение ”если два объекта равны, то они имеют одинаковые свойства” можно представить так;

$(\forall P)(\forall X)(\forall Y)(PABHO(X,Y) \rightarrow (P(X) \leftrightarrow P(Y)))$.

Но это выражение не является формулой первого порядка, так как квантифицируется предикат.

Покажем, каким образом можно выявлять логическую структуру мысли. Приведем несколько сложных суждений, структуру которых надо выразить в виде формул, используя введенные логические термины.

1. Если у меня будет свободное время (a) и я сдам экзамены по математике (b) и физике (c), то поеду отдыхать в Крым (d) или на Кавказ (e).

Формула: $(a \wedge b \wedge c) \rightarrow (d \vee e)$.

2. «Если человек с детства и юности своей не давал нервам властвовать над собой, то они не привыкнут раздражаться и будут ему послушны» (К.Д.Ушинский).

Формула: $(\neg a \wedge \neg b) \rightarrow (\neg c \wedge d)$.

Здесь буква a обозначает суждение: «человек с детства давал нервам властвовать над собой». А так как у нас имеется отрицание («не давал»), то запишем $\neg a$.

3. Если добродетель неправильно приложат, то она может стать пороком.

Формула: $a \rightarrow b$.

4. Если у меня будет свободное время, то я почитаю книгу или посмотрю телевизор.

Формула: $a \rightarrow (b \vee c)$.

5. Некоторые элементарные частицы имеют положительный заряд.

Формула: $(\exists X)(S(X) \wedge P(X))$.

Правилом вывода называют процедуру, которая из одной или нескольких ППФ производит другие ППФ. Задавая фиксированное множество правил вывода, можно рассматривать следующее семейство проблем: исходя из выбранного множества ППФ применением некоторого числа раз правил вывода, можно получить заранее заданную ППФ.

Исходные ППФ называют *аксиомами*, а ППФ, полученные из правил вывода, называют *теоремами*. Цель применения правил вывода, ведущих от аксиомы к теореме, называют *доказательством теоремы*.

*Интерпретация формул***.** Формула имеет определенный смысл, т. е. обозначает некоторое высказывание, если существует какая-либо интерпретация. Интерпретировать формулу – это значит связать с ней определенное непустое множество D, т. е. конкретизировать предметную область, называемую также *областью интерпретации,* и указать:

для каждой константы в формуле – конкретный элемент из D;

для каждой n-местной функциональной буквы в формуле – конкретную n-местную функцию на D;

для каждой n-местной предикатной буквы в формуле – конкретное отношение между n элементами из D.

Свойства правильно построенных формул. При заданной интерпретации значения истинности правильно построенную формулу (ППФ) можно вычислить по правилам, объединенным в таблице истинности.

Если F и G – любые две ППФ, то значения истинности составного выражения, построенного из этих ППФ, даются таблицей истинности (табл. 1).

Таблица 1. Таблица истинности

F	G	¬F	F∨G	F∧G	F→G	F↔G
И	И	Л	И	И	И	И
Л	И	И	И	Л	И	Л
И	Л	Л	И	Л	Л	Л
Л	Л	И	Л	Л	И	И

Если истинности двух ППФ независимо от интерпретации совпадают, то говорят, что эти ППФ являются эквивалентными. Пользуясь таблицей истинности, легко установить следующие эквивалентности (табл. 2.).

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что называется предикатом?
2. Из чего состоит алфавит языка предикатов первого порядка?
3. Что называют формулой алгебры логики?
4. Что такое элементарная формула?
5. Какую процедуру называют правилом вывода?
6. Что значить интерпретировать правильно построенную формулу?

Таблица 2. Таблица эквивалентности

Эквивалентные формулы		
¬(¬F)	F	
F→G	¬F∨G	
F↔G	(F→G) ∧(G→F)	
¬ (F∧G)	¬F∨¬G	Законы де Моргана
¬ (F∨G)	¬F∧¬G	

$F\wedge(G\vee H)$	$(F\wedge G)\vee(F\wedge H)$	Законы дистрибутивности
$F\vee(G\wedge H)$	$(F\vee G)\wedge(F\vee H)$	
$F\vee G$	$G\vee F$	Законы коммуникативности
$F\wedge G$	$G\wedge F$	
$(F\wedge G)\wedge H$	$F\wedge(G\wedge H)$	Законы ассоциативности
$(F\vee G)\vee H$	$F\vee(G\vee H)$	
$F\rightarrow G$	$\neg G\rightarrow\neg F$	
$F\wedge\neg F$	0	
$F\vee\neg F$	1	
$F\wedge 0$	0	
$F\wedge 1$	F	
$F\vee 0$	F	
$F\vee 1$	1	
$\forall XF(X)$ $\exists XF(X)$	$\forall YF(Y)$ $\exists YF(Y)$	
$\neg(\exists XF(X))$ $\neg(\forall XF(X))$	$\forall X(\neg F(X))$ $\exists X(\neg F(X))$	
$\forall X(F(X)\wedge G(X))$ $\exists X(F(X)\vee G(X))$	$\forall X\ F(X)\wedge\forall X\ G(X)$ $\exists X\ F(X)\vee\exists X\ G(X)$	

ЗАДАНИЕ

Выразить в символической форме следующие сложные суждения:

1. Если встать рано на рассвете и пойти в сад или парк, то можно услышать чудесные песни птиц.
2. Если эта фигура квадрат, то диагонали в ней равны, взаимно перпендикулярны и в точке пересечения делятся пополам.
3. Некоторые спортсмены не являются мастерами спорта.
4. «Некоторые лекарства опаснее самих болезней» (Сенека).
5. Некоторые люди не изучают логику.
6. Если мне дадут отпуск летом, то я поеду отдыхать к морю или по туристической путевке в Канаду.
7. Добро не умрет, а зло пропадет.

8. «Видеть несправедливость и молчать – это значит самому участвовать в ней» (Ж.-Ж. Руссо).
9. Если вы любите детей, полны жажды познания, имеете доброе сердце, мечтаете посвятить себя интересному творческому труду, то смело выбирайте профессию учителя.
10. «Если больному после разговора с врачом не становится легче, то это не врач» (В. М. Бехтерев).
11. «Если верный конь, поранив ногу,
    Вдруг споткнулся, а потом опять,
    не вини коня, вини дорогу
    и коня не торопись менять» (Р. Гамзатов).

ПРАКТИЧЕСКАЯ РАБОТА № 2

ПРЕОБРАЗОВАНИЕ ПРАВИЛЬНО ПОСТРОЕННЫХ ФОРМУЛ В ПРЕДЛОЖЕНИЕ

Аппарат логики предикатов используется для представления задачи в виде теоремы: формула H логически следует из формулы G, т.е. $G \rightarrow H$. Доказательство этой теоремы состоит в том, чтобы показать, что каждая интерпретация, удовлетворяющая G, удовлетворяет и H, или, что $\neg G \vee H$ истинно для любой интерпретации; так как $\neg(\neg G \vee H) = G \wedge \neg H$, то на практике обычно доказывают невыполнимость множества предложений $G \wedge \neg H$.

Чтобы упростить доказательства теоремы, все формулы представляются в виде дизъюнкций литералов. *Литералом* (или *литерой*) называют атом или его отрицание. Формулу, представляющую собой дизъюнкцию литералов, называют *предложением* (или *дизъюнктом*).

Например: $R(Z,a,g(X)) \vee (\neg T(U)) \vee (\neg U(b,k(c)))$.

Иными словами, в каждой формуле необходимо исключить все логические операции (включая кванторы), кроме дизъюнкции, и уменьшить область действия знака отрицания до одной предикатной буквы.

Преобразование ППФ в предложение. Перед тем как объяснить процесс резолюции, покажем, что любую ППФ исчисления предикатов первого порядка

можно преобразовать во множество предложений, используя законы эквивалентности. Процесс такого преобразования состоит из последовательных этапов, которые покажем на следующем примере ППФ:

$(\forall X)(((P(X)\wedge Q(X,a))\rightarrow R(X,b))\wedge((\forall Y)((\forall Z)(R(Y,Z)\rightarrow T(X,Y)))))\vee((\forall X)S(X))$.

Исключение символов эквивалентности и импликации. Для этого воспользуемся следующими законами эквивалентности: $(F\rightarrow G)$ заменим на $(\neg F \vee G)$. В нашем примере такая подстановка даст:

$(\forall X)((\neg(P(X)\wedge Q(X,a))\vee R(X,b))\wedge((\forall Y)(\neg(\forall Z)(R(Y,Z)\vee T(X,Y)))))\vee((\forall X)S(X))$.

Уменьшение области действия знаков отрицания. Нужно выполнить такое преобразование, чтобы каждый знак отрицания применялся не более чем к одной атомной формуле. Для этого используем следующие законы эквивалентности:

$(\neg(\neg F))$ и F

$(\neg\ (F\wedge G))$ и $\neg F)\vee(\neg G)$

$(\neg\ (F\vee G))$ и $\neg F)\wedge(\neg G)$

$(\ \neg(\exists X)F(X))$ и $(\forall X)(\neg F(X))$

$(\neg(\forall X)F(X))$ и $(\exists X)(\neg F(X))$

Полученная нами ППФ преобразуется к виду:

$(\forall X)((\neg P(X)\vee\neg Q(X,a)\vee(R(X,b))\wedge((\forall Y)((\exists Z)(\neg R(Y,Z))\vee T(X,Y)))))\vee((\forall X)S(X))$

Разделение переменных. В пределах области действия квантора, переменная, связываемая с этим квантором, представляет собой немую переменную. Такую переменную в области действия квантора можно заменить любой другой (не встречающейся) переменной, при этом значение истинности ППФ не изменится. Стандартизация переменных в пределах ППФ означает переименование немых переменных с той целью, чтобы каждый квантор имел свою, свойственную только ему, немую переменную. Так, вместо записи

$$(\forall X)(P(X)\vee(\exists X)Q(X))$$

можно написать

$$(\forall X)(P(X)\vee(\exists Y)Q(Y)).$$

Полученная на предыдущем этапе ППФ преобразуется к виду:

$(\forall X)((\neg P(X)\vee\neg Q(X,a)\vee(R(X,b))\wedge((\forall Y)((\exists Z)(\neg R(Y,Z))\vee T(X,Y)))))\vee((\forall U)S(U))$

Исключение кванторов существования. Рассмотрим ППФ

$$(\forall X)(\exists Y)P(X,Y)),$$

которую можно прочитать как: "Для всех X существует некоторое Y (возможно, зависящее от X) такое, что P(X,Y)". Поскольку квантор существования находится в области некоторого квантора общности, то можно допустить, что "существующий" Y зависит от X. Пусть эта зависимость определяется функцией Y=g(X), отображающей каждое X в Y. Такая функция называется *сколемовской*, или *функцией Сколема*. Заменив Y на g(X), мы исключим квантор существования, и наша ППФ примет вид:

$$(\forall X)(P(X,q(X)).$$

Пусть ППФ $(\exists Y)P(Y)$ является подформулой другой ППФ, в которой $X_1,...,X_n$ квантифицированы универсально. Тогда удаляют Y и заменяют встречающуюся Y сколемовской функцией $f(X_1,...,X_n)$. Заметим, что эта функция будет содержать столько аргументов, сколько имеется универсальных квантификаторов слева от формулы $(\exists Y)P(Y)$. Эта функция, как мы уже говорили, просто выражает существование соответствия априори между совокупностью значений $(\exists Y)P(Y)$ и значениями Y. Поскольку эта функция заранее неизвестна, то в каждой замене Y мы должны записывать новую функциональную букву, которая отличается от уже имеющихся. Когда слева от символа "∃" отсутствуют символы "∀", то вводимая функция Сколема не будет содержать аргументов: следовательно, это будет новая константа, называемая константой Сколема. Так, ППФ $(\exists Y)P(Y)$ заменится на P(a), где про константу "a" известно, что она существует.

После выполнения замены Z функцией Сколема g(X,Y) наша ППФ будет иметь вид

$(\forall X)((\neg P(X)\vee\neg Q(X,a)\vee(R(X,b))\wedge((\forall Y)(\neg R(Y,g(X,Y))\vee T(X,Y)))))\vee((\forall U)S(U)).$

Перемещение всех квантификаторов общности в начало ППФ (без изменения их относительного порядка). Каждый квантор общности имеет свою переменную, и поэтому такие кванторы можно переместить в начало формулы, считая, что область действия каждого из них включает всю последующую часть ППФ. Говорят, что результирующая ППФ находится в префиксной форме. Она состоит из цепочки кванторов, называемой *префиксом*, и следующей за ней бескванторной формулы, называемой *матрицей*.

Префиксная форма для нашей ППФ имеет вид:

$(\forall X)(\forall Y)(\forall U)(((\neg P(X)\vee\neg Q(X,a)\vee R(X,b))\wedge(\neg R(Y,Z)\vee T(X,Y)))\vee S(U))$.

Приведение матрицы к конъюнктивной нормальной форме. Матрица находится в *конъюнктивной нормальной форме*, если она записана как конъюнкция конечного множества дизъюнкций литералов. Например:

$(P(X)\vee Q(X,Y))\wedge(P(Y)\vee\neg R(Y))$

$Q(X)\wedge R(X,Y)$

$\neg R(Y)$

Любую матрицу можно привести к конъюнктивной нормальной форме, используя законы ассоциативности и дистрибутивности операций $\vee$ и $\wedge$. После приведения матрицы нашего примера ППФ в конъюнктивную нормальную форму эта ППФ примет следующий вид:

$(\forall X)(\forall Y)(\forall U)(\neg P(X)\vee\neg Q(X,a))\vee(R(X,b)\vee S(U))\wedge$

$(\neg R(Y,g(X,Y))\vee T(X,Y)\vee S(U))$.

Исключение кванторов общности. Поскольку все переменные в ППФ должны быть связаны, то можно предположить, что для каждой из них имеется квантор общности. При этом порядок расположения этих кванторов роли не играет, так что можно их исключить из ППФ. После исключения кванторов из ППФ у нас остается лишь матрица в конъюнктивной нормальной форме.

