Лекция №1

Искусственный интеллект: основные понятия и история возникновения

1. Данные и знания.

Понятие искусственного интеллекта, задачи, решаемые с помощью методов искусственного интеллекта, и, как следствие, необходимость создания интеллектуальных систем (систем искусственного интеллекта) возникли практически сразу после создания А. Тьюрингом, Фон Нейманом и др. основополагающих принципов построения автоматических дискретных вычислений (ЭВМ, компьютеров).

Появление ЭВМ, работа которых происходит под управлением созданных человеком программ (т.е. с максимально полным привлечением интеллектуальных способностей человека), позволило автоматизировать самые разнообразные процессы обработки данных или, по-другому, самые разнообразные вычислительные процессы.

Здесь под автоматизацией вычислительных процессов или вычислений понимается выполнение их вычислительным устройством (компьютером) без непосредственного участия человека. При этом важным является то, что

а) вычислительный процесс должен представляться в виде последовательности (сколь угодно большой, но конечной длины) элементарных или «рутинных» операций;

б) формирование последовательности элементарных операций или, по-другому, составление алгоритма решения задачи, осуществляется непосредственно человеком (пользователем ЭВМ);

в) вычислительное устройство не может само (без участия человека) ни создавать, ни менять алгоритм, если это изменение не предусмотрено самим алгоритмом.

Поэтому принято говорить, что вычислительные устройства (в дальнейшем ЭВМ или компьютеры), построенные по классической фон-

неймановской схеме (а таковыми сейчас являются подавляющее большинство ЭВМ) реализуют т.н. «жесткие» вычисления. Термин «жесткие» вычисления обозначает организацию вычислений по заранее (до начала вычислений) разработанному человеком (пользователем ЭВМ) вполне определенному алгоритму.

Если обозначить через X – исходные данные для решения задачи, через Z – результат решения, то процедуру решения задачи на ЭВМ можно рассматривать как реализацию некоторого отображения исходных данных в конечный результат в соответствии с алгоритмом F решения задачи (рис.1).

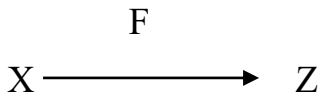


Рис.1.

По сути, в алгоритме F аккумулированы наши знания о тех или иных законах (математических, физических, химических и т.д.), привлекаемых для решения поставленной задачи (разработки алгоритма ее решения), а также новые приемы решения, специально для этой цели разработанные человеком – пользователем ЭВМ.

Таким образом, можно констатировать, что традиционная обработка информации на ЭВМ происходит по схеме «Данные» ----» «Данные» на основе (или с помощью) знаний человека – пользователя ЭВМ. (рис. 2)

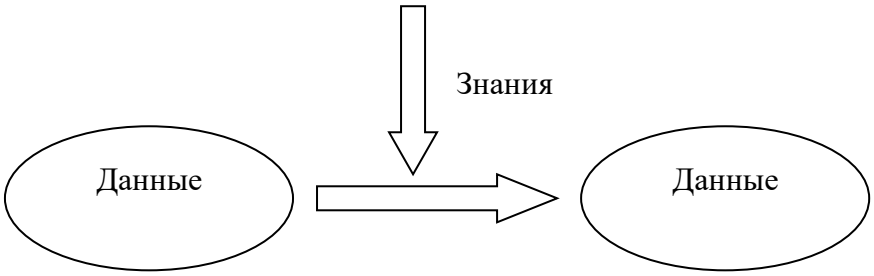


Рис. 2.

Поэтому говорят обработка информации на ЭВМ, понимаемая в общепринятом смысле, представляет собой обработку данных. в то же время, характерным признаком интеллектуальных систем является обработка знаний. При этом **Данными** называют информацию фактического характера, описывающую объекты, процессы и явления конкретной предметной области. Как правило, эта информация не требует при своем дальнейшем использовании более глубокого осмысления и анализа. К примеру, в качестве данных могут быть координаты материальной точки (x_i, y_i) измеренные в процессе ее плоского движения (точнее вращения вокруг начала координат) и соответствующей данным координатам моменты времени t_i (рис. 3).

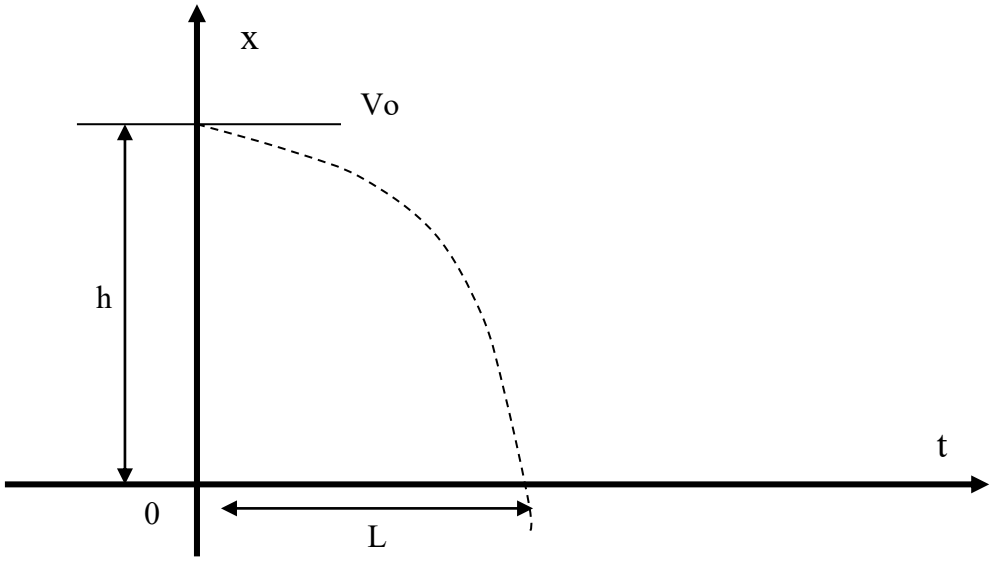


Рис.3.

Обычно данные представляются в виде таблиц, диаграмм, графиков. Так, данные о движении точки (рис. 3) можно представить в ивде следующей таблицы: Таблица 1

x	**t**
x_1	t_1
x_2	t_3
….	…
x_N	t_N

Знания являются более сложной категорией информации по сравнению с данными. Знания описывают не только отдельные факты, но и взаимосвязи между ними. Поэтому, знания иногда структурированными данными. Знания могут быть получены:

А) на основе обработки экспериментальных данных (данных эксперимента);

Б) в результате мысленной деятельности человека.

Интеллектуальные системы позволяют производить автоматическую (без участия человека) обработку данных в условиях существенной априорной неполноты знаний о том, как нужно вести эту обработку для получения требуемого результата. Очевидно. Что для этого интеллектуальные системы должны быть способны сами генерировать (получать) недостающие знания путем:

А) логического (дедуктивного) вывода;

Б) обучения

В) поиска

Г) обработки экспериментальных данных

В первых, трех случаях обработка информации происходит по схеме «знания» --- «знания», «знания» --- «данные».

С помощью существующих на настоящий момент времени методов искусственного интеллекта (нечеткой логики, нейронных сетей и генетических алгоритмов) в последнем случае – по схеме «данные» --- «знания» с помощью нейросетевых методов аппроксимации и интерпретации данных. (рис. 4)

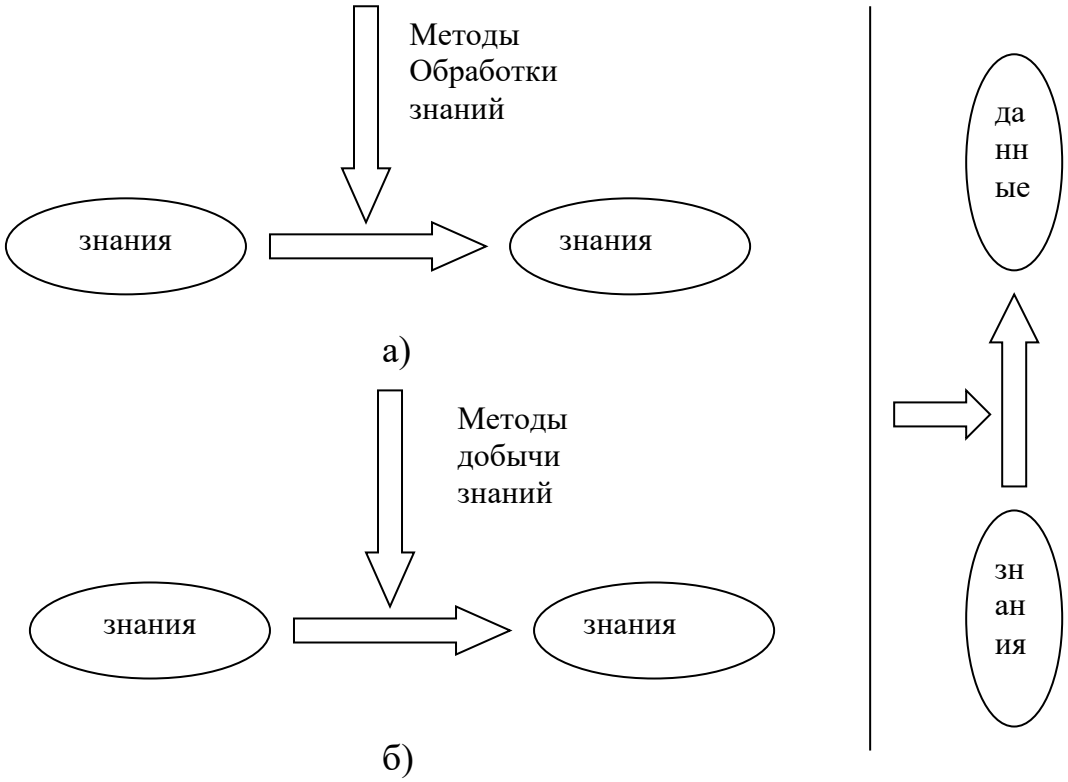


Рис. 4.

Так, если аппроксимировать приведенные в табл.1 данные о движении материальной точки, брошенной с высоты h под углом $\alpha = 0$ к горизонту, то получим следующую функциональную зависимость высоты x от времени t движения $x = h - \frac{gt^2}{2}$ (1)

Где g=9.8 – гравитационная постоянная.

Выражение (1) можно трактовать как запись (на языке математических формул) наших знаний о законе движения материальной точки под действием силы земного притяжения.

Сопоставляя табл. 1. и формулу (1), можно отметить, что запись информации о движении материальной точки с помощью знаний дает (по сравнению с совокупностью данных о ее движении) более целостную и системную картину движения.

2. Общая характеристика задач решаемых методами ИИ

Рассмотрим главные или, что более правильно, сущностные отличия задач, решаемых на ЭВМ с помощью методов искусственного интеллекта, от обычных задач, решаемых традиционными методами и способами.

Степень использования человеческого интеллекта.

Как известно, традиционно решаемые на ЭВМ задачи требуют максимально полного использования интеллекта (способностей, знаний) при выборе метода и составлении алгоритма решения задачи.

При этом на ЭВМ возлагается лишь задача правильного выполнения (или реализации) разработанного человеком алгоритма.

Напротив, решение задач с привлечением методов искусственного интеллекта (или задач, решаемых системами искусственного интеллекта) основывается не только на использовании знаний человека, но и дополнительных знаниях, полученных самой ЭВМ.

Методы и структурные решения, лежащие в основе получения (вывода) знаний, являются предметом рассмотрения сравнительно молодой науки (ей не более 50 лет), называемой искусственным интеллектом.

Полнота априорной информации

Традиционно решаемые на ЭВМ задачи (разумеется речь идет не о простых задачах, а о достаточно сложных) требуют для своего успешного решения большого объема априорной информации о закономерностях поведения исследуемого объекта или процесса. Например, если рассматривается движение летательного аппарата в атмосфере, то должны быть точно известны:

а) физические законы, определяющие силы, действующие на летательный аппарат;

б) полученные на их основе математические состояния (математическая модель объекта), определяющие реакцию летательного аппарата (изменение его высоты, скорости полета и т.п.) на эти силы и на управляющие воздействия со стороны системы управления летательным аппаратом.

На практике это весьма сложно обеспечить, учитывая существенную нестационарность условий полета (внешних и внутренних). Действительно, для современных летательных аппаратов характерен большой диапазон

изменения характеристик атмосферы, возможность возникновения нештатных (или критических) ситуаций как в атмосфере (грозы, смерчи, турбулентные потоки и т.п.), так и на борту летательного аппарата (отказы оборудования, неправильные действия летчиков). Заранее все это при разработке алгоритма управления сложно предусмотреть. Поэтому, «жесткие» алгоритмы управления современными летательными аппаратами не обеспечивают требуемой эффективности (в том числе боевой) их применения.

Интеллектуальные или «мягкие» алгоритмы управления, основанные на применении методов искусственного интеллекта («мягких» вычислений), существенно снижают требования к объему необходимой априорной информации за счет ее доопределения интеллектуальной системой непосредственно в процессе функционирования (в режиме on-line).

«Продвинутость» задач.

На сегодняшний день практически все ЭВМ имеют фон-неймановскую архитектуру, основанную на функциональных принципах построения дискретных вычислений, изложенных в работах Ч. Бэббиджа, Поста, А.Тьюринга. Это накладывает определенные ограничения на класс задач, решаемых на ЭВМ. В частности, решение задач должно допускать возможность описания его с помощью некоторого алгоритма.

В свою очередь, это означает, что на ЭВМ могут быть реализованы только алгоритмические процедуры, допускающие представление в виде совокупности базовых (элементарных) операций (в современных ЭВМ это сложение и сдвиг).

Однако с помощью алгоритмов и алгоритмических процедур в классическом понимании можно автоматизировать решение только т.н. «рутинных» задач, не связанных с получением качественно новой информации (новых знаний), а связанных с организацией вычислительной процедуры их решения при условии, что априорно имеется вся необходимая для этого информация.

Решение же более содержательных по смыслу (более интеллектуальных задач) только с помощью алгоритмических процедур невозможно. Например, невозможно с помощью алгоритмов описать процессы, реализующие

а) решение задач, начиная от словесной постановки и кончая получением результата решения;

б) перевода текстов с одного языка на другой;

в) игру в шахматы, карты и т.п.

г) диагностики болезней;

д) доказательства математических теорем и др.

Для этого нужно располагать качественно новым математическим аппаратом и вычислительными машинами, позволяющими моделировать процесс мышления человека.

Рассмотрим характерные особенности данного процесса.

1. Деятельность человека всегда целесообразна, т.е. связана с достижениями некоторой цели. Это означает, что мыслительные процессы человека направлены на достижение цели (цель заставляет человека думать).
2. Человеческий мозг хранит огромное количество фактов и правил их использования. Для достижения определенной цели надо только обратиться к нужным фактам и правилам.
3. Принятие решений всегда осуществляется на основе специального механизма упрощения, позволяющего отбрасывать ненужные (малосущественные) факты и правила. Не имеющие отношения к решаемой в данный момент задаче и, наоборот, выделять главные, наиболее значимые факты и правила, нужные для достижения цели.
4. Достигая цели, человек не только приходит к решению поставленной перед ним задачи, но и одновременно приобретает новые знания. Та часть интеллекта, которая позволяет ему делать соответствующие заключения (выводы) на основании правил, сформулированных

человеком, а также генерировать новые факты из уже существующих, называется механизмом ***логического вывода.***

Так, типовая схема решения математической задачи часто выглядит следующим образом. Выбираются неизвестные величины, подлежащие определению. На основании анализа условий (ограничений), содержащихся в исходной формулировке задачи, составляется система уравнений, связывающих указанные неизвестные. Далее, применяя какой либо из стандартных методов решения полученных уравнений, находим искомое решение задачи. Заметим, то, решив один раз конкретную задачу по описанной схеме, мы решим (и гораздо быстрее) другую подобную (и даже более сложную) задачу, отличающуюся значениями исходных данных, числом неизвестных, формой представления условий и т.д.

Поскольку система ИИ принимает решения аналогично тому, как это делает человек, то она должна включать в себя следующие ключевые элементы – цель, факты и данные. Правила, механизмы вывода и упрощения. Все эти компоненты системы ИИ показаны на рис. 5.. на этом же рисунке выделена база знаний, которая содержит всю располагаемую информацию о внешнем мире (моделях решаемых задач). Условно она может быть разделена на три части (или области), называемые базой целей, базой правил и базой данных. Первая область содержит информацию о целях, для достижения которых предназначена система ИИ. Вторая область включает в себя сведения, которые отражают закономерности, характерные для решаемого класса задач. Это правила, механизмы упрощения и вывода, которые позволяют не только выводить новые факты, не зафиксированные ранее в базе данных, но и приобретать новые знания в ходе функционирования системы или на этапе ее обучения. В третьей области содержатся в некотором упорядоченном виде качественные данные, необходимые для решения данной задачи. В силу той особой роли, которую играет база знаний в процессе формирования решений, системы ИИ называют системами основанными на знаниях.

