1. ПРИКЛАДНЫЕ СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

1.1. Логический и нейрокибернетический подходы к созданию систем ИИ

С самого начала исследований в области моделирования процесса мышления (конец 40-х годов) выделились два до недавнего времени практически независимых направления: логическое и нейрокибернетическое.

Первое было основано на выявлении и применении в интеллектуальных системах различных логических и эмпирических приемов (эвристик), которые применяет человек для решения каких-либо задач. В дальнейшем с появлением концепций "экспертных систем" (ЭС) (в начале 80-х годов) это направление вылилось в научно-технологическое направление информатики "инженерия знаний", занимающееся созданием т.н. "систем, основанных на знаниях" (Knowledge Based Systems). Именно с этим направлением обычно ассоциируется термин "искусственный интеллект" (ИИ).

Второе направление – нейрокибернетическое – было основано на построении самоорганизующихся систем, состоящих из множества элементов, функционально подобных нейронам головного мозга. Это направление началось с концепции формального нейрона Мак-Каллока-Питтса и исследований Розенблатта с различными моделями перцептрона – системы, обучающейся распознаванию образов. В связи с относительными успехами в логическом направлении ИИ и низким технологическом уровнем в микроэлектронике нейрокибернетическое направление было почти забыто с конца 60-х годов до начала 80-х, когда появились новые удачные теоретические модели (например, "модель Хопфилда") и сверхбольшие интегральные схемы.

Логическое направление можно рассматривать как моделирование мышления на уровне сознания или вербального мышления. Его достоинствами являются:

- возможность относительно легкого понимания работы системы;
- легкость отображения процесса рассуждений системы на ее интерфейс с пользователем на естественном языке или каком-либо формальном языке;
- достижимость однозначности поведения системы в одинаковых ситуациях.

Недостатками этого подхода являются:

- трудность и неестественность реализации нечетких знаков (образов) (см.2.4);
- трудность (или даже невозможность) реализации адекватного поведения в условиях неопределенности (недостаточности знаний, зашумленности данных, не точно поставленной цели и т.п.);
- трудность и неэффективность распараллеливания процесса решения задач.

Нейрокибернетическое направление можно рассматривать как моделирование мышления на подсознательном уровне (моделирование интуиции, творческого воображения, инсайта). Его достоинства – это отсутствие недостатков, свойственных логическому направлению, а недостатки – отсутствие его досто-

инств. Кроме того, в нейрокибернетическом направлении привлекает возможность (быть может, иллюзорная), задав базовые весьма простые алгоритмы адаптации и особенности структуры искусственной нейронной сети, получить систему, настраивающуюся на поведение сколь угодно сложное и адекватное решаемой задаче. Причем его сложность зависит только от количественных факторов модели нейронной сети.

Еще одним достоинством в случае аппаратной реализации нейронной сети является ее живучесть, т.е. способность сохранять приемлемую эффективность решения задачи при выходе из строя элементов сети. Это свойство нейронных сетей достигается за счет избыточности. В случае программной реализации структурная избыточность нейронных сетей позволяет им успешно работать в условиях неполной или зашумленной информации.

1.2. Интеллектуальные роботы

"Господин сначала создал
людей – самый не сложный вид,
который легче всего производить.
Постепенно он заменил их роботами.
Это был шаг вперед".

А. Азимов. Логика.
Из книги "Я – робот"

Интеллектуальные роботы (иногда говорят "интеллектные" или роботы с искусственным интеллектом) явились развитием простейших программируемых промышленных роботов, которые появились в 60-х годах. Тогда же были заложены основы современных и будущих интеллектуальных роботов в исследованиях, связанных с координацией программирования роботов-манипуляторов и технического зрения на основе телевизионной камеры, планирования поведения мобильных роботов, общения с роботом на естественном языке.

Эксперименты с первыми интеллектуальными роботами проводились в конце 60-х – начале 70-х годов в Стэндфордском университете, Стэндфордском исследовательском институте (Калифорния), Массачусетском технологическом институте (Массачусетс), Эдинбургском университете (Великобритания), в Электротехнической лаборатории (Япония).

Типичный интеллектуальный робот состоит из одной или двух рук (манипуляторов) и одной или двух телевизионных камер, размещенных на неподвижной тумбе либо на перемещающейся тележке.

На рис. 1 показана обобщенная структура информационной системы интеллектуального робота. Здесь надо иметь в виду, что на подсистему восприятия поступает большой объем разнотипной информации от датчиков различ-

ных типов: зрительных, слуховых, тактильных, температурных, лазерных или ультразвуковых дальномеров и других более специализированных.

Рис. 1. Структура информационной системы интеллектуального робота

Под синтаксисом понимается структура в пространстве и во времени этой разнотипной информации. Под семантикой – результат ее восприятия как множества возможных типовых ситуаций или образов, требующих какой-либо дальнейшей обработки. Под миром понимается описание окружения робота как результат работы его подсистемы восприятия.

Под действием понимается достаточно сложный двигательный акт, например, перемещение детали из входного бункера в шпиндель станка и ее закрепление там, в отличие от движения как результата срабатывания какой-либо одной степени свободы робота, например, вращение робота вокруг вертикальной оси на заданный угол.

1.3. Интеллектуальный доступ к данным

В настоящее время растет количество информации, хранимой в электронном виде. Компьютерные сети представляют пользователям огромные массивы информации, причем со временем экспоненциально растет как количество этой информации, так и число людей, получивших к ней доступ, благодаря сети Internet.

Появилось также огромное количество поисковых систем, облегчающих доступ к ней. Как правило, они используют ту или иную модификацию поиска по ключевым словам. Большое количество информации хранится в реляционных таблицах различных типов, к которым доступ осуществляется посредством специальных языков типа SQL.

Для того, чтобы избавиться от неудобств, связанных с ограниченностью языка SQL и сложностью поиска информации по ключевым словам в локальных и распределенных в Internet базах данных, разрабатываются средства доступа к данным на естественном языке.

Применительно к локальной базе данных комплекс из таких средств и самой БД может быть назван интеллектуальным банком данных. Его обобщенная структура показана на рис. 2.

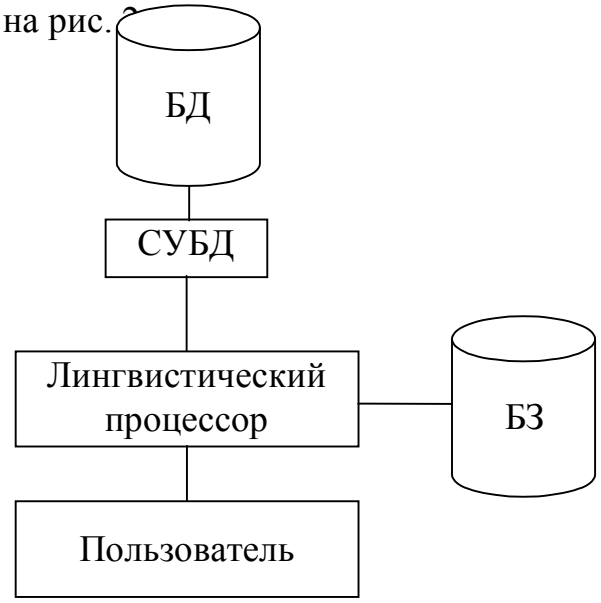

Рис. 2. Структура интеллектуального банка данных

База знаний содержит знания о языке общения, а также о предметной области, необходимые для понимания запроса к базе данных. Лингвистический процессор должен обеспечивать синтаксический, семантический анализ и прагматический анализ запроса (вопроса) на естественном языке. В идеале он должен реализовывать "активный диалог" с пользователем, в ходе которого инициатива должна переходить от пользователя к системе и обратно с целью уточнения вопроса.

Примером программного обеспечения для доступа к базам данных на естественном языке является пакет InBase, разработанный в Российском научно-исследовательском институте искусственного интеллекта (Москва – Новосибирск).

1.4. Интеллектуальные системы обработки текстовой информации

В настоящее время все чаще появляются прикладные программы для автоматизации офисной деятельности, претендующие на право называться интеллектуальными, т.е. использующими методы искусственного интеллекта. На исследования в области искусственного интеллекта с целью создания таких программ ведущие компании, производящие ПО, в частности Microsoft, тратят миллионы и миллиарды долларов.

Этот класс прикладных систем искусственного интеллекта можно разделить на следующие типы программ:

- текстовые редакторы со встроенными средствами проверки орфографии и стилистики (например, всем известная программа Word фирмы Microsoft);
- программы-переводчики (например, Stylus и ПРОМТ фирмы ПРОМТ);
- программы для распознавания и ввода печатных и рукописных документов (программные продукты GuniForm и FineReader российских фирм Cognitive Technologies и ABBYY, соответственно);
- программы для поиска информации в электронных документах по смыслу, в том числе, в Internet (например, программный продукт "Следопыт" российской фирмы MediaLingua);
- программы для реферирования текстовых документов (например, TextAnalist фирмы "Микросистемы").
- программы для обработки и классификации по смыслу электронной почты (например, программа MLExpert фирмы MediaLingua).

Те или иные из перечисленных выше типов программ встраиваются в современные системы документооборота (например, ЕВФРАТ фирмы Cognitive Technologies).

1.5. Экспертные системы

Экспертные системы – это прикладные системы ИИ, в которых база знаний представляет собой формализованные эмпирические знания высококвалифицированных специалистов (экспертов) в какой-либо узкой предметной области. Экспертные системы предназначены для замены при решении задач экспертов в силу их недостаточного количества, недостаточной оперативности в решении задачи или в опасных (вредных) для них условиях.

Обычно экспертные системы рассматриваются с точки зрения их применения в двух аспектах: для решения каких задач они могут быть использованы и в какой области деятельности. Эти два аспекта накладывают свой отпечаток на архитектуру разрабатываемой экспертной системы.

Можно выделить следующие основные классы задач, решаемых экспертными системами:

- диагностика;

- прогнозирование;
- идентификация;
- управление;
- проектирование;
- мониторинг.

Наиболее широко встречающиеся области деятельности, где используются экспертные системы:

- медицина;
- вычислительная техника;
- военное дело;
- микроэлектроника;
- радиоэлектроника;
- юриспруденция;
- экономика;
- экология;
- геология (поиск полезных ископаемых);
- математика.

Примеры широко известных и эффективно используемых (или использованных в свое время) экспертных систем:

DENDRAL – ЭС для распознавания структуры сложных органических молекул по результатам их спектрального анализа (считается первой в мире экспертной системой);

MOLGEN – ЭС для выработке гипотез о структуре ДНК на основе экспериментов с ферментами;

XCON – ЭС для конфигурирования (проектирования) вычислительных комплексов VAX 11 в корпорации DEC в соответствии с заказом покупателя;

MYCIN – ЭС диагностики кишечных заболеваний;

PUFF – ЭС диагностики легочных заболеваний;

MACSYMA – ЭС для символьных преобразований алгебраических выражений;

YES/MVS – ЭС для управления многозадачной операционной системой MVS больших ЭВМ корпорации IBM;

DART – ЭС для диагностики больших НМД корпорации IBM;

PROSPECTOR – ЭС для консультаций при поиске залежей полезных ископаемых;

POMME – ЭС для выдачи рекомендаций по уходу за яблоневым садом;

набор экспертных систем для управления планированием, запуском и полетом космических аппаратов типа "челнок";

AIRPLANE – экспертная система для помощи летчику при посадке на авианосец;

ЭСПЛАН	–	ЭС для планирования производства на Бакинском нефтеперерабатывающем заводе;
МОДИС	–	ЭС диагностики различных форм гипертонии;
МИДАС	–	ЭС для идентификации и устранения аварийных ситуаций в энергосистемах;
NetWizard	–	ЭС для проектирования локальных систем.

2. МЕТОДЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

2.1. Виды знаний

"Не в совокупности ищи
единства, но более в
единообразии разделения".
Козьма Прутков

В основе представления знаний лежат понятия семиотики – науки о знаковых системах (искусственных и естественных языках). В семиотике различают следующие разделы: синтактику, имеющую дело со структурой (синтаксисом) знаковых систем, семантику, рассматривающую смысл (интерпретацию) знаковых систем (другими словами, соответствие знаковой системы другой знаковой системе), прагматику, имеющую дело с целенаправленностью знаковых систем. Другими словами синтактика отвечает на вопрос: как выглядит знание (как оно структурировано или как формализовано), семантика отвечает на вопрос "что означает знание", прагматика отвечает на вопрос "зачем" или "почему" необходимо (или передается) знание. Таким образом, имея дело со "знанием" необходимо уметь различать его синтаксис, семантику и прагматику. В этом разделе (методы представления знаний) рассматривается в основном синтаксис (структура знаний). Методы обработки знаний в основном характеризуют их семантику. Прагматика знаний в основном выражается в архитектуре интеллектуальной системы – особенностях реализации методов представления и обработки знаний, реализации интерфейсов системы с внешним миром.

С точки зрения глубины различают экстенсиональные (конкретные, поверхностные) и интенсиональные (абстрактные, глубинные) знания. Экстенсиональные знания представляют собой факты об объектах реального мира. Примером их являются реляционная база данных, утверждения вида:

"Иванов имеет автомобиль";
"Расстояние от Земли до Солнца 150 млрд. км";
"Треугольник – есть геометрическая фигура".

Интенсиональные знания представляют собой правила, связывающие между собой факты, или закономерности реального мира. Примерами их являются продукционная база знаний (см. 2.3), утверждения вида:

"человек может иметь собственность, например, автомобиль";
"планета, на которой может быть жизнь, должна находиться от Солнца на расстоянии 100-300 млрд. км";
"если геометрическая фигура имеет три угла, то это треугольник".

Иногда разделение на экстенсиональные и интенсиональные знания условно и зависит от уровня абстрагирования. Например, при необходимости оперировать с правилами типа "если ... то" как с фактами, они выступают в роли экстенсиональных знаний, а интенсиональные знания, предназначенные для

этого, обычно называют метазнаниями. Примером метазнания может быть утверждение

> "Если мы имеем дело с Евклидовой геометрией, то справедливо утверждение "Сумма углов треугольника равна 180 градусов".

По используемым методам представления знания подразделяются на декларативные и процедурные.

Декларативные знания содержат описание объектов и отношений между ними. Их интерпретация или обработка осуществляется программами. Говорят, что при декларативном представлении семантические и синтаксические знания отделены друг от друга, что придает этой форме представления большую по сравнению с другими универсальность и общность.

Процедурные знания содержат в явном виде описание процедур, т.е. являются самоинтерпретируемыми. В этих процедурах могут быть запрограммированы действия, связанные с изменением предметной области и ее модели в базе знаний. При этом текущее состояние представляется в виде набора специализированных процедур, обрабатывающих определенный участок базы знаний. Это позволяет отказаться от хранения описаний всех возможных состояний предметной области, требуемых для работы системы ИИ, и ограничиться хранением исходного состояния и процедур, обеспечивающих преобразование модели предметной области, т.е. порождение всех других состояний из исходного. Процедурные представления реализуются, как правило, специальными языками программирования (например, PLANNER [19]). Включение семантики в базу знаний позволяет повысить эффективность поиска решений с использованием базы знаний. Но это достигается ценой специализации базы знаний, ориентации ее на особенности решаемых задач. Процедурные знания уступают декларативным в возможностях для накопления и коррекции знаний.

Разделение методов представления знаний на процедурные и декларативные является в определенной степени условным. "Чисто" декларативные или процедурные знания практически не используются. Пример чисто процедурного знания – программа, написанная на алгоритмическом языке (но не объектно-ориентированном). Пример чисто декларативного знания – реляционная база данных. В конкретных реализациях баз знаний присутствуют элементы и тех и других способов представления знаний.

Наиболее распространенными вариантами декларативного представления являются семантические сети и фреймы. Иногда говорят о реляционных базах знаний, не отличая их существенно от соответствующих баз данных. Среди процедурных методов представления знаний выделяются средства работы со списковыми структурами и исчисление высказываний первого порядка.