Исключение символов конъюнкции. Теперь можно исключить символы $\wedge$, путем замены выражения типа $(P\wedge Q)$ на множество ППФ $\{P,Q\}$. В результате выполнения таких замен получим конечное множество ППФ, каждая из

которых – дизъюнкция литералов. А любая ППФ, содержащая дизъюнкцию литералов, как мы условились, является предложением.

Наш пример ППФ преобразуется в следующее множество предложений:

{¬P(X)∨¬Q(X,a)∨(R(X,b)∨S(U)}

{¬R(Y,g(X,Y))∨T(X,Y)∨S(U)}.

Переименование (разделение) переменных. Символы переменных должны быть изменены так, чтобы каждый появился не более чем в одном предложении. Напомним, что $(\forall X(P(X) \wedge Q(X))$ эквивалентно $(\forall X)P(X) \wedge (\forall Y)Q(Y)$. Теперь наши предложения приобретают вид:

¬P(X)∨¬Q(X,a)∨(R(X,b)∨S(U)

¬R(Y,g(Z,Y)) ∨T(A,B) ∨S(C)

Вместо переменных в литералах могут находиться термы, не содержащие переменных. Тогда говорят, что имеет место основной частный случаи. Например, P(a,f(g(b))) является основным частным случаем P(X,Y).

По структуре высказывания делятся на простые (они имеют логическую форму «S есть P» либо «S не есть P») и сложные (грамматически выражаются сложными предложениями). Простые высказывания бывают

атрибутивные (в них выражается принадлежность или непринадлежность какого-то свойства объекту или классу объектов);

об отношениях между несколькими объектами;

о существовании или несуществовании какого-либо объекта или явления).

В атрибутивные высказывания часто включаются кванторные связки.

По качеству простые высказывания делятся на утвердительные и отрицательные. С количественной точки зрения высказывания делятся на единичные, частные и общие.

Перечислим основные типы высказываний и их обозначения [5]:

A – общеутвердительные («Всякий S есть P»),

E – общеотрицательные («Всякий S не есть P»),

I – частноутвердительные («Некоторый S есть P»),

O – частноотрицательные («Некоторый S не есть P»).

В частных суждениях слово «некоторые» не исключает варианта «все».

Пример.

Преобразовать высказывание "Все дети имеют матерей" в предложение, т.е. в дизъюнкцию литералов. Это высказывание можно представить следующей ППФ:

$\forall X \exists Y(\textit{ребенок}(X) \rightarrow \textit{мать}(Y) \wedge \textit{относится}(X,Y))$.

Для удаления квантора существования заменим переменную Y функцией Сколема g(A), т.е. Y=g(X), откуда получим:

$\forall X(\textit{ребенок}(X) \rightarrow \textit{мать}(g(X)) \wedge \textit{относится}(X,g(X)))$.

Теперь можно исключить квантор общности, который распространяет свое действие на все выражения. Кроме того, заменим символ "импликация" дизъюнкцией, используя закон $P \rightarrow Q \equiv \neg P \vee Q$, откуда получаем:

$\neg \textit{ребенок}(X) \vee (\textit{мать}(g(X)) \wedge \textit{относится}(X,g(X)))$.

Используя закон

$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$,

можем написать

$\neg \textit{ребенок}(X) \vee \textit{мать}(g(X))$,

$\neg \textit{ребенок} \vee \textit{относится}(X,g(X))$.

Два полученные предложения можно интерпретировать так: "Если X – ребенок, то он относится к матери, выражаемой функцией g(X)". Из них следует:

ребенок(X)→ мать(g(X)) и

ребенок(X)∨ относится(X,g(X)).

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение литерала.
2. Что называют дизъюнктором?
3. Какая функция называется функцией Сколема?

4. Как любую ППФ можно преобразовать во множество предложений?
5. Когда ППФ находится в префиксной форме?
6. Какая ППФ называется матрицей?
7. Когда матрица находится в конъюктивной нормальной форме?

ЗАДАНИЕ

Преобразовать высказывания в дизъюнкцию литералов.

1. Некоторые растения являются грибами.
2. Все предложения со второстепенными членами являются распространенными.
3. Все одноклеточные не являются червяками.
4. Некоторые космонавты – летчики.
5. Некоторые студенты – не лодыри.
6. Все дельфины – теплокровные.
7. Ни один кит не является рыбой.
8. Все свидетели дают истинные показания.
9. Ни одна балалайка не является клавишным инструментом.
10. Некоторые люди не любят природу.

ПРАКТИЧЕСКАЯ РАБОТА № 3

СУЩНОСТЬ ПРИНЦИПА РЕЗОЛЮЦИЙ.

ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ ПРИНЦИПА РЕЗОЛЮЦИЙ

Резольвенты [2]. Говорят, что литерал является *конкретным*, если он не содержит никакой переменной. Например, ¬P(a) или Q(a,f(b)) являются конкретными литералами, в то время как ¬P(X) или Q(X,f(y)) не являются таковыми. Конкретным предложением является дизъюнкция конкретных литералов.

Пусть имеются два конкретных предложения

$P1 \vee P2 \vee .... \vee P_n$ и $\neg P1 \vee Q2 \vee .... \vee Q_n$

Здесь литерал ¬P1 является отрицанием литерала P1. Из этих двух предложений можно вывести новое предложение, называемое *резольвентой,*

или *резолюцией*. Резольвентой является дизъюнкция этих предложений с последующим исключением пары P1 и ¬P1.

Из предложений P1 и¬P1 получается *пустое предложение*, которое является признаком противоречия и обозначается символом .

Два конкретных предложения могут не иметь резольвенты или могут иметь множество резольвент. Например, из $P \vee Q \vee R$ *и* $\neg P \vee \neg Q \vee S$ получаются резольвенты $Q \vee \neg Q \vee R \vee S$ *или* $P \vee \neg P \vee R \vee S$ (которые в действительности являются эквивалентными).

Сущность принципа резолюций. На практике невыполнимость множества предложений устанавливается посредством принципа резолюций. Речь идет о процедуре логического вывода новых предложений из множества исходных.

Принцип резолюций состоит:

1) в получении новых предложений на основании множества исходных и вновь получаемых предложений;

2) в отыскании частных случаев формул $F(t_1,\ldots,t_n)$ из $F(V_1,...,V_n)$ при подстановке вместо V_i произвольных термов t_i, т.е. $F(V_1,...,V_n) \rightarrow F(t_i,...,t_n)$.

Резольвенту двух предложений можно получить следующим образом:

1) переименовать переменные двух предложений так, чтобы последние стали одинаковыми;

2) найти подстановку, преобразующую литерал одного предложения в дополнительный по отношению к некоторому литералу другого предложения и произвести эту замену в обоих предложениях;

3) вычеркнуть дополнительные друг к другу литералы;

4) удалить одинаковые литералы в предложении кроме одного;

5) дизъюнкция литералов, оставшихся в обоих предложениях, является резольвентой.

Если некоторая последовательность резолюций, применяемых к исходному множеству предложений E и множеству резольвент, полученных в

процессе резолюции, приводит к пустому предложению, то множество E является *невыполнимым.*

Для доказательства невыполнимости множества предложений пользуются *опровержением*. Доказательство выполнимости множества G→H, что эквивалентно ¬G∨H, – это опровержение его невыполнимости, или, что то же самое, доказательство невыполнимости ¬(¬G∨H), что эквивалентно G∧¬H.

Общий алгоритм опровержения с помощью резолюций может иметь следующий вид:

резолюция (G,H)

1) C: – множество предложений, полученных путем преобразования формул множества G;
2) добавить к множеству C предложения, полученные из ¬H;
3) пока пустое предложение не появится в C, **выполнить:**

начало: выбрать два различных предложения в C;

если они имеют резольвенты, то найти одну резольвенту и добавить ее к множеству C.

конец.

Примеры использования метода резолюций.

Пример 1. Предположим, что покупательная способность людей падает, если цены растут на товары и услуги. Предположим также, что люди несчастны, когда их покупательная способность падает. Предположим, что цены растут. Из этого можно заключить, что люди несчастны.

Применим следующие обозначения:

1) P – цены растут;
2) S – покупательная способность уменьшается;
3) U – люди несчастны.

В этом примере четыре утверждения:

1) если цены растут, то покупательная способность падает;
2) если покупательная способность падает, то люди несчастны;

3) цены растут;

4) люди несчастны.

Эти утверждения можно записать в виде следующих формул:

1) $P \rightarrow S$,

2) $S \rightarrow U$,

3) P,

4) U.

Преобразуем эти формулы в предложения:

1) $\neg P \vee S$,

2) $\neg S \vee U$,

3) P,

4) U.

Докажем путем опровержения, что U – логическое следствие из 1), 2), 3). Отрицаем 4) и получаем следующее доказательство:

1) $\neg P \vee S$,

2) $\neg S \vee U$,

3) P,

4) $\neg U$ отрицание заключения,

5) S резольвента 3) и 1),

6) U резольвента 5) и 2),

7) резольвента 6) и 4).

Пример 2. Все доктора имеют пациентов и некоторые пациенты любят своих докторов. Ни один пациент не любит знахаря. Следовательно, никакой доктор не является знахарем.

Обозначим:

1) P(X): X – пациент;

2) D(X): X – доктор;

3) Q(X): X – знахарь;

4) L(X,Y): X любит Y.

Тогда факты и заключения можно записать следующим образом:

F1: $(\exists X)(P(X)\wedge(\forall Y)(D(Y) \rightarrow L(X,Y)))$,

F2: $(\forall X)(P(X) \rightarrow (\forall Y)(Q(Y) \rightarrow \neg L(X,Y)))$,

G: $(\forall X)(D(X) \rightarrow \neg Q(X))$.

Преобразуем эти формулы в предложения:

1) $P(a)$ из F1,

2) $\neg D(Y)\vee L(a,Y)$ из F1,

3) $\neg P(X)\vee\neg Q(Y)\vee\neg L(X,Y)$ из F2,

4) $D(b)$ из G,

5) $Q(b)$ из G.

Используя метод резолюций, получим следующее доказательство:

6) $L(a,b)$ резольвента 4) и 2),

7) $\neg Q(Y)\vee\neg L(a,Y)$ резольвента 3) и 1),

8) $\neg L(a,b)$ резольвента 5) и 7),

9) резольвента 6) и 8).

Поясним естественность этого доказательства.

Из F1 можно предположить, что существует пациент a, который любит каждого доктора 1) и 2).

Предположим, что заключение неверно, т.е. b – одновременно и доктор, и знахарь.

Так как пациент любит каждого доктора, то a любит b 6).

Так как a – пациент, то a не любит никакого знахаря 7).

Однако b – знахарь. Следовательно, a не любит b 8).

Это невозможно из-за 6).Таким образом, мы закончили доказательство.

Пример 3. Допустим, что если Верховный Совет отказывается принимать новые законы, то забастовка не будет закончена, если только она не длится более года и зарплата не повышается. Закончится ли забастовка, если Верховный Совет отказывается действовать и забастовка только началась?

Применим следующие обозначения:

1) P – Верховный Совет отказывается действовать;

2) Q – забастовка заканчивается;

3) R – зарплата повышается;

4) S – забастовка длится более года.

Тогда приведенные выше утверждения можно представить следующими формулами:

1) F1: $(P \rightarrow (\neg Q \vee (R \wedge S)))$ – если Верховный Совет отказывается принимать новые законы, то забастовка не будет закончена, если она не длится более года и зарплата не повышается;

2) F2: P – Верховный Совет отказывается действовать;

3) F3: $\neg S$ – забастовка только началась.

Требуется доказать, что $\neg Q$ – логическое следствие $F1 \wedge F2 \wedge$иF3. Отрицаем Q и преобразуем F1, F2 и F3 в предложения:

1) $\neg P \vee \neg Q \vee R$ из F1,

2) $\neg P \vee \neg Q \vee S$ из F1,

3) P из F2,

4) $\neg S$ из F3,

5) Q отрицание заключения.

Используя резолюции, получим следующее доказательство:

6 6) $\neg Q \vee S$ резольвента из 3) и 2),

7) S резольвента из 6) и 5),

8) резольвента из 7) и 4).

Пример 4. Таможенники обыскивают каждого, кто въезжает в страну, кроме высокопоставленных лиц. Некоторые люди, способствующие провозу наркотиков, въезжали в страну и были обысканы людьми, также способствующими провозу наркотиков. Никто из высокопоставленных лиц не способствовал провозу наркотиков. Доказать, что некоторые из таможенников способствовали провозу наркотиков.

Примем следующие обозначения:

1) E(X): X въезжал в страну;

2) V(X): X – высокопоставленное лицо;

3) S(X,Y): Y обыскивает X;

4) C(X): X – таможенник;

5) P(X): X способствует провозу наркотиков.

Заключение представляется формулой:

$$(\exists X)(P(X)\wedge C(X)).$$

Отрицаем заключение и преобразуем формулы в предложения:

1) $\neg E(X)\vee V(X)\vee S(X,f(X))$,

2) $\neg E(X)\vee V(X)\vee C(f(X))$,

3) $P(a)$,

4) $E(a)$,

5) $\neg S(a,Y)\vee P(Y)$,

6) $\neg P(X)\vee\neg V(X)$,

7) $\neg P(X)\vee\neg C(X)$.

Доказательство методом резолюций выглядит следующим образом:

8) $\neg V(a)$ из 3) и 6),

9) $V(a)\vee C(f(a))$ из 2) и 4),

10) $C(f(a))$ из 8) и 9),

11) $V(a)\vee S(a,f(a))$ из 1) и 4),

12) $S(a,f(a))$ из 8) и 11),

13) $P(f(a))$ из 12) и 50,

14) $\neg C(f(a))$ из 13) и 7)

15) из 10) и 14).

Заключение доказано.

Пример 5. Каждый, кто хранит деньги в кассе, получает проценты. Докажем, что если не существует процентов, то никто не хранит денег.

Примем следующие обозначения:

1) S(X,Y): X хранит деньги Y;

2) M(X): X есть деньги;

3) I(X): X есть проценты;

4) E(X,Y): X получает Y проценты.