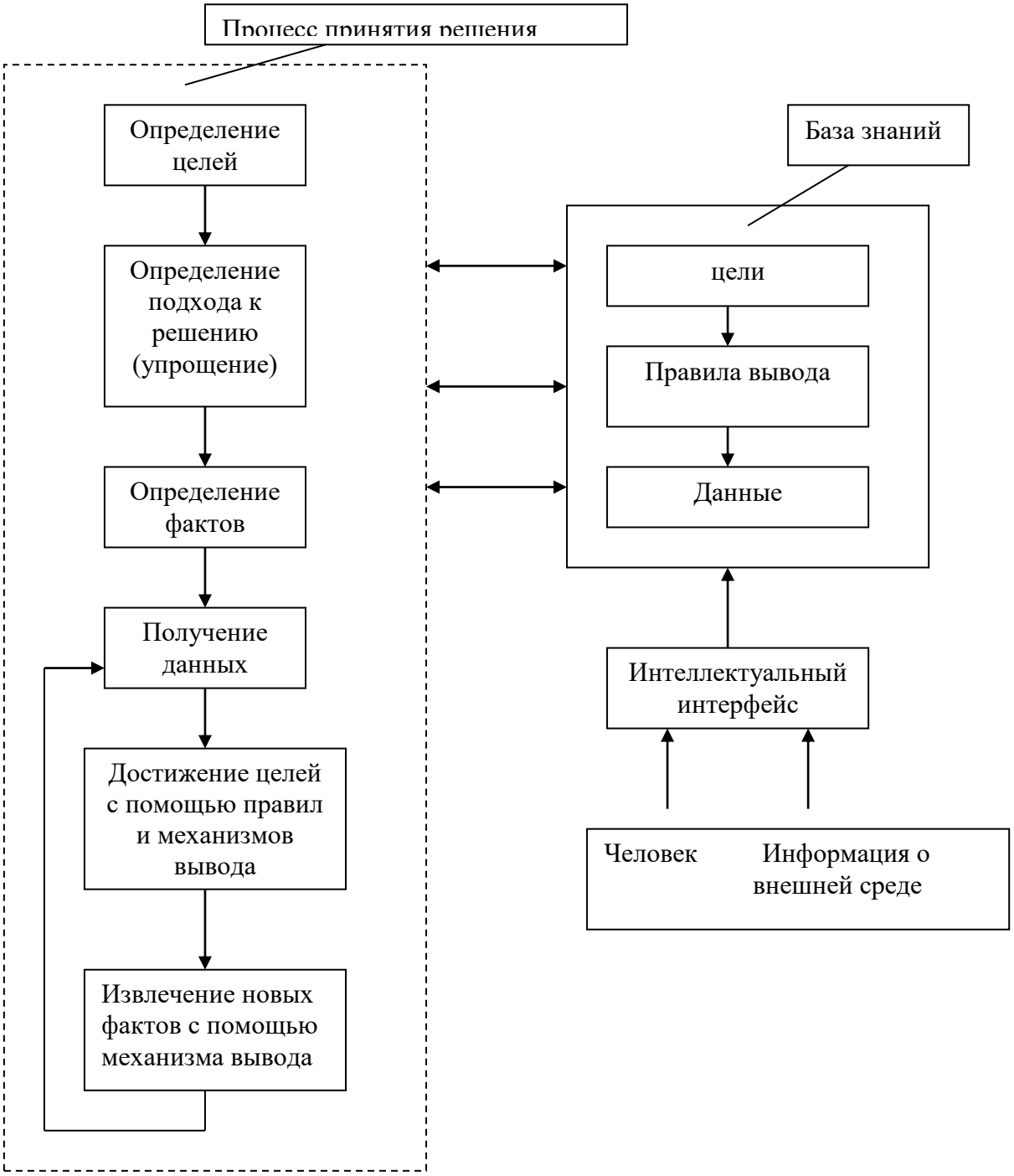


Рис. 5. Компоненты системы ИИ.

Проектирование БЗ информационной системы

Техническую поддержку многих систем экономически более выгодно оказывать удаленно, в автоматическом режиме, то есть использовать формализованные знания эксперта. Знания, обеспечивающие техническую поддержку для различных предметных областей, имеют схожую структуру. Для их представления рационально использовать онтологии, и другие технологии Semantic Web.

Пользователь всякой системы технической поддержки, как правило, не знает всех причинно-следственных связей используемой им технической системы, и имеет дело лишь с «симптомами». Задача экспертной системы технической поддержки – по неполной информацию о системе, однозначно определить её состояние и выдать пользователю рекомендации по переводу системы в целевое состояние. Причем интеллектуальность системы зависит от того насколько быстро система «поставит
диагноз».

При создании базы знаний экспертной системы технической поддержки предлагается разделить понятия, которыми оперирует экспертная система при принятии решения:

состояние
поддерживаемой системы, факторы, влияющие на него и т.п.;

и термины конкретной
предметной области, которые можно отнести к этим понятиям.

Например,
«автомобиль» находится в состоянии «не заводится», на это может влиять то, что«аккумулятор» находится в состоянии «разряжен».

Такой подход позволит повторно использовать одну и туже машину логического вывода (уже с интерфейсом) в качестве экспертной системы для поддержки различных предметных областей. Для этого необходимо «разметить» конкретную предметную область в терминах предложенного словаря.

База знаний

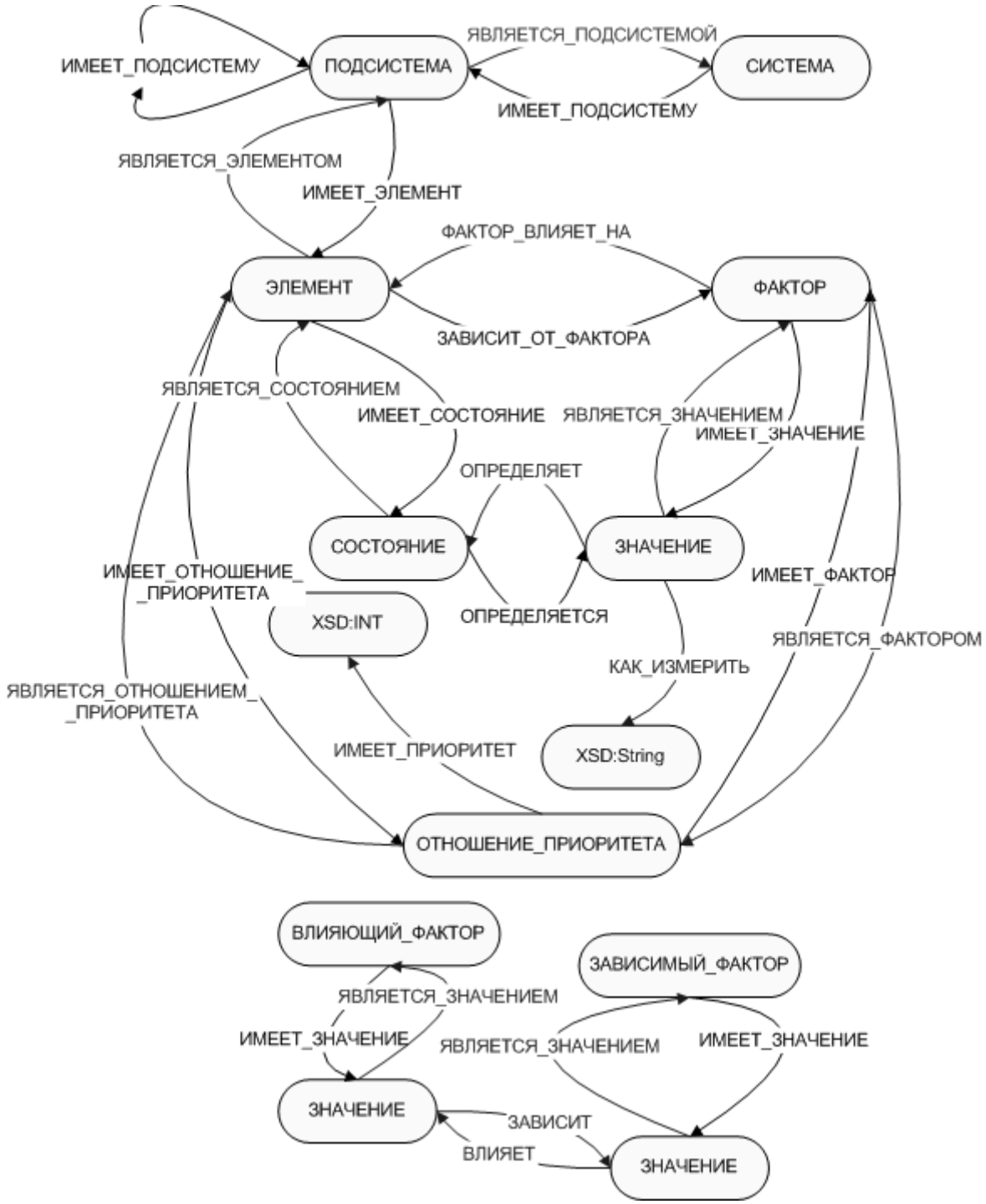
В качестве базы знаний экспертной системы предлагается использовать OWL онтологию – компонент технологии Semantic Web. Принцип разделения понятий машины вывода и терминов конкретной предметной области разделяет базу знаний на онтологию логики принятия решений, в которой определяются понятия, с которыми оперирует машина логического вывода, и

онтологию предметной области – формальное описание предметной области в терминах онтологии логики принятия решений.

Далее прописными буквами обозначаются ключевые элементы онтологии логики принятия решений.

Пользователь имеет дело с ЭЛЕМЕНТАМИ СИСТЕМЫ или её ПОДСИСТЕМ. Каждый ЭЛЕМЕНТ имеет своё СОСТОЯНИЕ. На состояние элемента влияет набор ФАКТОРОВ. Фактор имеет ряд разрешенных ЗНАЧЕНИЙ. Причем один фактор может влиять на несколько элементов, а на значение самого фактора могут влиять другие факторы. Состояние всякого элемента однозначно определяется значениями всех влияющих на него. В рамках одного элемента каждый фактор имеет ПРИОРИТЕТ. Для реализации отношения элемент-фактор-приоритет предлагается расширить триплетную модель данных OWL. Фактору определено КАК_ОПРЕДЕЛИТЬ_ИЗМЕНИТЬ.

Ниже представлена ER-диаграмма онтологии логики принятия решений, разработанной с помощью редактора онтологий Protégé 3.1.



Уровень логического вывода

Состояние системы определяется состоянием её элементов, пользователь указывает элементы, состояние которых его не устраивает, и определяет целевое состояние. Экспертная система определяет набор всех факторов, способных перевести систему в целевое состояние.

Если пользователь уже определил значения некоторых факторов, то ввиду того, что значение одного фактора может определяться значением другого, из набора

всех факторов, способных перевести систему в целевое состояние, возможно, потребуется устранить факторы, запрещенные другими. Таким образом, образуется иерархия факторов.

Интеллектуальная задача системы состоит в том чтобы, определить фактор, изменив значение которого пользователь переведет систему в целевое состояние, при том что пользователю будет задано минимум вопросов (предложено определить значения минимального количества факторов). Экспертная система выбирает фактор по следующим критериям:

в иерархии факторов выбирается верхний (влияющийна наибольшее количество других факторов);

фактор, влияющий на наибольшее количество элементов, состояние которых не устраивает пользователя;

фактор с наибольшим приоритетом.

Пользователь определяет или изменяет значение предложенного фактора, и если система не переходит в целевое состояние, то по указанным критериям выбирается следующий фактор, и пользователю предлагается определить или изменить его значение.

Изначально полагается, что поддерживаемая система не обладает «памятью», то есть порядок изменения значений факторов не имеет значения.

Предполагается, что в онтологии предметной области указаны все факторы, способные перевести систему в целевое состояние. Это означает, что рано или поздно пользователю будет предложен фактор, изменение значение которого, переведет систему в целевое состояние.

В следствие, кросс-платформености Semantic Web приложений, предлагается использовать платформу Java2. Механизм логического вывода реализуется с помощью Jena Semantic Web Framework - библиотеки Java классов для работы с RDF и OWL онтологиями.

В коде приложения следует разделять классы, обеспечивающие загрузку онтологии в память, классы, которые проводят с онтологией указанные процедуры логического вывода, и классы непосредственно связанные с интерфейсом пользователя. Это позволит развивать систему модульно.

В частности предложенная онтология лишь определяет основные понятия работы эксперта, при этом простота OWL онтологий позволяет ввести,

например понятие, обеспечивающее самообучение, лишь на уровне онтологии. Для реализации нескольких моделей вывода, например, еще нечеткого, предлагается ввести RDF-словарь самой экспертной системы с профилем логического вывода.

Интерфейс

Для реализации интерфейса предлагается использовать технологии JSP страниц и сервлетов, компонентов спецификации J2EE, которая позволяет разрабатывать копоративные веб приложения. Таким образом, экспертная система представляет собой веб приложение.

Классы интерфейса пользователя позволяют визуализировать элементы онтологии и осуществлять диалог с пользователем.

Важным моментом в успехе внедрения экспертной системы является осуществление через тот же веб интерфейс таких функций, как администрирование, модификация онтологии предметной области.

Заключение

Предложенные онтология логики принятия решений и машина логического являются одним из немногих российских приложений, использующих технологии Semantic Web. Используя данный подход, созданы онтологии для двух предметных областей: компьютер, автомобиль. Более подробно с соответствующими приложениями можно познакомиться на сайте http://semanticweb.dev.juga.ru.

Авторы благодарят М.А. Марценюка за обсуждение работы.

Ссылки

1. Tim Barners Lee, Semantic Web, SCIENTIFIC AMERICAN 2001.

2. OWL Web Ontology Language Use Cases and Requirements http://www.w3.org/TR/2004/REC-webont-req-20040210

3. OWL Web Ontology Language Guide http://www.w3.org/TR/2004/REC-owl-guide-20040210/

4. Prot.eg.e home page http://protege.stanford.edu/

5. The Prot.eg.e OWL Plugin: for Semantic Web Applications Holger Knublauch, Ray W. Fergerson, Natalya F. Noy and Mark A. Musen Stanford Medical Informatics, Stanford School of Medicine

6. Ontology-Driven Software Development in the Context of the Semantic Web: An Example Scenario with Prot.eg.e/OWL

7. Defining N-ary Relations on the Semantic Web http://www.w3.org/TR/2006/NOTE-swbp-n-aryRelations-20060412/

8. Jena Semantic Web Framework homepage http://jena.sourceforge.net/

9. http://semweb.krasu.ru

10. http://www.ksl.stanford.edu/people/dlm/webont/wineAgent/

Лекция 3. Информационные компоненты интеллектуальных приложений

Базы данных, ориентированные на искусственный интеллект

Термин «искусственный интеллект», безусловно, привлекает внимание каждого, хотя бы поверхностно интересующегося вычислительной техникой. Пока самым большим достижением в области ИИ можно считать «экспертные системы». В этой главе мы остановимся на некоторых особенностях их разработки.

В России исследования и разработки в области ЭС включены в ряд государственных и отраслевых научно-технических программ. Системы с базами знаний не только стали находить практическое применение в бизнесе и в решении серьезных информационных задач, но и продемонстрировали ощутимый эффект от их использования. Например, чрезвычайно эффективными с точки зрения применения ЭС оказались системы поддержки организационного управления и планирования распределения ресурсов. Основными областями их применения являются: медицина, электроника, вычислительная техника, геология, математика, космос, сельское хозяйство, управление, финансы, юриспруденция и т.д.

Экспертные системы и их особенности

Эксперты на основе собственного опыта или глубокого изучения проблемы осмысливают большое число факторов и устанавливают правила, их объясняющие. Выработанные ими правила упорядочиваются и хранятся в памяти ЭВМ. К ЭВМ обращаются за консультацией другие специалисты.

Но возможности экспертной системы шире. Подобно базе данных, она позволяет проводить поиск по ряду признаков одновременно и дает возможность оценить вероятность событий, которые могут использоваться в качестве условий поиска. Такие системы применяются на практике, например, при определении диагноза заболевания – ЭС MYCIN (середина 1970-х гг., Стэнфордский университет) ставила диагноз при инфекционных заболеваниях крови.

Приведем примеры других известных прикладных ЭС, ставших сегодня классическими, это:

- DENDRAL (середина 1960-х гг., Стэнфордский университет) – ЭС расшифровки данных масс-спектрографического анализа;
- PROSPECTOR (1974–1983 гг., Стэнфордский университет) – ЭС обнаружения полезных ископаемых;
- SOPHIE – ЭС обучения диагностированию электрических цепей;
- XCON – ЭС конфигурирования оборудования системы VAX;
- PALLADIO – ЭС проектирования и тестирования СБИС;
- JUDITH – ЭС оказания помощи специалистам по гражданским делам и юристам, предлагающая различные варианты подходов к разрешению дела на основе его фактических и юридических предпосылок;

- LRS – ЭС оказания помощи в подборе и анализе информации о судебных решениях и правовых актах в области кредитно-денежного законодательства, связанного с использованием векселей и чеков;
- «Ущерб» – созданная на основе российского трудового законодательства ЭС, обеспечивающая юридический анализ ситуации привлечения рабочих и служащих к материальной ответственности при нанесении предприятию материального ущерба.

Многие из систем сегодня получили развитие. Например, HEARSAY, HEARSAY-2, HEARSAY-3, AGE [18]. Первые две системы этого ряда являются развитием интеллектуальной системы распознавания слитной человеческой речи, слова которой берутся из заданного словаря. Они отличаются оригинальной структурой, основанной на использовании доски объявлений - глобальной базы данных, содержащей текущие результаты работы системы. В дальнейшем на основе этих систем были созданы инструментальные системы HEARSAY-3 и AGE (Attempt to Generalize – попытка общения) для построения ЭС.