По степени формализации различают логические и эвристические методы представления знаний. Логические методы могут быть описаны в виде формальной теории (системы)

$$S=<B,F,A,R>,$$

где: B – алфавит,
F – формулы-факты,
A – формулы-аксиомы,
R – правила-вывода.

Примеры логических методов представления знаний: исчисление высказываний (логика предикатов) 1-го порядка, различные псевдофизические логики, продукционные модели представления знаний.

Эвристические методы представления знаний основаны на применении ряда приемов, принципов или подходов для описания знаний в удобном для понимания человеком или обработки компьютером виде. Примерами таких методов являются: семантические сети, фреймы, объектно-ориентированное представление. Эвристические методы являются более высокоуровневыми методами представления знаний и, как правило, могут быть описаны с помощью какого-либо логического метода с потерей ряда свойств, например, таких как наглядность, обозримость, универсальность.

Несколько в стороне от логических и эвристических методов представления и обработки знаний находятся нейронные сети. Их обычно не относят к методам инженерии знаний, а рассматривают в рамках нейрокибернетики (нейроинформатики). Их принципиальным отличием от методов инженерии знаний является то, что в них знания содержатся не в формализованном и локализованном виде, а в виде состояния множества нейроподобных элементов и распределены между этими элементами. Нейронная сеть обладает, как говорят, голографическими свойствами, т.е. порча какого-либо элемента распределенного в них знания не приводит к порче всего знания, а всего лишь – к некоторому ухудшению его характеристик.

2.2. Логика предикатов первого порядка

ЛОГИКА – искусство мыслить и рассуждать в строгом соответствии с ограниченностью и несостоятельностью человеческих заблуждений.
А. Бирс. Словарь Сатаны

Логика предикатов является развитием алгебры логики (или логики высказываний). В логике высказываний для обозначения фактов используются буквы (имена или идентификаторы), не имеющие структуры, и принимающие значения "1" или "0" ("да" или "нет"). В логике предикатов факты обозначаются n-арными логическими функциями – предикатами $F(x_1,x_2,...,x_m)$, где F – имя предиката (функтор) и x_i – аргументы предиката. Имена предикатов неделимы, т.е. являются так называемыми атомами. Аргументы могут быть атомами или функциями $f(x_1,x_2,...,x_m)$, где f – имя функции, а $x_1,...,x_m$, так же как и аргументы

предикатов являются переменными или константами предметной области. В результате интерпретации (по-другому, конкретизации) предиката функторы и аргументы принимают значения констант из предметной области (строк, чисел, структур и т.д.). При этом следует различать интерпретацию на этапе описания предметной области (создания программ и баз знаний) и на этапе решения задач (выполнения программ с целью корректировки или пополнения баз знаний). В дальнейшем при работе с предикатами мы будем иметь дело с результатом их интерпретации в первом смысле, т.е. с их привязкой к некоторой предметной области.

Предикат с арностью n > 1 может использоваться в инженерии знаний для представления n-арного отношения, связывающего между собой n сущностей (объектов) – аргументов предиката. Например, предикат отец("Иван", "Петр Иванович") может означать, что сущности "Иван" и "Петр Иванович" связаны родственным отношением, а именно, последний является отцом Ивана или наоборот. Уточнение семантики этого предиката связано с тем, как он используется, т.е. в каких операциях или более сложных отношениях он участвует и какую роль в них играют его 1-й и 2-й аргументы. Предикат компьютер(память, клавиатура, процессор, монитор) может обозначать понятие "компьютер" как отношение, связывающее между собой составные части компьютера.

Предикат с арностью n=1 может представлять свойство сущности (объекта), обозначенного аргументом или характеристику объекта, обозначенного именем предиката. Например, кирпичный(дом), оценка(5), улица("Красный проспект"), дата_рождения("1 апреля 1965 г."), быстродействие("1 Мфлопс").

Предикат с арностью n=0 (без аргументов) может обозначать событие, признак или свойство, относящееся ко всей предметной области. Например, "конец работы".

При записи формул (выражений) помимо логических связок "конъюнкция" (&), "дизъюнкция" ($\vee$), "отрицание" ($\neg$), "следование" ("импликация") ($\rightarrow$), заимствованных из логики высказываний, в логике предикатов используются кванторы всеобщности ($\forall$) и существования ($\exists$). Например, выражение $\forall$(x,y,z) (отец(x,y) & (мать(x,z)) $\rightarrow$ родители(x,y,z) означает, что для всех значений x,y,z из предметной области справедливо утверждение "если y – отец и z – мать x , то y и z – родители x; выражение ($\exists$x) (студент(x) & должность(x,"инженер") означает, что существует хотя бы один студент, который работает в должности инженера.

Переменные при кванторах называются связанными переменными в отличие от свободных переменных. Например, в выражении

($\forall$x) (владелец(x,y) $\rightarrow$ частная_собственность(y))

x – связанная переменная, y – свободная переменная.

Логика предикатов 1-го порядка отличается от логик высших порядков тем, что в ней запрещено использовать выражения (формулы) в качестве аргументов предикатов.

Решение задач в логике предикатов сводится к доказательству целевого утверждения в виде формулы или предиката (теоремы), используя известные утверждения (формулы) или аксиомы.

В конце 60-х годов Робинсоном для доказательства теорем в логике предикатов был предложен метод резолюции, основанный на доказательстве "от противного". Целевое утверждение инвертируется, добавляется к множеству аксиом и доказывается, что полученное таким образом множество утверждений является несовместным (противоречивым). Для выполнения доказательства методом резолюции необходимо провести определенные преобразования над множеством утверждений, а именно, привести их к совершенной конъюнктивной нормальной форме (СКНФ).

Приведение формул к СКНФ состоит из следующих этапов, легко реализуемых на ЭВМ.

1. Устранение импликации с помощью замены ее на отрицание и дизъюнкцию применением формулы

$$A \rightarrow B = \sim A \vee B;$$

2. Ограничение области действия символов отрицания, т.е. продвижение отрицания внутрь формулы с помощью закона де Моргана, т.е. применяя формулы

$$\neg(A \& B) = (\neg A) \vee (\neg B);$$
$$\neg(A \vee B) = (\neg A) \& (\neg B);$$
$$\neg(\exists x) F(x) = (\forall x) \neg F(x);$$
$$\neg(\forall x) F(x) = (\exists x) \neg F(x).$$

3. Стандартизация или разделение переменных.

На этом этапе в каждой формуле переименовываются связанные переменные так, чтобы они стали уникальными для каждого квантора, с которым они связаны. Это делается на основании того факта, что связанную переменную в области действия квантора можно заменить на любую другую, не встречающуюся переменную, не изменив этим значение истинности формулы.

Например, $(\forall x)\,(P(x) \vee Q(y)) \,\&\, (\forall x)\,(F(x))$
преобразуется в $(\forall x)\,(P(x) \vee Q(y)) \,\&\, (\forall z)\,(F(z))$.

4. Исключение кванторов существования.

На этом этапе квантор существования заменяется так называемой функцией Сколема g(x) или перечислением предикатов с аргументами – константами из области определения переменной-аргумента.

Примеры преобразования:

исходная формула	результирующая формула
$(\forall x)\ y$	g(x)
$(\forall x)\,(\forall y)\ z$	g(x,y)
$(\exists x)\,(F(x))$	F(a),F(b)...,

где a и b – константы. Функция Сколема задает отображение областей определения других переменных на область определения переменной, связанной с квантором существования.

5. Вынесение кванторов всеобщности в начало формулы.

6. Исключение кванторов всеобщности.

Исключение достигается просто удалением кванторов в предположении, что если в формуле есть некоторая переменная x, то формула справедлива для всех ее значений из области определения.

7. Собственно приведение формулы к СКНФ применением закона дистрибутивности

$$A \vee (B \& C) = (A \vee B) \& (A \vee C).$$

8. Исключение символов &. Это достигается заменой формулы вида (A&B) на множество формул вида {A,B}.

Предположим надо привести к СКНФ формулу

$$(\forall x)\{P(x) \to \{(\forall y)[P(y) \to P(f(x,y))] \& \neg(\forall y)[Q(x,y) \to P(y)]\}\}.$$

Исключив импликацию, получим

$$(\forall x)\{\neg P(x) \vee \{(\forall y)[\neg P(y) \vee P(f(x,y))] \& \neg(\forall y)[\neg Q(x,y) \vee P(y)]\}\}.$$

Продвинув внутрь отрицание, получим

$$(\forall x)\{(\neg P(x) \& \{(\forall y)[\neg P(y) \vee P(f(x,y))] \& [(\exists y)[Q(x,y) \& \neg P(y)]\}\}.$$

Разделив переменные, получим

$$(\forall x)\{(\neg P(x) \& \{(\forall y)[\neg P(y) \vee P(f(x,y))] \& [(\exists w)[Q(x,w) \& \neg P(w)]\}\}.$$

Заменив переменную w на сколемовскую функцию g(x), исключим квантор существования и получим

$$(\forall x)\{(\neg P(x) \& \{(\forall y)[\neg P(y) \vee P(f(x,y))] \& [Q(x,g(x)) \& \neg P(g(x))]\}\}.$$

Вынесем кванторы всеобщности в начало формулы и получим

$$(\forall x)(\forall y)\{(\neg P(x) \& \{[\neg P(y) \vee P(f(x,y))] \& [Q(x,g(x)) \& \neg P(g(x))]\}\}.$$

Применив закон дистрибутивности, получим

$$(\forall x)(\forall y)\{[P(x) \vee \neg P(y) \vee P(f(x,y))] \& [\neg P(x) \vee Q(x,g(x))] \&$$
$$\& [\neg P(x) \vee \neg P(g(x))]\}.$$

Исключив кванторы всеобщности и заменив конъюнкцию формул их множеством, получим множество следующих формул (предложений):

$$\neg P(x) \vee \neg P(y) \vee P[f(x,y)],$$
$$\neg P(x) \vee Q[x, g(x, g(x)],$$
$$\neg P(x) \vee \neg P[g(x)].$$

Логика предикатов 1-го порядка легла в основу языков логического программирования, самым распространенным из которых является Prolog (различные его диалекты). Точнее, язык Prolog основан на модифицированной логике предикатов 1-го порядка (логике Хорна или логике дизъюнктов). Логика Хорна отличается от классической логики предикатов 1-го порядка тем, что она оперирует уже почти преобразованными к применению метода резолюции формулами без кванторов всеобщности и существования, представляющими собой множество дизъюнктов (предложений или клауз Хорна). "Почти" объясняется тем, что клаузы Хорна содержат импликацию и выглядят как

$$A \to B ,$$

где A – предикат,

B – предикат или конъюнкция или дизъюнкция предикатов (такое представление части B предложения возможно, т.к. конъюнкция и дизъюнкция рассматриваются как частные случаи предикатов).

Доказательство некоторого утверждения (целевого предиката) в логическом программировании сводится к процессу унификации, с помощью которого происходит рекурсивный перебор всех возможных подстановок значений переменных в целевом предикате, управляемый ограничениями, заданными множеством предложений. Множество предложений обычно в Прологе называется базой данных Пролога. База данных состоит из предложений-правил вывода вида $A \to B$ и предложений-фактов, представляющих собой отдельные предикаты. При этом в предикатах-фактах параметрами могут только константы, а в предикатах-правилах – константы и неконкретизированные переменные. Последнее относится и к целевому предикату. В этом случае, если параметром является переменная, то это означает, что ее значение необходимо найти при доказательстве целевого предиката.

Унификация основана на сравнении (сопоставлении с образцом) целевого предиката, который надо доказать, с предикатами-фактами и предикатами-правилами из базы данных Prolog-программы. При этом успешность сопоставления двух предикатов определяется следующими условиями, упорядоченными в порядке их проверки:

- имена предикатов совпадают;
- количество параметров у предикатов совпадает;
- каждая пара сравниваемых параметров сопоставима.

Последнее условие для некоторой пары параметров истинно при трех вариантах:

- параметры являются константами (любого типа) и они равны;
- один параметр из пары является константой, а другой – переменной, в этом случае переменной присваивается значение константы;
- оба параметра являются неконкретизированными переменными, в этом случае эти переменные становятся "связанными", т.е. в дальнейшем при интерпретации программы рассматриваются как одна и та же переменная.

При унификации целевого предиката с правилом, сопоставлению подвергается сначала левая часть правила, а затем, в случае успешной унификации, последовательно проверяются предикаты, находящиеся в правой части. Т.е. правило в Прологе с точки зрения унификации рассматривается как предикат с именем "::=" (импликация) и с параметрами A и B, а B рассматривается в свою очередь как предикат "," (конъюнкция) или ";" (дизъюнкция) с параметрами-предикатами правой части правила.

Следующий фрагмент программы (на Эдинбургской версии Пролога) описывает поведение гипотетического робота-манипулятора и может являться частью программного обеспечения его системы управления.

```
/* описание объектов, с которыми работает робот */
        куб("кубик", 10).
        цилиндр("прут", 100, 3).
/* описание их местоположения    */
        объект("кубик", "стол").
        объект("прут", "коробка").
/*  описание некоторых действий робота  (команд)  */
        взять(_):-   в_схвате(_),   /*  проверка занятости схвата  */
                     write("Схват занят"),
                     nl,
                     !.
        взять(X):-   объект(X,Y), /*определение положения объекта X*/
                     повернуть_к(Y),
                     выбрать_объект(X,Coord_X,Coord_Y),
                     позиционировать_схват(Coord_X,Coord_Y),
                     взять,     /* включение схвата  */
                     assert(в_схвате(X)).  /*запоминание что в схвате*/
        положить(X,Y):- в_схвате(Z), /* определение, что в схвате*/
                        Z<>X,
                        write("Схват занят"),
                        nl,
                        !.
        положить(X,Y):- в_схвате(X),
                        повернуть_к(Y),
                        центр(Y, Coord_X,Coord_Y),
                        позиционировать_схват(Coord_X,Coord_Y),
                        отпустить,
                        retract(в_схвате(X)),
                        !.
        положить(X,Y):- not(в_схвате(_)),
                        объект(X,Y),
                        write("Объект "),
```

```
                write(X),
                write(" уже находится в "),
                        write(Y),
                nl,
                !.
положить(X,Y):- not(в_схвате(_)),
                взять(X),
                положить(X,Y).
```

Здесь операторы (встроенные предикаты) assert(X) и retract(X) используются для добавления и удаления предиката-факта X, соответственно, оператор ! используется для аннулирования попыток альтернативной унификации при неуспешной текущей унификации.

К недостаткам логики предикатов 1-го порядка как метода представления знаний можно отнести следующее:

- монотонность логического вывода, т.е. невозможность пересмотра полученных промежуточных результатов (они считаются фактами, а не гипотезами);
- невозможность применения в качестве параметров предикатов других предикатов, т.е. невозможность формулирования знаний о знаниях;
- детерминированность логического вывода, т.е. отсутствие возможности оперирования с нечеткими знаниями.

Но логику предикатов 1-го порядка можно использовать как основу для конструирования более сложных и удобных логических методов представления знаний. В этом качестве она используется в модальных и псевдофизических логиках.

2.3. Модальные логики

Первой попыткой расширить возможности логики 1-го порядка явилось появление множества модальных логик, в которых вводились различные кванторы (модальности) и аксиомы, отражающие тот или иной аспект реального мира. Наиболее известны модальные логики "возможного-необходимого" (алетическая логика), деонтическая логика (модальности "разрешено-обязательно"), эпистемическая логика (логика знания-веры), временная модальная логика (модальности "всегда-никогда", "часто-иногда").

Для интерпретации модальных логик возможностей предикатов, имеющих всего два значения (двузначной семантики), было недостаточно. Поэтому, сначала появилась трехзначная логика Лукасевича (логика Лукасевича), где логические переменные могут принимать значения 0, 1, 2, а затем семантика возможных миров (4-значная логика).

Ниже приведена четырехзначная семантика возможных миров (рис. 3).

Но для представления нечетких знаний модальные логики не годятся, т.к. они базируются на детерминированной семантике и являются, по существу,

некоторым улучшением логики предикатов 1-го порядка со всеми ее недостатками как метода для представления знаний.