Посылка записывается в виде:

$$(\forall X)(\exists Y)(S(X,Y)\wedge M(Y)) \rightarrow (\exists Y)(I(Y)\wedge E(X,Y))),$$

а заключение в виде:

$$\neg(\exists X)I(X) \rightarrow (\forall X)(\forall Y)(S(X,Y)) \rightarrow \neg M(Y)).$$

Преобразуем посылку в предложения:

1) $\neg S(X,Y)\vee\neg M(Y)\vee I(f(X))$,

2) $\neg S(X,f(X))\vee\neg M(Y)\vee E(X,f(X))$.

Преобразованное в предложение отрицание заключения имеет вид:

3) $\neg I(z)$,

4) $S(a,b)$,

5) $M(b)$.

Доказательство заключения:

6) $\neg S(X,Y)\vee\neg M(Y)$ из 3) и 1),

7) $\neg M(b)$ из 6) и 4),

8) из 7) и 5).

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем смысл процедуры резолюции?
2. Что такое резольвента?
3. Какое предложение называется пустым?

ЗАДАНИЕ

Используя метод резолюций, доказать истинность заключения.

1. Лошадь есть животное, поэтому голова лошади есть голова животного.

2. Полиция обыскивает всех въехавших в страну, за исключением дипломатов. Шпион въехал в страну, однако распознать личность шпиона может только шпион. Шпион не является дипломатом. Среди полицейских имеется шпион.

3. Благородный труд заслуживает уважения, так как благородный труд способствует прогрессу общества. Труд учителя есть благородный труд, так как

труд учителя заключается в обучении и воспитании подрастающего поколения. Труд учителя заслуживает уважения.

4. Все тюльпаны – цветы. Все цветы – растения. Все растения используют для питания углекислый газ атмосферы и выделяют в нее кислород. Все растения, использующие для питания углекислый газ атмосферы и выделяющие в нее кислород, содержат хлорофилл. Все тюльпаны содержат хлорофилл.

5. Все, что требует мужества и героизма, есть подвиг. Первый полет человека в космос требовал мужества и героизма. Первый полет человека в космос есть подвиг.

6. Если животное млекопитающее, то оно относится к типу хордовых. Это животное не является млекопитающим. Это животное не относится к типу хордовых.

7. Всякое преступление карается законом, поскольку оно общественно опасно. Грабеж есть преступление, так как грабеж – это открытое хищение личного имущества граждан. Грабеж карается законом.

8. Если должностное лицо получает взятку, то оно совершает преступление. Должностное лицо не получает взятку. Данное должностное лицо не совершает преступления.

9. Если на металле появились следы ржавчины, то началась коррозия. Коррозия не началась. На металле не появились следы ржавчины.

10. Если эта машина – двигатель внутреннего сгорания, то она является тепловым двигателем. Если эта машина является тепловым двигателем, то в ней топливо сжигается внутри цилиндра. Если эта машина – двигатель внутреннего сгорания, то в ней топливо сжигается внутри цилиндра.

11. Некоторый остров заселен исключительно рабами и сеньорами. На этом острове царствует некая принцесса. Сеньоры всегда говорят правду, а рабы все время лгут. Некоторые сеньоры бедны, другие богаты. Рабы также делятся на богатых и бедных. Принцесса желает выйти замуж за богатого раба. Каким же образом раб может всего лишь одним заявлением убедить принцессу

в том, что он достоин быть ее супругом (какая фраза могла бы убедить принцессу в том, что говорящий эту фразу является богатым рабом)?

ПРАКТИЧЕСКАЯ РАБОТА № 4

ПРЕДСТАВЛЕНИЕ ЗНАНИЙ ПРАВИЛАМИ И ЛОГИЧЕСКИЙ ВЫВОД

Наиболее распространённым методом представления знаний являются правила продукций, или продукционные правила. Идея этого метода, широко используемого в разработке информационных систем, принадлежит Э. Посту (1943).

Продукционная система состоит из трех основных компонентов. Первый из них – это набор правил, используемый как база знаний, поэтому его еще называют базой правил. Следующим компонентом является рабочая память (или память для кратковременного хранения), в которой хранятся предпосылки, касающиеся конкретных задач предметной области, и результаты выводов, полученных на их основании, и, наконец, следует механизм логического вывода, использующий правила в соответствии с содержимым рабочей памяти.

Основным элементом продукции является ее ядро: A $\Rightarrow$ B. Интерпретация ядра продукции может быть различной и зависит от того, что стоит слева и справа от знака секвенции $\Rightarrow$. Обычное прочтение ядра продукции выглядит так: ЕСЛИ A, ТО B. Более сложные конструкции ядра допускают в правой части альтернативный выбор, например, ЕСЛИ A, ТО B1, ИНАЧЕ B2. Секвенция может истолковываться в обычном логическом смысле как знак логического следования B из истинного A (если A не является истинным выражением, то о B ничего сказать нельзя). Возможны и другие интерпретации ядра продукции, например A описывает некоторое условие, необходимое для того, чтобы можно было совершить действие B.

Пусть правило продукции имеет вид:

ЕСЛИ<условия>ТО<действие>.

Слова в угловых скобках являются посылками, или утверждениями.

Рассмотрим несложный пример. Допустим, что данные, записываемые в рабочую память, представляют собой образцы в виде наборов символов. Например, «намерение – отдых», «место отдыха – горы» и т. п. Правила, накапливаемые в базе правил, отражают содержимое рабочей памяти. В их условной части находятся либо одиночные образцы, либо несколько условий, соединенных предлогом «и», а в заключительной части – образцы, дополнительно регистрируемые в рабочей памяти. Рассмотрим два примера подобных правил.

Правило 1.

ЕСЛИ «намерение – отдых» и «дорога ухабистая»

ТО «использовать джип»

Правило 2.

ЕСЛИ «место отдыха – горы»

ТО «дорога ухабистая»

После того, как в рабочую память записываются образцы «намерение – отдых» и «место отдыха – горы», рассматривается возможность применения этих правил. Сначала механизм вывода сопоставляет образцы из условной части правила с образцами, хранимыми в рабочей памяти. Если все образцы имеются в рабочей памяти, условная часть считается истинной, в противном случае – ложной. В данном примере образец «намерение – отдых» существует в рабочей памяти, а образец «дорога ухабистая» отсутствует, поэтому его условная часть считается ложной. Что касается правила 2, то его условная часть истинна. Поскольку в данном случае существует только одно правило с истинной условной частью, то механизм вывода сразу же выполняет его заключительную часть и образец «дорога ухабистая» заносится в рабочую память. При попытке вторично применить эти правила получается, что можно применить лишь правило 1, поскольку правило 2 уже было применено и выбыло из числа кандидатов. К этому времени содержимое рабочей памяти было дополнено новым образцом – результатом применения правила 2, поэтому условная часть правила 1 становится истинной и содержимое рабочей

памяти пополняется образцом его заключительной части – «использовать джип». В итоге правил, которые можно было бы применять, не остается, и система останавливается.

Если в памяти системы хранится некоторый набор продукций, то они образуют систему продукций. В системе продукций должны быть заданы специальные процедуры управления продукциями, с помощью которых происходит актуализация продукций и выбор для выполнения той или иной продукции из числа актуализированных. В ряде систем ИИ используют комбинации сетевых и продукционных моделей представления знаний. В таких моделях декларативные знания описываются в сетевом компоненте модели, а процедурные знания – в продукционном. В этом случае говорят о работе продукционной системы над семантической сетью.

Классификация ядер продукции. Ядра продукции можно классифицировать по различным основаниям. Прежде всего все ядра делятся на два больших типа: детерминированные и недетерминированные. В детерминированных ядрах при актуализации ядра и при выполнимости А правая часть ядра выполняется обязательно; в недетерминированных ядрах В может выполняться и не выполняться. Таким образом, секвенция $\Rightarrow$ в детерминированных ядрах реализуется с необходимостью, а в недетерминированных – с возможностью. Интерпретация ядра в этом случае может, например, выглядеть так: ЕСЛИ А, ТО ВОЗМОЖНО В.

Возможность может определяться некоторыми оценками реализации ядра. Например, если задана вероятность выполнения В при актуализации А, то продукция может быть такой: ЕСЛИ А, ТО С ВЕРОЯТНОСТЬЮ Р РЕАЛИЗОВАТЬ В. Оценка реализации ядра может быть лингвистической, связанной с понятием терм – множества лингвистической переменной, например: ЕСЛИ А, ТО С БОЛЬШЕЙ ДОЛЕЙ УВЕРЕННОСТИ В. Возможны иные способы реализации ядра.

Детерминированные продукции могут быть однозначными и альтернативными. Во втором случае в правой части ядра указываются

альтернативные возможности выбора, которые оцениваются специальными весами выбора. В качестве таких весов могут использоваться вероятностные оценки, лингвистические оценки, экспертные оценки и т.п.

Особым типом являются прогнозирующие продукции, в которых описываются последствия, ожидаемые при актуализации А, например: ЕСЛИ А, ТО С ВЕРОЯТНОСТЬЮ Р МОЖНО ОЖИДАТЬ В.

В настоящее время используются следующие формы продукционных правил:

ЕСЛИ<предшествующий>ТО<последующий>

ЕСЛИ<основание>ТО<гипотеза>

Таким образом, правило состоит из двух частей. Часть ЕСЛИ указывает "условия", "предшествующий" или "основание", а часть ТО представляет соответственно "заключение", "последующий" или "гипотезу". Перечисленные элементы второй части генерируются при истинности условий первой части.

Примером правила продукции может служить следующее:

ЕСЛИ клиент работает на одном месте более двух лет

ТО клиент имеет постоянную работу.

Так, в примере, представленном выше, если возраст клиента равен или больше 18 лет, то мы можем заключить, что клиент имеет право претендовать на получение кредита.

Гипотетический силлогизм означает, что когда заключение одного правила является посылкой другого правила, то можно установить третье правило с посылкой из первого правила и заключением из второго. Другими словами:

Условие ЕСЛИ X ТО Y

условие ЕСЛИ *Y* ТО Z

заключение ЕСЛИ *X* ТО Z

Например:

ЕСЛИ клиент женат

ТО имущество является совместной собственностью

ЕСЛИ имущество является совместным

ТО закладная на кредит может быть подписана.

Мы можем заключить, что

ЕСЛИ клиент женат

ТО закладная на кредит может быть подписана.

Модус поненс и гипотический силлогизм являются двумя правилами вывода в логике.

Рассмотрим базу знаний, написанную на примере закладной для получения кредита банка под залог недвижимого имущества (ссуды). Для получения кредита клиент должен иметь постоянную работу, приемлемый доход, хороший кредитный рейтинг и приемлемое имущество (собственность).

Если клиент не имеет постоянной работы, тогда он должен иметь адекватные активы. Величина кредита не может быть больше 80 % стоимости его имущества и клиент должен иметь 20 % ценности в кассе.

Определение "постоянная работа" состоит в том, что клиент должен иметь одну и ту же работу более двух лет. Определение "адекватные активы" заключается в том, что имущество клиента должно быть оценено как десятикратная величина кредита или клиент должен иметь ликвидные активы, оцениваемые как пятикратная величина кредита. "Приемлемое имущество" – это имущество, находящееся на территории расположения банка.

Проверка кредитного рейтинга проводится вручную.

Смысл "адекватного дохода" состоит в следующем. Если клиент *одиночка,* то оплата закладной должна быть менее чем 70 % его чистого дохода. Если клиент женат, то оплата закладной должна быть менее чем 60 % чистого дохода семьи.

В продукционных правилах мы будем использовать соединители И, ИЛИ и НЕ.

1. ЕСЛИ клиент имеет постоянную работу

   И клиент имеет адекватный доход

И имущество приемлемо

И клиент имеет хороший кредитный рейтинг

И величина кредита меньше 80% стоимости имущества

И клиент имеет 20% стоимости имущества в кассе

ТО одобрить кредит.

2. ЕСЛИ клиент имеет адекватные активы

И клиент имеет адекватный доход

И имущество приемлемо

И клиент имеет хороший кредитный рейтинг

И величина кредита меньше 80% стоимости имущества

И клиент имеет 20% стоимости имущества в кассе

ТО одобрить кредит.

3. ЕСЛИ клиент имеет работу

И клиент имел более чем в течении двух лет эту работу

ТО клиент имеет постоянную работу.

4. ЕСЛИ имущество находится в зоне расположения банка

ИЛИ имущество в запрещенном списке

ТО имущество приемлемо.

5. ЕСЛИ доход имеется и адекватен

ИЛИ доход одиночки адекватен

ТО клиент имеет адекватный доход.

6. ЕСЛИ клиент женат

И оплата закладной меньше 60% чистого дохода семьи

ТО доход семьи адекватен.

7. ЕСЛИ клиент не женат

И оплата закладной меньше 70% чистого дохода клиента

ТО доход одиночки адекватен.

8. ЕСЛИ стоимость имущества клиента больше десятикратной величины кредита

ИЛИ ликвидные активы клиента больше пятикратной величины кредита

ТО клиент имеет адекватные активы.

Популярность продукционных правил объясняется несколькими причинами:

1. Большинство человеческих знаний можно представить в виде правил продукций.

2. Модульность продукции позволяет добавлять в систему новые продукции без изменения прежних.

3. Продукции могут реализовать любые алгоритмы и, следовательно, любые процедурные знания.

4. Параллелизм и асинхронность продукций делают их удобной моделью вычислений, отвечающей подобным требованиям новых поколений ЭВМ.

Недостаток продукции состоит в том, что при их большом количестве становится трудоемкой проверка непротиворечивости системы продукций, – например, при добавлении новых правил.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что является фактом в продукционном правиле?
2. Какая часть правила называется посылкой, какая выводом?
3. Что означает гипотетический силлогизм?
4. Чем объясняется популярность продукционных правил?
5. В чем состоит недостаток продукций?
6. Какова классификация ядер продукции?

ЗАДАНИЕ

Представить знания о заданной предметной области в виде набора правил (не менее 20).

1. Диагностика неисправностей персонального компьютера. Экспертная система должна исследовать ситуацию и попытаться определить на общем уровне, допускает ли ошибки пользователь или, действительно, имеется

неисправность в системном блоке, на диске, в мониторе и т.д. Возможный путь проектирования – беседа с мастером-профессионалом. При оценке ситуации подразумеваются грубые функциональные тесты, без глубокого анализа электронных элементов.