Экспертная система XCON, созданная фирмой DEC, служит для определения или изменения конфигурации компьютерных систем типа VAX в соответствии с требованиями покупателя. В настоящее время фирма DEC разрабатывает более мощную систему XSEL, включающую базу знаний системы XCON, с целью оказания помощи покупателям при выборе вычислительных систем с нужной конфигурацией. В отличие от XCON система XSEL является интерактивной.

Среди современных коммерческих систем хочется выделить экспертную систему-оболочку G2 американской фирмы Gensym (США) [16] как непревзойденную экспертную коммерческую систему для работы с динамическими объектами. Работа в реальном масштабе времени с малыми интервалами ответа необходима при анализе критических ситуаций, возникающих в корпоративных информационных сетях, на атомных реакторах, в космических полетах и других задачах, требующих принятия решения в течение миллисекунд с момента их возникновения.

Многие предприятия используют ЭС для принятия решений в таких областях как торги на фондовой бирже, автоматическое понимание новостей, кредитный анализ, управление рисками, построение портфелей кредитов и инвестиций, оценка рейтинга банков, автоматизация аудита, предсказание изменений на финансовом рынке и др.

Приложением теории экспертных систем в экономике является использование их для организации бизнес-процесса реинжиниринга (БПР).

БПР – это анализ деятельности компании с целью выявления ее слабых мест и создание на основе его результатов максимально эффективной модели функционирования предприятия.

В практике построения экспертных систем используются следующие технологии искусственного интеллекта:

- набор инструментов программирования для представления знаний;

- стратегии обработки знаний, т.е. их преобразования, представления и управления их применением в процессе решения задач в рамках проблемной области;
- методологии проектирования, обеспечивающие создание таких экспертных систем, «прозрачных» как для пользователя, так и для инженеров по знаниям, сопровождающих систему в процессе эксплуатации.

Эти технологии оформлены в виде блоков-компонентов, из которых формируется архитектура экспертной системы (см. рис. 1.1).

Если данные существуют на жестком носителе в БД, то знания существуют только в рабочей области памяти в момент функционирования системы. В базе знаний представлены описания знаний (каркасы знаний) – модели, которые, попав в рабочую область памяти, заполняются текущими данными из БД прикладной системы или информационного приложения (ИП). Таким образом, осуществляется наполнение блока логического вывода для формирования рекомендаций и комментариев пользователю.

Современные экспертные системы основываются на знаниях экспертов, специалистов в заданной проблемной области. Программа работает с базой данных, где собраны различные факты, статистические данные, и извлекает из информации, хранимой в базе знаний, набор правил. Правила могут быть сформулированы в виде суждений на языке, близком к естественному, на основе которых можно принять решение или выработать его автоматически.

Например, можно ожидать, что база знаний, в которой хранятся сведения о торговых операциях фирмы, просмотрит записи о клиентах и попытается выявить характеристики потенциальных задолжников. А с помощью базы знаний, ориентированной на медицинские исследования, можно попытаться прогнозировать исход сердечных приступов или причины острых инфекционных заболеваний.

Во многих случаях базы знаний основываются на информации, собираемой для решения обычных задач. Вывод правил на основе этой информации проводится параллельно обработке данных. В других случаях данные будут собираться в процессе научного исследования. Подходы к проектированию БЗ и ЭС включают ряд альтернатив: от классического статистического анализа, байесовского логического анализа, в котором используются не только ответы типа «да» и «нет», «истинно» или «ложно», до теории вероятности, с генерацией правил вывода, нечеткая логика, нейросетевые и эволюционные технологии.

Таким образом, экспертные системы, или системы, основанные на знаниях, предназначены для решения плохо или слабо формализованных задач. Трудноформализуемые задачи обладают ошибочностью, неполнотой, неоднозначностью и противоречивостью, как исходных данных, так и знаний о решаемой задаче.

ЭС используются там, где существует враждебная человеку среда, отсутствует алгоритм решения задачи, или для решения задачи требуется достаточно много времени (машинного) и алгоритм трудоемкий, или имеется недостаток в числе экспертов для решения поставленной задачи.

Эксперт – это человек, являющийся профессионалом высокой квалификации в проблемной области, для которой предназначена разработка экспертной системы. Его знания лежат в основе системы.

ЭС разрабатывается в том случае, если ее разработка, во-первых, необходима, во-вторых, оправдана и задача, которую предполагается решать с ее помощью, должна быть вполне под силу эксперту-человеку.

Чаще всего экспертные системы используются как правило для решении так называемых NP-задач [15]. NP-задачи – это недетерминированные полиномиальные задачи, которые могут не сойтись при конечном количестве итераций. К ним относятся слабо формализованные или плохо структурированные задачи, а также задачи, для которых может не существовать точного решения. Подобные задачи призваны решать проблемы в условиях неполной, нечеткой или недостоверной информации, а также при достаточно большом объеме обрабатываемых данных, т.е. при угрозе комбинаторного взрыва.

Работающий совместно с экспертом специалист по инженерии знаний, выявляющий и формализующий экспертные знания, называется когнитологом. Часто когнитолога называют также инженером по знаниям.

Дополним определение.

Экспертная система – это система, основанная на знаниях о заданной проблемной области, в которой знания слабо структурированы, в которой решаются сложные NP-задачи, осуществляется взаимодействие с естественным языком на основе рассуждений и комментирования своих действий с целью обучения пользователя при самообучении системы.

Структура экспертной системы должна быть «прозрачна» для конечного пользователя. Конечными пользователями экспертной системы могут являться и необученный пользователь, и эксперт в заданной предметной области, и прикладной программист, и когнитолог.

Отличительными особенностями разработки экспертных систем в настоящее время являются использование естественного языка и объектно-ориентированное представление информации.

Основные типы задач, решаемых с помощью экспертных систем

Области применения систем, основанных на знаниях, могут быть сведены к нескольким основным типам.

Интерпретация – это процедура анализа данных с целью определения их смысла. Интерпретатор должен быть в состоянии обрабатывать информацию, представленную частично, и выдвигать гипотезы о доверии данным. При ненадежных данных интерпретация также будет ненадежной, поэтому для достижения доверия необходимо определить, какая информация была неточной или неопределенной. Так как цепочки рассуждений в ИнС могут быть достаточно длинными, интерпретатору необходимо располагать средствами объяснения того, как интерпретация обусловлена имеющимися данными.

Интерпретирующие системы обладают способностью получать определенные заключения на основе результатов наблюдения. Например,

система PROSPECTOR – одна из наиболее известных систем интерпретирующего типа – объединяет знания девяти экспертов. Используя сочетания девяти методов экспертизы, системе удалось обнаружить залежи руды стоимостью около 1 млн USD, причем наличие этих залежей не предполагал ни один из девяти экспертов. Другая интерпретирующая система HASP/SIAP позволяет определять местоположение и типы судов в Тихом океане по данным акустических систем слежения.

Планирование или процедура составления планов. Планирующие системы предназначены для достижения конкретных целей при решении задач с большим числом переменных. Планирование должно быть условным, т.е. иметь возможность коррекции при поступлении новых сведений. Процедура планирования должна предусматривать возможность совершать пробные шаги, сравнивать различные варианты планы, а также уметь сосредотачивать внимание на наиболее важных гипотезах и работать в условиях неопределенности.

В качестве примеров таких систем можно привести ЭС составления перспективного плана капиталовложения компании; ЭС планирования потребности материалов и комплектующих для основного производства предприятия на определенный период, а также упомянутую выше экспертную систему XCON для конфигурирования компьютерных систем типа VAX.

Прогнозирование – это определение хода событий в будущем на основании модели прошлого и настоящего. Прогнозирующие системы предсказывают возможные результаты или события на основе данных о текущем состоянии объекта. Прогнозирование действий осуществляется на определенный (заданный) промежуток времени. Ключевыми проблемами задачи является требование соединения в единое целое неполную имеющуюся информацию. Прогнозирование должно рассматривать различные варианты будущего и указывать их чувствительность к изменению входных данных. Решаемая задача прогнозирования должна носить условный характер, поскольку вероятность определенных событий в будущем будет зависеть от более близких, но не предсказуемых событий.

В то же время, например прогнозирование местоположения Юпитера через два года после ближайшего четверга, в случае имеющейся полной информации к задаче ИИ не относится.

Мониторинг – это непрерывное оповещение о состоянии системы или процесса. Это непрерывная интерпретация сигналов и выдача оповещений при возникновении ситуаций, требующих вмешательства.

Проектирование – это использование экспертной системы для исключения профессионала из задачи проектирования или выполнение рутинных действий по обработке информации в конкретной прикладной системе.

Ключевыми проблемами проектирования являются:

- отсутствие исчерпывающей информации, позволяющей увязать ограничения проектирования с принимаемыми решениями;
- взаимодействие подзадач (одна оказывает влияние на другую);
- умение видеть картину в интересах ухода из локально-оптимальных точек в пространстве проекта;
- оценивание последствий принимаемых решений.

В таких ЭС структуры представления информации, отражающие сведения о круге проектируемых объектов, представляются в виде морфологических таблиц и семантических сетей, имеющих в частном случае вид и/или деревьев, гиперграфов и наборов продукций.

Диагностика – это процесс поиска неисправностей в системе или определение стадии заболевания в медицине, основанный на интерпретации данных (возможно зашумленных). Такие системы используются для установления связи между нарушениями деятельности в системе или организме и их возможными причинами. К основным проблемам, возникающим при решении подобных задач, можно отнести: недоступность или малодоступность некоторых данных; сочетание не вполне совместимых частных моделей объектов или процессов; наложение симптомов других неисправностей (в некоторых ИП этой проблемой пренебрегают).

Наиболее известна диагностическая система MYCIN, которая предназначена для диагностики и наблюдения за состоянием больного при бактериальных инфекциях. Ее первая версия была разработана в Стэнфордском университете в середине 1970-х гг. В настоящее время эта система ставит диагноз на уровне врача-специалиста. Она имеет расширенную базу знаний, благодаря чему может применяться и в других областях медицины. Диагностические ЭС незаменимы при ремонте механических и электрических машин и при устранении неисправностей и ошибок в аппаратном и программном обеспечении компьютеров.

Обучение рассматривается в ЭС в двух аспектах: обучение пользователя и самообучение системы, причем как на этапе приобретения знаний, так и в процессе работы ИП.

Системы, основанные на знаниях, могут входить составной частью в компьютерные системы обучения. Система получает информацию о деятельности некоторого объекта (например, студента) и анализирует его поведение. База знаний изменяется в соответствии с поведением объекта. Одним из примеров обучающих ЭС является разработанная Д. Ленатом система EURISCO [30], которая использует простые эвристики.

Большинство ЭС включают знания, по содержанию которых их можно отнести одновременно к нескольким типам. Например, обучающая система может также обладать знаниями, позволяющими выполнять диагностику и планирование. Она определяет способности обучаемого по основным направлениям курса, а затем с учетом полученных данных составляет учебный план.

Контроль и управление – это системы, основанные на знаниях, которые могут применяться в качестве контролирующих и обеспечивающих поддержку принятия решений, анализируя данные, поступающие от нескольких источников. Системы подобного типа широко используются на атомных электростанциях, в управлении воздушным движением и медицинском контроле. Также они могут быть полезны при регулировании финансовой деятельности предприятия и в процессе выработки решений в критических ситуациях. Такие системы обычно называют системами поддержки принятия решений. В большей степени в них используются статистические методы обработки информации и интеллектуальный анализ данных.

В целом для задач, решаемых в среде ИнС характерны большие пространства представления и поиска решений, условные рассуждения, изменяющиеся во времени и зашумленные данные.

Приобретение знаний

Приобретение знаний для системы, основанной на знаниях, осуществляется у эксперта в заданной прикладной области, когнитологом (инженером по знаниям).

Известны следующие способы приобретения знаний:

- наблюдение за работой эксперта;
- опрос эксперта;
- интервьюирование и др.

Приобретение знаний может осуществляться на основе:

- четко представленной информации в виде конкретных значений данных и описывающих их понятий;
- информации, представленной в виде таблиц, графиков, гистограмм;
- слабо структурированной информации, представленной в виде эвристик эксперта в заданной проблемной области.

Функции когнитолога могут выполнять человек, программа распознавания естественного языка или экспертная система.

На этапе приобретения знаний осуществляются функции:

- описание (данных, ситуаций и т.д.);
- концептуализация – описание объектов, свойств, рассуждений;
- формализация – представление информации в заданных структурах;
- реализация – разработка информационного приложения, осуществляемая с проверкой корректности и непротиворечивости поступающей информации.

Комплексный подход к проектированию систем искусственного интеллекта

Комплексное применение рассмотренных интеллектуальных методов обработки информации позволяет существенно повысить эффективность разрабатываемых ИнС.

Возможность использования в рамках одной системы как символического, так и субсимволического подхода (обычно считающихся

взаимно исключающими), привело к появлению так называемых гибридных систем. Такие системы потенциально являются мощным инструментом решения сложных проблем, которые не под силу отдельным «чистым» подходам.

Например, генетические алгоритмы могут быть использованы для обучения нейронной сети [2], а нечеткая система реализована в виде нечеткой НС [11].

Предстоит еще очень много сделать в теории систем ИИ, прежде чем такие системы смогут в достаточно полной мере эмулировать способность к постоянному совершенствованию, которой обладает человек-эксперт. В этих целях на сегодняшний день исследователям и разработчикам следует разрешить еще целый ряд проблем.

Например, на VIII Международной научно-технической конференции «Интеллектуальные системы» в разработке систем искусственного интеллекта определены следующие основные направления дальнейшего развития в области искусственного интеллекта:

- параллелизм в логическом выводе;
- экспертные системы и вывод в условиях неопределенности;
- аргументация и абдуктивный выход;
- квазиаксиоматические системы;
- машинное обучение и индуктивный вывод;
- мягкие вычисления: нечеткая логика и приближенные вычисления;
- нейронные сети;
- генетические алгоритмы;
- системы когнитивной графики;
- системы семантического web и онтологии;
- агентно-ориентированное и распределенное решение проблем;
- понимание естественного языка.

Бизнес-реинжиниринг

Под бизнес-реинжинирингом будем понимать широкий подход, подразумевающий осуществление изменений на предприятии, предназначенных для повышения эффективности производства и скорости реакции предприятия на изменения рынка (требований потребителей, действий конкурентов и др.). В общий реинжиниринг включается и реинжиниринг предприятия как такового, и постепенное совершенствование процессов. В первую очередь будет рассматриваться реинжиниринг бизнес-процессов, не предполагая, что другие формы не учитываются.
В аналитических и обзорных работах (К. Саймон, П. Страссман и др.) как синонимы приводятся термины: Business Reengineering, Business Process Reengineering, Business Process Redesign, Business Process Improvement. Все же, в дальнейших разделах при использовании термина BPR имеется в виду подход именно М.Хаммера и Дж.Чампи.
К видам деятельности, осуществляемым в офисе, потенциально являющихся

предметом реинжиниринга и требующих информационной поддержки, необходимо в первую очередь отнести:

основную деятельность офиса: принятие решений разных уровней, контрактование, планирование и контроль выполнения заданий и др.,

стратегическое и тактическое планирование основной деятельности,

собственно документооборот и организацию делопроизводства,

обслуживание входных информационных потоков разнообразных видов, офис-менеджмент,

техническую поддержку бизнес-процессов как процессов специфического рода (длительных, вариантных, распределенных и др.).

Лекция 4

Представление неопределенности в информационных приложениях с базами знаний

Организация логического вывода в экспертных системах предполагает наличие процедур описания процесса или его моделирования, организацию диалога с пользователем, а также обработку неполной и неточной информации. Это связано с тем, что описание процесса моделирования определяется не только данными, составляющими информационное ядро проекта, но и сведениями из БЗ, а также полученными в результате диалога с пользователем.

Неопределенность (см. рис.) можно определить как степень соответствия процесса или состояния характеристикам реального мира. Например, в системах искусственного интеллекта для обработки изображений неопределенностью является само изображение. Для его определения используются не только детерминированные, но и статистические и эвристические процедуры.