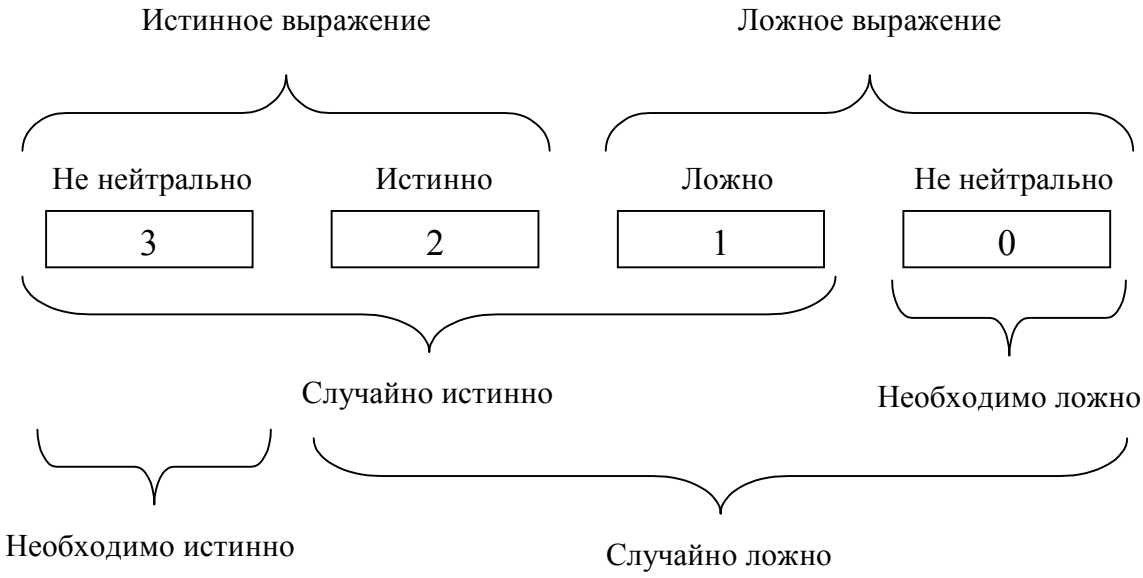

Рис 3. Семантика возможных миров

2.4. Теория нечетких множеств и нечеткая логика

Для представления нечетких понятий и оперирования с ними американский ученый Л. Заде в 60-х годах придумал теорию нечетких множеств, а затем – нечеткую логику, базирующуюся на ней. В основе теории нечетких множеств лежит интерпретация факта принадлежности элемента *a* множеству *A* как факта, который может быть истинным или ложным с некоторой оценкой истинности $\mu_A(a)$, пробегающей значения от 0 до 1. Эта оценка истинности называется функцией принадлежности элемента *a* множеству *A*.

Операции включения и равенства в теории нечетких множеств определяются обычно следующим образом (по Заде):

$$F \subseteq G:$$
$$\forall a, \mu_F(a) \le \mu_G(a)$$
$$F = G:$$
$$\forall a, \mu_F(a) = \mu_G(a).$$

Дополнение множества F к G определяется так, что

$$\forall a, \mu_F(a) = 1 - \mu_G(a).$$

Пересечение и объединение множеств определяются следующим образом:

$$\forall a, \mu_{F \cap G}(a) = \min(\mu_F(a), \mu_G(a)),$$
$$\forall a, \mu_{F \cup G}(a) = \max(\mu_F(a), \mu_G(a)).$$

Эти определения не единственные, хотя они не противоречат интуитивным представлениям о соответствующих операциях над нечеткими множест-

вами. Частным случаем теории нечетких множеств (при μ=1 или 0) является классическая теория множеств. Однако встречаются и другие определения операций над нечеткими множествами.

Так же как на основе классической теории множеств строится двоичная (булева) логика, так и на базе теории нечетких множеств строится теория нечетких множеств. Она оперирует с высказываниями, для которых функция принадлежности, описанная ранее, определена на множестве истинных высказываний. Функция принадлежности интерпретируется как мера истинности, уверенности или достоверности и отражает нечеткость знаний.

Предположим, существуют следующие высказывания:

"Иванов – хороший человек" с μ =0.8,

"Политик – хороший человек" с μ=0.3.

Тогда конъюнкция этих двух высказываний (имеющая смысл как уточнение мнения об Иванове, когда стало известно, что он – политик) определяется функцией принадлежности μ=0.3, а дизъюнкция – μ=0.8.

Можно развить логику нечетких высказываний до логики нечетких предикатов, которая обычно рассматривается в рамках псевдофизических логик (см. 2.5).

В теории нечетких множеств функция принадлежности может интерпретироваться как субъективное представление об истинности высказываний или объективная нечеткость знаний (информации). В первом случае описание нечетких высказываний является как бы снимком состояния некоторой интеллектуальной системы, обученной на примерах взаимодействия с внешней средой или заполненной субъективными знаниями экспертов. Во втором случае нечеткость является следствием каких-либо помех при поступлении информации в систему и интерпретации ее в виде знаний. В обоих случаях функцию принадлежности можно интерпретировать как вероятностную меру истинности и применять теорию вероятности к ее обработке и анализу. Это справедливо, т.к. интеллектуальная система работает с множеством разных субъектов, имеющих разные субъективные представления об истинности высказываний, или с множеством разных ситуаций, в которых разные помехи создают вероятностное описание истинности информации (знаний).

2.5. Псевдофизические логики

Недостатки классической логики и основанной на ней логики предикатов первого порядка как метода представления знаний об окружающем мире привели к появлению псевдофизических логик. В их основе лежит представление нечетких или размытых понятий в виде так называемых лингвистических переменных, придуманных Заде [9] для того, чтобы приблизить семантику (смысл) денотата (знака) к семантике, которая вырабатывается в мозгу человека в процессе его обучения (опыта). Для этого множество образов (десигнатов), с которыми должна оперировать интеллектуальная система, представляется в

виде точек на шкалах. Например, можно рассматривать шкалы "возраст" (в годах), "расстояние до объекта" (в м или км) и т.п. С каждой шкалой связано множество знаковых значений лингвистической переменной. Например, со шкалой "возраст" могут быть связаны следующие значения одноименной лингвистической переменной: "юный", "молодой", "зрелый", "пожилой", "старый", "дряхлый". Со шкалой "расстояние" – "вплотную", "очень близко", "близко", "рядом", "недалеко", "далеко", "очень далеко", "у черта на куличиках". Взаимосвязь между этими двумя представлениями (множеством точек на шкале и множеством знаковых значений) задается с помощью функции принадлежности μx(t), где x – значение лингвистической переменной, t – значение на шкале. Значение функции принадлежности интерпретируется как вероятность того, что значение t на шкале можно заменить знаком x или наоборот. Очевидно, что можно пронормировать значения функции принадлежности в соответствии с формулой

$$\sum_x \mu_x(t) = 1$$

или в соответствии с

$$\sum_t \mu_x(t) = 1 .$$

На рис. 4 приведен пример описания лингвистической переменной возраст. Здесь каждая кривая описывает ее одно символьное значение.

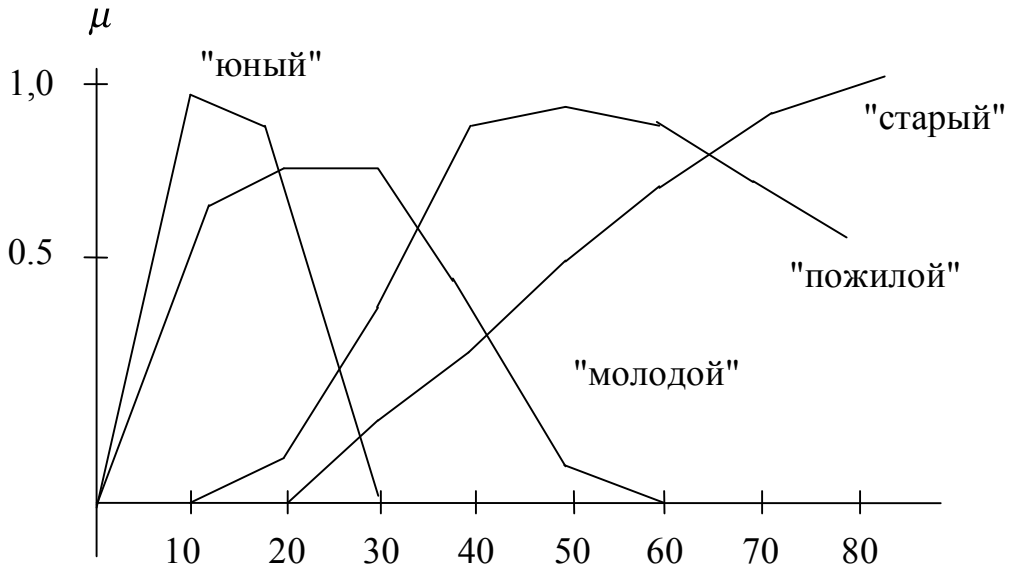

Рис. 4. Описание лингвистической переменной "Возраст"

Наиболее используемыми псевдофизическими логиками являются пространственная, временная и каузальная (причинно-следственная).

На рис. 5 показана структура составляющих пространственной логики.

Логики взаимного расположения объектов, расстояний и направлений делятся на метрическую и топологическую логики. В отличие от метрической топологическая логика не связана с метрической шкалой.

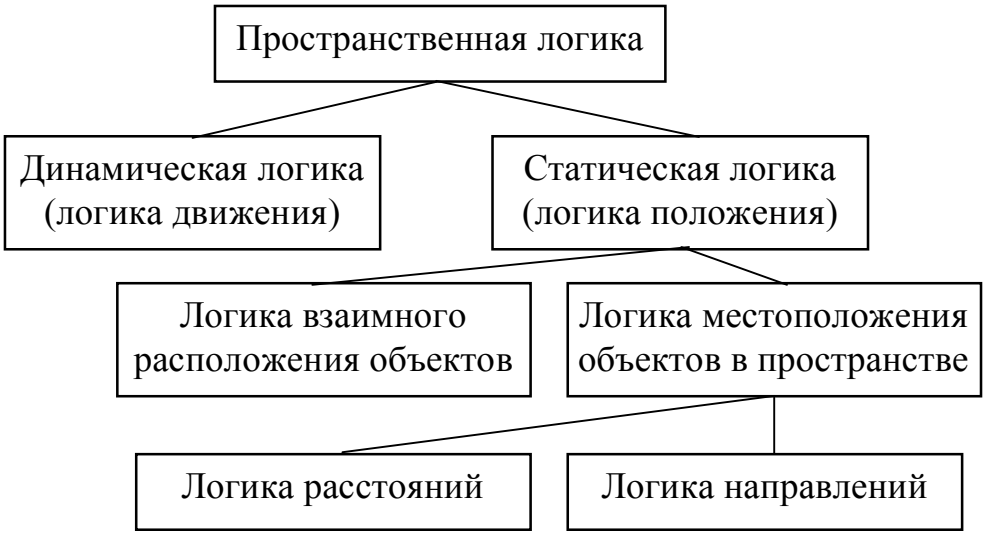

Рис. 5. Пространственная логика

Метрические шкалы подразделяются на экзоцентрические и эндоцентрические, относительные и абсолютные. Экзоцентрические шкалы имеют началом координат точку, связанную с самой интеллектуальной системой. Примером такой шкалы является шкала для описания лингвистической переменной "Расстояние до объекта" в логике расстояний. Ее символьными значениями могут быть следующие: "совсем рядом", "рядом", "очень-очень близко", "очень близко", "близко", "не очень близко", "не близко", "недалеко", "не далеко, но и не очень близко", "не очень далеко", "неблизко", "далеко", "очень далеко" и т.п. Эндоцентрическая шкала имеет началом координат точку вне системы. Примером такой шкалы является шкала для описания лингвистической переменной "расстояние между двумя объектами" в той же логике расстояний. Относительные шкалы имеют изменяемую точку отсчета (начало координат), а абсолютные – неизменную (обычно, подразумеваемую, т.е. явно не заданную).

Логика направлений оперирует с понятиями "справа", "слева", "впереди", "сзади" или "на восток", "на запад" и т.п.

В логике взаимного расположения объектов описываются следующие базовые отношения: унарные – "иметь горизонтальное положение", "иметь вертикальное положение", бинарные – "находиться внутри", "находиться вне", находиться на поверхности", "находиться в центре", "находиться в середине", "быть там же, где..", "быть ненулевой проекцией", "находиться в ε-окрестности", "быть частью", "находиться на одной прямой", "находиться вокруг", "быть на краю", "быть параллельно", "быть перпендикулярно", "быть симметрично", "находиться в n единицах от..", "иметь точку опоры на..", "иметь точку подвеса на..", "соприкасаться", "быть выше", "быть ниже", "нахо-

диться на одинаковом уровне", "быть дальше", "быть ближе", "быть равноудаленными", n-арное отношение – "быть между".

Из базовых отношений с помощью логических связок строятся производные отношения, такие как "не соприкасаться" (отрицание "соприкасаться"), "быть вместе.." (следствие от "находиться там же.."), "висеть" (конъюнкция "иметь вертикальное положение" и "висеть на…"), "стоять" (конъюнкция "иметь вертикальное положение" и "иметь точку опоры на..") и т.п.

2.6. Правила-продукции

Правило-продукция (или просто правило) в общем случае можно представить в виде

$$<I, S, P, A \rightarrow B, F>,$$

где I – идентификатор правила (обычно порядковый номер);
: S – область применимости;
P – условие применимости;
A – посылка правила;
B – заключение;
F – постусловие правила.

A→B является ядром правила-продукции и может по-разному интерпретироваться. Наиболее часто используемая форма интерпретации – логическая, при которой A является множеством элементарных условий, связанных логическими связками "И", "ИЛИ" и "НЕТ", B – множеством элементарных заключений. При этом правило считается сработавшим (выполняется заключение B), если посылка A истинна. Другой формой интерпретации ядра является вероятностная интерпретация, при которой правило срабатывает с некоторой вероятностью, зависящей от истинности посылки.

В качестве заключения обычно применяется операция добавления факта в базу данных интеллектуальной системы с указанием меры достоверности получаемого факта. В качестве постусловия могут использоваться какие-либо дополнительные действия или комментарии, сопровождающие правило.

Обычно при описании баз знаний или экспертных систем правила представляются в более наглядном виде, например:

```
ПРАВИЛО 1:
ЕСЛИ
        Образование=Высшее И
        Возраст=Молодой И
        Коммуникабельность=Высокая
ТО
        Шансы найти работу=Высокие КД=0.9.
```

При срабатывании этого правила в базу данных интеллектуальной системы (например, экспертной системы) добавляется факт, означающий, что шансы

найти работу высоки с достоверностью 0.9 или 90% (значение коэффициента достоверности КД). Понятия "Образование", "Возраст", "Коммуникабельность" служат для задания условия (в данном случае, конъюнкции), при котором срабатывает правило.

Факты хранятся в базе данных продукционной системы в форме

(Объект, значение, КД)

или

(Объект, атрибут, значение, КД).

Но могут использоваться и другие структуры для хранения фактов, такие как семантические сети или фреймы (см. 2.6 и 2.7). В этом случае говорят о комбинации разных методов представления знаний или о гибридных интеллектуальных (экспертных) системах. При интерпретации (выполнении) правила в ходе проверки условия система проверяет факты, находящиеся уже в базе данных, и, если соответствующего факта нет, обращается за ним к источнику данных (пользователю, базе данных и т.д.) с вопросом (или запросом).

Кроме правил в продукционных базах знаний могут использоваться метаправила для управления логическим выводом. Пример метаправила для гипотетической базы знаний, пример из которой был приведен ранее:

ЕСЛИ

Экономика = развивается

ТО

Увеличить приоритет правила 1

Для представления нечетких знаний факты и правила в продукционных системах снабжаются коэффициентами достоверности (или уверенности), которые могут принимать значения из разных интервалов в разных системах (например, <0,1>, <0, 100>, <-1,+1>). Во втором случае можно говорить об уверенности в процентах, а в последнем случае – о задании коэффициентом уверенности меры ложности или истинности факта.