2. Подбор субоптимальной конфигурации персонального компьютера с учетом субъективных и объективных потребностей заказчика.
3. Подбор субоптимальной конфигурации локальной компьютерной сети с учетом множества эксплуатационных, финансовых и прочих важных критериев.
4. Диагностика широко распространенных заболеваний человека по совокупности симптомов. Диагностируется не менее 20 болезней с учетом 15 типовых симптомов. Каждый симптом может указывать на несколько болезней (возможно, с разной степенью уверенности).
5. Консультация в отношении покупки автомобиля с учетом субъективных факторов, объективных потребностей и платежеспособности клиента, сезона и др.
6. Консультация в отношении покупки недвижимости с учетом связанных с этим важных факторов (надежность продавца, платежеспособность покупателя, страхование сделки, изменение цен и банковских процентных ставок и др.).
7. Диагностика причины зависания в процессе загрузки операционной системы ПЭВМ.
8. Подбор членов экипажа с учетом их психологической совместимости и возраста. При этом учитываются индивидуальные особенности претендентов, пожелания тех или иных кандидатов работать вместе, а также их возраст (командир должен быть старше всех остальных членов экипажа).
9. Информация об авиалиниях, связывающих Москву с некоторыми городами РФ. С какими городами Москва связана прямыми беспосадочными авиалиниями: с указанием всех городов, из которых

имеются обратные прямые беспосадочные рейсы.

10. Консультация в отношении приема лекарств при болезнях родственных вирусной инфекции или гриппу. В зависимости от конкретных симптомов заболевания решить, обращаться ли к врачу.

ПРАКТИЧЕСКАЯ РАБОТА № 5.

СЕМАНТИЧЕСКИЕ СЕТИ

Разработка семантических сетей относится к 1960 г., когда они использовались для моделирования обработки естественного языка.

"Семантика" определяется как значение, смысл слова, переданные с помощью каких-либо представлений и выражений. Другими словами, семантика означает определенные отношения между символами и объектами, представленными этими символами.

Семантическая сеть [6] – структура для представления знаний в виде узлов, соединенных дугами. Самые первые семантические сети были разработаны в качестве языка-посредника для систем машинного перевода, а многие современные версии по своим характеристикам до сих пор сходны с естественным языком. Однако последние версии семантических сетей стали более мощными и гибкими, они составляют конкуренцию фреймовым системам, логическому программированию и другим языкам представления.

С конца 50 гг. были созданы и использованы на практике десятки вариантов семантических сетей. Несмотря на то, что терминология и их структура различаются, существуют элементы сходства, присущие практически всем семантическим сетям:

1) узлы семантических сетей представляют собой концепты предметов, событий, состояний;

2) различные узлы одного концепта относятся к различным значениям, если не помечено, что они относятся к одному концепту;

3) дуги семантических сетей создают отношения между узлами-концептами (пометки над дугами указывают на тип отношения);

4) некоторые отношения между концептами представляют собой лингвистические падежи, – такие как агент, объект, реципиент и инструмент (другие означают временные, пространственные, логические отношения и отношения между отдельными предложениями);
5) концепты организованы по уровням в соответствии со степенью обобщенности – как, например, сущность, живое существо, животное, плотоядное.

Однако имеются и различия: понятие значения с точки зрения философии; методы представления кванторов общности и существования в логических операторах; способы манипулирования сетями и правила вывода, терминология. Все это у различных авторов варьируется. Несмотря не некоторые различия, сети удобны для чтения и обработки компьютером, а также достаточно мощны, чтобы представить семантику естественного языка.

Семантические сети предназначены для семантического моделирования реальной действительности с возможностью представления как экстенсиональной, так и интенсиональной информации. Семантическую сеть можно рассматривать как маркированный ориентированный граф с помеченными узлами и дугами. Узлам соответствуют некоторые объекты, а дугам – семантические отношения между этими объектами. Метки, приписываемые узлам, выделяют множество рассматриваемых объектов и выступают в качестве их имени. В роли таких имен могут быть слова естественного языка. Метки, приписываемые дугам, соответствуют элементам множества отношений, заданных на графе.

В семантической сети могут быть выделены подграфы определенной структуры, называемые *высказываниями*. Каждый такой подграф представляет собой граф, корнем которого является предикатный узел, остальные узлы называются концептуальными. Высказывание является минимальной единицей информации, вводимой и хранящейся в семантической сети. Разделение узлов на предикатные и концептуальные возможно лишь в пределах одного высказывания. В структуре семантической сети один и тот же узел может быть

предикатным относительно одного высказывания и концептуальным относительно другого.

В качестве структурной модели долговременной памяти психолог Куиллиан предложил модель понимания смысла слов, получившую название TLC-модель (доступный механизм понимания языка). Для описания структуры долговременной памяти в ней была использована сетевая структура как способ представления семантических отношений между концептами (словами).

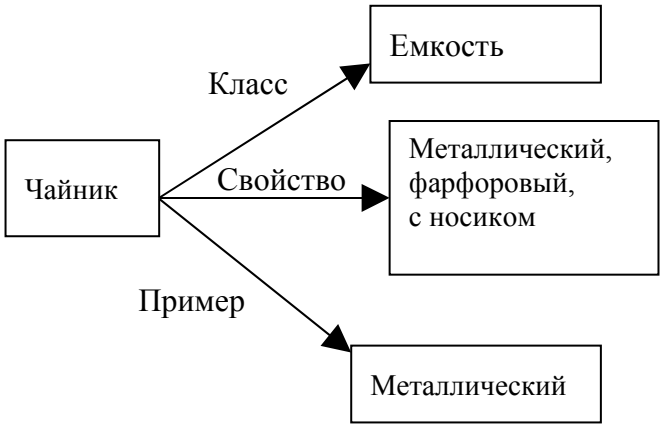


Рис.1. Образец семантической сети.

В модели Куиллиана концептуальные объекты представлены ассоциативными сетями, состоящими из вершин, показывающих концепты, и дуг, показывающих отношения между концептами. Подобная ассоциативная структура называется плоскостью, описываемые концепты объекта – называются вершинами типа, а связанные с ними соответствующие отдельные слова – вершинами лексем. В любой плоскости существует одна вершина типа и только необходимое для определения концептов, описывающих его, число вершин лексем. Вершины лексем определяют всевозможные сущности, имеющие место в реальном мире.

В семантических сетях используются четыре основных типа объектов: понятия, события, свойства и значения.

Понятия представляют собой сведения об абстрактных или физических объектах предметной области. Они могут быть заданы множеством доменов (параметров или констант).

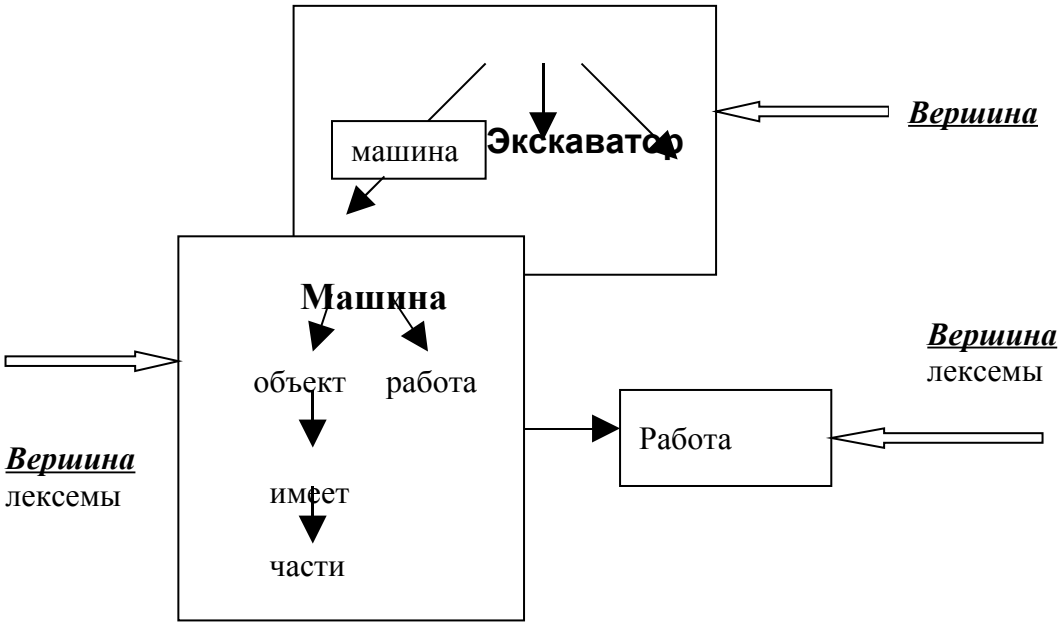


Рис.2. Представление знаний в долговременной памяти для TLC-моделей Куиллиана.

События – это действия, способные перевести предметную область из текущего состояния в некоторое новое. Можно перевести предметную область в определенное желаемое (целевое) состояние, поставить задачу, отыскать на семантической сети путь событий, приводящих к целевому состоянию.

Свойства используются для характеристики (уточнения) понятий, событий, процессов и т.д. Свойствами понятий могут быть цвет, размеры, качество, а свойствами действий – время, продолжительность, место и т.д.

Значения – это значения из какого-нибудь множества, которые могут принимать свойства.

Рассмотрим пример семантической сети (рис. 3), которая соответствует действию "Иванов руководит Петровым". Она представляется одним событием РУКОВОДИТ и двумя вершинами понятий ИВАНОВ и ПЕТРОВ. Дуги, соединяющие вершины-события и вершины-понятия, указывают роли понятий в событиях. Так, в событии РУКОВОДИТ понятие ИВАНОВ играет роль агента, а ПЕТРОВ – объекта воздействия. Свойством понятия ИВАНОВ является ДОЛЖНОСТЬ со знанием ПРОФЕССОР.

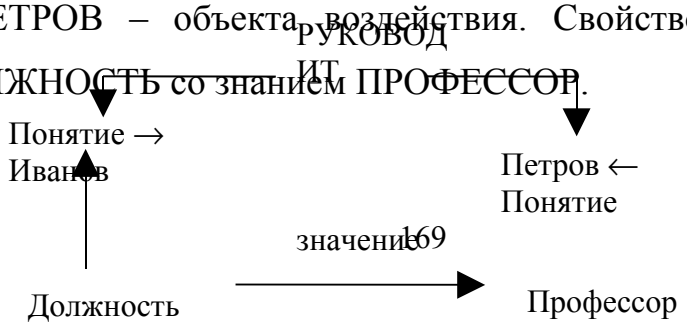


Рис. 3. Пример семантической сети

Основными связями для семантических сетей, с помощью которых формируются понятия, являются:

структурные – класс, к которому принадлежит данное понятие;

отличительные свойства, выделяющие понятие из всех прочих понятий этого класса;

примеры данного понятия;

связи типа "часть – целое" ("класс – подкласс", "элемент – множество" и т.п.);

функциональные связи (определяемые обычно глаголами "выполняет", "влияет"…);

количественные ("больше", "меньше", "равно");

пространственные ("близко", "далеко", "за", "перед", "внутри");

временные ("раньше", "позже", "во время");

атрибутивные ("имеет свойство", "имеет значение");

логические (И, ИЛИ, НЕ).

На самой СС принадлежность элемента к некоторому классу или части к целому передается с помощью связок "это есть" и "часть от" соответственно. Свойства описываются связками "есть" и "имеет". Отбор понятий и связок определяется предметной областью и назначением СИИ.

В качестве примера может быть приведена часть семантической сети, относящейся к понятию "фрукты".

Существует несколько классификаций семантических сетей, связанных с типами отношений между понятиями.

По количеству типов отношений:

однородные (с единственным типом отношений);

неоднородные (с различными типами отношений).

По типам отношений:

бинарные (отношения связывают два объекта).

N-арные (в которых есть специальные отношения, связывающие более двух понятий).

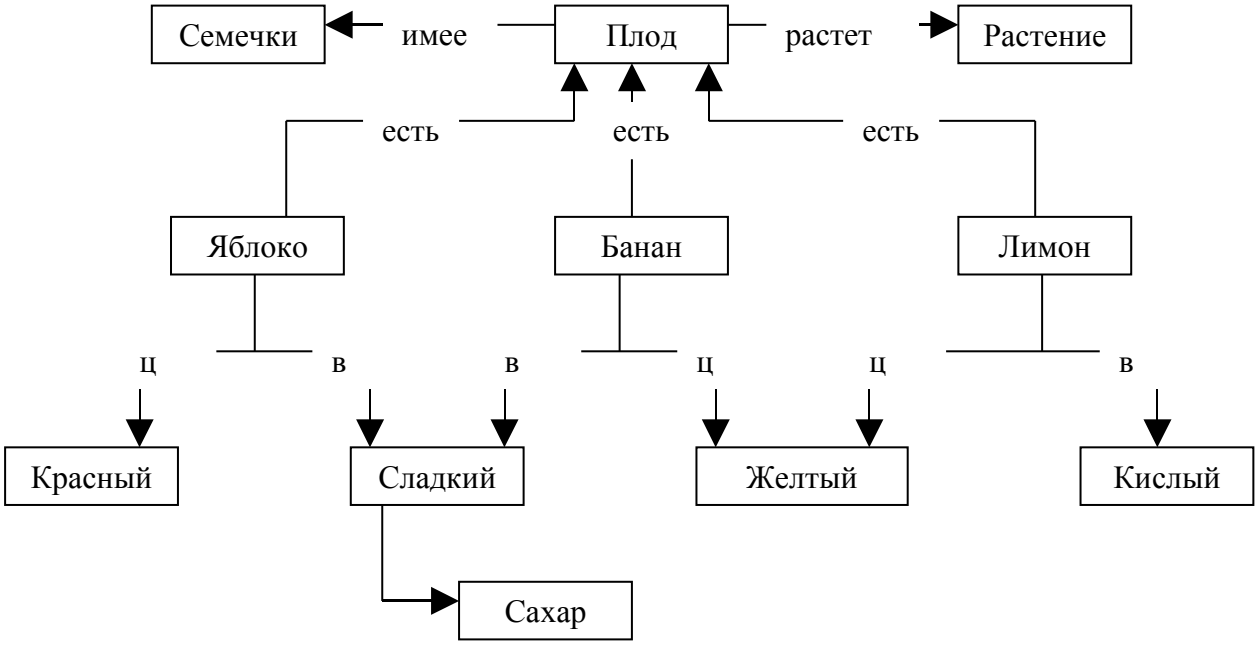


Рис. 4. Фрагмент семантической сети «Фрукты».