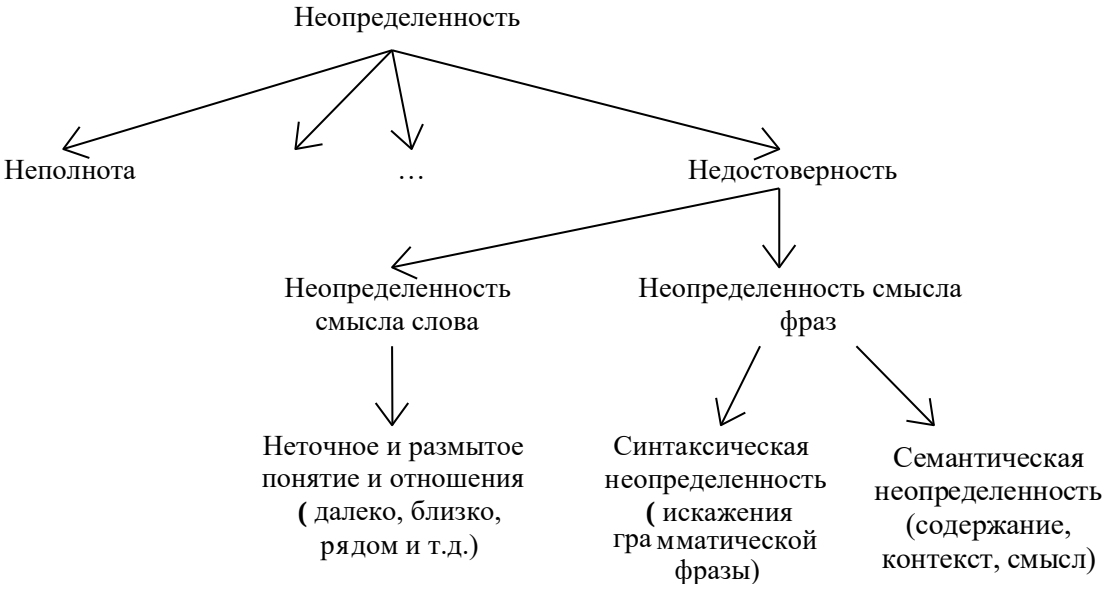


Рис. Фрагмент классификация неопределенностей

Для исключения семантической неопределенности слов используются вероятностные характеристики и аппарат нечетких множеств.

Неопределенность смысла фраз идентифицируется с использованием лингвистического анализатора текста, причем отдельно анализируется как синтаксическая неопределенность (за счет грамматических искажений), так и семантическая (смысловые искажения).

Оценка семантической составляющей неопределенности может быть получена следующими методами:

- с помощью вероятностных показателей различного рода;
- с помощью коэффициентов уверенности и мощности правил в методе редукции;
- с помощью введения интервалов значений и вероятностей попадания в заданный интервал;
- использованием формулы Байеса;
- использованием лингвистических переменных;
- использованием переменных неопределенности и др.

Мощность правила – это апостериорная характеристика, определяемая по формуле:

$$P_{ПР} = K_{УФ} * K_{УПР}\ , \qquad (1)$$

где $K_{уф}$ – коэффициент уверенности факта в условии,

$K_{упр}$ – коэффициент уверенности правила.

В зависимости от назначения системы коэффициенты уверенности либо определяются разработчиком системы, либо вносятся в систему пользователем в процессе ее эксплуатации.

Интервалы в определении неопределенности задаются как [-1, +1]. Конкретные значения показателей являются вероятностными характеристиками, однако: «-1» – всегда «ложь», а «+1» – всегда «истина».

Формула Байеса имеет следующий вид:

$$P(H_i|E) = \frac{P(E|H_i) \cdot P(H_i)}{P(E)}, \qquad (2)$$

где $P(H_i)$ – априорная вероятность гипотезы H_i;

$P(H_i \mid E)$ – вероятность гипотезы H_i при наступлении события E (апостериорная вероятность);

$P(E \mid H_i)$ – вероятность наступления события E при истинности гипотезы H_i;

$P(E)$ – вероятность наступления события E.

Апостериорная вероятность некоторого события E определяется через совокупность попарно несовместных гипотез H_i, образующих полную группу событий. Событие E существует, если существует гипотеза H_i. Вероятность вычисляется для каждой пары «гипотеза H_i – событие E».

Использование лингвистических переменных и переменных неопределенности характерно для систем, связанных с обработкой естественного языка. Переменная неопределенности описывается следующим кортежем:

$$X_{\text{НЕОПР.}} := < a, X, m >, \qquad (3)$$

где a – наименование переменной,

X – область определения значений переменных,

m – нечеткое множество, в которое попадает X значение нечеткой переменной или переменной неопределенности.

Лингвистическая переменная является следствием использования переменной неопределенности или используется вместо нее в системах обработки текстов. Лингвистическая переменная определяется кортежем:

$$X_{\text{Л}} := < \beta, \alpha, X, S, T >, \qquad (4)$$

где β – наименование лингвистической переменной;

α – интервал значений, в который попадает лингвистическая переменная, областью для которой является X;

S – синтаксическая процедура использования лингвистической переменной;

T – семантическая процедура использования лингвистической переменной.

Интеллектуальные системы в финансово-экономической сфере

Данные в информационной системе поддержки решений в экономической сфере могут включать документы, иллюстрации, карты, звуки и анимацию. Эти данные могут быть сохранены и организованы различными путями до и после их использования. Они также включают понятия, предметы и мнения (оценки). Данные могут быть предварительные, необработанные или обобщенные. Многие прикладные системы поддержки решений используют обобщенные или извлеченные данные, которые получают из трех основных источников: внутренних, внешних и персональных.

Внутренние данные хранятся в одном или более местах в корпорации .Это данные о людях, продукции, услугах и процессах. Информационная управляющая система может использовать как необработанные, так и обработанные данные (такие, как отчеты и сводки). Внутренние данные доступны через компьютерные сети организации.

Существует много источников внешних данных. Например, коммерческие базы данных, Интернет, спутниковая информация, фильмы, музыка, звуковая информация, иллюстрации, диаграммы, атласы, телевидение.

Постановления, нормативные акты и отчеты правительства являются главными источниками внешних данных.

Информация торгово – промышленных палат, локальных банков, исследовательских институтов, финансово – аналитических структур, биржевых сводок и другая, подобно наводнению обрушивается на пользователя информационной системы, вызывая у него информационные перегрузки.

Большинство внешних данных являются не относящимися к деятельности конкретной информационной системы. Поэтому осуществляется целенаправленный мониторинг данных с целью извлечения необходимой информации и минимизации возможности пропуска и недооценки важности информации.

Пользователи информационных систем или другие сотрудники корпорации или предприятия могут использовать свои собственные экспертные знания и информацию для создания персональных данных. Они включают субъективные оценки продаж, мнения о возможных действиях конкурентов, интерпретации рыночной или производственной информации, прогнозные оценки и т.д.

Типичными методами сбора данных являются изучение во времени (посредством наблюдения), обследования (с использованием анкетирования), наблюдение (например, используя видеокамеры) и информация от экспертов (например, с использованием интервью).

Общеизвестна необходимость в достоверных и точных данных для любой системы поддержки решений. Однако в реальной жизни пользователи сталкиваются со слабоструктурированными задачами в зашумленных предметных областях с высоким уровнем неопределенности.

Данные должны быть доступны системе или система должна включать подсистему извлечения данных.

Как отмечалось, внешние данные стекаются в организацию из многих источников. Некоторые данные поступают на постоянной основе посредством межмашинного обмена по каналам связи между организациями, другие – посредством Интернет, который делает возможным доступ ко многим тысячам баз данных во всем мире.

Развитие Web – систем привело к использованию Web – браузеров для доступа к жизненно – важной информации для сотрудников и покупателей.

Другие Web – системы включают исполнительные информационные системы, системы поддержки, развернутые посредством Web - браузеров и

системы управления базами данных (СУБД), которые обеспечивают данными непосредственно в формате, представляемом web – браузером с передачей посредством Интернет или интранет.

Большая тройка продавцов реляционных СУБД – компании Informix, Oracle и Sybase переработали свои основные продукты с целью приспособления клиент – серверных и Интернет интранет приложений, которые включали бы нетрадиционные или мультимедийные типы данных.

Сложность большинства корпоративных БД иногда делает стандартные операционные системы (ОС) компьютеров неадекватными эффективному интерфейсу между пользователем и БД. СУБД созданы для дополнения стандартных ОС возможностями более полной интеграции данных, сложных структур файлов, быстрого поиска и обмена, лучшей защиты данных. СУБД – это часть программного обеспечения для пополнения информации в БД и модернизации, удаления, манипулирования, хранения и поиска информации. СУБД в сочетании с языком моделирования является типичным инструментом развития системы, который используется при разработке информационной системы поддержки решений.

Отношения между многими индивидуальными записями, хранящимися в БД могут быть выражены несколькими логическими структурами.

СУБД для выполнения своих функций разрабатываются с использованием таких структур.

Тремя основными структурами являются реляционная, иерархическая и сетевая. Более новыми структурами являются объектно – ориентированные БД и мультимедийные БД.

Рассмотрим две последние структуры подробнее.

Информационные системы поддержки решений в таких сложных предметных областях как интегрированное производство, требуют возможности доступа к сложным данным, которые могут включать иллюстрации и сложные отношения.

Ни иерархическая, ни сетевая, на даже реляционная архитектура не может эффективно справляться с такими БД. Даже когда для создания и доступа в реляционной БД используется SQL, решения могут быть неэффективными.

Названные три типа БД являются алфавитно - числовыми. Но иногда для достижения лучших результатов требуется графическое представление.

Объектно – ориентированное управление данными базируется на принципах объектно – ориентированного программирования. Системы с объектно – ориентированными БД объединяют характеристики объектно – ориентированных языков, таких как Smalltalk или C++ с механизмом хранения данных и доступа к ним. Объектно – ориентированная СУБД позволяет анализировать данные на концептуальном уровне, который делает упор на естественные отношения между объектами.

Абстракция используется для установления наследственных иерархий, а описание и представление в сжатой форме позволяет проектировщику БД хранить обычные и процедурные коды внутри одних и тех же объектов.

Объектно – ориентированная СУБД определяет данные как объекты и представляет данные в сжатой форме в соответствии с их подходящей структурой и поведением.

Система использует иерархию классов и подклассов объектов. Структура (в терминах отношений) и поведение (в терминах методов и процедур) содержаться внутри объекта.

Объектно – ориентированные СУБД особенно полезны в распределенных информационных системах поддержки решений для очень сложных приложений и предметных областей.

Мультимедийные СУБД управляют данными в различных форматах (в дополнение к стандартному тексту или числовым полям). Эти форматы включают следующие образы: цифровые фотографии и формы компьютерной графики, такие как карты и .pic файлы; гипертекстовые образы; видеоклипы; звук и виртуальную реальность (многомерные образы).

Хранилище данных (DW).

Современным организациям присуще использование как старых централизованных систем, так и новых распределенных систем. Широкое разнообразие технологий обеспечено также большим числом продавцов программных продуктов. Сталкиваясь с таким технологическим и коммерческим окружением, менеджеры должны использовать новые понятия в управляющих информационных технологиях. Одним из таких понятий является складирование данных (или хранение данных).

Определение понятия «хранилище данных» начинается с физического разделения оперативного окружения, поддерживающего решения. В сердцевине многих компаний используется хранилище оперативных данных, обычно извлекаемых из неавтономных систем обработки транзакций в режиме онлайн (OLTP – online transaction processing – оперативная обработка транзакций) и базирующихся на головных компьютерах (фейн – фрейм; mainframe).

OLTP – системы, например, для финансов, инвентаризации запасов или управления, также производят оперативные данные. В оперативном окружении доступ к данным, прикладные логические задачи и логика представления данных тесно взаимодействуют вместе, обычно в нереляционных БД. Эти нереляционные хранилища данных не очень способствуют эффективному поиску данных при поддержке решений.

Целью хранилища данных является установление такого репозитария данных, который делает оперативные данные доступными в форме, которая приемлема для приложений в информационных системах поддержки решений. Как часть этого нового уровня доступности, процесс должен преобразовать детализированные по уровням оперативные данные в реляционную форму, которая делает их более подходящими для обработки при поддержки решений.

Хранение данных (или хранение информации) – это понятие, предложенное и разработанное для обеспечения решения проблемы

эффективного доступа к данным, описанным выше. Хранилище данных объединяет различные источники данных в простые источники для доступа конечного пользователя.

Существует несколько базовых структур для хранения данных. Основными являются двухрядные и трехрядные структуры. Вариант трехрядной архитектуры представлен на рис. 1.

Перед размещением в хранилище данные, поступающие из внутренних (связанных) и внешних источников извлекаются, очищаются, фильтруются и суммируются посредством специального ПО. Далее данные снова обрабатываются и помещаются в дополнительную специальную многомерную БД (третий ряд в архитектуре), организованную для легкого многомерного представления. Пользователи информационной системы поддержки решений могут запрашивать сервер и осуществлять анализ.

Рис. 1. Трехрядная архитектура хранилища данных.

В двухрядной архитектуре отсутствует многомерная БД или сервер.

Подобное хранение данных наиболее подходит для организаций, где:

- данные хранятся в различных системах;
- используется информационно – аналитический подход к менеджменту;
- имеется большая и разнообразная покупательская и клиентская база;
- одни и те же данные представлены по – разному в различных системах;
- данные хранятся в высокотехнических, трудных для расшифровки форматах.

OLAP: оперативная аналитическая обработка данных.

В течение многих лет информационные технологии концентрировались на построении систем поддержки обработки корпоративных транзакций. Такие системы должны быть визуально отказоустойчивыми и обеспечивать быстрый отклик. Эффективное решение было обеспечено OLTP, которые сосредотачивались на распределенном реляционном окружении БД.

Более поздним достижением в этой области явилось добавление архитектуры клиент – сервер. Было издано много инструментов для развития OLTP приложений.

Доступ к данным часто требуется как OLTP приложениям, так и информационным системам поддержки решений. К сожалению, попытка обслужить оба типа запросов может быть проблематична. Поэтому некоторые компании избрали путь разделения БД на OLTP тип и OLAP тип.

OLAP (Online Analytical Processing – оперативная аналитическая обработка) – это информационный процесс, который дает возможность пользователю запрашивать систему, проводить анализ и т.д. в оперативном режиме (онлайн). Результаты генерируются в течении секунд.

С другой стороны, в OLTP системе огромные объемы данных обрабатываются так скоро, как они поступают на вход.

OLAP системы выполнены для конечных пользователей, в то время как OLTP системы делаются для профессиональных пользователей ИС. В OLAP предусмотрены такие действия, как генерация запросов, запросы нерегламентированных отчетов, проведение статистического анализа и построение мультимедийных приложений.

Для обеспечения OLAP необходимо работать с хранилищем данных (или многомерным хранилищем), а также с набором инструментальных средств, обычно ч многомерными способностями. Этими средствами могут быть инструментарий запросов, электронные таблицы, средства добычи данных (Data Mining), средства визуализации данных и др.

В основе концепции OLAP лежит принцип многомерного представления данных. Э. Кодд рассмотрел недостатки реляционной модели, в первую очередь указав на невозможность объединять, просматривать и анализировать данные с точки зрения множественности измерений, то есть самым понятным для корпоративных аналитиков способом, и определил общие требования к системам OLAP, расширяющим функциональность реляционных СУБД и включающим многомерный анализ как одну из своих характеристик [83].

В большом числе публикаций аббревиатурой OLAP обозначается не только многомерный взгляд на данные, но и хранение самих данных в многомерной БД. Вообще говоря, это неверно, поскольку сам Кодд отмечает, что реляционные БД были, есть и будут наиболее подходящей технологией для хранения корпоративных данных. Необходимость существует не в новой технологии БД, а скорее, в средствах анализа, дополняющих функции существующих СУБД и достаточно гибких, чтобы предусмотреть и автоматизировать разные виды интеллектуального анализа, присущие OLAP.

По Кодду, многомерное концептуальное представление представляет собой множественную перспективу, состоящую из нескольких независимых измерений, вдоль которых могут быть проанализированы определенные совокупности данных. Одновременный анализ по нескольким измерениям определяется как многомерный анализ. Каждое измерение включает

направления консолидации данных, состоящие из серии последовательных уровней обобщения, где каждый вышестоящий уровень соответствует большей степени агрегации данных по соответствующему измерению. Так измерение Исполнитель может определяться направлением консолидации, состоящим из уровней обобщения «предприятие – подразделение – отдел - служащий». Измерение Время может даже включать два направления консолидации – «год – квартал – месяц - день» и «неделя - день», поскольку счет времени по месяцам и по неделям несовместим. В этом случае становится возможным произвольный выбор желаемого уровня детализации информации по каждому из измерений. Операция спуска соответствует движению от высших ступеней консолидации к низшим; напротив, операция подъема означает движение от низших уровней к высшим.

Кодд определил 12 правил, которым должен удовлетворять программный продукт класса OLAP. Эти правила:

1. Многомерное концептуальное представление данных.
2. Прозрачность.
3. Доступность.
4. Устойчивая производительность.
5. Клиент – серверная архитектура.
6. Равноправие измерений.
7. Динамическая обработка разреженных матриц.
8. Поддержка многопользовательского режима.
9. Неограниченная поддержка кроссмерных операций.
10. Интуитивное манипулирование данными.
11. Гибкий механизм генерации отчетов.
12. Неограниченное количество измерений и уровней агрегации.

Набор этих требований, послуживший фактическим определением OLAP, следует рассматривать как рекомендательный, а конкретные продукт оценивать по степени приближения к идеально полному соответствию всем требованиям.

Интеллектуальный анализ данных.

Интеллектуальный анализ данных (ИАД), или Data Mining, - термин, используемый для описания открытия знаний в базах данных, выделения знаний, изыскания данных, исследования данных, обработки образцов данных, очистки и сбора данных; здесь же подразумевается сопутствующее ПО. Все эти действия осуществляются автоматически и позволяют получать быстрые результаты даже непрограммистам.

Запрос производится конечным пользователем, возможно на естественном языке. Запрос преобразуется в SQL – формат. SQL запрос по сети поступает в СУБД, которая управляет БД или хранилищем данных. СУБД находит ответ на запрос и доставляет его назад. Пользователь может затем разрабатывать презентацию или отчет в соответствии со своими требованиями.