Существуют разные методы обработки нечеткости при интерпретации правил. Обычно для оценки истинности условия используются правила нечеткой логики (см. 2.3). Для оценки истинности факта, полученного при срабатывании правила, обычно также используется правило из нечеткой логики для оценки конъюнкции, аргументами которой являются условие и факт в заключении со своими коэффициентами достоверности.

Более разнообразные подходы для оценки истинности используются при формировании правилом факта, уже существующего в базе данных интеллектуальной системы. Ниже приводятся формулы, используемые в таком случае в экспертной системе MYCIN (в ней коэффициент принадлежности пробегает значения из интервала <-1,+1>):

$$КД = \begin{cases} ИП + РП(1 - ИП); ИП, РП > 0 \\ -(|ИП| + |РП|(1 - |ИП|); ИП, РП < 0 \\ \dfrac{|ИП| + |РП|}{1 - \min(|ИП|, |РП|)}; ИП * РП < 0, \end{cases}$$

где: КД – новое значение факта,

ИП – показатель истинности уже существующего факта (исходный показатель),

РП – показатель факта, формируемый исходя из истинности условия и заключения правила (результирующий показатель).

Легко проверить, что получающиеся значения не входят в противоречие с интуитивным представлением, о том, как должна меняться истинность факта при срабатывании правила, подтверждающего или опровергающего его.

Достоинствами продукционного метода представления знаний являются следующие.

1. Наглядность и понятность знаний (по крайней мере, на уровне одного правила).

2. Возможность реализации немонотонного логического вывода и обработки противоречивых фактов.

3. Возможность введения различных модификаций в интерпретацию правил в соответствии с особенностями решаемых системой задач.

4. Возможность легкого наращивания базы знаний путем добавления новых правил.

Недостатками этого метода представления являются следующие.

1. Необозримость большой базы знаний и ее структуры.

2. Возможность легкого внесения серьезных искажений в базу знаний, приводящих к неправильному функционированию системы (если в системе нет развитых средств проверки целостности базы знаний).

3. Ориентация на последовательную обработку правил.

2.7. Семантические сети

В третий раз забросил старик невод.

Принес невод золотую рыбку.

А.С. Пушкин. Сказка о рыбаке и рыбке

Семантической сетью называется ориентированный граф с помеченными вершинами и дугами, где вершинам соответствуют конкретные объекты, дугам – отношения между ними.

В семантических сетях используются три основных типа объектов: понятия, события и свойства.

Понятия представляют собой сведения об абстрактных или конкретных (физических) объектах предметной области.

События – это действия, которые могут внести изменения в предметную область, т.е. изменить состояние предметной области.

Свойства используются для уточнения понятий и событий. Применительно к понятиям свойства описывают их особенности или характеристики, например – цвет, размер, качество. Применительно к событиям свойства – продолжительность, место, время и т.д.

Семантические отношения условно делятся на четыре класса: лингвистические, логические, теоретико-множественные и квантифицированные. К наиболее распространенным лингвистическим отношениям относятся падежные и атрибутивные отношения. Падежными (или ролевыми) отношениями могут являться следующие:

– агент, отношение между событием и тем, что (или кто) его вызывает, например, отношение между "завинчиванием" (гайки) и рукой;

– объект, отношение между событием и тем, над чем производится действие, например, между "завинчиванием" и "гайкой";

– условие, отношение, указывающее логическую зависимость между событиями, например, отношение между "завинчиванием" (гайки) и "сборкой" (узла);

– инструмент, отношение между событием и объектом, с помощью которого оно совершается, например, между "завинчиванием" и "верстаком".

Атрибутивные отношения – это отношения между объектом и свойством, например, цвет, размер, форма, модификация и т.д. На рис. 6 приведен пример семантической сети с использованием атрибутивных отношений.

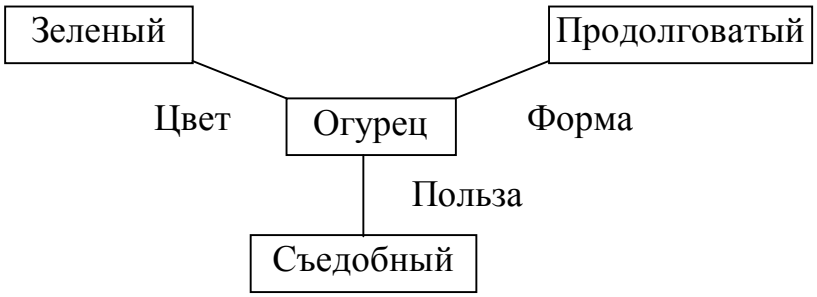

Рис. 6. Пример атрибутивных отношений

Логические отношения – это операции, используемые в исчислении высказываний: дизъюнкция, конъюнкция, импликация, отрицание.

Теоретико-множественные отношения – это отношения между элементом множества (подмножества) и множеством, отношение части и целого и т.п. Этот тип отношений используется для хранения в базе знаний сложных (составных или иерархических) понятий. Этот тип отношений иллюстрируется рис. 7.

Квантифицированные отношения – это логические кванторы общности и существования. Они используются для представления знаний типа: "любой студент должен посещать лабораторные занятия", "существует хотя бы один язык программирования, который должен знать любой выпускник НГТУ".

К базе знаний представленной семантической сетью, возможны следующие основные типы запросов:

1) запрос на существование;

2) запрос на перечисление.

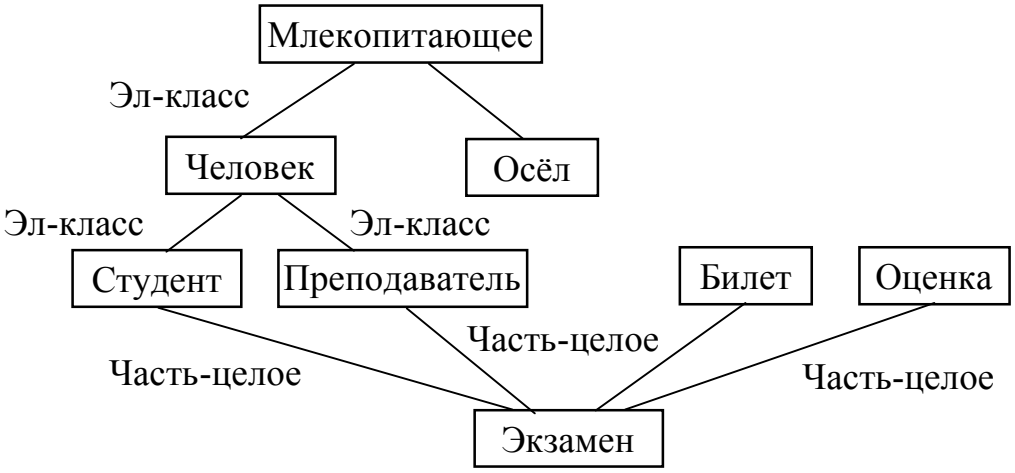

Рис. 7. Теоретико-множественные отношения

При построении интеллектуальных банков знаний обычно используют разделение интенсиональных и экстенсиональных знаний. Экстенсиональная семантическая сеть (или К-сеть) содержит информацию о фактах, о конкретных объектах, событиях, действиях. Интенсиональная семантическая сеть (или А-сеть) содержит информацию о закономерностях, потенциальных взаимосвязях между объектами, неизменяемую информацию об объектах, т.е. модель мира. Экстенсиональные (конкретные) знания создаются и обновляются в процессе работы с банком данных, а интенсиональные (абстрактные) изменяются редко. Первые можно назвать экземпляром, а последние – моделью (схемой) базы данных.

Запрос к банку знаний, обрабатываемый системой управления базой знаний, представляет собой набор фактов (ситуацию), при описании которого допускается использование переменных вместо значений атрибутов, имен понятий, событий и отношений. Запрос можно представить в виде графа, в котором вершины, соответствующие переменным, не определены.

Поиск ответа сводится к задаче изоморфного вложения графа запроса (или его подграфа) в семантическую сеть.

Запрос на существование не содержит переменных и требует ответа типа ДА, если изоморфное вложение графа запроса в семантическую сеть удалось, и НЕТ – в противоположном случае. При обработке запроса на перечисление происходит поиск всех возможных изоморфных графу запроса подграфов в

семантической сети, а также присваивание переменным в запросе значений из найденных подграфов.

Достоинством семантических сетей является их универсальность, достигаемая за счет выбора соответствующего применению набора отношений. В принципе с помощью семантической сети можно описать сколь угодно сложную ситуацию, факт или предметную область.

Недостатком семантических сетей является их практическая необозримость при описании модели мира реального уровня сложности. При этом появляется проблема размещения семантической сети в памяти ЭВМ. Если ее размещать всю в оперативной (виртуальной) памяти, на ее сложность накладываются жесткие ограничения. Если размещать во внешней памяти, появляется проблема, как подгружать необходимые для работы участки.

Частично эта проблема структуризации семантических сетей решается выделением фрагментов семантической сети, называемых обычно высказываниями. Ниже приводится фрагмент семантической сети, состоящей из двух высказываний, связанных отношением импликации (следования). Смысл этого фрагмента семантической сети можно выразить высказыванием (рис. 8):

"Если студент учится в НГТУ, значит он – умный".

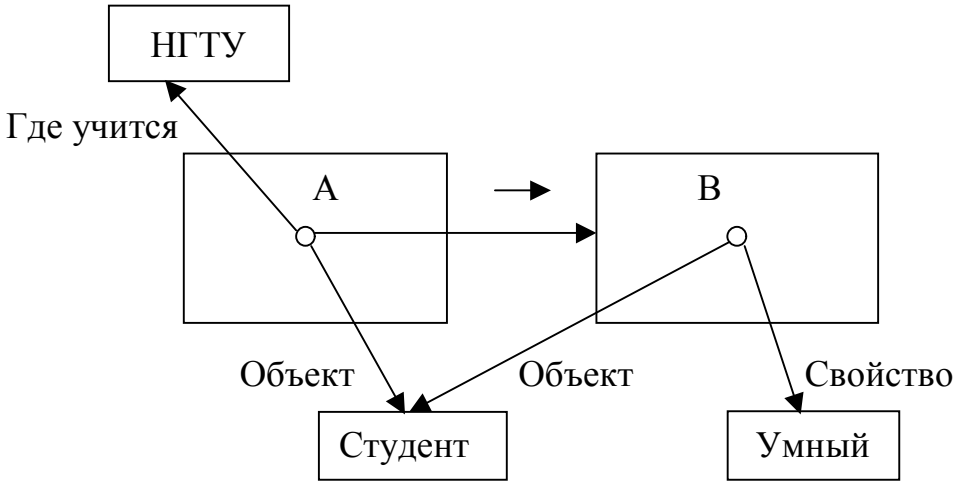

Рис. 8. Пример представления двух высказываний, связанных импликацией

Кроме того, в семантических сетях нельзя явно задавать наследование свойств, т.к. отношение "элемент класса – класс" является одним из многих типов отношений и его обработка (участие в процедуре поиска подходящих фрагментов) ничем не отличается от обработки других отношений.

Эта проблема структуризации семантических сетей и необходимости задания в них наследования свойств привела к идее структуризации семантических сетей, приведшей к появлению концепции фреймов.

2.8. Фреймы

"Держите себя в рамках!"
Реплика

В основе теории фреймов лежит восприятие фактов посредством сопоставления полученной извне информации с конкретными элементами и значениями, а также, с рамками, определенными для каждого концептуального объекта в памяти. Структура, представляющая эти рамки, называется фреймом.

Другими словами, фрейм – это структура, описывающая фрагмент базы знаний, который в какой-то степени рассматривается и обрабатывается обособленно от других фрагментов. Другие фрагменты, с которыми он связан, во фрейме представлены только их именами (идентификаторами) так же как и он в них.

В виде фрейма может описываться некоторый объект, ситуация, абстрактной понятие, формула, закон, правило, визуальная сцена и т.п.

Понятие фрейма неразрывно связано с абстрагированием и построением иерархии понятий.

Из понятия "фрейм", появившегося в конце 60-х годов в работах М. Минского, выросло в дальнейшем понятие объекта и объектно-ориентированного программирования с его идеями инкапсуляции данных в объекте, наследования свойств и методами, привязанными к описанию объектов.

Так как фрейм является более общей и гибкой концепцией, чем "объект", в дальнейшем будем использовать терминологию, сложившуюся при описании фреймов, подразумевая, что с некоторыми ограничениями они могут быть перенесены в среду объектно-ориентированного программирования.

Фреймы подразделяются на два типа: фреймы-прототипы (или классы) и фреймы-примеры (или экземпляры). Фреймы-прототипы используются для порождения фреймов-примеров.

В общем виде фрейм можно описать как структуру, состоящую из имени фрейма, множества слотов, характеризующихся именами и значениями, и множества присоединенных процедур, связанных с фреймом или со слотами:

$$F = (NF, P, (NS_1, VS_1, P_1), ..., (NS_i, VS_i, P_i), ..., (NS_n, VS_n, P_n)),$$

где NF – имя фрейма, NS – имя слота, VS – значение слота, P – присоединенная процедура.

Во фреймах различают два типа присоединенных процедур: процедуры-демоны и процедуры-слуги. Первые из них запускаются автоматически при наличии некоторых условий или событий. Процедуры-слуги запускаются явно.

В реальных системах, базирующихся на фреймах, структура, описанная выше, может быть более сложной за счет иерархической структуры значений слотов (наличия более глубоких уровней в описании фрейма).

Ниже приводится пример фрейма-прототипа, описывающего земельный участок в виде многоугольника:

Имя фрейма: Земельный участок
Количество сторон: (4)
Длины сторон:
Размеры углов:
Площадь: IF_NEEDED: Вычисление площади
IF_ADDED: Вычисление цены
Цена:

Здесь слот "Количество сторон" имеет значение "по умолчанию", равное 4, т.к. подавляющее большинство земельных участков имеет форму четырехугольника.

К слоту "Площадь" присоединены процедуры-демоны вычисляющие площадь и цену, соответственно, запускаемые при запросе слота (событие "IF_NEEDED") и добавлении значения слота "Площадь" (событие "IF_ADDED").

В инженерии знаний существуют языки представления знаний, базирующиеся на концепции фреймов. Наиболее известные из них: KRL и FRL. Оба эти языка являются расширением языка LISP – функционального языка программирования.

В инженерии знаний существуют языки представления знаний, базирующиеся на концепции фреймов. Наиболее известные из них: KRL и FRL. Оба эти языка являются расширением языка LISP – функционального языка программирования.

2.9. Нейронные сети

В нейронных сетях знания содержатся в состояниях множества так называемых нейроподобных элементов (или просто нейронов) и связей между ними. Направление, которое во главу угла ставит связи между нейронами, называется коннективизмом.

Формальная модель нейрона Мак-Каллока-Питтса, которая и сейчас является наиболее применяемым формализмом для описания отдельного нейрона в нейронной сети, показана на рис. 9.

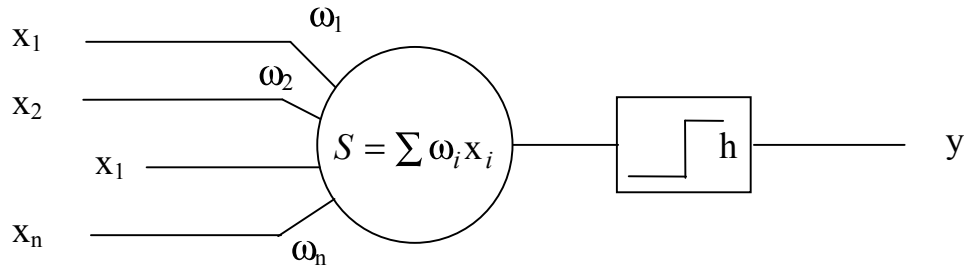

Рис. 9. Формальная модель нейрона Мак-Каллока – Питтса

Здесь: x_i – сигнал на i-м входе (синапсе) нейрона;
ω_i – вес i-го входа (синапса) нейрона;
y – выход нейрона;
h – порог срабатывания нейрона.