Правила построения семантических сетей.

Семантические сети получаются из концептуальных графов с помощью специальных правил.

Правило конъюнкции. Рассмотрим набор из трех фраз:

1 – я: Завод “Интеграл” производит Схему_14;

2 – я: “Интеграл” поставляет Схему_14 фирме “Луч”;

3 – я: Фирма “Луч” использует Схему_14 для сборки часов.

Каждую из этих фраз можно представить концептуальным графом. Используя правило конъюнкции, можно получить один концептуальный граф *g* из двух графов *g1* и *g2*.

Это правило состоит в следующем:

если узлы-концепты *c1* и *c2* соответственно в графах *g1* и *g2* идентичны, то граф *g* получается удалением *c2* и соединением с *c1* всех связывающих узлов, которые были связаны с c2 и *g2.*

Применим правило конъюнкции к приведенным выше трем фразам и получим семантическую сеть (рис. 5).

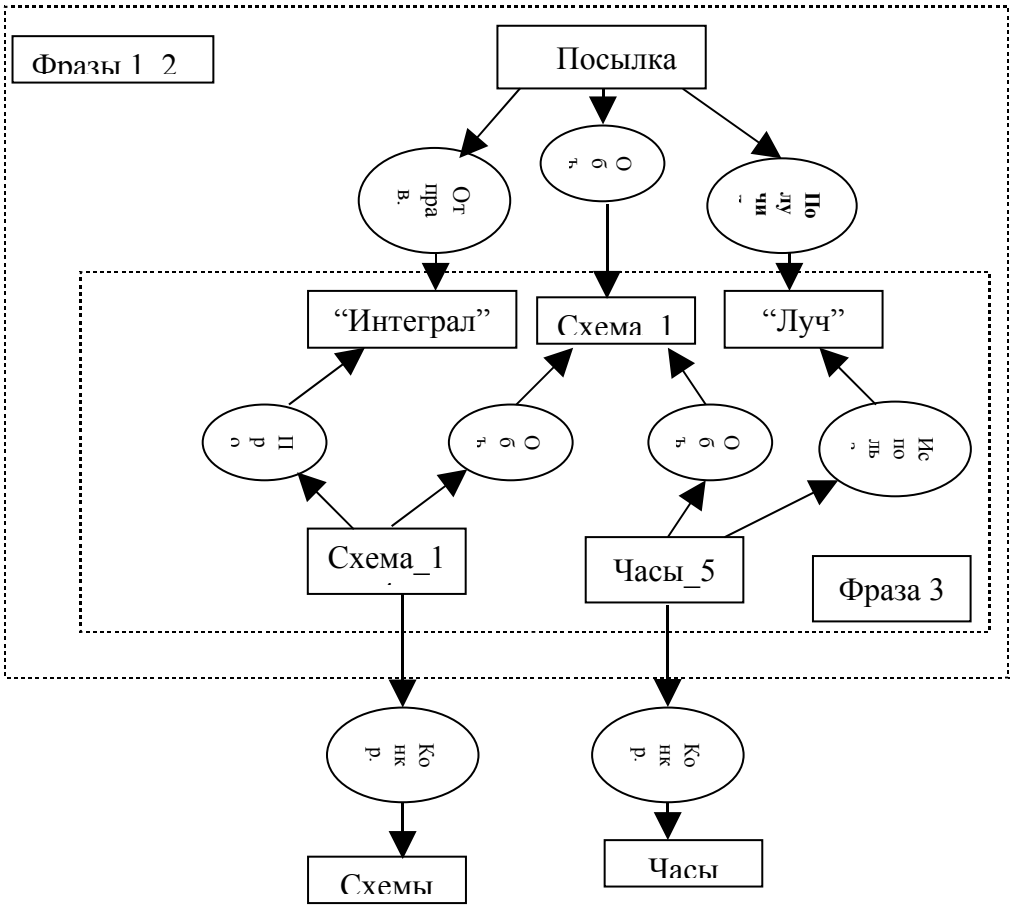


Рис.5. Концептуальный граф и контекстное представление

Правило упрощения. Если концептуальный граф *g* содержит два идентичных (соединенных с одними и теми же узлами-концептами) связывающих узла, то можно удалить один из них со связанными с ним стрелками.

Правило копирования. Граф *g* есть копия графа *g1*.

Использование типов. Отношение принадлежности типу представляется связывающим узлом КОНКР (от слова "конкретизация"). Так, "Схема_14" относится к типу "схема", что указано узлом КОНКР. Отношение принадлежности множеству представляется связывающим узлом ЭЛЕМ (от

слова "элемент"). Так, завод “Интеграл” принадлежит множеству заводов электроники ("электрозавод").

Отношение принадлежности подмножества некоторому множеству представляется узлом ПОДМН.

Связь между первым типом и более общим вторым типом осуществляется связующим узлом ЭТО.

Представление «совокупность – ссылка». Семантические сети образуется последовательностью трех узлов, соединенных так: за узлом, представляющим конкретное значение (Схема_14), следует связывающий узел КОНКР, сопровожденный узлом-совокупностью (схема). Эти три узла можно сгруппировать и изобразить одним узлом-прямоугольником, состоящим из двух полей: из «поля-совокупности», содержащего некоторый тип, и следующего за ним «поля ссылки», конкретизирующего тип из первого поля. Например,

Схема: Схема -	или просто	Схема: 14

В поле конкретизации может находиться переменная. Так, [схема: x] означает просто какой-то объект типа «схема». Отметим, что переменную можно просто опустить, условившись, что представление [схема] эквивалентно [схема: x].

Канонические графы. Синтаксически правильно построенные концептуальные графы могут семантически быть некорректными, как собственно и предложения, которым они соответствуют, т. е. некоторые комбинации узлов бессмысленны. Чтобы исключить графы нереальных (невозможных) ситуаций предметной области, определяют так называемые канонические (семантически корректные) графы разрешенных комбинаций слов. Следовательно, знания представляются каноническими графами исходя из личного опыта. Новые, канонические графы строятся из уже имеющихся по правилам: конъюнкции, упрощения, копирования и ограничения.

Правила конъюнкции были рассмотрены выше.

Правило ограничения заключается в следующем. Тип любого концепта с концептуального графа *g* можно записать неким подтипом, т.е.

конкретизировать концепт *c*. Проиллюстрируем это на примере концептуального графа, представленного на рис. 6.

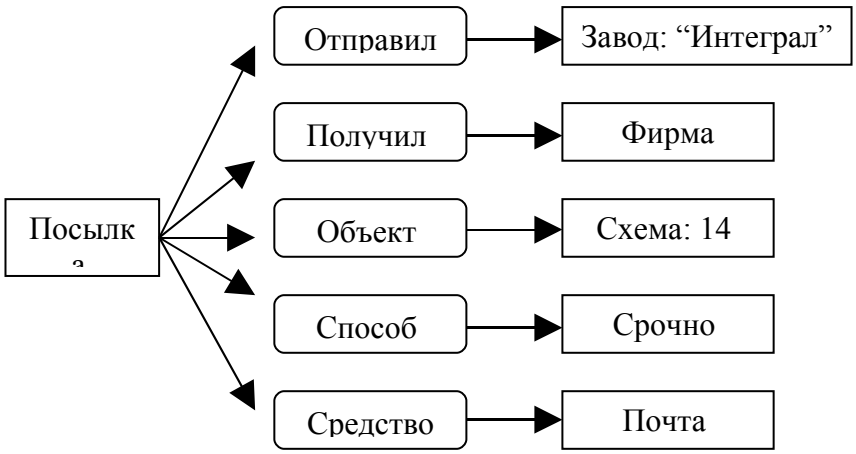


Рис. 6. Пример построения семантической сети.

Граф без элементов c(2) интерпретируется фразой ~Интеграл срочно поставляет схему кому-то ~. Граф без элементов c(1) означает ~Интеграл поставляет схему фирме “Луч” почтой~. Применения правила конъюнкции к этим двум графам дает весь граф на рис.6. Можно было бы применить к последнему графу правило ограничения, заменяя обозначенную *X* совокупность ~кому-то~ меньшей совокупностью ~часовые заводы~. Правило ограничения сужает концепты, а конъюнкции – добавляет условия на граф. На рис.7. представлен пример семантической сети для моделирования закладной под кредит.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. На чем основано представление знаний с помощью семантической сети?
2. Из чего состоит семантическая сеть?
3. Чем отличаетс… …ых моделей представления знаний?
4. Какие ти… …ошений используются в… …ки… …ссифицируются сем… …требуе… …кие с…

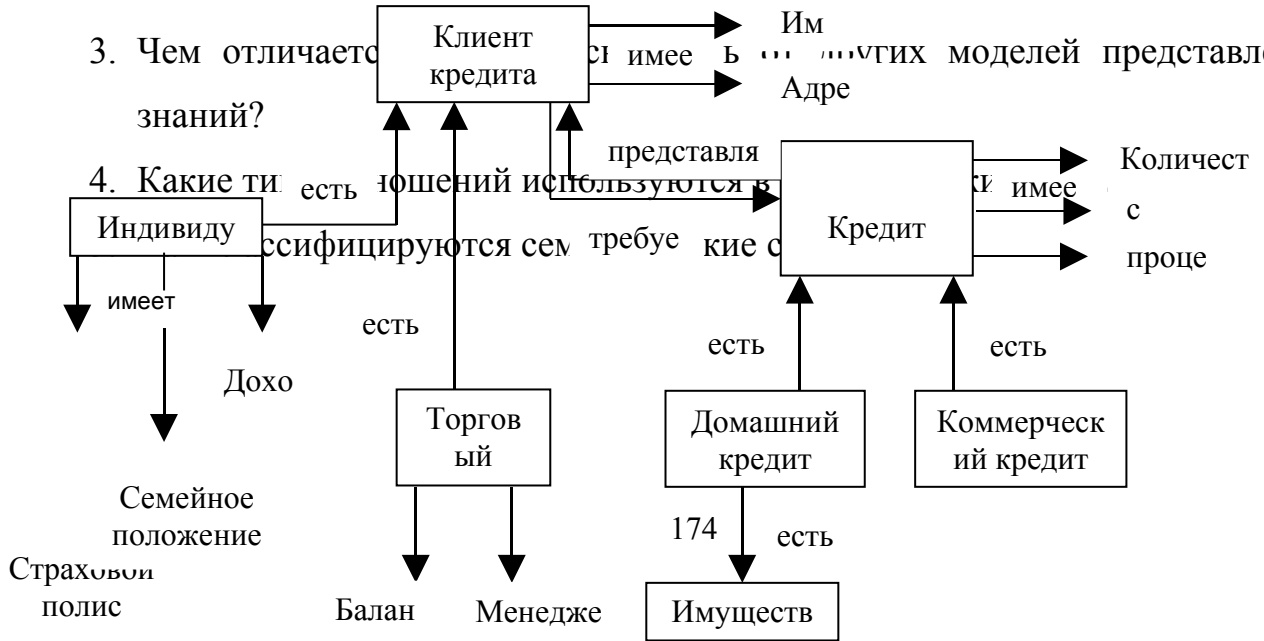


Рис. 7. Семантическая сеть для моделирования закладной под кредит.

ЗАДАНИЕ

Представить знания о заданной предметной области в виде семантической сети (по вариантам заданий, приведенных в практической работе № 4). Привести графическую интерпретацию.

ПРАКТИЧЕСКАЯ РАБОТА № 6

ПРЕДСТАВЛЕНИЕ ЗНАНИЙ ФРЕЙМАМИ И ВЫВОДЫ

Теорию фреймов опубликовал в 1975 г. М. Минский. Она относится к психологическим понятиям и касается способов понимания того, как мы воспринимаем (видим, слышим) явления, процессы, объекты и т.п.

В основе теории фреймов лежит восприятие фактов посредством полученной извне информации о некотором явлении с уже имеющимися данными, накопленными опытным путем или полученными в результате вычислений. Когда человек попадает в новую ситуацию, он вызывает из своей памяти основную структуру, называемую фреймом. *Фрейм* (рамка) – это единица представления знаний, заложенных в прошлом, детали которой могут быть изменены согласно текущей ситуации. Класс некоторых объектов (процессов) может определяться одним типичным (базовым) объектом, включающим наиболее существенные характеристики объектов данного класса. Так, некоторую характеристику объекта можно представить тройкой (Объект, атрибут_ j значение_ j).

Собрав все тройки, касающиеся данного объекта, получим объектное представление области рассуждений относительно данного объекта. Общая форма этого представления такова:

Объект(атрибут_i значение _ i), j= 1 ,...,*m*

Таким образом, вместо построения различных независимых формул строим более крупную структуру полной информации об объекте, которую называют фреймом. Если требуется информация о некотором объекте, то обращаются к соответствующему фрейму, внутри которого находятся свойства и факты относительно рассматриваемого объекта.

Фрейм содержит как информационные, так и процедурные элементы, которые обеспечивают преобразование информации внутри фрейма и связи его с другими фреймами. Элементами фрейма являются так называемые слоты, которые могут быть пустыми и заполняться в процессе активизации фрейма в соответствии с определенными условиями. Таким образом, фреймы представляют собой декларативно-процедурные структуры, т.е. совокупность описаний и (возможно) связанных с ними процедур, доступ к которым выполняется прямо из фрейма.

Структура данных фрейма

Фрейм можно представить в виде таблицы (отношения), строки которой соответствуют слотам фрейма, а столбцы – атрибутам (характеристикам) объекта. Имя таблицы является именем фрейма, и оно уникально. Каждый слот содержит следующие атрибуты: имя слота наследования, указатель наследования, тип данных, значение слота, демон.

ТИП ДАННЫХ определяет, что слот либо имеет численное значение, либо является именем другого фрейма. К типам данных относится: FRAME (указатель); INTEGER (целый); REAL (действительный); BOOL (булев); LISP (присоединенная процедура); TEXT (текст); LIST (список); TABLE (таблица); EXPRESSION (выражение) и др.