Многие важные решения в почти любой области бизнеса и социально сферы основываются на анализе больших и сложных БД. ИАД может быть очень полезным в этих случаях.

Методы интеллектуального анализа данных тесно связаны с технологиями OLAP и технологиями построения хранилищ данных. Поэтому наилучшим вариантом является комплексный подход к их внедрению.

Для того чтобы существующие хранилища данных способствовали принятию управленческих решений, информация должна быть представлена аналитику в нужной форме, то есть он должен иметь развитые инструменты доступа к данным хранилища и их обработки.

Очень часто информационно – аналитические системы, создаваемые в расчете на непосредственное использование лицами, принимающими решения, оказываются чрезвычайно просты в применении, но жестко ограничены в функциональности. Такие статические системы называются Информационными системами руководителя. Они содержат в себе предопределенные множества запросов и, будучи достаточными для повседневного обзора, неспособны ответить на все вопросы к имеющимся

данным, которые могут возникнуть при принятии решений. Результатов работы такой системы, как правило, являются многостраничные отчеты, после тщательного изучения которых у аналитика появляется новая серия вопросов. Однако каждый новый запрос, непредусмотренный при проектировании такой системы, должен быть сначала формально описан, закодирован программистом и только затем выполнен. Время ожидания в таком случае может составлять часы и дни, что не всегда приемлемо. Таким образом, внешняя простота статистических ИС поддержки решений, за которую активно борется большинство заказчиков информационно – аналитических систем, оборачивается потерей гибкости.

Динамические ИС поддержки решений, напротив, ориентированы на обработку нерегламентированных (ad hoc) запросов аналитиков к данным. Работа аналитиков с этими системами заключается в интерактивной последовательности формирования запросов и изучения их результатов.

Но динамические ИС поддержки решений могут действовать не только в области оперативной аналитической обработки (OLAP). Поддержка принятия управленческих решений на основе накопленных данных может выполняться в трех базовых сферах.

1. Сфера детализированных данных. Это область действия большинства систем, нацеленных на поиск информации. В большинстве случаев реляционные СУБД отлично справляются с возникающими здесь задачами. Общепризнанным стандартом языка манипулирования реляционными данными является SQL. Информационно – поисковые системы, обеспечивающие интерфейс конечного пользователя в задачах поиска детализированной информации, могут использоваться в качестве надстроек как над отдельными базами данных транзакционных систем, так и над общим хранилищем данных.

2. Сфера агрегированных показателей. Комплексный взгляд на собранную в хранилище данных информацию, ее обобщение и агрегация и многомерный анализ являются задачами систем OLAP. Здесь можно или

ориентироваться на специальные многомерные СУБД, или оставаться в рамках реляционных технологий. Во втором случае заранее агрегированные данные могут собираться в БД звездообразного вида, либо агрегация информации может производится в процессе сканирования детализированных таблиц реляционной БД.

3. Сфера закономерностей. Интеллектуальная обработка производится методами интеллектуального анализа данных главными задачами которых являются поиск функциональных и логических закономерностей в накопленной информации, построение моделей и правил, которые объясняют найденные аномалии и/или прогнозируют развитие некоторых процессов.

Полная структура информационно – аналитической системы построенной на основе хранилища данных, показана на рис. 2. В конкретных реализациях отдельные компоненты этой схемы часто отсутствуют.

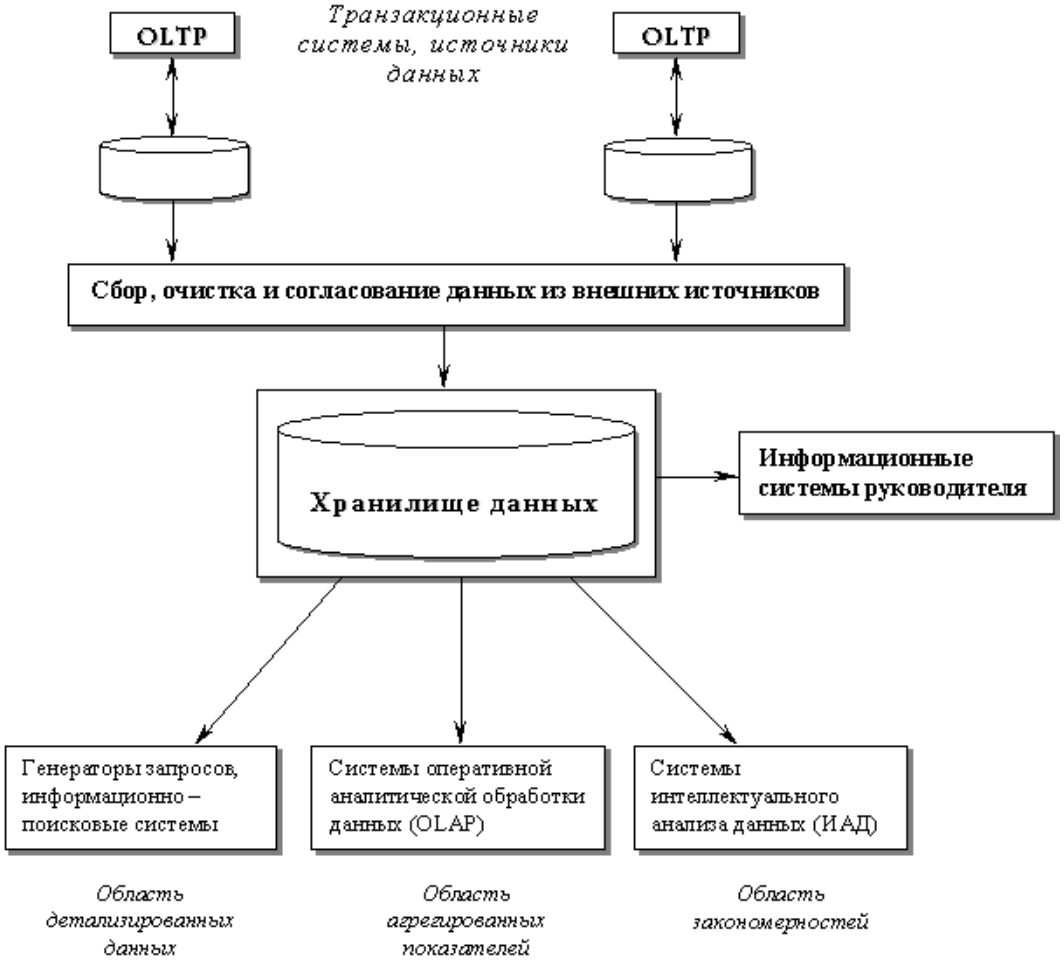


Рис. 2. Структура корпоративной информационно – аналитической системы.

Интеллектуальные базы данных.

Развитие приложений ИС требует реализации более легкого и удобного доступа к базам данных.

Технологии ИИ, особенно ЭС и искусственные нейронные сети (ИНС), могут сделать доступ и манипуляции в сложных БД проще. Одним из путей является усиление роли СУБД в обеспечении этого, совместно со способностью выведения заключений, что в результате получило общее название интеллектуальная БД.

Одним из вариантов интеграции ЭС и БД показан на рис. 3.

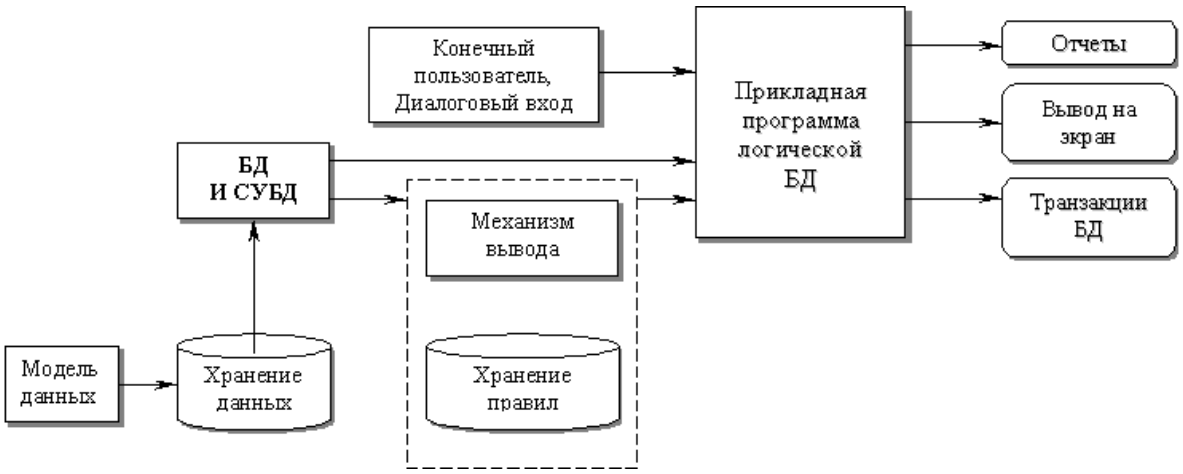


Рис. 3 Структура интеллектуальной базы данных, представляющая один из способов интеграции ЭС и БД.

Трудности в соединении ЭС с большими БД являются главной проблемой даже для больших корпораций. Многие продавцы ПО, осознавая важность такой интеграции, развивают свою программную продукцию для ее поддержки. Примером такого продукта является реляционная СУБД компании Oracle, которая объединяет функциональность ЭС с БД и представляет в форме оптимизатора запросов, которые отбирает наиболее эффективные пути следования запросов БД.

Оптимизация важна для пользователей, т.к. с такой способностью им нужно знать только несколько правил и команд для использования БД.

Одним из главных текущих направлений в разработке коммерческих программ ИИ компании IBM является обеспечение подсистемы обработки знаний для работы с БД, которая дает возможность пользователям выделить информацию из БД и передать ее в базу правил ЭС в нескольких различных структурах представления знаний.

Другой продукт – это KEE Connection (Intelli Corporation), который переводит команды KEE (KEE – Knowledge Engineering Environment) в запросы БД и автоматически поддерживает тракт данных, флуктуирующих туда и обратно между базой знаний KEE и реляционной БД, использующей SQL. Другими преимуществами такой интеграции являются способности использовать символьное представление данных и улучшения в конструкции, операциях и поддержании СУБД.

Некоторые программные инструменты для добычи данных включают интеллектуальные системы, которые поддерживают интеллектуальный поиск. Интеллектуальная добыча и анализ данных (ИАД) позволяет открыть информацию в хранилищах данных, когда запросы и отчеты не могут быть обнаружены.

Инструменты ИАД находят образцы в данных и выводят из них правила. Эти образцы и правила могут быть использованы для руководства при принятии решений и прогнозировании результатов этих решений. ИАД может ускорить анализ путем сосредоточения внимания на наиболее важных переменных.

Пять типов информации может быть применено при ИАД: ассоциации, последовательности, классификации, кластеры и прогнозирование.

Основными типами программных инструментариев, используемых в ИАД, являются:

- рассуждения на основе прецедентов;
- нейронные вычисления;
- интеллектуальные агенты;

другие средства: деревья решений, ролевая индукция, визуализация данных

Лекция 4. Управление неопределенностью. Источники неопределенных знаний
Пакеты для моделирования технологий ИИ

1. Интеллектуальные системы, основанные на нечеткой логике

Описанные выше положения можно применять для логического вывода утверждений.

Пример.

Известно: что $x \in A$, $A \subset B$,

тогда в соответствии с аксиомой вывода: $x \in B$.

Пусть A – множество народных депутатов, а B – множество пользующихся правом бесплатного проезда в общественном транспорте.

Тогда утверждение $A \subset B$ трансформируется в правило вывода: «Если лицо является народным депутатом, то оно пользуется правом бесплатного проезда в общественном транспорте».

Если множества не сравнимы непосредственно, может потребоваться дополнительное функциональное преобразование, которое позволит рассматривать одно множество как подмножество другого.

Нечеткое правило можно сформулировать как условное высказывание: «если X есть A, то Y есть B», где A и B нечеткие множества. На языке математики это записывается в виде упорядоченной пары

$$(A, B), \tag{1}$$

где A – нечеткое подмножество пространства входных значений X,

B – нечеткое подмножество пространства выходных значений Y,

либо как отношение (оператор):

$$R\colon R = A \to B. \tag{2}$$

Отношение R можно рассматривать как нечеткое подмножество прямого (декартова) произведения $X \times Y$ множества предпосылок X и множества следствий Y.

2. Нейронные сети

Еще в середине 1980-х гг. многие исследователи обратили внимание, что системы искусственного интеллекта ввиду их слабой способности к самообучению, встретившись с ситуацией, не предусмотренной разработчиком, либо формируют сообщение об ошибке, либо дают совершенно неправильные результаты. Для преодоления подобных проблем было предложено использовать искусственные нейронные сети.

Под искусственными НС подразумеваются вычислительные структуры, моделирующие биологические процессы, обычно ассоциируемые с процессами, происходящими в человеческом мозге. НС представляют собой распределенные параллельные системы, способные к адаптивному обучению путем анализа положительных и отрицательных воздействий. Элементарным преобразователем в данных сетях является искусственный нейрон, названный так по аналогии с биологическим прототипом.

Биологический и искусственный нейроны. Нервная система и мозг человека состоят из нейронов, соединенных между собой нервными волокнами, способными передавать электрические импульсы. Процессы восприятия и передачи сигналов от органов чувств (кожи, ушей, глаз) к мозгу, мышление и управление действиями − все это реализовано в живом организме в виде обмена электрическими импульсами между нейронами. Нервная клетка или нейрон является особой биологической клеткой (рис. 1).

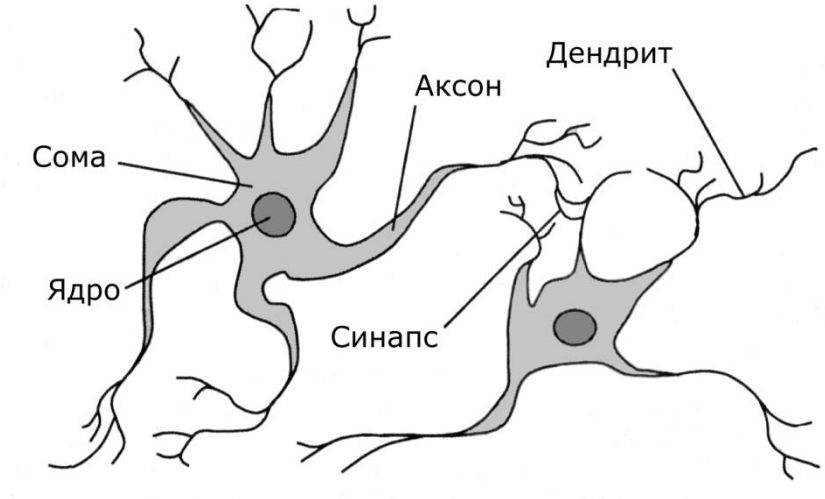


Рис. 1. Упрощенная структура биологического нейрона

Он состоит из тела или сомы, а также отростков нервных волокон двух типов: дендритов, принимающих импульсы, и единственного аксона, по которому нейрон может передавать импульс. Тело нейрона включает ядро и плазму. Нейрон получает сигналы (импульсы) от аксонов других нейронов через дендриты (приемники) и передает сигналы, сгенерированные телом клетки, вдоль своего аксона (передатчика), который в конце разветвляется на волокна. На окончаниях этих волокон находятся специальные образования – синапсы, влияющие на амплитуду импульсов.

Синапс является элементарной структурой и функциональным узлом между двумя нейронами (волокно аксона одного нейрона и дендрит другого). Под воздействием приходящего импульса в синапсе высвобождаются химические вещества, называемые нейротрансмиттерами. Нейротрансмиттеры диффундируют через синаптическую щель, возбуждая или затормаживая, в зависимости от типа синапса, способность нейрона-приемника генерировать электрические импульсы. Результативность передачи импульса синапсом может настраиваться проходящими через него сигналами так, что синапсы могут обучаться в зависимости от активности процессов, в которых они участвуют. Эта зависимость от предыстории действует как память. Важно отметить, что веса синапсов могут изменяться со временем, а значит, меняется и поведение соответствующих нейронов.

Другими словами, каждый нейрон характеризуется внутренним состоянием и порогом возбудимости, а его входы делятся на возбуждающие и тормозящие. Поступивший на возбуждающий вход сигнал, повышает степень активности нейрона, а на тормозящий – наоборот, снижает ее. Если сумма сигналов на возбуждающих и тормозящих входах превышает порог возбудимости, то нейрон формирует выходной сигнал, поступающий на входы связанных с ним других нейронов, т.е. происходит распространение возбуждения (сигнала) по нейронной сети.

Кора головного мозга человека содержит около 10^{11} нейронов и представляет собой протяженную поверхность толщиной 2–3 мм и площадью около 2200 см^2. Каждый нейрон связан с 10^3–10^4 другими нейронами. Таким образом, мозг человека в целом содержит приблизительно от 10^{14} до 10^{15} взаимосвязей.