В модели взвешенная сумма сигналов на входах нейрона сравнивается с пороговым значением h, и на выходе есть сигнал, если она превышает порог. В современных моделях нейронов пороговая функция в общем случае заменяется на нелинейную функцию $y = f(S)$, называемую передаточной функцией или функцией активации нейрона. В качестве этой функции может использоваться, одна из сигмоидальных функций, например, рациональная сигмоида

$$f(S) = \frac{S}{S + \alpha}$$

Параметр α обычно называется смещением. Таким образом, иногда говорят, что нейрон состоит из умножителей (на веса), сумматора и нелинейного элемента.

Из связанных определенным образом нейронов (узлов) строится нейронная сеть с определенным количеством входов и выходов. Обычно различают три типа узлов (нейронов) – входные (входной слой нейронов или Input layer), выходные (выходной слой или Output layer) и скрытые слои нейронов (Hidden layers) (рис. 10).

Функционирование нейронной сети состоит из двух этапов: обучения сети "правильному" или адекватному реагированию на входную информацию (входной вектор) и использования обученной сети для распознавания входных векторов. Последний этап часто называют тестированием. Другими словами, сеть учится распознаванию входных векторов, т.е. формированию выходных векторов, соответствующих распознанному классу входных векторов. При этом знания о соответствии входных векторов выходным сохраняются в весах синапсов

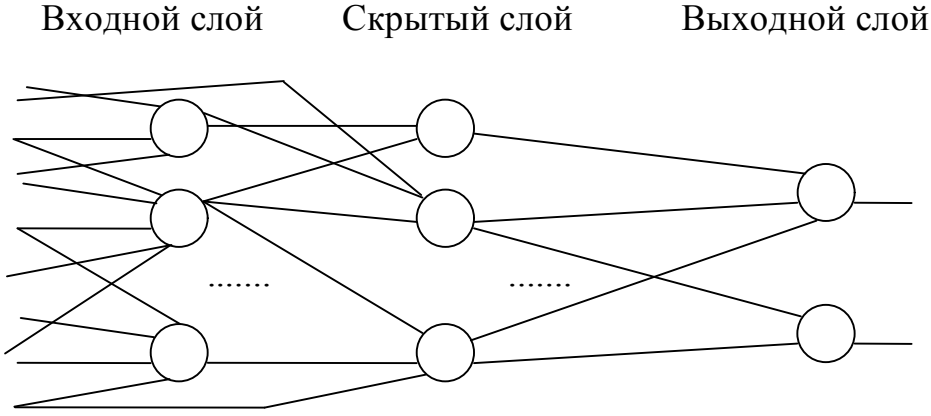

Рис. 10. Нейронная сеть с прямыми связями

и порогах нейронов. В случае коннективистского подхода – только в весах синапсов. Иногда под входным вектором понимается конкатенация входного и выходного вектора и не все разряды этого вектора могут задаваться при обучении и тестировании сети. В некоторых моделях нейронных сетей (например, модели Хопфилда) входные и выходные сигналы не различаются и соответствующие им входы (выходы) сети могут меняться ролями в процессе функционирования сети.

Функционирование нейронной сети часто описывают в терминах задач оптимизации. Ее обучение можно представить как формирование n-мерной гиперповерхности (где n – размерность входного вектора), определенной целевой функции, обычно называемой энергией сети. Описание этой гиперповерхности хранится в карте весов синапсов и в порогах нейронов. Например, в случае модели Хопфилда энергетическая функция описывается формулой

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\, x_i\, x_j$$

где N – количество нейронов (размерность входного вектора),
: w_{ij} – вес связи между i-м и j-м нейронами,
x_i – состояние i-го нейрона.

При тестировании (использовании) обученной нейронной сети происходит процесс поиска ближайшего минимума целевой функции. При этом происходит восстановление искаженных разрядов входного вектора или "вспоминание" неизвестных разрядов, ассоциативно связанных с заданными (известными) разрядами.

В настоящее время существует большое разнообразие моделей нейронных сетей. Их различают по структуре сети (связей между нейронами), особенностям модели нейрона, особенностям обучения сети.

По структуре нейронные сети можно разделить на неполносвязные (или слоистые) и полносвязные, со случайными и регулярными связями, с симметричными и несимметричными связями.

Неполносвязные нейронные сети описываются неполносвязным ориентированным графом. Наиболее распространенным типом таких нейронных сетей являются перцептроны: однослойные (простейшие перцептроны) и многослойные, с прямыми, перекрестными и обратными связями. В нейронных сетях с прямыми связями нейроны j-ого слоя по входам могут соединяться только с нейронами i-ых слоев, где $j > i$, т.е. с нейронами нижележащих слоев. В нейронных сетях с перекрестными связями допускаются связи внутри одного слоя, т.е. выше приведенное неравенство заменяется на $j \geq i$. В нейронных сетях с обратными связями используются и связи j-ого слоя по входам с i-ым при $j < i$. Кроме того, по виду связей различают перцептроны с регулярными и случайными связями.

По используемым на входах и выходах сигналам нейронные сети можно разделить на аналоговые и бинарные.

По моделированию времени нейронные сети подразделяются на сети с непрерывным и дискретным временем. Для программной реализации применяется, как правило, дискретное время.

По способу подачи информации на входы нейронной сети различают:

- подачу сигналов на синапсы входных нейронов,
- подачу сигналов на выходы входных нейронов,
- подачу сигналов в виде весов синапсов входных нейронов,
- аддитивную подачу на синапсы входных нейронов.

По способу съема информации с выходов нейронной сети различают:

- съем с выходов выходных нейронов,
- съем с синапсов выходных нейронов,
- съем в виде значений весов синапсов выходных нейронов,
- аддитивный съем с синапсов выходных нейронов.

По организации обучения разделяют обучение нейронных сетей с учителем (supervised neural networks) и без учителя (nonsupervised). При обучении с учителем предполагается, что есть внешняя среда, которая предоставляет обучающие примеры (значения входов и соответствующие им значения выходов) на этапе обучения или оценивает правильность функционирования нейронной сети и в соответствии со своими критериями меняет состояние нейронной сети или поощряет (наказывает) нейронную сеть, запуская тем самым механизм изменения ее состояния. Под состоянием нейронной сети, которое может изменяться, обычно понимается:

- веса синапсов нейронов (карта весов – map) (коннекционистский подход);
- веса синапсов и пороги нейронов (обычно в этом случае порог является более легко изменяемым параметром, чем веса синапсов);
- установление новых связей между нейронами (свойство биологических нейронов устанавливать новые связи и ликвидировать старые называется пластичностью).

По способу обучения разделяют обучение по входам и по выходам. При обучении по входам обучающий пример представляет собой только вектор входных сигналов, а при обучении по выходам в него входит и вектор выходных сигналов, соответствующий входному вектору.

По способу предъявления примеров различают предъявление одиночных примеров и "страницы" примеров. В первом случае изменение состояния нейронной сети (обучение) происходит после предъявления каждого примера. Во втором – после предъявления "страницы" (множества) примеров на основе анализа сразу их всех.

Если рассматривать нейронную сеть как способ представления знаний, то в ней хранятся знания об ассоциативных связях между стимулами (входными векторами) и откликами (выходными векторами).

Знания хранятся (формируются в процессе обучения) обычно в форме весов связей между нейронами.

Недостатками нейронных сетей в качестве метода представления знаний являются:

- трудности вербализации результатов работы нейронной сети и объяснений, почему она приняла то или иное решение,
- невозможность гарантировать повторяемость и однозначность получения результатов

Преимущества нейронных сетей в качестве метода представления знаний:

- отсутствие необходимости формализации знаний, формализация заменяется обучением на примерах,
- естественное представление и обработка нечетких знаний примерно так, как это осуществляется в естественной интеллектуальной системе – мозге
- ориентация на параллельную обработку, что при соответствующей аппаратной поддержке обеспечивает возможность работы в реальном времени
- отказоустойчивость и живучесть при аппаратной реализации нейронной сети
- возможность обработки многомерных (размерности больше трех) данных и знаний так же (без увеличения трудоемкости) как и небольшой размерности (но в этом случае затруднено объяснение результатов, т.к. человек с трудом воспринимает многомерность)

В настоящее время наметились тенденции объединения в одной системе логических и/или эмпирических методов представления знаний (вербальных или символьных) с ассоциативными, использующими нейронные сети. В ходе этих исследований появились такие парадигмы как:

- семантические нейронные сети,
- нечеткие нейронные сети,
- "двухполушарные" экспертные системы.

Целью этих исследований является создание систем ИИ, способных обучаться примерно так, как это делает человек, создавая при обучении ассоциативные связи между параметрами внешней среды и символьными понятиями и иерархии понятий, и решать задачи, комбинируя и чередуя ассоциативный поиск и логический вывод.

3. МЕТОДЫ ОБРАБОТКИ ЗНАНИЙ

3.1. Дедуктивный логический вывод

"Странная игра, – сказал Джошуа. – Единственная выигрышная стратегия
– не играть вообще".

Д. Бишоф. Недетские игры

Для решения задач в продукционной интеллектуальной системе существует два основных метода дедуктивного логического вывода: обратный и прямой. Может использоваться и комбинация этих двух методов. При обратном логическом выводе процесс интерпретации правил начинается с правил, непосредственно приводящих к решению задачи. В них в правой части находятся заключения с фактами, являющимися решением (целевыми фактами). При интерпретации этих правил в процесс решения могут вовлекаться другие правила, результатом выполнения которых являются факты, участвующие в условиях конечных правил и т.д.

В самом общем виде алгоритм обратного логического вывода, записанный на псевдокоде в виде функции, выглядит так (условие ограничено конъюнкцией элементарных условий):

функция Доказана_Цель(Цель): boolean;
Поместить Цель в стек целей.
пока стек целей не пуст
цикл
Выбор цели из стека целей и назначение ее текущей.
Поиск множества правил, в правой части которых
находится текущая цель (множества подходящих правил).
Считать, что Цель не доказана.
пока множество походящих правил не пусто
и Цель не доказана
цикл
Выбор из этого множества одного текущего правила
с использованием определенной стратегии.
Считать текущим элементарным условием первое.
пока не проверены все элементарные условия правила
и не надо прервать проверку условия
цикл
если в текущем элементарном условии
участвует факт,
встречающийся в правой части
какого-то правила

```
                    то
                      если не Доказана_Цель(Этот факт)
                        то
                                Надо прервать проверку условия
                          конец_если
                  иначе
                     Запросить информацию о факте.
                     Проверить элементарное условие.
                     если  элементарное условие истинно
                       то
                         Добавить факт в базу данных.
                         Перейти к следующему элементарному
                              Условию.
                       иначе
                          Надо прервать проверку условия.
                     конец_если
                  конец_если
               конец_цикла
            если условие правила истинно
              то
                  Выполнить заключение.
                  Исключить Цель из стека целей.
                  Считать, что Цель доказана.
            конец_если
         конец_цикла
      конец_цикла
конец_функции.
```

Существует много различных стратегий выбора правила из подходящих. Наиболее простой и часто встречающейся стратегией является "первая попавшаяся". При этой стратегии решение задачи зависит от порядка расположения (перебора) правил в базе знаний. Другие используемые стратегии выбора:

"стопки книг" (LIFO),

"по приоритету",

"наиболее свежие данные",

"наиболее длинное условие".

В прямом методе логического вывода интерпретация правил начинается от известных фактов, т.е. сначала выполняются правила, условия которых можно проверить с использованием фактов, уже находящихся в базе данных.

В общем виде алгоритм прямого вывода выглядит так:

```
пока Цель не доказана
  цикл
      Формирование множества подходящих правил
      (по их условиям и наличию фактов).
```

Выбор одного правила из этого множества (с использованием определенной стратегии выбора).
Считать текущим элементарным условием первое.
пока не проверены все элементарные условия правила
и не надо прервать проверку условия
цикл
если элементарное условие истинно
то
Перейти к следующему элементарному условию.
иначе
Надо прервать проверку условия.
конец_если
конец_цикла
Выполнить заключение.
если при формировании заключения появился целевой факт
то
Считать, что Цель доказана.
конец_если
конец_цикла.

Метод прямого логического вывода можно применять тогда, когда факты появляются в базе данных не зависимо от того, какую задачу сейчас требуется решить (какой целевой факт доказать) и в разные моменты времени. В этом случае можно говорить о том, что факты управляют логическим выводом (решением задачи). Кроме того, этот метод целесообразно применять для формирования вторичных признаков (фактов) из первичных для подготовки решения задачи в дальнейшем с применением обратного логического вывода.

Метод обратного логического вывода можно применять тогда, когда необходимо минимизировать количество обращений к источнику данных (например, пользователю), исключив из рассмотрения заведомо ненужные для решения задачи факты.

3.2. Методы поиска релевантных знаний

"Иди туда, не знаю куда,
ищи то, не знаю что".
Описание проблемы будущего
искусственного интеллекта
в русских народных сказках

В основе практически всех методов решения задач в искусственном интеллекте лежит какой-либо из вариантов сопоставления фрагмента базы знаний (запроса) с образцами, хранящимися в базе знаний (или поиск релевантных

знаний). Этот процесс в общем случае состоит из процедуры сравнения двух фрагментов знаний (собственно, сопоставление) и процедуры перебора вариантов для сопоставления.

По способу сравнения двух фрагментов знаний различают следующие виды сопоставлений:

- синтаксическое;
- параметрическое;
- семантическое.

При синтаксическом сопоставлении происходит сравнение на полное тождество двух структур данных, представляющих знания. Единственное различие, допускаемое в этих структурах и приводящее к успешному сопоставлению, возможно в неопределенных переменных, включенных в сравниваемые структуры. Примером такого сопоставления является унификация предикатов в Прологе, при которой сначала проверяются на тождество имена предикатов, затем, проверяется количество аргументов в них, а затем попарно сравниваются аргументы (при этом неопределенным переменным, если они есть в одном из предикатов, присваиваются значения из другого).

При параметрическом сопоставлении возможно неполное тождество сравниваемых структур данных, представляющих фрагменты знаний. При этом результатом сопоставления является некоторая вычисляемая при сравнении мера их синтаксического различия (различия в структуре данных), которая может состоять из множества параметров, таких как различие в количестве аргументов предиката, количественное различие в написании имен предикатов или аргументов (или названий или значений слотов) и т.п.

При семантическом сопоставлении происходит сравнение семантики (смысла) сравниваемых фрагментов знаний. Обычно это приводит к необходимости просмотра структур данных, связанных со сравниваемыми фрагментами (других фрагментов знаний), т.е. описывающих их семантику и, в конечном итоге, к синтаксическому или параметрическому сопоставлению фрагментов знаний, из которых они состоят. Последние фрагменты являются более элементарными (базовыми) фрагментами. Их можно сравнить с терминальными символами при грамматическом разборе, а фрагменты, для сопоставления которых используется семантическое сравнение, – с нетерминальными символами. При семантическом сопоставлении обычно используют понятие семантической близости понятий (фрагментов знаний). Иногда семантическая близость может быть вычислена непосредственно, например, при сопоставлении двух лингвистических переменных, описанных на одной и той же метрической шкале. В более сложном случае семантическая близость может оцениваться в процессе просмотра описаний семантик, описывающих сравниваемые фрагменты знаний. При этом могут использоваться результаты параметрического сопоставления базовых фрагментов знаний. Универсальных и достаточно хорошо математически обоснованных алгоритмов семантического сопоставления (в отличие от синтаксического) не существует.

Несколько в стороне находится поиск релевантных знаний, реализуемый в нейронных сетях, который можно назвать ассоциативным поиском (см. 2.8). В отличие от методов, описанных выше, знания в этом случае представляются в виде сигналов или численных значений состояний нейронов, а не в виде символьных обозначений.

3.3. Основные понятия о методах приобретения знаний

"Многие вещи нам не понятны, не потому, что наши понятия слабы, но потому, что сии вещи не входят в круг наших понятий".

Козьма Прутков

"... человек (или животное) воспринимает окружающую среду в той мере, в какой он может взаимодействовать с этой средой".