ЗНАЧЕНИЕ СЛОТА соответствует указанному типу данных этого слота; кроме того, должно выполняться условие наследования. Значениями

слотов могут быть имена других фреймов, что обеспечивает связь (вложенность) между фреймами.

ДЕМОН – автоматически запускаемая процедура при обращении к слоту и выполнении некоторого условия. Например, демон типа IF-NEEDED запускается, если в момент обращения к слоту его значение не было установлено; IF-ADDED запускается при подстановке в слот значения, IF-REMOVED – при стирании значения слота.

Присоединенная процедура может использоваться в качестве значения слота и запускаться по сообщению, переданному из другого фрейма. Когда мы говорим, что фреймы, как модели представления знаний, объединяют декларативные и процедурные знания, то считаем демоны и присоединенные процедуры процедурными знаниями.

Фрейм состоит из произвольного числа слотов, среди которых имеются системные слоты и слоты, определяемые пользователем. Каждый слот характеризуется определенной структурой и уникальным именем внутри данного фрейма. В качестве системных могут быть, например, определены следующие слоты: указатель фрейма-родителя, указатель прямого дочернего фрейма, пользователь фрейма, дата определения фрейма и его последней модификации, а также некоторые другие. Системные слоты используются при редактировании баз данных и управлении выводом.

Понятие «наследование свойств» позволяет фреймам заимствовать свойства, которые имеют другие фреймы. УКАЗАТЕЛИ НАСЛЕДОВАНИЯ касаются только фреймовых систем иерархического типа, основанных на отношениях «абстрактное – конкретное». Они показывают, какую информацию об атрибутах слотов во фрейме верхнего уровня наследуют слоты с такими же именами во фрейме нижнего уровня. Типичные указатели наследования:

U (unique – уникальный); *S* (same – такой же); *R* (range – установление границ); *O* (override – переопределить) и др. Указатель наследования *U* показывает, что каждый фрейм должен иметь слоты с различными значениями; *S* – все слоты имеют одно значение; *R* – значения слотов фрейма нижнего

уровня должны находиться в пределах указанных значений слотов фрейма верхнего уровня; *О* – при отсутствии указания значение слота фрейма верхнего уровня становится значением слота фрейма нижнего уровня.

Свойства фреймов

Рассмотрим основные свойства фреймов.

1. Базовый тип. Базовые фреймы используются для указания наиболее важных объектов, позволяют добиться быстрого понимания сущности данного предмета. На основании базовых фреймов строятся фреймы для новых состояний. При этом каждый фрейм содержит слот-указатель подструктуры, что позволяет различным фреймам совместно использовать одинаковые части.

2. Процесс сопоставления. Во фреймовой системе осуществляется поиск фрейма, который соответствует (релевантен) цели (данной ситуации). Другими словами, сопоставляются значения (ограничения) слота фрейма во фреймовой системе со значениями атрибутов цели. Процесс сопоставления осуществляется следующим образом:

а) сначала с помощью предложения и интуиции выбирается некоторый базовый фрейм с учетом выявленных особенностей, релевантности, т.е. посредством подфреймов. Данный фрейм подтверждает или не подтверждает свою релевантность. При этом в соответствии с текущей целью определяется, какое ограничение слота следует использовать при сопоставлении. Если фрейм подходит, то процесс сопоставления завершается, в противном случае выполняется пункт б);

б) если в данном фрейме имеется слот, значение которого отрицательно влияет на сопоставление, то нужно присвоить слоту надлежащее значение;

в) если два предыдущих шага не дают результата, то управление передается другому надлежащему фрейму этой или другой фреймовой системы. Если последнее сопоставление заканчивается безрезультатно, то задача не имеет решения.

3. Иерархическая структура. Фрейм обычно соответствует некоторой иерархической структуре, особенность которой состоит в том, что значения

атрибутов фрейма верхнего уровня совместно используются всеми фреймами нижних уровней, связанных с верхними (рис. 8). Такая структура позволяет удобно систематизировать и записать схожие понятия, добавлять новые понятия или знания в соответствующие позиции иерархии, упрощает обнаружение противоречий в знаниях, просмотр знаний и делает фреймовую систему более гибкой.

Пример 1. Рассмотрим пример фрейма в системе моделирования процесса выдачи банковского кредита под залог (рис.8). Эта система содержит фрейм «заемщика», который представляет собой класс клиентов и принадлежит к суперклассу заемщиков кредита. Он содержит характеристики, относящиеся к человеку-клиенту, такие как цель использования кредита, информацию об обеспечении клиента, его ответственности, имуществе и т. д.

Наследование свойств у фреймов представляется отношениями каждого фрейма с другими фреймами [illegible]ивидуальный клиент" имеет в качестве его суперкл[illegible]аследует его характеристики.

Клиент кредита

Имя:
Адрес:
Цель кредита:

Важным является то, [illegible]кную структуру данных, в которой атрибуты фрейма сами являются фреймами. Например, "[illegible] фрейма [illegible]яется ф[illegible]бутами.

Индивидуальный кредит
Прилагается к закладной (фрейм)

Обеспечивает документы кредита (фрейм)

Ответственность: оплата кредита

Залог домашний (фрейм)

Суперкласс

Залог под кредит

Количество

Качество

Ставка

Тип: фиксированный или переменный

Месячная оплата: вычисленная исходя из количества, ставки и срока

Документы кредита

Домашнее имущество

Рис.8. Пример фрейма

В фреймовых системах отсутствует специальный механизм управления выводом, поэтому пользователь может реализовать данный механизм с помощью присоединенной процедуры. В то же время это дополнительная нагрузка для пользователя, поэтому язык представления знаний фреймами ориентирован на специалистов по искусственному интеллекту.

Основной причиной того, что фреймовые системы являются наиболее благоприятной средой для исследований по искусственному интеллекту, следует считать то, что они в целом соответствуют многочисленным требованиям, касающимся представления знаний:

1. Для систематизированного управления сложными знаниями большого объема желательно организовать эти знания на основе концептуальных объектов.
2. В целях увеличения гибкости системы следует сделать возможным представление в виде комбинации декларативных и процедурных знаний.

3. Объекты представляют собой иерархическую структуру.

Существуют три способа управления выводом во фреймовой системе:

1. Способ, при котором фреймовая система используется как база данных, основанная на фактах.
2. Когда во фреймовой системе ограничена роль присоединенной процедуры. Фреймовая система используется внешней базой правил и механизмом ввода и отвечает на сообщения. Этот способ применяется мощными продукционными системами с базой данных фреймового типа, снабженной механизмом наследования и демона.
3. Способ с обменом сообщениями. Основан на постепенном продвижении к цели посредством поочередной передачи сообщений между фреймами. Однако способ требует тщательного проектирования всей системы.

Параллельно с языками фреймов существуют объектно-ориентированные программные языки, используемые для составления программ , но имеющие некоторые свойства языков фреймов (использование слотов для детальной, доскональной классификации объектов и пр.). Отличие их от языков фреймов в том, что фреймовые языки направлены на более обобщенное представление информации об объекте.

Одна из трудностей представления знаний и языка фреймов – является отсутствие формальной семантики. Это затрудняет сравнение свойств представления знаний различных языков фреймов, а также полное логическое объяснение языка фреймов.

Пример 2. Знания о предметной области представлены в виде фреймовой структуры.

Фрейм "Компьютер" – фрейм класса

Слот наследования	Указатель наследования	Тип данных	Значение слота	Демон
Слот наследования	Уникальные атрибуты	Родительский фрейм	Встречаются впервые	Родителей не имеет, является базовым фреймом
Описание	Уникальные	Текст	Встречают	Компьютер – класс

	атрибуты		ся впервые	ПВЭМ
Свойства компьютера	Уникальные атрибуты	Текст	Встречаются впервые	Имеет монитор, клавиатуру, мышь, МП, материнскую плату, элементы памяти, винчестер

Фрейм "Работа в Internet" – фрейм шаблон

Слот наследования	Указатель наследования	Тип данных	Значение слота	Демон
Слот наследования	Уникальные атрибуты	Указатель на родительский фрейм	Унаследовано от фрейма "Компьютер"	Родитель – фрейм компьютер
Описание	Уникальные атрибуты	Текст	Унаследовано от фрейма "Компьютер"	Компьютер – класс ПЭВМ
Свойства компьютера	Унаследованные атрибуты	Текст	Унаследовано от фрейма "Компьютер"	Имеет монитор, клавиатуру, мышь, МП, материнскую плату, элементы памяти, винчестер.
Свойства мультимедиа-компьютера	Унаследованные атрибуты	Указатель на фрейм "Мультимедиа-компьютер"	Унаследовано от фрейма "Мультимедиа-компьютер"	Имеет звуковую карту, колонки,CD-ROM.
Свойства работы в Internet	Уникальные атрибуты	Текст	Встречаются впервые	Имеет модем

Фрейм "Принтер" – фрейм шаблон

Слот наследования	Указатель наследования	Тип данных	Значение слота	Демон
Слот наследования	Уникальные атрибуты	Указатель на родительский фрейм	Унаследовано от фрейма "Компьютер"	Родитель – фрейм компьютер
Описание	Уникальные атрибуты	Текст	Унаследовано от фрейма "Компьютер"	Компьютер – класс ПЭВМ

Свойства компьютера	Унаследованные атрибуты	Текст	Унаследовано от фрейма "Компьютер"	Имеет монитор, клавиатуру, мышь, МП, материнскую плату, элементы памяти, винчестер.
Свойства графической станции	Унаследованные атрибуты	Указатель на фрейм "Графическая станция"	Унаследовано от фрейма "Графическая станция"	Имеет 3D-видеокарту, световое перо, записывающий CD-ROM
Свойства принтера	Уникальные атрибуты	Текст	Встречаются впервые	Имеет принтер

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое фрейм?
2. Какие основные свойства фреймов вы знаете?
3. Какова структура данных фрейма?
4. Что такое «наследование» свойств?
5. Как осуществляется механизм вывода во фрейме?
6. Какую роль играет иерархическая структура во фрейме?

ЗАДАНИЕ

Представить знания о заданной предметной области в виде фреймовой структуры (по вариантам заданий приведенных в практической работе № 4).

Литература

1. Рыжиков Ю.И. Информатика. Лекции и практикум. – СПб., Корона принт, 2000. – 256 с.
2. Змитрович А.И. Интеллектуальные информационные системы. – Минск, ТетраСистемс, 1997. – 367 с.
3. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний нтеллектуальных систем/ Учебник для технических вузов – СПб., Питер, 2000. – 384 с.

4. Шемакин Ю.И. Введение в информатику. – М., Финансы и статистика, 1985. – 190 с.
5. Гетманова А.Д. Учебник по логике – М.: ЧеРо, 2000.
6. Представление и использование знаний: Пер. с япон./Под ред. Х. Уэно, М. Исидзука. - М., Мир, 1989. – 220 с.

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОРГАНИЗАЦИИ МЕЖСЕССИОННОГО КОНТРОЛЯ ЗНАНИЙ СТУДЕНТОВ

1. Межсессионная аттестация студентов проводится дважды в семестр на 7 и 13 неделях 5-го семестра.

2. Аттестационная оценка складывается из оценок, полученных по результатам промежуточного тестирования.

3. Организация аттестации студентов, проводится в соответствии с положением АмГУ о курсовых экзаменах и зачетах.

10. ФОНД ТЕСТОВЫХ ЗАДАНИЙ ДЛЯ ОЦЕНКИ КАЧЕСТВА ЗНАНИЙ ПО ДИСЦИПЛИНЕ

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Тестовые задания по проверке остаточных знаний по дисциплине
«Методы искусственного интеллекта»
для специальностей 230102 – «Автоматизированные системы обработки информации и управления»
Утверждено на заседании кафедры Кафедра ИУС

«23» октября 2006 г. Заведующий кафедрой

«УТВЕРЖДАЮ» _____ А.В.Бушманов

Инструкция: Тест состоит из 20 заданий, время тестирования – 40 минут.

ВАРИАНТ № 1

1. Проблемами искусственного воспроизведения тех структур и процессов, которые характерны для живого человеческого мозга и которые лежат в основе процесса решения задач человеком занимается:

1) программно – прогматическое направление;
2) бионоческое направление;
3) нейрофизиологическое направление;
4) программное направление.

2. Контактный, процедурный и когнитивные слои – это проблемы извлечения знаний:

1) психологического аспекта; 2) лингвистического аспекта;
3) гносеологического аспекта; 4) когнитивного аспекта.

3. Выявление знаний из источников, преобразование знаний в нужную форму и перенос знаний в базу знаний искусственного интеллекта называется:

1) извлечением знаний; 2) приобретением знаний;
3) формированием знаний. 4) получением знаний.

4. Если представление знаний обладает способностью распознавать все отличия, которые Вы закладываете в исходную сущность, это означает:

1) побочный эффект; 2) естественность нотации;
3) логическую адекватность; 4) эвристическую мощность.

5. Алгоритм, который отыскивает решение, путь к которому на графе – кратчайший, если таковое существует, называется:

1) пространством решений; 2) алгоритмом поиска в ширину:
3) алгоритмом поиска в глубину; 3) комбинаторным взрывом.

6. Традиционно семиотика включает совокупность правил построения языка или отношения между знаками. Такой раздел семиотики называется:

1) синтаксисом; 2) семантикой;
3) прагматикой; 4) пирамидой знаний.

7. Одна из возможных классификаций людей по психологическим характеристикам делит всех на несколько типов. …. ориентированы на интеллектуальную работу, учебу, теоретические обобщения и обладают такими характеристиками когнитивного стиля, как поленезависимость и рефлексивность:

1) собеседники; 2) мыслители;
3) практики; 4) теоретики.

8. Методы извлечения знаний, которые охватывают методы и процедуры контактов инженера по знаниям с непосредственным источником знаний – экспертом, называются:

1) текстологические методы; 2) анализ документов;

3) анализ литературы; 4) коммуникативные методы.

9. Выполнение Пролог – программы есть:

1) доказательство теорем с использованием логического аспекта языка;
2) вывод следствий из программы;
3) обоснование противоречий между поставленным вопросом и множеством фактов;
4) выполнение цели.

10. Декларативный смысл Пролог – программы определяет:

1) что должно быть результатом программы;
2) как результат был достигнут;
3) как результат будет достигнут;
4) что будет результатом.