Нейроны взаимодействуют короткими сериями импульсов продолжительностью, как правило, несколько миллисекунд. Сообщение передается посредством частотно-импульсной модуляции. Частота может изменяться от нескольких единиц до сотен герц, что в миллион раз медленнее, чем в быстродействующих электронных схемах. Тем не менее, сложные задачи распознавания человек решает всего за несколько сотен миллисекунд, что не доступно большинству современных ЭВМ. Процесс принятия решения контролируется сетью нейронов, затрачивающих на выполнение одной операции всего несколько миллисекунд. Единственным объяснением такого феномена стало предположение, что для решения таких сложных задач мозг «запускает» параллельные программы, каждая из которых содержит около 100 шагов. Явление получило название «массовый параллелизм». Основываясь на таком подходе, можно обнаружить, что количество информации, посылаемое от одного нейрона другому, должно быть очень малым (несколько бит). Из чего следует, что основная часть информации не передается непосредственно, а захватывается и распределяется в связях между нейронами

Искусственный нейрон (далее просто – нейрон) представляет собой искусственную структуру, моделирующую свойства биологического нейрона. Одной из наиболее простых и общих моделей нервной клетки, является так называемая модель МакКаллока-Питса представленная на рис. 2.

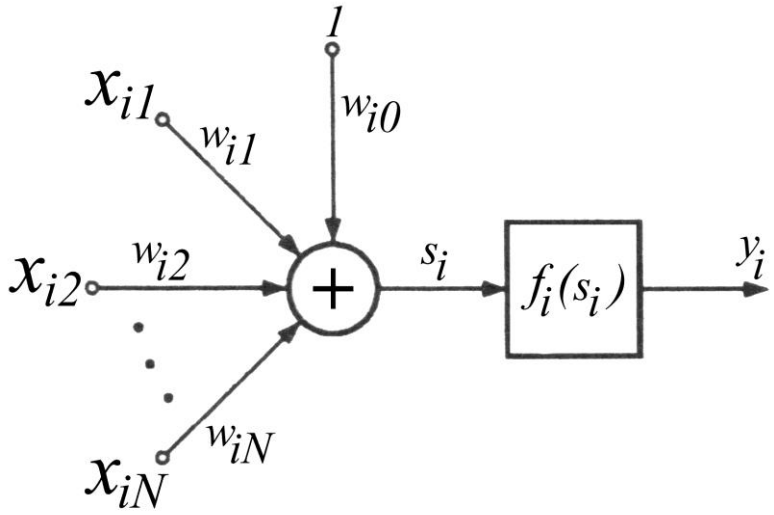


Рис. 2. Модель нейрона по МакКаллоку-Питсу

Математически модель нейрона можно записать следующим образом:

$$\left.\begin{aligned} s_i &= \sum_{j=1}^{N} w_{ij} \cdot x_{ij}(t) + w_{i0} \\ y_i &= f_i(s_i) \end{aligned}\right\}, \qquad (1)$$

где x_{ji} – совокупностью сигналов на входе нейрона,

w_{ij} – совокупностью весов входных сигналов,

s_i – суммарный сигнал или функция состояния нейрона,

f_i – функция активации нейрона,

y_i – выходной сигнал нейрона,

N – количество входов нейрона.

Одной из первых искусственных нейронных сетей, является так называемый персептрон Розенблатта [< лат. perceptio получение, собирание]. Персептроном называют однослойную нейронную сеть, состоящую из нейронов с пороговой функцией активации .

Понятие функции активации является фундаментальным в теории нейронных сетей. Функция активизации f_j определяет реакцию нейрона на совокупность внешних воздействий, выраженную величиной выходного сигнала, как функции от его текущего состояния.

В настоящее время при моделировании в нейросетевом базисе используется большое разнообразие функций активации, различающихся, главным образом, видом переходной характеристики. Наиболее часто встречающиеся функции активации приведены в табл. 1.

Таблица 1

Функции активации нейронов

Название	Формула	Примечание
Ступенчатая пороговая (персептрон)	$f_j(s_i) = \begin{cases} 0, \text{при } s_i < a \\ 1, \text{при } s_i \geq a \end{cases}$	a – величина порога
Линейная пороговая	$f_j = \begin{cases} 0, \text{при } s_i < a_1 \\ k \cdot s_i + b, \text{при } a_1 < s_i < a_2 \\ 1, \text{при } s_i \geq a_2 \end{cases}$	a_1, a_2 – величины порогов, k, b – параметры функции
Линейная	$f_j = k \cdot s_i + b$	k, b – параметры функции
Сигмоидальная униполярная (логистическая)	$f_j = \frac{1}{1+e^{-\beta \cdot s_i}}$	β – параметр, влияющий на форму функции

Окончание таблицы

Название	Формула	Примечание
Сигмоидальная биполярная (гипреболический тангенс)	$f_j = \frac{e^{\beta \cdot s_i} - e^{-\beta \cdot s_i}}{e^{\beta \cdot s_i} + e^{-\beta \cdot s_i}}$	β – параметр, влияющий на форму функции
Гауссиана	$f_j = e^{-k \cdot (s_i - a)^2}$	k, a – параметры функции
Экспоненциальная	$f_j = e^{-\beta \cdot s_i}$	β – параметр, влияющий на форму функции

3. Эволюционные вычисления

Ведущую роль в области ИИ играют алгоритмы поиска решений. Многие задачи диагностики, прогнозирования, классификации и распознавания образов могут быть интерпретированы как реализация процедуры поиска в абстрактном пространстве решений. Если целью поиска является оптимизация, то возникает вопрос о выборе оптимальных решений.

Процедуру принятия правильного решения в большинстве случаев можно свести к генерации всех возможных вариантов и выбору из числа возможных такого, который с учетом всех разнообразных факторов и противоречивых требований будет в максимальной степени способствовать достижению поставленной цели.

Формализация задачи, как правило, предполагает описание всех важных факторов, в наибольшей степени влияющих на достижение цели, порядка их взаимодействия, ограничительных условий и критерия качества принимаемого решения, на основе которого можно осуществлять выбор варианта решения.

Обычно в качестве критерия выступает некая целевая функция, аргументами которой являются количественные характеристики, описывающие факторы, влияющие на достижение цели в решаемой задаче. При этом решению, приводящему к наилучшему результату, как правило, соответствует экстремальное значение целевой функции, то есть точка ее максимума или минимума. Все возможные комбинации аргументов при этом образуют пространство поиска задачи, размерность которого определяется числом аргументов целевой функции, а каждая из указанных комбинаций образует точку в данном пространстве.

Эволюционные вычисления [англ. evolutionary computation] – термин, используемый для общего описания алгоритмов поиска, оптимизации или обучения, основанных на некоторых формализованных принципах естественного эволюционного процесса. Основное преимущество эволюционных вычислений в этой области заключается в возможности

решения многомодальных (имеющих несколько локальных экстремумов) задач с большой размерностью за счет сочетания элементов случайности и детерминированности, подобно тому, как это происходит в природной среде.

Детерминированность этих методов заключается в моделировании природных процессов отбора, размножения и наследования, происходящих по строго определенным правилам. Основным правилом при этом является закон эволюции – «выживает сильнейший», обеспечивающий улучшение находимого решения. Другим важным фактором эффективности эволюционных вычислений является моделирование размножения и наследования. Рассматриваемые варианты решений могут по определенному правилу порождать новые решения, которые будут наследовать лучшие черты своих «предков».

В качестве случайного элемента в методах эволюционных вычислений используется моделирование процесса мутации, при котором характеристики того или иного решения могут быть случайно изменены, что приводит к новому направлению в процессе эволюции решений и может ускорить процесс выработки лучшего решения.

История эволюционных вычислений началась с разработки ряда различных независимых моделей эволюционного процесса, среди которых можно выделить три основных направления исследований:

- генетические алгоритмы;
- эволюционные стратегии;
- эволюционное программирование.

Парадигма генетических алгоритмов была предложена Джоном Холландом (Holland J.H.) в начале 1960-х гг.. Основное отличие генетических алгоритмов заключается в представлении любого варианта решения в виде битовой строки фиксированной длины, манипуляции с которой производятся в отсутствие всякой связи с ее смысловой интерпретацией. То есть в данном случае применяется единое универсальное представление любой задачи.

Эволюционные стратегии, напротив, оперируют объектами, тесно связанными с решаемой задачей. Каждая из альтернатив решения представляется единым массивом численных параметров, за каждым из которых скрывается, по сути, аргумент целевой функции. Воздействие на данные массивы осуществляется, в отличие от генетических алгоритмов, с учетом их смыслового содержания и направлено на улучшение значений входящих в них параметров. Парадигму эволюционных стратегий предложили Реченберг (Rechenberg I.) и Шефель (Schwefel H.-P.) в 1973 и 1977 гг. соответственно.

В основе направления эволюционного программирования лежит идея представления альтернатив в виде универсальных конечных автоматов, способных реагировать на стимулы, поступающие из окружающей среды. Соответствующим образом разрабатывались и операторы воздействия на них. Идеи эволюционного программирования были предложены в 1966 году Фогелем, Оуэнсом и Уолшем (Fogel L.J., Owens A.J., Walsh M.J.).

4. Машинное обучение

- Это подраздел искусственного интеллекта. т Его цель заключается в том, чтобы научить компьютеры обучаться самостоятельно. С помощью алгоритма обучения компьютерная программа может определять закономерности в указанных данных, выполнять построение модели и предсказывать вещи без явно запрограммированных правил и моделей.
- Том Митчелл - определение: “Компьютерная программа обучается на основе опыта E по отношению к некоторому классу задач T и меры качества P, если качество решения задач из T, измеренное на основе P, улучшается с приобретением опыта E.”
- где:
- E- опыт
- T – класс задач
- P – мера качества.

Обучение задачей не является. Обучение это средство, благодаря которому можно выполнить решение некоторой задачи.

Например, обучение робота ходьбе.

Лекция 6. Нечеткие логики.

Система нечеткого логического вывода представляет собой композицию нечетких правил [27]:

$$\bigcup_{p=1}^{k_j}\left[w_{jp} \cdot \left(\bigcap_{i=1}^{n} x_i = a_{i,jp} \right) \right] \rightarrow y = b_j,\ j = \overline{1,m} \quad , \qquad (1)$$

где m – количество нечетких термов, степень принадлежности к которым требуется определить,

k_j – количество правил вывода, необходимых для определения степени принадлежности к нечеткому терму b_j,

n – количество условий, реализующих правило вывода,

w_{jp} – вес правила,

x_i – входное значение, принадлежащее нечеткому терму $a_{i,jp}$,

y – выходное значение.

Правила, входящие в (1) обычно имеют вид:

«Если цена велика и спрос низкий, то оборот мал», (2)

где «цена» и «спрос» – входные переменные,

«оборот» – выходное значение,

«велика», «низкий» и «мал» – функции принадлежности (нечеткие множества), определенные на множествах значений «цены», «спроса» и «оборота» соответственно.

Пример.

Система кондиционирования может быть описана правилами: «Если температура в комнате высокая, то скорость вращения вентилятора высокая» и «Если температура в комнате низкая, то скорость вращения вентилятора низкая». Результатом преобразования посылки «Температура в комнате 30° С» для кондиционера может служить указание «Включить вентилятор». Все значения температур, при

которых необходимо его включение, образуют подмножество во множестве условий, приводящих к включению вентилятора. Преобразование производится функцией управления, роль которой в данном случае может выполнять термостат.

Нечеткие правила вывода образуют базу правил. Следует особо отметить, что в нечеткой экспертной системе, в отличие от традиционной, работают все правила одновременно, однако степень их влияния на выход может быть различной. Таким образом, в основе нечетких экспертных систем лежит принцип суперпозиции множества правил при оценке их влияния на конечный результат.

Процесс обработки нечетких правил вывода в экспертной системе состоит из четырех этапов:

1) вычисление степени истинности левых частей правил (между «если» и «то») – определение степени принадлежности входных значений нечетким подмножествам, указанным в левой части правил вывода;

2) модификация нечетких подмножеств в правой части правил вывода (после «то») в соответствии со значениями истинности, полученными на первом этапе;

3) объединение (суперпозиция) модифицированных подмножеств;

4) скаляризация результата суперпозиции, т.е. переход от нечетких подмножеств к скалярным значениям.

Для определения степени истинности левой части каждого правила, нечеткая экспертная система вычисляет значения функций принадлежности нечетких подмножеств от соответствующих значений входных переменных. Например, для правила (5.4) определяется степень вхождения конкретного значения переменной «цена» в нечеткое подмножество «велика», то есть истинность предиката «цена велика». К вычисленным значениям истинности могут применяться логические операции. Наиболее часто используются следующие определения операций нечеткой логики:

$$\text{truth}\,(\sim X) = 1 - \text{truth}(X),$$

$$\text{truth}(X \,\&\, Y) = \min\{\text{truth}(X), \text{truth}(Y)\}, \qquad (3)$$

$$\text{truth}(X \vee Y) = \max\{\text{truth}(X), \text{truth}(Y)\},$$

где X и Y – высказывания,

truth (Z) – степень истинности высказывания Z.

Полученное значение истинности предназначено для модификации нечеткого множества, указанного в правой части правила. Для выполнения такой модификации применяют метод «минимума» (Correlation-min Encoding), либо метод «произведения» (Correlation-product Encoding).

Первый метод ограничивает функцию принадлежности множества, указанного в правой части правила, значением истинности левой части (рис. 1).

Во втором методе значение истинности левой части используется как коэффициент, на который умножаются значения функции принадлежности (рис. 2).

Результатом выполнения правила является нечеткое множество. Говоря более строгим языком, происходит ассоциирование переменной и функции принадлежности, указанной в правой части.

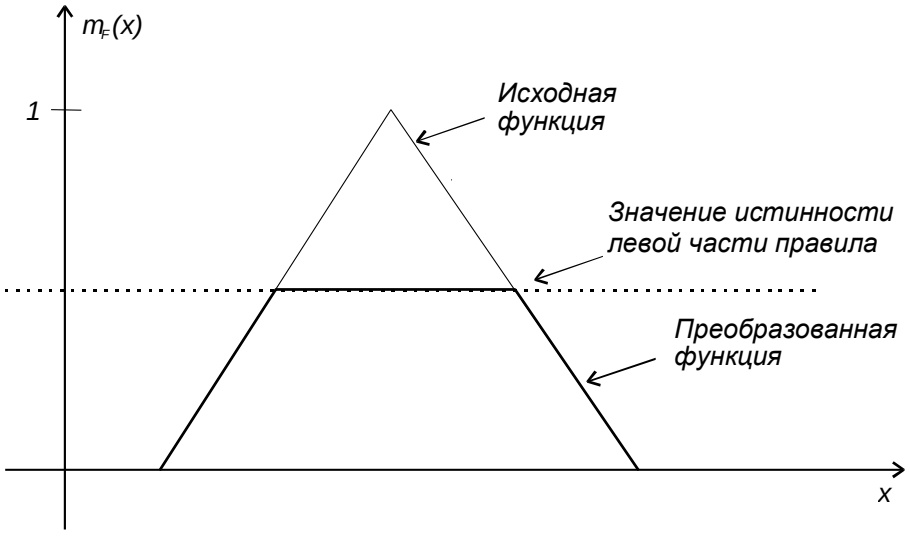


Рис. 1. Метод «минимума»

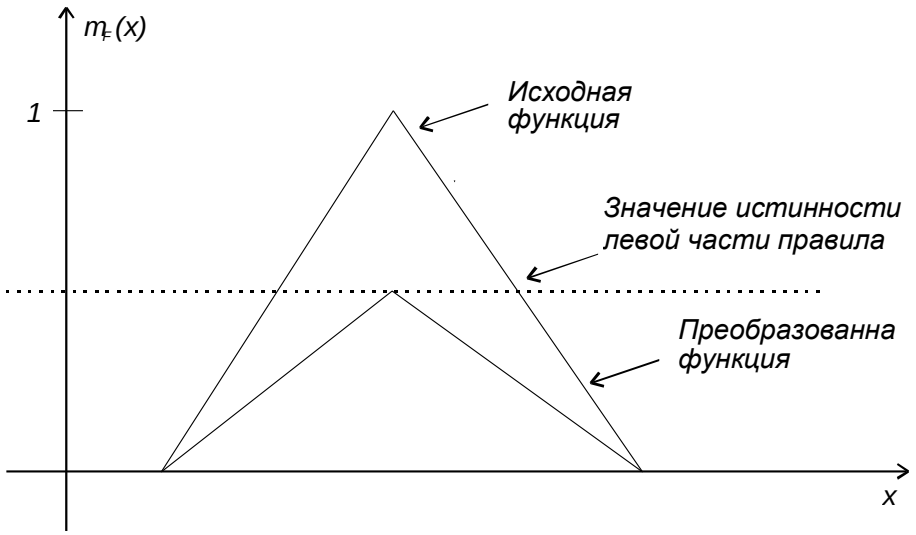


Рис. 2. Метод «произведения»

Выходы всех правил вычисляются нечеткой экспертной системой отдельно, однако в правой части нескольких из них может быть указана одна и та же нечеткая переменная. Как было сказано выше, при определении обобщенного результата необходимо учитывать все правила. Для этого система осуществляет суперпозицию нечетких множеств, связанных с каждой из таких переменных. Эта операция называется нечетким объединением правил вывода.

Например, правая часть правил «Если цена мала, то спрос велик» и «Если цена велика, то спрос мал» содержит одну и ту же переменную – «спрос». Два нечетких подмножества, получаемые при выполнении этих правил, должны быть объединены экспертной системой.