Я. Сентоготаи, М.Арбиб "Концептуальные модели нервной системы"

Основной проблемой при разработке современных экспертных систем является проблема приобретения знаний, т.е. преобразование разного вида информации (данных) из внешнего представления в представление в виде знаний, пригодное для решения задач, для которых создается экспертная система. Эту проблему часто называют проблемой извлечения знаний из данных (в более общем виде, из внешнего мира), которая сводится к задаче обучения интеллектуальной системы.

Примерами задач извлечения знаний являются:

1) выявление причинно-следственных связей между атрибутами реляционной базы данных и формирование их в виде правил в продукционной экспертной системе;

2) формирование программы (или правил) решения задачи (например, планирования производственного процесса или поведение робота) на основе примеров удачного планирования, вводимых в компьютер;

3) выявление информативных признаков для классификации объектов, существенных с точки зрения решаемой задачи.

Обучающиеся системы можно классифицировать по двум признакам: уровень, на котором происходит обучение и применяемый метод обучения. По первому признаку различают обучение на символьном уровне (SLL – symbol

level learning), при котором происходит улучшение представления знаний на основе опыта, полученного при решении задач, и обучение на уровне знаний (KLL – knowledge level learning), при котором происходит формирование новых знаний из существующих знаний и данных.

На символьном уровне обучение сводится к манипулированию уже существующими структурами, представляющими знание, например, корректировка коэффициентов достоверности правил-продукций, изменение порядка расположения (просмотра) правил-продукций в базе знаний вводимого пользователем описания решения задачи на достаточно формализованном языке, не сильно отличающимся от языка, на котором представляются знания в системе.

На уровне знаний обучение сводится к выявлению и формализации новых знаний. Например, из фактов

журавль умеет летать,
воробей умеет летать,
синица умеет летать,
журавль есть птица,
воробей есть птица,
синица есть птица

система может сформулировать правило-продукцию

Если X есть птица
то X умеет летать.

По признаку применяемого метода обучения различают системы, в которых используются аналитические или эмпирические методы обучения. Аналитические, в свою очередь, делятся на использующие глубинные (knowledge-rich) или поверхностные (knowledge-drizen) знания.

Эмпирические делятся на использующие знания (knowledge-learning) или данные (data-drizen).

С другой стороны в инженерии знаний известны три основных подхода к приобретению знаний: индуктивный вывод, вывод по аналогии и обучение на примерах. В основе индуктивного вывода лежит процесс получения знаний из данных и/или других знаний (в продукционных системах – правил из фактов и/или других правил). Вывод по аналогии основан на задании и обнаружении аналогий между объектами (ситуациями, образами, постановками задачи, фрагментами знаний) и применением известных методов (процедур) к аналогичным объектам. В основе обучения на примерах лежит демонстрация системе и запоминание ей примеров решения задач. Резкой границы между этими методами не существует, т.к. все они базируются на обобщении, реализованной в той или иной форме, т.е. реализуют переход от более конкретного знания (фактов) к более абстрактному знанию.

На рис. 11 показана классификация обучающихся систем и взаимосвязи между понятиями, связанными с приобретением знаний.

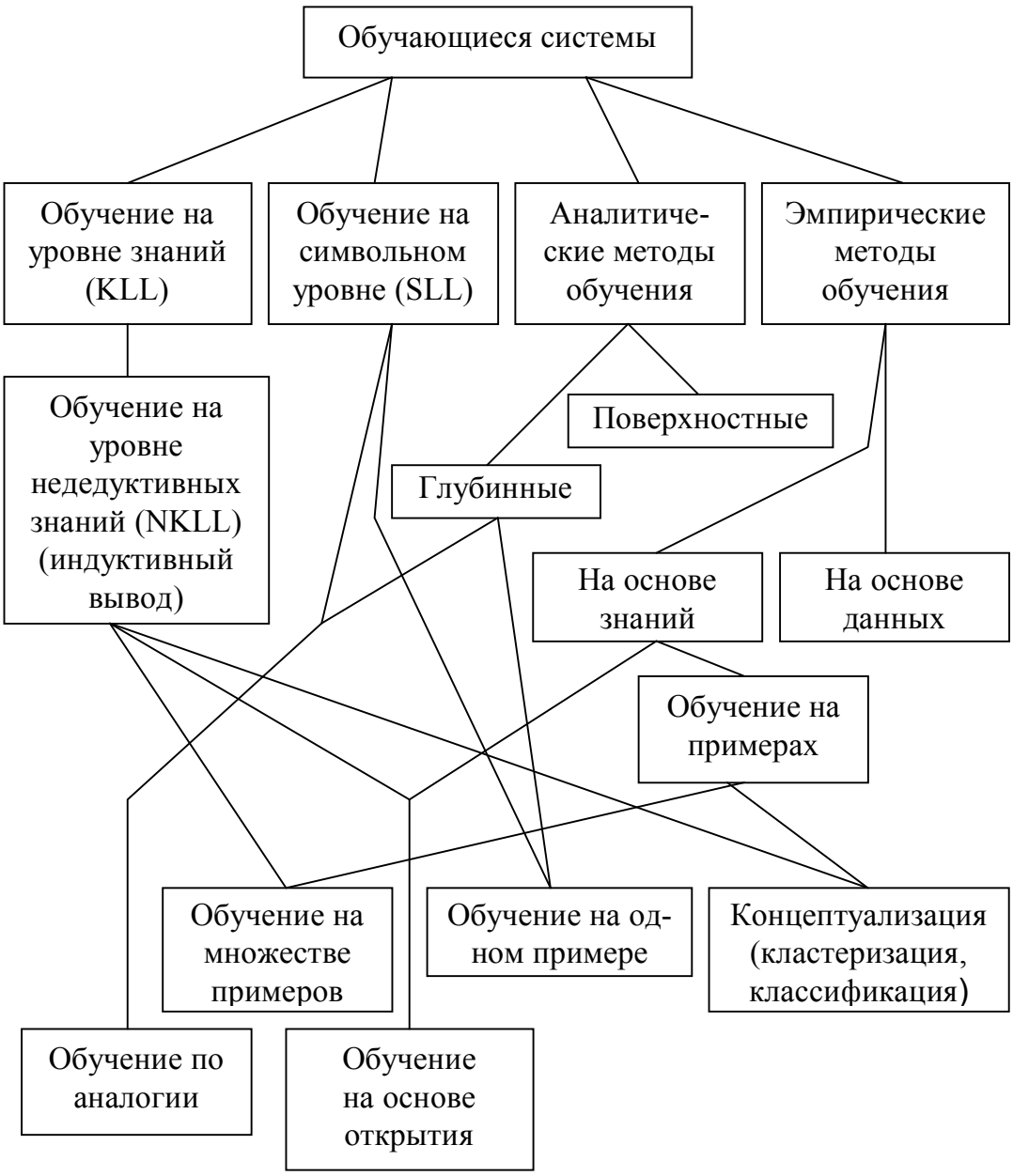

Рис. 11. Классификация обучающихся систем и взаимосвязи ее с используемой терминологией

4. ЭКСПЕРТНЫЕ СИСТЕМЫ

"Узкий специалист – это человек,
который не разбирается ни в чем,
в чем разбираются другие".

Шутка

4.1. Структура экспертных систем

На рис. 12 изображена обобщенная структура экспертной системы.

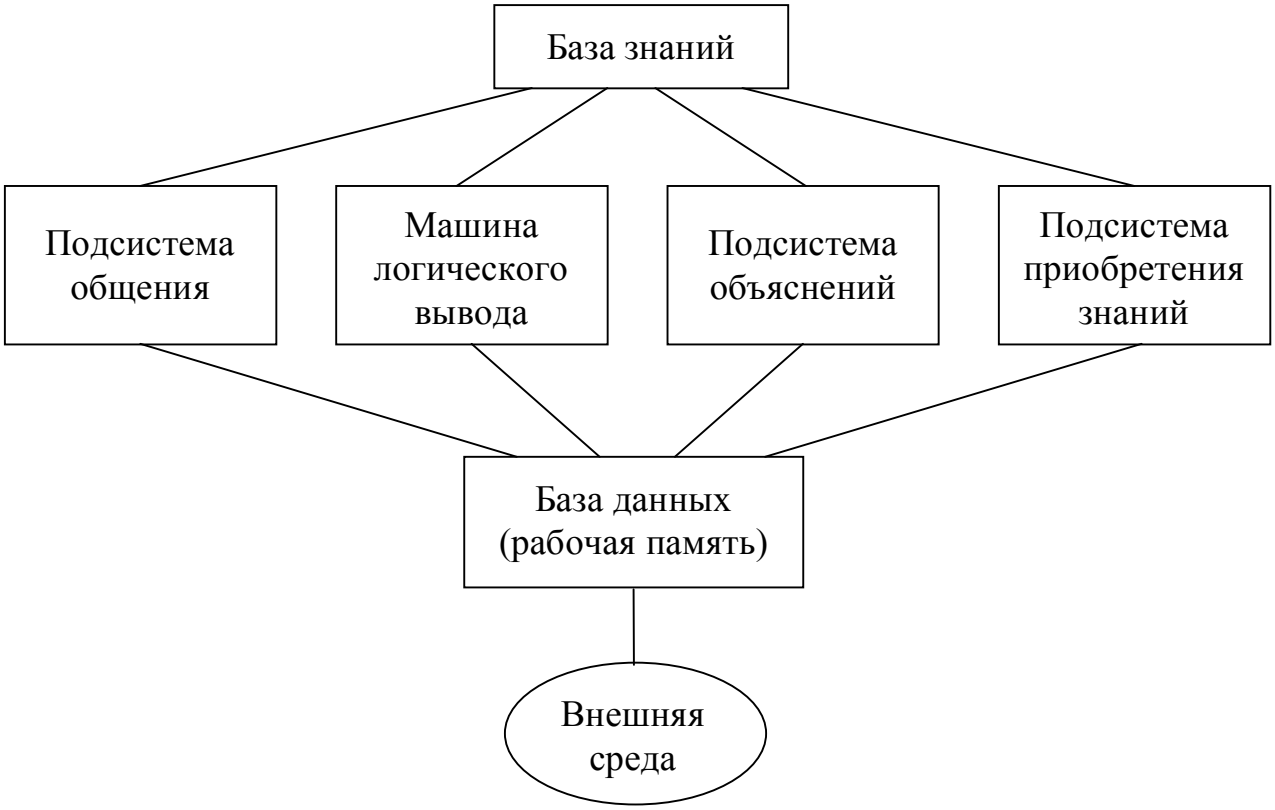

Рис. 12. Структура экспертной системы

База знаний предназначена для хранения экспертных знаний о предметной области, используемых при решении задач экспертной системой.

База данных предназначена для временного хранения фактов или гипотез, являющихся промежуточными решениями или результатом общения системы с внешней средой, в качестве которой обычно выступает человек, ведущий диалог с экспертной системой.

Машина логического вывода – механизм рассуждений, оперирующий знаниями и данными с целью получения новых данных из знаний и других данных, имеющихся в рабочей памяти. Для этого обычно используется программно реализованный механизм дедуктивного логического вывода (какая-либо его

разновидность) или механизм поиска решения в сети фреймов или семантической сети.

Машина логического вывода может реализовывать рассуждения в виде:

1) дедуктивного вывода (прямого, обратного, смешанного);
2) нечеткого вывода;
3) вероятностного вывода;
4) унификации (подобно тому, как это реализовано в Прологе);
5) поиска решения с разбиением на последовательность подзадач;
6) поиска решения с использованием стратегии разбиения пространства поиска с учетом уровней абстрагирования решения или понятий, с ними связанных;
7) монотонного или немонотонного рассуждения;
8) рассуждений с использованием механизма аргументации;
9) ассоциативного поиска с использованием нейронных сетей;
10) вывода с использованием механизма лингвистической переменной.

Подсистема общения служит для ведения диалога с пользователем, в ходе которого ЭС запрашивает у пользователя необходимые факты для процесса рассуждения, а также, дающая возможность пользователю в какой-то степени контролировать и корректировать ход рассуждений экспертной системы.

Подсистема объяснений необходима для того, чтобы дать возможность пользователю контролировать ход рассуждений и, может быть, учиться у экспертной системы. Если нет этой подсистемы, экспертная система выглядит для пользователя как "вещь в себе", решениям которой можно либо верить, либо нет. Нормальный пользователь выбирает последнее, и такая ЭС не имеет перспектив для использования.

Подсистема приобретения знаний служит для корректировки и пополнения базы знаний. В простейшем случае это – интеллектуальный редактор базы знаний, в более сложных экспертных системах – средства для извлечения знаний из баз данных, неструктурированного текста, графической информации и т.д.

4.3. Когда целесообразно использование экспертных систем

Экспертные системы целесообразно использовать тогда, когда: *1) разработка ЭС возможна; 2) оправдана; 3) методы инженерии знаний соответствуют решаемой задаче.*

Рассмотрим более подробно эти условия.

Разработка ЭС возможна когда:

- существуют эксперты в данной области;
- эксперты должны сходиться в оценке предлагаемого решения;
- эксперты должны уметь выразить на естественном языке и объяснить используемые методы;
- задача требует только рассуждений, а не действий;

– задача не должна быть слишком трудной, ее решение должно занимать у эксперта до нескольких часов или дней, а не недель или месяцев;

– задача должна относиться к достаточно структурированной области;

– решение не должно использовать в значительной мере здравый смысл (т.е. широкий спектр общих сведений о мире и о способе его функционирования).

Разработка ЭС оправдана, если:

– решение задачи принесет значительный эффект;

– использовать человека-эксперта невозможно из-за ограниченного количества экспертов или из-за необходимости выполнения экспертизы одновременно во многих местах;

– при передаче информации эксперту происходит значительная потеря времени или информации;

– необходимо решать задачу в окружении, враждебном человеку.

Методы инженерии знаний соответствуют задаче, если задача обладает следующими характеристиками:

– может быть естественным образом решена посредством манипуляции с символами, а не с числами;

– имеет эвристическую природу, т.е. не годится задача, которая может быть решена гарантированно с помощью некоторых формальных процедур;

– должна быть достаточно сложной, чтобы оправдать затраты, но не чрезмерно сложной;

– должна быть достаточно узкой, но практически значимой.

4.4. Этапы создания экспертных систем

В проектировании экспертных систем можно выделить следующие этапы.

1. Идентификация.

1.1. Определение участников и их ролей в процессе создания и эксплуатации экспертной системы.

В процессе создания экспертной системы могут участвовать следующие специалисты: инженеры по знаниям, эксперты, программисты, руководитель проекта, заказчики (конечные пользователи). При реализации сравнительно простых экспертных систем программистов может не быть. Роль инженера по знаниям – выуживание профессиональных знаний из экспертов и проектирование базы знаний экспертной системы и ее архитектуры. Программист необходим при разработке специализированного для данной экспертной системы программного обеспечения, когда подходящего стандартного (например, оболочки для создания экспертных систем) не существует или его возможностей не достаточно и требуются дополнительные модули.

В процессе эксплуатации могут принимать участие конечные пользователи, эксперты, администратор.

1.2. Идентификация проблемы.

На этом этапе разработчики должны ответить на ряд вопросов, определяющих особенности решаемых экспертами, а, следовательно, будущей экспертной системой, задач. Эти особенности определят и особенности архитектуры экспертной системы, формируемой на последующих этапах. К этим вопросам относятся следующие:

- какой класс задач должна решать ЭС;
- как эти задачи могут быть охарактеризованы или определены;
- какие можно выделить подзадачи;
- какие исходные данные должны использоваться для решения;
- какие понятия и взаимосвязи между ними используются при решении задачи экспертами;
- какой вид имеет решение и какие концепции используются в нем;
- какие аспекты опыта эксперта существенны для решения задачи;
- какова природа и объем знаний, необходимых для решения задачи;
- какие препятствия встречаются при решении задач;
- как эти помехи могут влиять на решение задачи.

1.3. Определение необходимых ресурсов – временных, людских, материальных.

1.4. Определение целей.

В качестве целей, преследуемых при создании экспертных систем, могут быть: повышение скорости принятия решения, повышение качества решений, тиражирование опыта экспертов и т.п.