11. Совокупность фактов в Прологе называют:

1) базой знаний; 2) базой данных;
3) базой фактов; 4) базой фактов и правил.

12. Если при изъятии отсечения из программы изменился ее декларативный смысл, то такое отсечение называют:

1) зеленым; 2) красным;
3) белым; 4) черным.

13. Функция от любого числа аргументов, принимающая истинностные значения: истинно (1) и ложно (0). Аргументы принимают значения из произвольного, конечного или бесконечного множества D, называемого предметной областью, называется:

1) правильно построенной формулой; 2) предикатом;
3) атомом; 4) пропозициональной формой.

14. Укажите правильный вариант высказывания "Если у меня будет отпуск летом и я куплю автомобиль и катер, то поеду отдыхать в Крым или на Кавказ", записанного с использованием логики предикатов первого порядка:

1) $(a \vee b \vee c) \leftrightarrow (d \vee e)$. 2) $(a \wedge b \wedge c) \rightarrow (d \wedge e)$.
3) $(a \vee b \wedge c) \leftrightarrow (d \vee e)$. 4) $(a \wedge b \wedge c) \rightarrow (d \vee e)$.

15. Укажите правильный вариант высказывания "Некоторые элементарные частицы имеют положительный заряд" записанного с использованием логики предикатов первого порядка:

1) $(\forall X)(S(X) \vee P(X))$ 2) $(\forall X)(S(X) \wedge P(X))$
3) $(\exists X)(S(X) \wedge P(X))$ 4) $(\exists X)(S(X) \vee P(X))$

16. Восприятие фактов посредством полученной извне информации о некотором явлении с уже имеющимися данными, накопленными опытным путем или полученными в результате вычислений. Когда человек попадает в новую ситуацию, он вызывает из своей памяти основную структуру, называемую:

1) правилом продукции; 2) логической единицей;
3) фреймом; 4) семантической сетью.

17. Заключение одного правила является посылкой другого правила, и можно установить третье правило с посылкой из первого правила и заключением из второго. Другими словами это есть:

1) модус поненс;
2) гипотетический силлогизм;
3) предложение Хорна;
4) ядро правила.

18. X является элементом списка, если в соответствии с первым предложением X – это голова списка, или в соответствии со вторым предложением X – это элемент хвоста:

1) member(X,[X|_]).
   member(X,[_|Y]):- member(X,Y).
2) member(X,[]).
   member(X,[X|Y]):- member(X,Y).
3) member(_,[_|Y]).
   member(X,[X|_]):- member(X,Y).
4) member(_,[_|Y]).
   member(X,[_|_]):- member(X,Y).

19. Отношение на Прологе имеет вид:

```
DOMAINS
LIST = INTEGER*
DATABASE - dba1
fact1(INTEGER,STRING,LIST)
CLAUSES
fact1(1,"fact1",[1,2,3]).
fact1(2,"fact2",[1,3]).
```

Определить ответ на следующий запрос: Goal: retract(X,dba1)

1) X=fact1(1,"fact1",[1,2,3])
   X=fact1(2,"fact2",[1,3])
   2 Solutions
2) X=fact1(1,"fact1",[1,2,3])
   1 Solutions
3) X=fact1(2,"fact2",[1,3])
   1 Solutions
4) X=fact1(1,"fact1",[3,2,1])
   1 Solutions

20. Отношение на Прологе имеет вид:

```
constants
bgi_path=""
predicates
win(integer,integer)
clauses
win(X,Y):-S=30,C=S+5,
makewindow(1,7,15,"Окно1",Y,X,6,S),
makewindow(2,7,12,"Окно2",Y,C,6,S),
readln(_), removewindow,hiftwindow(1).
```

Определить ответ на следующий запрос: win(5,6)

1) Окно 1 – белого цвета, Окно 2 – красного цвета, курсор в первом окне.
2) Окно 1 – красного цвета, Окно 2 – белого цвета, курсора нет в окнах.
3) Окно 1 – белого цвета, Окно 2 – красного цвета, курсор во втором окне.
4) Окно 1 – синего цвета, Окно 2 – белого цвета, курсор в первом окне.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Тестовые задания по проверке остаточных знаний по дисциплине
«Методы искусственного интеллекта»
для специальностей 230102 – «Автоматизированные системы обработки информации и управления»

Утверждено на заседании кафедры
«23» октября 2006 г.
кафедрой
«УТВЕРЖДАЮ» _____
А.В.Бушманов

Кафедра ИУС
Заведующий

Инструкция: Тест состоит из 20 заданий, время тестирования – 40 минут.

ВАРИАНТ № 2

1. Бионическое направление развития искусственного интеллекта не изучает:

1) нейробионический подход;
2) нейронные сети;
3) структурно – эвристический подход;
4) алгоритмические языки.

2. Глобальной схемой, которая может быть положена в основу концептуального анализа структуры знаний любой предметной области является:

1) иерархия понятий;
2) иерархия абстракций;
3) концептуализация.
4) пирамида знаний.

3. Общий код, понятийная структура, словарь пользователя – это составные части инженерии знаний:

1) психологического аспекта;
2) лингвистического аспекта;
3) гносеологического аспекта;
4) когнитивного аспекта.

4. Если наряду с наличием выразительного языка представления знания существует некоторое средство использования представлений, сконструированных и интерпретируемых таким образом, чтобы с их помощью можно было решить проблему, это означает:

1) побочный эффект;
2) естественность нотации;
3) логическую адекватность;
4) эвристическую мощность.

5. Множество решений, которые удовлетворяют условию проверять, не является ли образовавшееся состояние конечным решением, иногда называют:

1) пространством решений;
2) алгоритмом поиска в ширину:
3) алгоритмом поиска в глубину;
3) комбинаторным взрывом.

6. Традиционно семиотика включает связь между элементами языка и их значениями или отношения между знаками и реальностью. Такой раздел семиотики называется:

1) синтаксисом;
2) семантикой;
3) прагматикой;
4) гештальтпсихологией.

7. Одна из возможных классификаций людей по психологическим характеристикам делит всех на несколько типов. …. – это общительные, открытые люди, готовые к сотрудничеству:

1) собеседники;
2) мыслители;
3) практики;
4) теоретики.

8. Методы извлечения знаний, которые включают методы извлечений знаний из документов, методик, пособий, и специальной литературы, называются:

1) текстологические методы;
2) пассивные методы;
3) активные методы;
4) коммуникативные методы.

9. Простейшим видом утверждения на Прологе является:

1) вопрос;
2) факт;
3) правило;
4) отношение.

10. У языков логического программирования основным отличием от классических алгоритмических языков является то, что:

1) алгоритмы получения определенных результатов непосредственно не задаются;
2) имеется наличие отношения релевантности информационных единиц;
3) имеется наличие подмножества формальной логики;
4) алгоритмы получения определенных результатов задаются непосредственно.

11. Семантика, определяющая результат работы программы с учетом действия программы, называется:

1) декларативной;
2) процедурной;
3) примитивной;
4) алгоритмической.

12. Язык Пролог, основанный на логике предикатов первого порядка, манипулирует предложениями Хорна, которые имеют вид:

1) $\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_n \vee P_m$;
2) $\neg P_1 \vee P_1 \vee \neg P_2 \vee P_2 \vee \ldots \vee \neg P_n \vee P_n$;
3) $\neg P_1 \wedge \neg P_2 \wedge \ldots \wedge \neg P_n \wedge P_m$.
4) $\neg P_1 \wedge P_1 \vee \neg P_2 \wedge P_2 \vee \ldots \vee \neg P_n \wedge P_n$;

13. Всякое высказывание, составленное из некоторых исходных высказываний посредством логических операций, называют:

1) формулой алгебры логики;
2) правильно построенной формулой;
3) предикатом;
4) атомом.

14. Укажите правильный вариант высказывания "Если у меня будет свободное время и я сдам экзамены по математике и физике, то поеду отдыхать в Крым или на Кавказ", записанного с использованием логики предикатов первого порядка:

1) $(a \vee b \vee c) \leftrightarrow (d \vee e)$.
2) $(a \wedge b \wedge c) \rightarrow (d \wedge e)$.
3) $(a \vee b \wedge c) \leftrightarrow (d \vee e)$.
4) $(a \wedge b \wedge c) \rightarrow (d \vee e)$.

15. Укажите правильный вариант высказывания "Некоторые рыбы являются китами" записанного с использованием логики предикатов первого порядка:

1) $(\forall X)(S(X) \vee P(X))$
2) $(\forall X)(S(X) \wedge P(X))$
3) $(\exists X)(S(X) \wedge P(X))$
4) $(\exists X)(S(X) \vee P(X))$

16. Какой из компонентов не является основным в продукционной системе:

1) набор правил;
2) рабочая память;
3) логическая единица;
4) механизм логического вывода.

17. Структура для представления знаний в виде узлов, соединенных дугами, называется:

1) правилом продукции;
2) логикой предикатов первого порядка;
3) фреймом;
4) семантической сетью.

18. Предикат длина_списка(L,N) истинен, если список L содержит N элементов.

1) длина_списка([],0).
   длина_списка(H,T,N):- длина_списка(T,M), N = M+1 .
2) длина_списка([],0).
   длина_списка([X|L],N):- длина_списка(L,M), N=M+1.
3) длина_списка([_],0).
   длина_списка([X|L],N):-длина_списка(X,M), N=N1+1.
4) длина_списка([],_).
   длина_списка(H,T,N):- длина_списка(T,M), N = M-1 .

19. Отношение на Прологе имеет вид:

```
DOMAINS
  LIST = INTEGER*
DATABASE - dba2
  fact2(INTEGER,STRING)
CLAUSES
  fact2(1,"one").
  fact2(1,"one once more").
  fact2(2,"two").
```

Определить ответ на следующий запрос: Goal: retract(fact2(1,X),dba2)

1) fact2(2,"two").
   1 Solutions
2) No Solutions
3) X=one
   X=one once more
   2 Solutions
4) fact2(1,"two").
   1 Solutions

20. Отношение на Прологе имеет вид:

```
code=2000
include "GRAPDECL.PRO"
predicates
    demo2
     Convertmode(Integer,Integer)
clauses
demo2:- detectgraph(GraphDriver,GraphMode),
    convertmode(GraphMode,NewMode),
    initgraph(GraphDriver,NewMode,_,_, ""),getmaxx(X),
    X2=X div 2, getmaxy(Y),
    bar(0,0,X2,Y), readchar(_), closegraph.
    convertmode(egaHi,egaLo):-!.
    /* Переключает, если режим EGA определен */
    convertmode(Mode,Mode).
Goal   demo2.
```

Определить, что будет выведено на экран при выполнении цели:

1) белый прямоугольник на черном фоне;
2) весь экран окрасится в белый цвет;
3) весь экран окрасится в черный цвет;

4) черный прямоугольник на белом фоне.

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Тестовые задания по проверке остаточных знаний по дисциплине
«Методы искусственного интеллекта»
для специальностей 230102 – «Автоматизированные системы обработки информации и управления»

Утверждено на заседании кафедры — Кафедра ИУС
«23» октября 2006 г. — Заведующий кафедрой
«УТВЕРЖДАЮ» _____
А.В.Бушманов

Инструкция: Тест состоит из 20 заданий, время тестирования – 40 минут.

ВАРИАНТ № 3.

1. Программно – прогматическое направление развития искусственного интеллекта не изучает:

1) информацию о мышлении и языке;
2) интеллектуальные программы;
3) функциональные механизмы организма;
4) автоматический синтез программ.

2. Процедура взаимодействия эксперта с источником знаний, в результате которой становится явным процесс рассуждений экспертов при принятии решения и структура их представлений о предметной области называется:

1) извлечение знаний;
2) приобретение знаний;
3) формирование знаний.
4) получение знаний.

3. На отношение между знаками, отношения между знаками и реальностью, отношения между знаками и их пользователями подразделяют:

1) гносеологию;
2) гештальдпсихологию;
3) семиотику.
3) семантику.

4. … следует рассматривать как некую добродетель системы, поскольку большинство приложений, построенных на базе экспертных систем, нуждается в накоплении большого объема знаний, а решить такую задачу довольно трудно, если соглашения в языке представлены слишком сложны:

1) побочный эффект;
2) естественность нотации;
3) логическую адекватность;
4) эвристическую мощность.

5. Алгоритм, который может быстрее найти решение, особенно, если при его выполнении используются эвристики для выбора очередной ветви, называется:

1) пространством решений;
2) алгоритмом поиска в ширину:
3) алгоритмом поиска в глубину;
3) комбинаторным взрывом.

6. Традиционно семиотика включает отношения между знаками и их пользователями. Такой раздел семиотики называется:

1) синтаксисом;
2) семантикой;
3) прагматикой;
4) гештальтпсихологией.

7. Одна из возможных классификаций людей по психологическим характеристикам делит всех на несколько типов. ….предпочитают действие разговорам, хорошо реализуют замыслы других, направлены на результативность работы:

1) собеседники; 2) мыслители;
3) практики; 4) теоретики.

8. Методы извлечения знаний, которые включают такие методы, где ведущая роль в процессе извлечения фактически передается эксперту, а инженер по знаниям только фиксирует рассуждения эксперта во время работы по принятию решения, называются:

1) текстологические методы; 2) пассивные методы;
3) активные методы; 4) коммуникативные методы.

9. Язык Пролог предназначен для решения:

1) формализованных задач; 2) оригинальных задач;
3) неформализованных задач; 4) математических задач.

10. Множество аксиом и правил, задающих отношения между объектами является:

1) вычислительной программой; 2) логической программой;
3) интеллектуальной программой; 4) классической программой.

11. Семантика, определяющая результат работы программы, не вдаваясь в подробности как это делается, называется:

1) декларативной; 2) общей;
2) процедурной. 4) математической.

12. Предложение Q ***является резольвентой двух конкретных предложений*** P ***и*** $\neg P \vee Q$***. Следовательно, принцип резолюций покрывает правило вывода, называемое:***

1) modus ponens; 2) гипотетический силлогизм;
3) эвристика; 4) эврика.