Суперпозиция функций принадлежности нечетких множеств обычно определяется методом максимума комбинации (Max Combination):

$$\forall x, m_{sumF}(x) = \max\{m_{iF}(x)\}, i = \overline{1,n} \quad , \qquad (4)$$

где $m_{sum\,F}(x)$ – результирующая функция принадлежности,

$m_{i\,F}(x)$ – нечеткие множества.

Реализация этого метода для двух функций представлена на рис. 3.

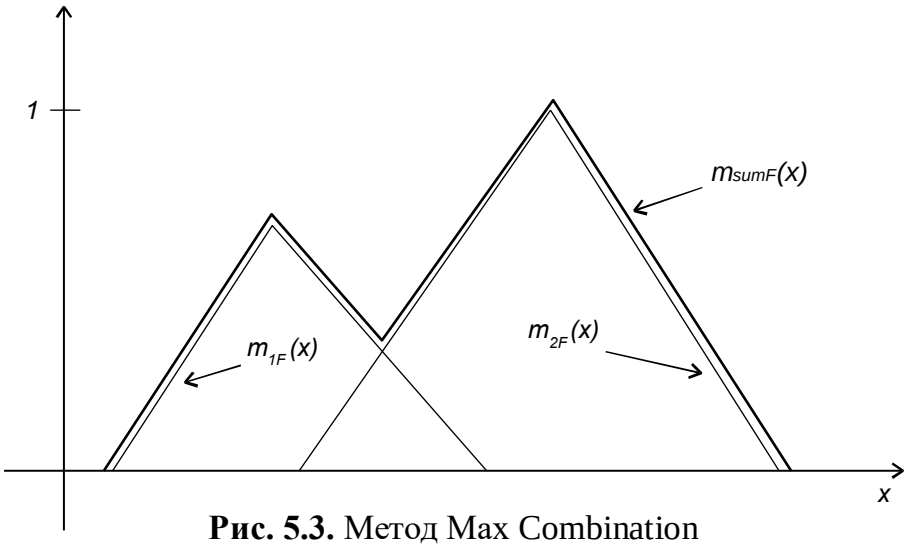


Рис. 5.3. Метод Max Combination

Рис. 3.

Другой метод суперпозиции, называемый Sum Combination, заключается в суммировании значений всех функций принадлежности (рис. 5.4):

$$\forall x, m_{sumF}(x) = \sum_{i=1}^{n} m_{iF}(x) \quad . \tag{5}$$

Самым простым (но и наименее часто используемым) является подход, при котором суперпозиция не производится, а выбирается одно из правил вывода, результат которого считается интегральным. Такой прямой метод используется для задания функции принадлежности четко измеримых понятий, например, таких как скорость, время, давление и т.д.

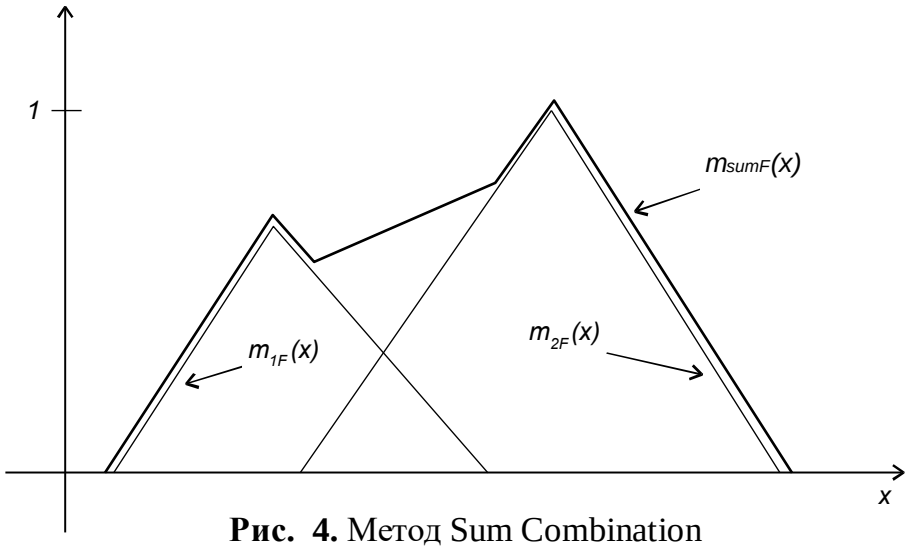


Рис. 4. Метод Sum Combination

Конечным этапом обработки базы правил вывода является переход от нечетких значений к скалярным. Процесс приведения нечеткого множества к некоторому единственному значению называется скаляризацией или дефаззификацией [< англ. defuzzification].

Обычно это значение определяется методом нахождения центра тяжести функции принадлежности (Centroid Defuzzification Method) или методом максимального значения функции принадлежности (Modal Defuzzification Method), проиллюстрированными на рис. 5 и 6 соответственно.

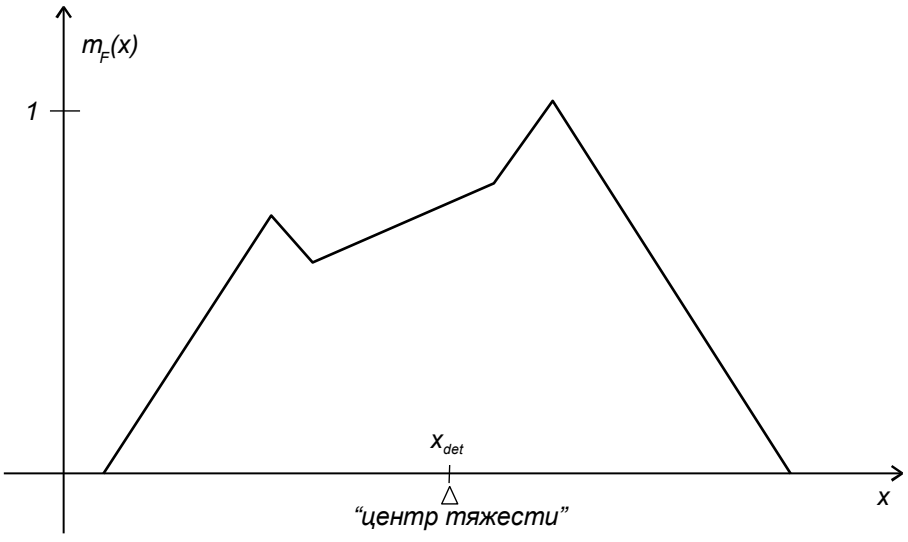


Рис. 5. Скаляризация методом нахождения центра тяжести

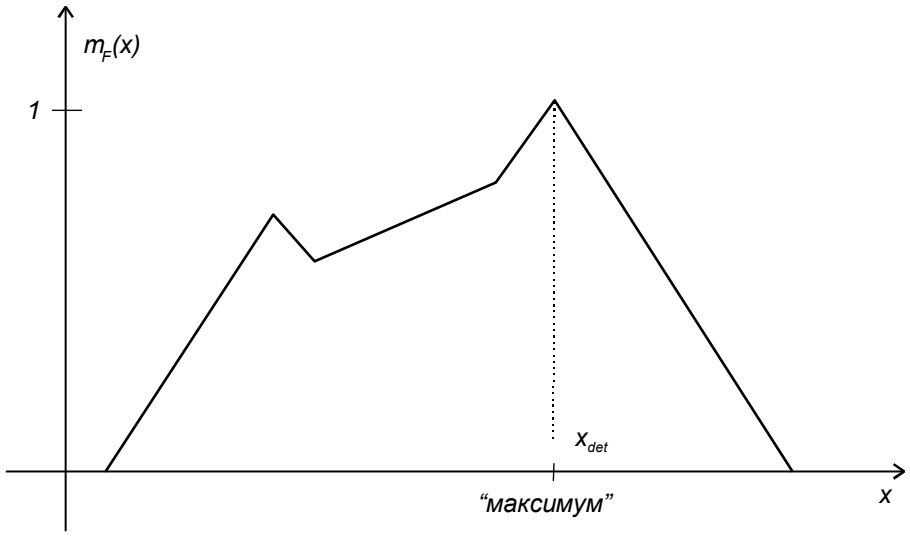


Рис. 6. Скаляризация методом нахождения максимума

Конкретный выбор методов суперпозиции и скаляризации осуществляется в зависимости от желаемого поведения нечеткой экспертной системы. В качестве инструментального средства для разработки нечеткой экспертной системы можно воспользоваться пакетом CubiCalc 2.0 компании Hyper Logic Corporation.

Лекция №6

Генетические алгоритмы

1. История появления генетических алгоритмов.

История эволюционных вычислений началась с разработки ряда различных независимых моделей. Основными из них были генетические алгоритмы и классификационные системы Голланда (Holland), опубликованные в начале 60-х годов и получившие всеобщее признание после выхода в свет книги, ставшей классикой в этой области, - "Адаптация в естественных и искусственных системах" ("Adaptation in Natural and Artifical Systems", 1975). В 70-х годах в рамках теории случайного поиска Растригиным Л.А. был предложен ряд алгоритмов, использующих идей бионического поведения особей. Развитие этих идей нашло отражение в цикле работ Букатовой И.Л. по эволюционному моделированию. Развивая идеи Цетлина М.Л. о целесообразном и оптимальном поведении стохастических автоматов, Неймарк Ю.И. предложил осуществлять поиск глобального экстремума на основе коллектива независимых автоматов, моделирующих процессы развития и элиминации особей. Большой вклад в развитие эволюционного программирования внесли Фогел (Fogel) и Уолш (Walsh). Несмотря на разницу в подходах, каждая из этих "школ" взяла за основу ряд принципов, существующих в природе, и упростила их до такой степени, чтобы их можно было реализовать на компьютере.

2. Основные понятия.

Генетические Алгоритмы - адаптивные методы поиска, которые в последнее время часто используются для решения задач оптимизации (поиска оптимального решения). Они основаны на генетических процессах биологических организмов: биологические популяции развиваются в течение нескольких поколений, подчиняясь законам естественного отбора и по принципу "выживает наиболее приспособленный", открытому Чарльзом

Дарвином. Подражая этому процессу генетические алгоритмы способны "развивать" решения реальных задач, если те соответствующим образом закодированы. Например, ГА могут использоваться, чтобы проектировать структуры моста, для поиска максимального отношения прочности/веса, или определять наименее расточительное размещение для нарезки форм из ткани. Они могут также использоваться для интерактивного управления процессом, например на химическом заводе, или балансировании загрузки на многопроцессорном компьютере.

Основные принципы ГА были сформулированы Голландом (Holland, 1975), и хорошо описаны во многих работах. В отличии от эволюции, происходящей в природе, ГА только моделируют те процессы в популяциях, которые являются существенными для развития. Точный ответ на вопрос: какие биологические процессы существенны для развития, и какие нет? - все еще открыт для исследователей.

В природе особи в популяции конкурируют друг с другом за различные ресурсы, такие, например, как пища или вода. Кроме того, члены популяции одного вида часто конкурируют за привлечение брачного партнера. Те особи, которые наиболее приспособлены к окружающим условиям, будут иметь относительно больше шансов воспроизвести потомков. Слабо приспособленные особи либо совсем не произведут потомства, либо их потомство будет очень немногочисленным. Это означает, что гены от высоко адаптированных или приспособленных особей будут распространятся в увеличивающемся количестве потомков на каждом последующем поколении. Комбинация хороших характеристик от различных родителей иногда может приводить к появлению "суперприспособленного" потомка, чья приспособленность больше, чем приспособленность любого из его родителя. Таким образом, вид развивается, лучше и лучше приспосабливаясь к среде обитания.

ГА используют прямую аналогию с таким механизмом. Они работают с совокупностью **особей**, каждая из которых представляет возможное решение данной проблемы.

На первом шаге алгоритма формируется исходная популяция особей, где каждая особь популяции представляет собой решение задачи, иначе говоря, является кандидатом на решение. Формирование исходной популяции, как правило, происходит с использованием какого-нибудь случайного закона, в ряде случаев исходная популяция может быть также результатом работы другого алгоритма. Необходимо отметить, что особи популяции могут состоять как из одной, так и из нескольких хромосом. Каждая хромосома особи в свою очередь состоит из генов, причем количество генов в хромосоме определяется количеством варьируемых параметров решаемой задачи (аргументов целевой функции).

Для применения генетических операторов значения этих параметров должны быть представлены в виде двоичной последовательности, т.е. двоичной строки, состоящей из нескольких бит. Количество бит при кодировании гена (хромосомы) зависит от требуемой точности решаемой задачи. Оптимальным считается такой выбор, когда выполняется соотношение

$$n \geq \log_2(\frac{x_{\max} - x_{\min}}{\Delta})$$

где $x_{\max}$ и $x_{\min}$ - максимальное и минимальное значение аргумента x целевой функции;

Δ - погрешность решения задачи;

n – количество бит, используемых для кодирования значения аргумента.

Количество особей M в популяции определяется, как правило, эмпирическим путем, желательно из интервала $n \leq M \leq 2n$.

На втором шаге работы генетического алгоритма происходит отбор или селекция наиболее приспособленных особей, имеющих наиболее

предпочтительные значения функции пригодности по сравнению с остальными особями. После чего к отобранным особям применяются операторы скрещивания и мутации.

В классическом генетическом алгоритме отбор наиболее приспособленных особей осуществляется случайным образом с помощью различных методов генерации дискретных случайных величин, имеющих различные законы распределения. Среди данных методов следует упомянуть отбор методом “колеса рулетки, пропорциональный отбор, отбор с вытеснением, равновероятный отбор и т.д. Каждый из перечисленных методов имеет как свои достоинства, так и недостатки, например, общим недостатком указанных методов является то, что при их использовании в некотором поколении наилучшие особи популяции могут быть потеряны. Одним из способов преодоления этого недостатка является использование элитного отбора, который предусматривает сохранение “наилучшей” особи в популяции (“лучшая” особь всегда переходит в следующее поколение).

На третьем этапе реализуется операция скрещивания особей. Смысл данной процедуры сводится к нахождению новых комбинаций генетического кода хромосом путем обмена случайным образом участков генетического кода у двух особей, прошедших отбор. Это способствует “получению” дополнительного числа новых особей, среди которых могут оказаться как более приспособленные, так и менее приспособленные особи. Это явление объясняется тем, что точка скрещивания (локус) выбирается случайным образом.

На четвертом этапе к особям популяции, полученным в результате скрещивания, применяется операция мутации. С помощью данной операции можно получать принципиально новые генотипы и фенотипы особи, что приводит к еще большему разнообразию особей в рассматриваемой популяции. Суть этого оператора заключается в следующем: в популяции случайным образом выбирается особь и так же случайно выбирается позиция гена, в

которой значение изменяется на противоположное ($0 \to 1$ или $1 \to 0$). Внесение таких случайных изменений позволяет существенно расширить пространство поиска приемлемых решений задачи целочисленной оптимизации.

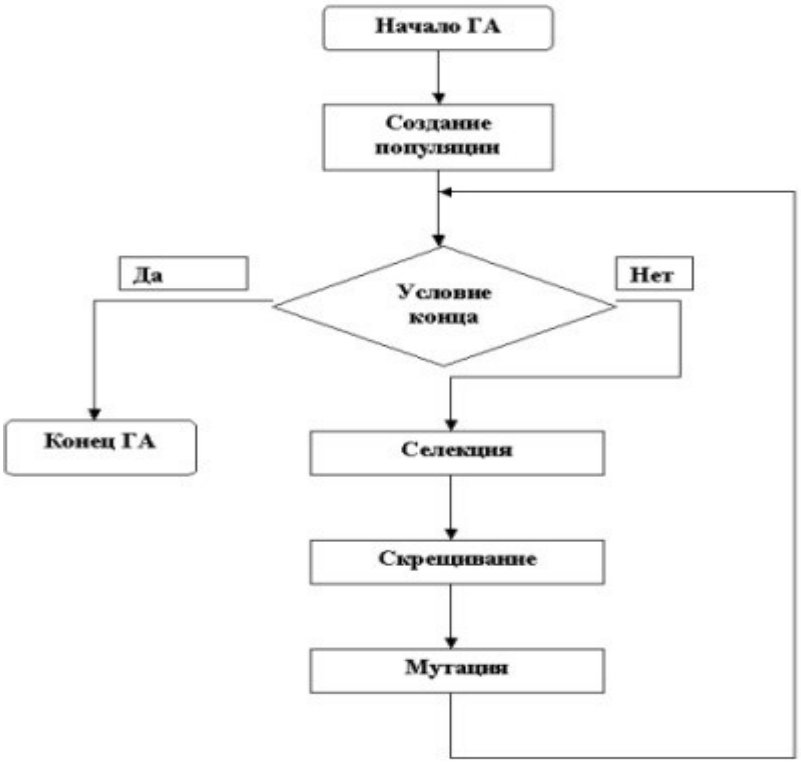
В процессе работы алгоритма все указанные операторы применяются многократно и ведут к постепенному изменению исходной популяции в направлении улучшения значения функции пригодности.

В качестве критериев останова работы генетического алгоритма принято рассматривать следующие условия

- сформировано заданное число поколений,
- популяция достигла заданного уровня качества (например, 80% особей имеют одинаковую генетическую структуру или одинаковое значение функции пригодности),
- достигнут некоторый уровень сходимости, при котором улучшение популяции не происходит.

3. Классический (традиционный) генетический алгоритм

Имеются много способов реализации идеи биологической эволюции в рамках ГА. Традиционным считается ГА, представленный на схеме.