2. Концептуализация.

На этом этапе разработчики должны ответить на следующие вопросы:

- какие типы данных нужно использовать;
- что из данных задано, а что должно быть выведено;
- имеют ли подзадачи наименования;
- имеют ли стратегии наименования;
- имеются ли ясные частичные гипотезы, которые широко используются.

3. Формализация.

4. Реализация прототипной версии.

5. Тестирование.

6. Перепроектирование прототипной версии.

4.5. Прототипы и жизненный цикл экспертной системы

По степени готовности к использованию и распространению различают четыре прототипа экспертных систем:

1) демонстрационный; предназначен для демонстрации возможностей будущей экспертной системы, основных архитектурных решений, пользовательского интерфейса, для уточнения требований к пользовательскому интерфейсу и функциям, выполняемым экспертной системой, содержит демонстрационную далеко неполную базу знаний;

2) исследовательский; предназначен для исследования направлений дальнейшего совершенствования экспертной системы и для пополнения базы знаний, может использоваться для решения реальных задач в ограниченных пределах;

3) промышленный; предназначен для использования, как правило, в организации, где был разработан, в нем возможны ограничения, условности, специализация, свойственные для данной организации;

4) коммерческий; предназначен для широкого распространения, обладает гибкостью, удобством в эксплуатации, адаптируемостью к конкретным задачам и требованиям пользователя.

Жизненный цикл экспертной системы состоит из этапов разработки и сопровождения. На этапе разработки создается программное обеспечение и база знаний экспертной системы, на этапе сопровождения происходит исправление выявленных ошибок и пополнение базы знаний без участия разработчиков (если последнее допускается архитектурой экспертной системы).

Применение экспертной системы с базой знаний, неизменяемой в процессе эксплуатации, возможно при достаточно стабильной в течение длительного времени предметной области, в которой решаются задачи. Примерами таких предметных областей являются разделы математического анализа, описание правил диагностики различных заболеваний.

Примерами областей применения, требующих гибкости со стороны создания и пополнения базы знаний, являются: планирование производства, проектирование и диагностика в области электроники, вычислительной техники и машиностроения.

4.6. Характеристики экспертных систем

Экспертные системы можно характеризовать следующими особенностями:

- область применения,
- класс решаемых задач,
- метод (методы) представления знаний,
- метод (методы) решения задач (поиска решений),
- структуризация данных (фактов) предметной области,
- структуризация/неструктуризация знаний о решении задач,
- четкость/нечеткость данных,
- четкость/нечеткость знаний,
- монотонность/немонотонность процесса решения задач,
- метод (методы) приобретения (пополнения) знаний,

- вид пользовательского интерфейса,
- динамическая или статическая предметная область,
- интеграция с другими программными системами (СУБД, системами моделирования, графическими пакетами и т.д.).

4.7. Инструментальные средства для разработки экспертных систем

Трудозатраты на разработку ЭС в значительной степени зависят от используемых инструментальных средств (ИС). Ниже приведены типы современных ИС, упорядоченные в соответствии с убыванием трудозатрат при создании экспертных систем.

1. Традиционные (в том числе объектно-ориентированные) языки программирования типа С, С++ (как правило, эти ИС используются не для создания ЭС, а для создания ИС).

2. Символьные языки программирования (например, Lisp, Prolog и их разновидности). Эти ИС в последнее время, как правило, не используются в реальных приложениях в связи с тем, что они плохо приспособлены к объединению с программами, написанными на языках традиционного программирования.

3. Инструментарий, содержащий многие, но не все компоненты ЭС. Эти средства предназначены для разработчика, от которого требуются знание программирования и умение интегрировать компоненты в программный комплекс. Примерами являются такие средства, как OPS 5, ИЛИС и др.

4. Оболочки ЭС общего назначения, содержащие все программные компоненты, но не имеющие знаний о конкретных предметных средах. Средства этого и последующего типов не требуют от разработчика приложения знания программирования. Примерами являются ЭКО, Leonardo, Nexpert Object, Kappa и др.

Подчеркнем, что в последнее время термин "оболочка" (shell) используется реже, его заменяют на более широкий термин "среда разработки" (development environment). Если хотят подчеркнуть, что средство используется не только на стадии разработки приложения, но и на стадиях использования и сопровождения, то употребляют термин "полная среда" (complete environment). Примерами таких средств для создания статических ЭС являются: Nexpert Object , ProKappa, ART*Enterprise, Level 5 Object и др.

5. Проблемно/предметно-ориентированные оболочки (среды):

- проблемно-ориентированные средства (problem-specific), ориентированные на некоторый класс решаемых задач и имеющие в своем составе соответствующие этому классу альтернативные функциональные модули (примерами таких классов задач являются задачи поиска, управления, планирования, прогнозирования и т.п.);

- предметно-ориентированные средства (domain-specific), включающие знания о некоторых типах предметных областей, что сокращает время разработки БЗ.

При использовании инструментария первого, второго и третьего типов в задачу разработчика входит программирование всех или части компонентов ЭС на языке довольно низкого уровня. При применении четвертого и пятого типов ИС разработчик приложения полностью освобождается от работ по созданию программ, его основные трудозатраты связаны с наполнением базы знаний общими и (или) специфическими знаниями. При использовании инструментария четвертого типа могут возникнуть следующие трудности:

1) управляющие стратегии, вложенные в механизм вывода инструментария, могут не соответствовать методам решения, которые использует эксперт, взаимодействующий с данной системой, что может привести к неэффективным, а возможно, и неправильным решениям;

2) язык представления знаний, принятый в инструментарии, может не подходить для данного приложения.

Значительная компенсация этих трудностей достигается применением проблемно/предметно-ориентированных средств (ИС пятого типа).

4.8. Инструментальное программное обеспечение ESWin для создания экспертных систем

4.8.1. Состав и назначение ПО

ESWin — инструментальное программное обеспечение для создания экспертных систем, решающих задачи диагностики, идентификации и классификации. В состав этого ПО входят пять программ: программная оболочка ESWin, предназначенная для интерпретации баз знаний (ориентированная на использование при разработке и отладке баз знаний), программная оболочка ESWinK для запуска баз знаний конечным пользователем, редактор-конструктор баз знаний EdKb, позволяющий в удобной форме конструировать, просматривать и редактировать базы знаний, программа KBView для просмотра и диагностики целостности баз знаний, программа KBOptim для редактирования и оптимизации баз знаний. Эти программы могут работать независимо друг от друга или в комплексе. ПО ESWin можно отнести к 4-му типу инструментальных средств для построения экспертных систем с элементами 5-го типа.

В пакете ESWin реализовано представление знаний в виде правил-продукций, фреймов и лингвистических переменных. Для решения задач в ней реализован нечеткий обратный логический вывод.

Программное обеспечение ESWin разработано совместно кафедрой Вычислительной техники НГТУ и фирмой "ИНСИКОМ" (Интеллектуальные Системы и Комплексы) (http://insycom.chat.ru) и реализовано в среде Delphi 5.0.

4.8.2. База знаний

База знаний содержит набор фреймов и правил-продукций.

Пример базы знаний (фрагмент экспертной системы для проектирования экспертных систем):

```
TITLE = для выбора метода представления знаний
FRAME = Цель
   Метод представления знаний: ()
ENDF
FRAME = Тип
   Решаемые задачи: (диагностика; проектирование)
ENDF
FRAME = Область
        Применение [Какова область применения?]: (медицина; вычис-
   лительная техника)
ENDF
FRAME = Количество
   Число правил в базе знаний (численный): ()
   Число объектов в базе знаний (численный): ()
ENDF
FRAME = Действие
   Сообщение: ()
ENDF
RULE 1
   > (Количество.Число правил в базе знаний; 50)
   < (Количество.Число правил в базе знаний; 100)
   < (Количество.Число объектов в базе знаний; 30)
DO
   = (Тип.Решаемые задачи; диагностика) 100
ENDR
RULE 2
   > (Количество.Число правил в базе знаний; 100)
   > (Количество.Число объектов в базе знаний; 30)
DO
   = (Тип.Решаемые задачи; проектирование) 100
ENDR
RULE 3
   = (Область.Применение; медицина)
   = (Тип.Решаемые задачи; диагностика)
DO
        = (Метод представления знаний; Правила-продукции с представ-
   лением нечетких знаний) 90
```

```
ENDR
RULE 4
   = (Область.Применение; вычислительная техника)
   = (Тип.Решаемые задачи; проектирование)
DO
   = (Метод представления знаний; Фреймы) 100
        = (Метод представления знаний; Правила-продукции с представлением нечетких знаний) 70
        = (Метод представления знаний; Семантические сети) 70
   MS(Действие.Сообщение; Доказано правило 4)
ENDR
```

К порядку следования фреймов и правил-продукций, нумерации правил-продукций жестких требований не предъявляется. Единственное ограничение – неизменность номера правила-продукции на протяжении всего сеанса работы с базой знаний. Начало нумерации и порядок нумерации правил-продукций может быть произвольным, но из соображений целесообразности лучше начинать нумерацию с единицы и нумеровать правила по порядку.

База знаний состоит из двух частей: постоянной и переменной. Переменная часть базы знаний называется базой данных и состоит из фактов, полученных в результате логического вывода. Факты в базе данных не являются постоянными. Их количество и значение зависит от процесса и результатов логического вывода.

До начала работы с экспертной оболочкой база знаний находится в текстовом файле. В файле с расширением *.klb (KnowLedge Base) хранятся фреймы и правила-продукции (база знаний). При начале работы с программной оболочкой наличие данного файла обязательно. Этот файл создается пользователем с помощью специального редактора или вручную. В файле с расширением *.dtb (DaTa Base) хранятся факты, полученные в процессе логического вывода (база данных). При начале работы с программной оболочкой наличие данного файла необязательно. Файл с базой данных создается программной оболочкой в процессе логического вывода. Первые части имен этих двух файлов совпадают. В файле с расширением *.lvd (Linguastic Variable Descriptor) описаны лингвистические переменные, используемые в базе знаний.

При работе с программной оболочкой (после загрузки в оперативную память баз знаний и данных) фреймы и правила-продукции, находившиеся в файле с расширением *.klb, остаются неизменными. Факты, находившиеся в файле с расширением *.dtb, могут изменяться в процессе логического вывода (появляться, удаляться или менять свое значение в результате срабатывания правил-продукций или диалога с пользователем).

База знаний может содержать специальную конструкцию SOURCE фреймоподобного типа:

SOURCE = ‹имя конструкции›

PARENT: ‹имя фрейма с описанием внешней базы данных›
‹имя слота 1› [‹арифметическое выражение›]: (‹имя поля 1 в БД)
‹имя слота 2› [‹арифметическое выражение ›]: (‹имя поля 2 в БД)
…
‹имя слота n› [‹арифметическое выражение ›]: (‹имя поля n в БД)
ENDS

Конструкция SOURCE используется для связи базы знаний с какой-либо стандартной базой данных. На ее основе автоматически формируется SQL-запрос. В нем задается отображение структуры одноименного фрейма на поля базы знаний. Имя внешней базы данных определяется во фрейме, имеющем имя, совпадающее с именем базы данных. Слот PARENT конструкции SOURCE ссылается на фрейм, имеющий имя, совпадающее с именем базы данных. Другие слоты данного фрейма определяют имя таблицы в базе данных, имена полей таблицы, SQL-запрос. Если слот, предназначенный для задания SQL-запроса, не имеет значения, в поле вопроса такого слота можно использовать вычислимое выражение, которое и будет использоваться в качестве значения слота.

Использование конструкции SOURCE и фрейма с описанием внешней базы данных позволяет в процессе логического вывода получать факты из внешней базы данных с помощью SQL-запроса.

Конструкция SOURCE должна удовлетворять следующим требованиям:

1) значение слота Parent должно соответствовать ALIAS в BDE для доступа к базе данных,

2) количество слотов должно соответствовать количеству слотов в одноименном с SOURCE фрейме и их имена тоже,

3) значения слотов соответствуют внутренним именам полей в базе данных (таблице), в качестве таблицы может выступать запрос в ACCESS,

4) если надо использовать вычислимое поле, значение слота должно быть пусто, а в качестве вопроса к слоту (в квадртных скобках) пишется выражение для значения поля.

Кроме структуры SOURCE в БЗ должен быть фрейм с именем, соответствующим ALIAS, в котором должен быть обязательно слот с именем Table и значением-именем таблицы в базе данных (или запроса в ACCESS). Могут быть и другие слоты, описывающие базу данных и носящие информативный характер или участвующие в диалоге (на усмотрение автора БЗ),

Во время логического вывода при обращении к слоту, содержащемуся во фрейме, связанном с внешней базой данных (одноименном с конструкцией SOURCE), происходит открытие формы для автоматизированного формирования SQL-запроса и чтения всех слотов фрейма из базы данных.

4.8.3. Фреймы

Фреймы используются в базе знаний для описания объектов, событий, ситуаций, прочих понятия и взаимосвязей между ними. Фрейм – это структура данных, состоящая из слотов (полей). Формат внешнего представления фреймов:

FRAME (‹тип фрейма›) = ‹имя фрейма›
PARENT: ‹имя фрейма-родителя›
‹имя слота 1› (‹тип слота›) [‹вопрос слота›?] {‹комментарий слота›}:
(‹значение 1›; ‹значение 2›; …; ‹значение k›)
‹имя слота 2› (‹тип слота›) [‹вопрос слота›?] {‹комментарий слота›}:
(‹значение 1›; ‹значение 2›; …; ‹значение l›)
…
‹имя слота n › (‹тип слота›) [‹вопрос слота›?] {‹комментарий слота›}:
(‹значение 1›; ‹значение 2›; …; ‹значение m›)
ENDF

Фрейм может принадлежать к одному из трех типов фреймов: фрейм-класс (тип описывается зарезервированным словом «класс»), фрейм-шаблон (тип описывается зарезервированным словом «шаблон»), фрейм-экземпляр (тип описывается зарезервированным словом «экземпляр»). В базе знаний содержатся фреймы-классы и фреймы-шаблоны. При создании базы знаний тип фрейма-класса можно не описывать, этот тип фрейма понимается по умолчанию. Явно следует описывать только тип фрейма-шаблона.

Среди фреймов-классов выделяется специальный фрейм-класс «Цель», задающий перечень целей логического вывода (то есть обозначений задач, решаемых экспертной системой).

База данных содержит только фреймы-экземпляры. Тип фрейма-экземпляра в базе данных понимается по умолчанию.

Имя фрейма, фрейма-родителя, слота – последовательность символов (кириллические и/или латинские буквы, цифры, пробелы, знаки подчеркивания).

Тип слота – символьный, численный или лингвистическая переменная. Обязательным является описание численного типа слота (описывается зарезервированным словом «численный») и лингвистической переменной (описывается зарезервированным словом «лп»). Слот без описания типа по умолчанию понимается как символьный. Описание типа слота заключается в круглые скобки ().

Вопрос слота – любая последовательность символов, заключенная в квадратные скобки []. Вопрос слота не является обязательным. При отсутствии вопроса будет использована формулировка: «Выберите значение» или «Введите значение».

Комментарий слота – имя текстового (*.txt) или графического файла (*.bmp), заключенного в фигурные скобки {}. Комментарий слота не является обязательным.

Значение слота – любая последовательность символов. Значения слота разделяются точками с запятыми. Список значений слота необязателен. Слот фрейма-экземпляра имеет единственное значение, слот фрейма-класса и фрейма-шаблона имеет неограниченное число значений.

4.8.4. Правила-продукции

Правила-продукции описывают отношения между объектами, событиями, ситуациями и прочими понятиями. На основе отношений, задаваемых в правилах, выполняется логический вывод (решение выбранной задачи). В условиях и заключениях правил присутствуют ссылки на фреймы и их слоты. Формат внешнего представления правил:

RULE ‹номер правила›
  ‹условие 1›
  ‹условие 2›
  …
  ‹условие m›
DO
  ‹заключение 1›
  ‹заключение 2›
  …
  ‹заключение n›
ENDR

Номер правила

Номер правила – целое число. Начало и порядок нумерации правил произвольный, предпочтительнее правила нумеровать по порядку и начинать с единицы.