13. Задавая фиксированное множество правил вывода, можно рассматривать следующее семейство проблем: исходя из выбранного множества ППФ применением некоторого числа раз правил вывода, можно получить заранее заданную ППФ. Процедура, которая из одной или нескольких ППФ производит другие ППФ называется:

1) доказательством; 2) преобразованием;
3) упрощением; 4) правилом вывода.

14. Укажите правильный вариант высказывания "Если у меня будет свободное время, то я почитаю книгу или посмотрю телевизор" записанного с использованием логики предикатов первого порядка:

1) $a \leftrightarrow (b \vee c)$. 2) $a \rightarrow (b \vee c)$.
3) $a \rightarrow (b \wedge c)$. 4) $a \leftrightarrow (b \wedge c)$.

15. Укажите правильный вариант высказывания "Некоторые спортсмены являются мастерами спорта" записанного с использованием логики предикатов первого порядка:

1) $(\forall X)(S(X) \vee P(X))$ 2) $(\forall X)(S(X) \wedge P(X))$

3) $(\exists X)(S(X) \wedge P(X))$ 4) $(\exists X)(S(X) \vee P(X))$

16. Детерминированная продукция в правой части ядра которой указываются возможности выбора, оценивающиеся специальными весами выбора (в качестве таких весов могут использоваться вероятностные оценки, лингвистические оценки, экспертные оценки и т.п.), называется:

1) однозначной; 2) альтернативной;
3) не однозначной; 4) не альтернативной.

17. В семантических сетях используются четыре основных типа объектов. Один из них, определенный как "сведения об абстрактных или физических объектах предметной области и задается множеством доменов (параметров или констант)", называется:

1) понятия; 2) события;
3) свойства; 4) значения.

18. Если L1 – пустой список, то результат добавления L1 к L2 будет список L3, иначе формируется список L3, голова которого совпадает с головой списка L1.

1) список(L1,L2,L3).
список([X|L1],L2,[X|L3]):-список(L1,L2,L3).
2) список([],L1,L1).
список([X|L1],L1,[X|L3]):-список(L1,L1,L3).
3) список([],L1,L1).
список([X|L1],L2,[X|L3]):-список(L1,L2,L3).
4) список([],[],L1).
список([X|L1],L1,[X|L3]):-список(L1,L1,L3).

19. Отношение на Прологе имеет вид:

```
DOMAINS
  LIST = INTEGER*

DATABASE - dba1
  fact1(INTEGER,STRING,LIST)
CLAUSES

  fact1(1,"fact1",[1,2,3]).
  fact1(2,"fact2",[1,3]).
  fact1(3,"fact2",[3,2,1]).
```

Определить ответ на следующий запрос: Goal: fact1(X,Y,Z)

1) X=1, Y=fact1, Z=[1,2,3]
X=2, Y=fact2, Z=[1,3]
X=3, Y=fact2, Z=[3,2,1]
3 Solutions

2) X=1, Y=fact1, Z=[1,2,3]
X=1, Y=fact2, Z=[1,3]
X=1, Y=fact2, Z=[3,2,1]
3 Solutions

3) No Solutions

4) X=1, Y=fact1, Z=[1,2,3]
1 Solutions

20. Отношение на Прологе имеет вид:

```
constants
  bgi_path=""
predicates
  rejim
clauses
```

```
rejim:-initgraph(0,0,_,_,"bgi_path"),
    setlinestyle(4, $fe8F, 3),
    pieslice(100,100,0,90, 50),
    pieslice(100,100,90,180, 50),
    pieslice(100,100,180,360, 50),
      readln(_).
```

Определить ответ на следующий запрос: rejim:

1) красная окружность на белом фоне;
2) белая окружность на черном фоне;
3) белая окружность на красном фоне;
4) черная окружность на белом фоне;

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Тестовые задания по проверке остаточных знаний по дисциплине
«Методы искусственного интеллекта»
для специальностей 230102 – «Автоматизированные системы обработки информации и управления»

Утверждено на заседании кафедры — Кафедра ИУС
«23» октября 2006 г. — Заведующий кафедрой
«УТВЕРЖДАЮ» _____
А.В.Бушманов

Инструкция: Тест состоит из 20 заданий, время тестирования – 40 минут.

ВАРИАНТ № 4.

1. Под искусственным интеллектом понимают научное направление, в рамках которого ставятся и решаются задачи:

1) аппаратного и программного моделирования;
2) классического моделирования;
3) объектно – ориентированного моделирования.
4) нейромоделирования.

2. Мерой «разговорности» речи при проведении профессионального разговора является:

1) когнитивная эквивалентность; 2) число Липшица;
3) число Ингве – Миллера; 4) число Ньютона.

3. Содержит основные понятия, используемые при описании предметной области; свойства всех отношений, используемых для установления связей между понятиями:

1) база знаний; 2) поле знаний;
3) база данных; 4) поле данных.

4. … означает, что представление знаний должно обладать способностью распознавать все отличия, которые Вы закладываете в исходную сущность:

1) естественность нотации; 2) логическая адекватность;
3) побочный эффект; 4) эвристическая мощность.

5. Множество решений, которые удовлетворяют условию проверять, не является ли образовавшееся состояние конечным решением, иногда называют:

1) пространством решений; 2) алгоритмом поиска в ширину:
3) алгоритмом поиска в глубину; 3) комбинаторным взрывом.

6. Традиционно семиотика включает совокупность правил построения языка или отношения между знаками. Такой раздел семиотики называется:

1) синтаксисом; 2) семантикой;
3) прагматикой; 4) гештальтпсихологией.

7. Одна из возможных классификаций людей по психологическим характеристикам делит всех на несколько типов. …. – это общительные, открытые люди, готовые к сотрудничеству:

1) собеседники; 2) мыслители;
3) практики; 4) теоретики.

8. Под …. будем понимать специфическую форму общения инженера по знаниям и эксперта, в которой инженер по знаниям задает эксперту серию заранее подготовленных вопросов с целью извлечения знаний о предметной области:

1) анкетированием; 2) свободным диалогом;
3) интервью; 4) игрой с экспертом.

9. Фундаментальным свойством языка Пролог являются:

1) механизм вывода с поиском и возвратом и встроенный механизм сопоставления с образцом;
2) работа в режиме компиляции и оконный интерфейс;
3) возможность создавать автономные выполняемые файлы;
4) оконный интерфейс.

10. Какие разделы не включает в себя Пролог - программа:

1) clauses; 2) goal;
3) domains; 4) trace;

11. В математических терминах Пролог рассматривает факты и правила в качестве множества аксиом, а вопрос – как:

1) новую аксиому; 2) теорему;
3) домен; 4) внутренний запрос.

12. Отсечение, введенное в программу, повышает эффективность программы, сокращает время перебора и объем памяти и влияет на декларативное чтение программы называется:

1) красным; 2) зеленым;
3) выполняемым; 4) пассивным.

13.. Если некоторая последовательность резолюций, применяемых к исходному множеству предложений E и множеству резольвент, полученных в процессе резолюции, приводит к пустому предложению, то множество E является:

1) выполнимым; 2)невыполнимым;
3) пустым; 4) пополняемым.

14. Укажите правильный вариант высказывания "Если у меня будет отпуск зимой, то я поеду на лыжную турбазу или горнолыжный курорт" записанного с использованием логики предикатов первого порядка:

1) $a \leftrightarrow (b \vee c)$.
2) $a \rightarrow (b \vee c)$.
3) $a \rightarrow (b \wedge c)$.
4) $a \leftrightarrow (b \wedge c)$.

15. Укажите правильный вариант высказывания " Все свидетели дают истинные показания" записанного с использованием логики предикатов первого порядка:

1) $(\forall X)(S(X) \vee P(X))$
2) $(\forall X)(S(X) \wedge P(X))$
3) $(\exists X)(S(X) \wedge P(X))$
4) $(\exists X)(S(X) \vee P(X))$

16. В детерминированных ядрах при актуализации ядра и при выполнимости A правая часть ядра выполняется обязательно. Таким образом, секвенция $\Rightarrow$ в детерминированных ядрах реализуется:

1) с необходимостью;
2) с возможностью;
3) с достаточностью;
4) с недостаточностью.

17. Форма представления знаний, которая рассматривается как маркированный ориентированный граф с помеченными узлами и дугами. Узлам соответствуют некоторые объекты, а дугам – отношения между этими объектами. Метки, приписываемые узлам, выделяют множество рассматриваемых объектов и выступают в качестве их имени. В роли таких имен могут быть слова естественного языка. Метки, приписываемые дугам, соответствуют элементам множества отношений, заданных на графе, называется:

1) правилом продукции;
2) логикой предикатов первого порядка;
3) фреймом;
4) семантической сетью.

18. Если X является головой списка, то результатом удаления будет хвост этого списка. Если X находится в хвосте списка, то его нужно удалить оттуда:

1) del([X|L],X,L).
   del([X|L],X,L):- del(L,X,L1).
2) del([],X,[]).
   del([_|L],X,L):- del(L,X,L).
3) del([X|L],_,L).
   del([X|L],X,L):- del(L,X,L1).
4) del([],[],[]).
   del([_|L],X,L):- del(L,X,L).

19. Отношение на Прологе имеет вид:

```
DOMAINS
  LIST = INTEGER*
DATABASE - dba2
  fact2(INTEGER,STRING)
CLAUSES
  fact2(1,"one").
  fact2(1,"one once more").
  fact2(2,"two").
```

Определить ответ на следующий запрос: Goal: retract(fact2(1,X))

1) X=one
   1 Solutions
2) X=one
   X=one once more

2 Solutions

3) X=one once more
1 Solutions

4) X=0
1 Solutions

20. Отношение на Прологе имеет вид:

```
constants
  bgi_path = ""
goal
  initgraph(0,0,_,_,bgi_path), ellipse(110,150,0,360,90,140),
  setcolor(10), ellipse(110,150,0,360,110,100),
  setcolor(1),readchar(CH).
```

Определить, что будет выведено на экран при выполнении цели:

1) не пересекающиеся эллипсы белого и зеленого цветов;
2) пересекающиеся эллипсы зеленых цветов;
3) пересекающиеся эллипсы белого и зеленого цветов;
4) пересекающиеся эллипсы белого цвета.

Лист ответов к тестам

по дисциплине «Методы искусственного интеллекта»

№ вопроса	1 вариант	2 вариант	3 вариант	4 вариант
1.	2	2	3	1
2.	1	2	1	3
3.	2	2	3	2
4.	3	4	2	2
5.	2	1	3	1
6.	1	2	3	1
7.	2	1	3	1
8.	4	1	2	3
9.	1	2	3	1
10.	1	1	2	4
11.	2	2	1	2
12.	2	1	1	1
13.	2	1	4	2
14.	4	4	2	2
15.	3	3	3	2
16.	3	3	2	1
17.	2	4	1	4
18.	1	2	3	1
19.	1	3	1	2
20.	3	1	2	3

11. КОНТРОЛЬНЫЕ ВОПРОСЫ К ЗАЧЕТУ

I часть:

1. Основные конструкции языка Пролог.
2. Переменные и константы языка Пролог.
3. Стандартные предикаты языка Пролог.
4. Факты и правила на Прологе.
5. Унификация термов.
6. Отсечение.
7. Использование рекурсии.
8. Работа с окнами.
9. Основные предикаты при работе с файлами.
10. Основные предикаты при работе с внутренней базой данных.
11. Основные предикаты при работе с внешней базой данных.
12. Предикаты для работы с цепочками.
13. Алгоритм создания случайных чисел.
14. Как найти в списке подсписок заданного размера и остаток?
15. Как вставить подсписок в список?
16. Алгоритм удаления элементов из списка
17. Алгоритм замены вхождения числа в списке на другое число.
18. Алгоритм сложения двух списков.
19. Инициализация графического режима.
20. Основные графические объекты на Прологе.

II часть:

1. Основные определения в логике предикатов.
2. Формулы первого и второго порядков в логике предикатов.
3. Правила вывода в логике предикатов.
4. Интерпретация формул.
5. Преобразование правильно построенных формул в предложения.
6. Сущность принципа резолюций.
7. Основные понятия продукционных правил.
8. Классификация ядер продукции.
9. Гипотетический силлогизм.
10. Основные понятия семантических сетей.
11. Основные типы объектов в семантических сетях.
12. Классификации семантических сетей.
13. Канонические графы.
14. Теория фреймов.
15. Структура данных фрейма.
16. Свойства фреймов.

III часть:

1. Этапы развития искусственного интеллекта.

2. Японский проект компьютеров пятого поколения.
3. Современные направления развития искусственного интеллекта.
4. Данные и знания.
5. Особенности знаний.
6. Модели представления знаний.
7. Компоненты продукционных систем.
8. Психологический аспект извлечения знаний
9. Лингвистический аспект извлечения знаний
10. Гносеологический аспект извлечения знаний.
11. Основные понятия методов работы со знаниями.
12. Системы приобретения знаний от экспертов.
13. Формализация качественных знаний.
14. Классификация методов извлечения знаний.
15. Коммуникативные методы извлечения знаний.
16. Текстологические методы извлечения знаний.
17. Языки семиотического моделирования.
18. Стадии структурирования.
19. Методы структурирования.
20. Психосемантика и методы многомерного шкалирования.

12. КАРТА КАДРОВОЙ ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ

Лектор – доцент Акилова Ирина Михайловна.

Руководитель практических работ – доцент Акилова Ирина Михайловна.

Руководитель лабораторных работ – доцент Акилова Ирина Михайловна.

СОДЕРЖАНИЕ

1. Выписка из государственного образовательного Стандарта высшего профессионального образования 3
2. Рабочая программа 4
3. График самостоятельной работы студентов 19
4. Методические рекомендации по проведению самостоятельной работы студентов 20
5. Перечень учебников, учебных пособий 21
6. Краткий конспект лекций 22
7. Методические указания по выполнению лабораторных работ 88
8. Методические указания по выполнению практических работ 137
9. Методические указания по организации межсессионного контроля знаний студентов 185
10. Фонд контрольных заданий для оценки качества знаний по дисциплине 186
11. Контрольные вопросы к зачету 199
12. Карта кадровой обеспеченности дисциплины 201

Ирина Михайловна Акилова,
доцент кафедры ИиУС АмГУ

Изд-во АмГУ. Подписано к печати Формат 60х84/16. Усл. печ. л .
Тираж Заказ