Формат записи условий и заключений одинаков и имеет следующий вид:
‹отношение› (‹имя слота›; ‹значение слота›) ‹коэффициент достоверности›

Отношения в условиях и заключениях могут быть:

EQ | = равно;
GT | > больше;
LT | < меньше;
DL удаление слота во фрейме-экземпляре;
EX запуск внешней программы;
FR вывод фрейма-экземпляра;
GO запуск правила;
MS выдача сообщения.

В заключениях правил используются только отношения EQ, DL, EX, FR, GO, и MS. Для строковых значений слотов могут быть использованы только отношения EQ, DL, EX, FR, GO, MS. Для лингвистических переменных допустимы все отношения, так как с ними связаны как строковые, так и численные значения.

Имя слота – локальное или глобальное. Локальное имя слота соответствует имени слота в некотором фрейме. Глобальное имя слота содержит имя фрейма и имя слота, разделенных точкой.

Значение слота – строка или число (определяется типом слота). Если в качестве значения слота используется имя фрейма-шаблона, то в процессе логического вывода выполняется одновременное определение значений для всех слотов данного фрейма.

Коэффициент достоверности – число от 0 до 100. Коэффициент достоверности в заключении используется при формировании значения слота фрейма-экземпляра при срабатывании правила. Коэффициент достоверности в условии в этой версии не используется. По умолчанию коэффициент достоверности принимает значение 100.

4.8.5. Лингвистические переменные

При формировании базы знаний для описания нечетких понятий используются лингвистические переменные в качестве слотов. Лингвистическая переменная позволяет при логическом выводе задавать как символьное, так и численное значение слота.

Лингвистическая переменная имеет одно или несколько символьных значений. Каждому символьному значению поставлена в соответствие функция принадлежности, которая определяет отношение между численным значением лингвистической переменной и коэффициентом достоверности для данного численного значения (соответствующего символьному значению). Для каждого символьного значения лингвистической переменной существует собственная функция принадлежности. Функция принадлежности определяется на отрезке метрической шкалы, одном и том же для всех символьных значений лингвистической переменной.

Описание лингвистических переменных хранится в текстовом файле (*.lvd – Linguistic Variable Description). Первая часть имени файла должна соответствовать именам файлов, содержащих базу знаний и базу данных (*.klb и *.dtb). Формат внешнего представления лингвистической переменной:

‹число лингвистических переменных›
‹имя лингвистической переменной 1›
‹нижнее значение границы метрической шкалы›
‹верхнее значение границы метрической шкалы›
‹шаг метрической шкалы›
‹число символьных значений лингвистической переменной 1›
‹символьное значение 1›
  ‹значение функции принадлежности 1›
  …
  ‹значение функции принадлежности m›
…
‹символьное значение n›
  ‹значение функции принадлежности 1›
  …

‹значение функции принадлежности m›

…

4.8.6. Интерпретация правил-продукций

Интерпретация правил начинается с выбора цели логического вывода или задачи. В качестве цели логического вывода используется один из целевых слотов, содержащихся во фрейме-классе со специальным именем «Цель».

Далее осуществляется поиск правила, в заключении которого присутствует выбранный целевой слот.

После нахождения правила начинается его интерпретация (перебор и проверка условий). При проверке условия ищется соответствующий слот. Первоначальный поиск выполняется в базе данных. Если слот имеет значение, то оно используется при проверке условия. Если значения нет, то значение слота запрашивается у пользователя, с использованием меню выбора символьных значений, или окна для ввода численного значения, или того и другого в случае слота лингвистического типа. Слот в условии может указываться своим локальным именем или глобальным (с указанием имени фрейма). При локальном имени слота поиск начинается с фрейма, использованного последним при логическом выводе. Такой фрейм считается текущим. Имя текущего фрейма хранится в качестве значения слота специального фрейма, описывающего контекст диалога. Этот фрейм всегда доступен для проверки условия в правилах.

При вводе пользователем значения слота лингвистического типа, формируется численное значение с коэффициентом достоверности равным 100, если пользователь ввел число. Если пользователь выбрал символьное значение, формируется также численное значение, равное значению на шкале лингвистической переменной с максимальным значением коэффициента достоверности. Если значение слота в правиле было символьным, а пользователем было введено численное значение, то коэффициент достоверности формируется как значение функции принадлежности лингвистической переменной (введенное пользователем число используется в качестве аргумента функции принадлежности).

Коэффициент достоверности набора условий вычисляется как коэффициент достоверности конъюнкции (минимальное значение из значений коэффициентов достоверности условий).

Коэффициент достоверности слота фрейма-экземпляра, формируемого на основе заключения, вычисляется как произведение коэффициента достоверности набора условий и коэффициента достоверности заключения. Если такой слот во фрейме-экземпляре уже есть, то его коэффициент достоверности меняется на новое значение, вычисляемое по формуле:

$$кд_{\text{результирующий}} = кд_{\text{исходного слота}} + кд_{\text{набора условий}} * (1 - кд_{\text{исходного слота}})$$

При проверке условия в правиле или при выработке решения (заключения правила) в случае, если коэффициент достоверности меньше определенной величины (20%), условие или заключение считаются не выполненными.

Общий вид основного окна программной оболочки ESWin представлен на рис. 11.

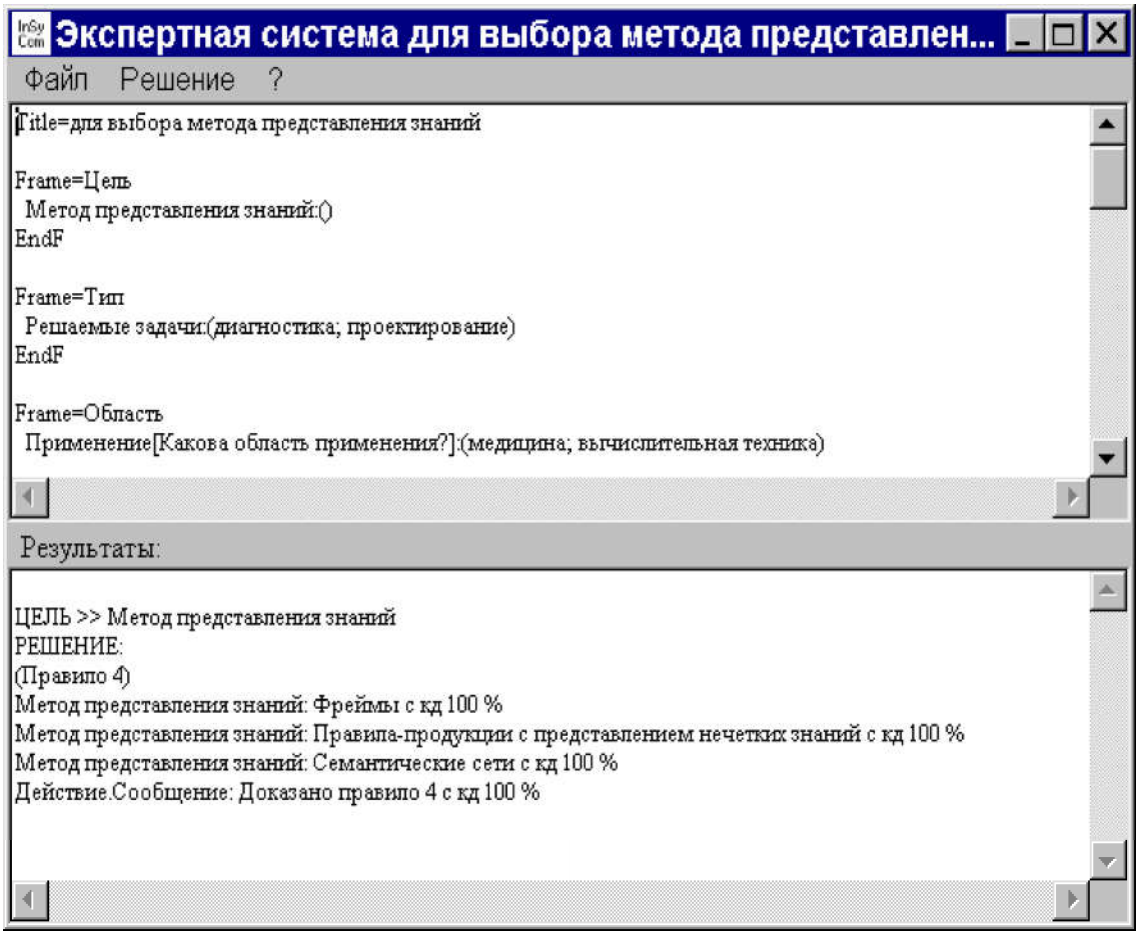

Рис. 11. Основное окно с результатами логического вывода

Ниже приведены вид главного меню основной формы редактора EdKB, форма для редактирования фрейма, форма для редактирования лингвистической переменной (рис. 12, 13, 14).

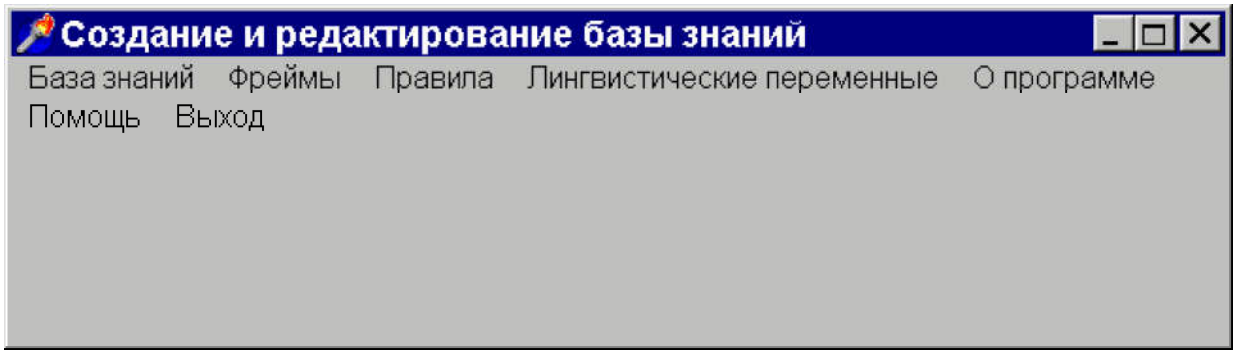

Рис. 12. Основное меню редактора баз знаний EdKB

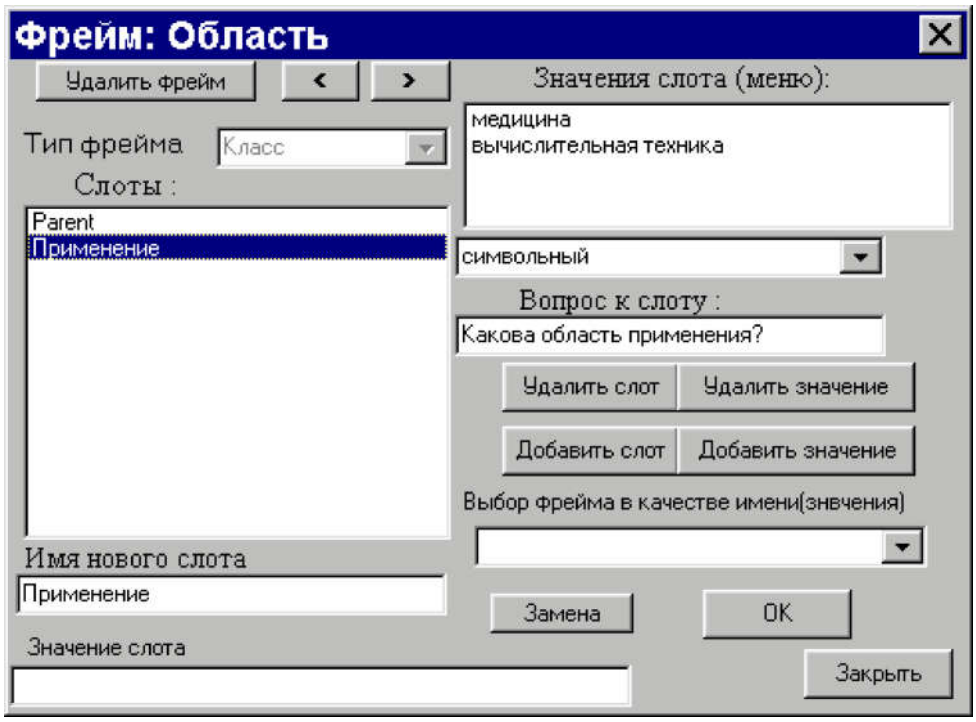

Рис. 13. Форма для редактирования фрейма

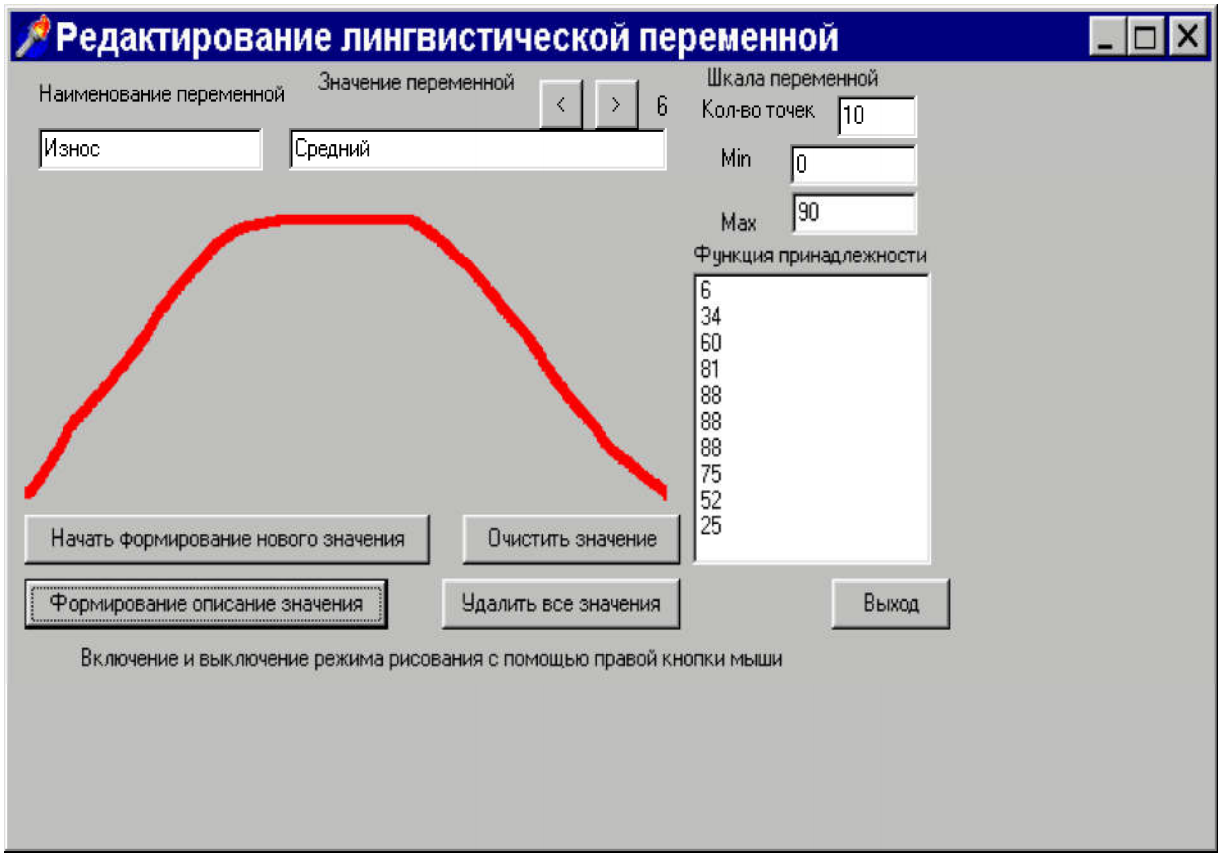

Рис. 14. Форма для редактирования лингвистической переменной