НАЧАЛО

Создать начальную популяцию

Оценить приспособленность каждой особи

останов := FALSE

ПОКА НЕ останов ВЫПОЛНЯТЬ

НАЧАЛО /* создать популяцию нового поколения */

ПОВТОРИТЬ (размер_популяции/2) РАЗ

НАЧАЛО /* цикл воспроизводства */

Выбрать две особи с высокой приспособленностью из предыдущего поколения для скрещивания

Скрестить выбранные особи и получить двух потомков

Оценить приспособленности потомков

Поместить потомков в новое поколение

КОНЕЦ

ЕСЛИ популяция сошлась ТО останов := TRUE

КОНЕЦ

КОНЕЦ.

Работа простого ГА

Простой ГА случайным образом генерирует начальную популяцию. Работа ГА представляет собой итерационный процесс, который продолжается до тех пор, пока не выполнятся заданное число поколений или какой-либо иной критерий остановки. На каждом поколении ГА реализуется отбор пропорционально приспособленности, одноточечный кроссинговер и мутация. Сначала, пропорциональный отбор назначает каждой структуре вероятность

Ps(i) равную отношению ее приспособленности к суммарной приспособленности популяции:

$$Ps(i) = \frac{f(i)}{\sum_{i=1}^{n} f(i)}$$

Затем происходит отбор (с замещением) всех *n* особей для дальнейшей генетической обработки, согласно величине Ps(i). Простейший пропорциональный отбор - рулетка - отбирает особей с помощью *n* "запусков" рулетки. Колесо рулетки содержит по одному сектору для каждого члена популяции. Размер i-ого сектора пропорционален соответствующей величине Ps(i). При таком отборе члены популяции с более высокой приспособленностью с большей вероятность будут чаще выбираться, чем особи с низкой приспособленностью.

После отбора, *n* выбранных особей подвергаются кроссинговеру (иногда называемому рекомбинацией) с заданной вероятностью Pc.

n строк случайным образом разбиваются на *n*/2 пары. Для каждой пары с вероятность Pc может применяться кроссинговер. Соответственно с вероятностью 1-Pc кроссинговер не происходит и неизмененные особи переходят на стадию мутации. Если кроссинговер происходит, полученные потомки заменяют собой родителей и переходят к мутации.

Одноточечный кроссинговер работает следующим образом. Сначала, случайным образом выбирается одна из *l*-1 точек разрыва. (Точка разрыва - участок между соседними битами в строке.) Обе родительские структуры разрываются на два сегмента по этой точке. Затем, соответствующие сегменты различных родителей склеиваются и получаются два генотипа потомков.

После того, как закончится стадия кроссинговера, выполняются операторы мутации. В каждой строке, которая подвергается мутации, каждый бит с вероятностью Pm изменяется на противоположный. Популяция, полученная после мутации записывает поверх старой и этим цикл одного

поколения завершается. Последующие поколения обрабатываются таким же образом: отбор, кроссинговер и мутация.

Пример. Найти максимум функции $f(x) = 2x^2 + 1$ (1) для целочисленной переменной x, принимающей значения от 0 до 31.

Решение:

1. Используя двоичную систему счисления, закодируем числа от 0 до 31. Тогда хромосомы приобретут вид двоичных последовательностей, состоящих из 5 битов (0 как 00000, 31 как 11111)

2. В роли функции приспособленности будет выступать функция $f(x) = 2x^2 + 1$. Тогда приспособленность хромосомы ch_i, $i = 1,2,...,N$,будет определяться значением функции $f(x)$ для x, равного фенотипу, соответствующему генотипу ch_i. Обозначим эти фенотипы как ch_i^*. Тогда значение функции приспособленности хромосомы ch_i будет равно $f(ch_i^*)$.

3. Выберем случайным образом исходную популяцию, состоящую из 6 кодовых последовательностей (N=6). Пусть выбраны хромосомы:

$ch_1 = [10011]$, $ch_2 = [00011]$, $ch_3 = [00111]$

$ch_4 = [10101]$, $ch_5 = [01000]$, $ch_6 = [11101]$

Соответствующие им фенотипы – это числа от 0 до 31:

ch_1^*=19, ch_2^*=3, ch_3^*=7, ch_4^*=21, ch_5^*=8, ch_6^*=29

По формуле (1) рассчитаем значения функции приспособленности для каждой хромосомы в популяции, получим:

$f(ch_1)=723,\ f(ch_2)=19,\ f(ch_3)=99,\ f(ch_4)=883,\ f(ch_5)=129$

$f(ch_6)=1683$

4. Селекция хромосом осуществляется методом рулетки.

Используя формулы (4.1) и (4.2) получим, что

$V(ch_1)=20{,}45$, $V(ch_2)=0{,}54$, $V(ch_3)=2{,}80$, $V(ch_4)=24{,}97$,

$V(ch_5)=3{,}65$, $V(ch_6)=47{,}60$

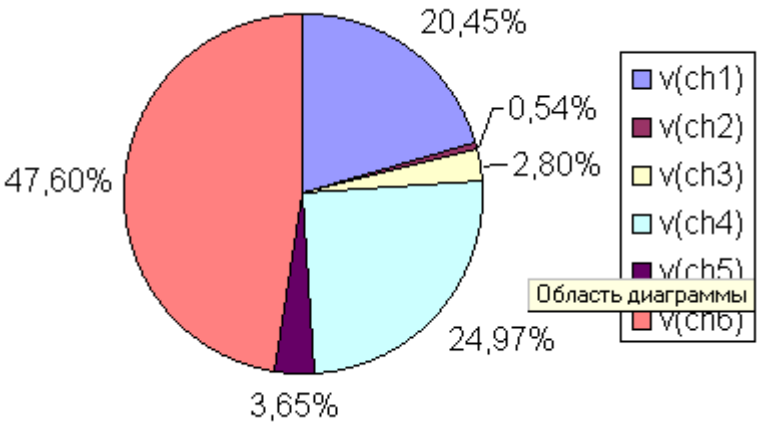


Рис 2

Розыгрыш с помощью колеса рулетки сводится к случайному выбору числа из интервала [0, 100], указывающего на соответствующий сектор на колесе, т.е. на конкретную хромосому (рис 2).

Допустим, что выбраны числа: 97, 26, 54, 13, 31, 88. Это означает выбор хромосом ch_6, ch_4, ch_6, ch_1, ch_4, ch_6.

5. Пусть скрещивание выполняется с вероятностью $p_c=1$. Допустим, что для скрещивания сформированы пары ch_1 и ch_4, ch_4 и ch_6, ch_6 и ch_6. Кроме того, случайным образом выбрана точка скрещивания, равная 3 для хромосом ch_1 и ch_4, а также точка скрещивания, равная 2 для хромосом ch_4 и ch_6. При условии, что вероятность мутации $P_m=0$, в новую популяцию включаются хромосомы

$ch_1=[10001]$, $ch_2=[10111]$, $ch_3=[10101]$, $ch_4=[11101]$,

$ch_5 = [11101]$, $ch_6 = [11101]$

Декодировав полученные последовательности, вычислим функции принадлежности данных хромосом:

$$ch_1^* = 17,\ ch_2^* = 23,\ ch_3^* = 21,\ ch_4^* = 29,\ ch_5^* = 29,\ ch_6^* = 29.$$

Продолжая данный процесс, уже на следующей итерации может быть получено хромосома [11111], с фенотипом равным 31, значение функции приспособленности которой будет наибольшим (равно 1923).

4. Настройка параметров генетического алгоритма.

В настоящее время исследователи ГА предлагают много других операторов отбора, кроссинговера и мутации. Вот лишь наиболее распространенные из них.

Турнирный отбор. Турнирный отбор реализует *n* турниров, чтобы выбрать *n* особей. Каждый турнир построен на выборке *k* элементов из популяции, и выбора лучшей особи среди них. Наиболее распространен турнирный отбор с k=2.

Элитные методы отбора гарантируют, что при отборе обязательно будут выживать лучший или лучшие члены популяции совокупности. При этом часть самых лучших особей без каких-либо изменений переходит в следующее поколение.

Двухточечный кроссовер и ***равномерный кроссовер*.**

В двухточечном кроссинговере выбираются две точки разрыва, и родительские хромосомы обмениваются участком генетического кода, который находится между двумя этими точками. В равномерном кроссинговере, каждый бит первого родителя наследуется первым потомком с заданной вероятностью; в противном случае этот бит передается второму потомку. И наоборот.

Размер популяции. Размер популяции является весьма важным элементом ГА. Если популяция слишком мала, генетического материала может не хватить для решения задачи. Размер популяции также влияет на коэффициент применения операторов скрещивания и мутации.

5. **Области применения генетических алгоритмов.**

Генетический алгоритм используется для решения многих задач оптимизации:

- составление расписаний (производственных, учебных и т.д.)
- задачи раскроя-упаковки
- аппроксимации и т.д.

Лекция 8.

Искусственные нейронные сети

Нейронная сеть представляет собой совокупность нейронов, определенным образом соединенных друг с другом и с внешней средой посредством связей, характеризуемых определенными весовыми коэффициентами. В зависимости от функций, выполняемых нейронами в сети, можно выделить три их типа:

- входные нейроны, на которые подается вектор входного воздействия или образ внешней среды. Обычно эти нейроны не осуществляют никаких вычислений, а лишь передают информации со своего входа на выход путем собственной активации;
- выходные нейроны, выходные значения которых представляют выходы нейронной сети. ;
- промежуточные нейроны составляют основу нейронных сетей.

В процессе функционирования сети осуществляется преобразование входного вектора в выходной, т.е. переработка информации. Конкретный вид преобразования, выполняемого сетью, обусловливается не только характеристиками составляющих ее нейронов, но и особенностями архитектуры сети, а именно:

- топологией межнейронных связей;
- выбором определенных подмножеств нейронов для ввода и вывода информации;
- способами обучения сети;
- наличием или отсутствием конкуренции между нейронами;
- способами управления и синхронизации процессом информационного обмена между нейронами.

С точки зрения топологии можно выделить три основных типа нейронных сетей (рис. 1): полносвязные, многослойные (слоистые) и слабосвязные (с локальными связями).

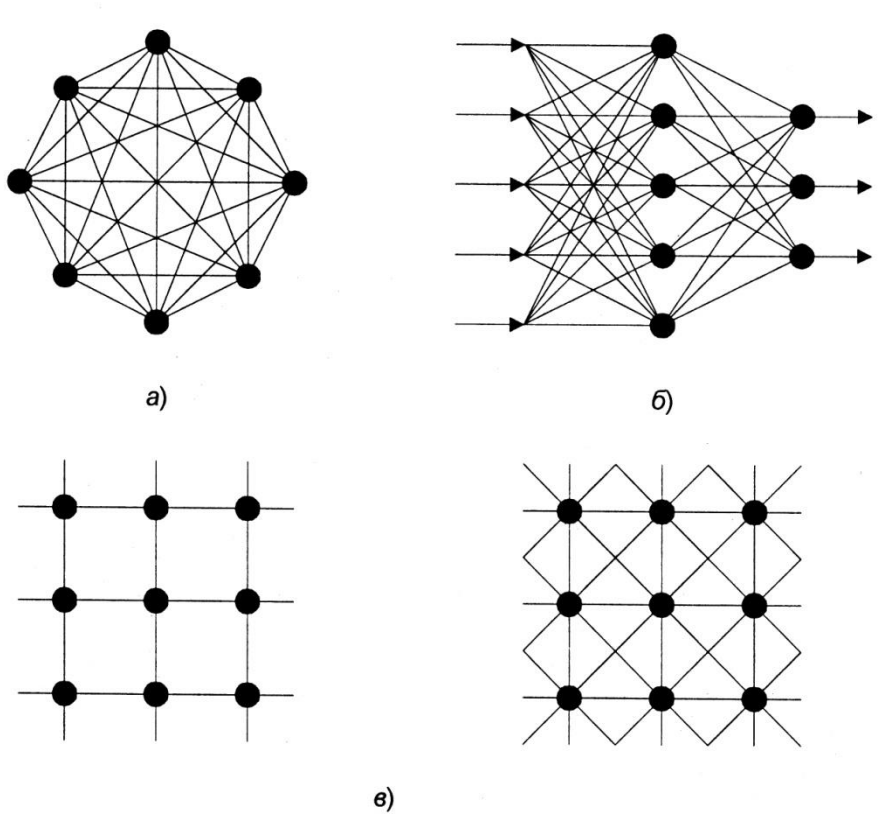


Рис. 1. Варианты архитектуры нейронных сетей:
а - полносвязная сеть, *б* - многослойная сеть с последовательными связями,
в - слабосвязные сети

В полносвязных нейронных сетях (рис. 1, *а*) каждый нейрон передает свой выходной сигнал остальным нейронам, в том числе и самому себе. Все входные сигналы подаются на входы всех нейронов. Выходными сигналами сети могут быть все или некоторые выходные сигналы нейронов после нескольких тактов функционирования сети.

В многослойных нейронных сетях (рис. 1, *б*) нейроны объединяются в слои. Каждый слой содержит совокупность нейронов с едиными входными сигналами. Число нейронов в слое может быть любым и не зависит от количества нейронов в других слоях. В общем случае сеть состоит из Q слоев, пронумерованных слева направо. Внешние входные сигналы подаются на входы нейронов входного (нулевого) слоя, а выходами сети являются выходные сигналы последнего слоя. Кроме входного и выходного слоев в многослойной нейронной сети есть один или несколько скрытых слоев.

Связи от выходов нейронов некоторого слоя q к входам нейронов следующего слоя (q+1) называются последовательными.

В свою очередь, в многослойных нейронных сетях выделяют следующие типы НС.

1. Монотонные НС являются частным случаем слоистых сетей, отличающиеся от последних рядом дополнительных условий, накладываемых на связи и нейроны. Каждый слой кроме последнего (выходного) разбит на два блока: возбуждающий и тормозящий. Связи между блоками также разделены на тормозящие и возбуждающие. Если от нейронов блока A к нейронам блока B ведут только возбуждающие связи, то это означает, что любой выходной сигнал блока является монотонной неубывающей функцией любого выходного сигнала блока A. Если же эти связи только тормозящие, то любой выходной сигнал блока B является невозрастающей функцией любого выходного сигнала блока A. Для нейронов монотонных сетей необходима монотонная зависимость выходного сигнала нейрона от параметров входных сигналов.

2. В сетях без обратных связей нейроны входного слоя получают входные сигналы, преобразуют их и передают нейронам первого скрытого слоя, и так далее вплоть до выходного, который выдает сигналы для интерпретатора и пользователя. Если не оговорено противное, то каждый выходной сигнал q-го слоя подастся на вход всех нейронов (q+1)-го слоя; однако возможен вариант соединения q-го слоя с произвольным (q+p)-m слоем.

Среди многослойных сетей без обратных связей различают полносвязанные (выход каждого нейрона q-го слоя связан с входом каждого нейрона (q+1)-го слоя) и частично полносвязанные. Классическим вариантом слоистых сетей являются полносвязанные сети прямого распространения (рис. 1).

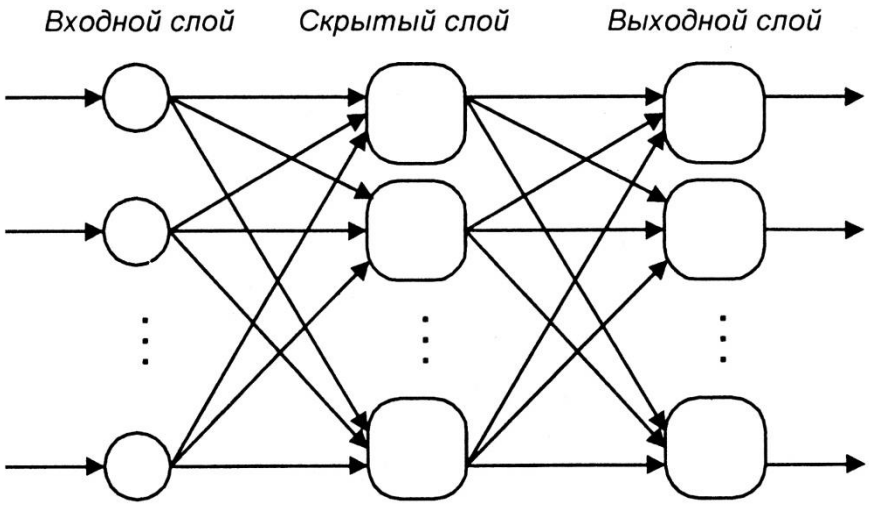


Рис. 5.10. Многослойная сеть прямого распространения

3. В сетях с обратными связями информация с последующих слоев может передаваться на предыдущие слои нейронов. Эти сети также могут быть разделены на следующие классы:

- слоисто-циклические – отличающиеся тем, что слои замкнуты в кольцо. Последний слой передает свои выходные сигналы первому, при этом все слои равноправны и могут, как получать входные сигналы, так и выдавать выходные;
- слоисто-полносвязанные – состоящие из слоев, каждый из которых представляет собой полносвязную сеть. Сигналы передаются как от слоя к слою, так и внутри слоя. В каждом слое цикл работы распадается на три этапа: прием сигналов с предыдущего слоя, обмен сигналами внутри слоя, выработка выходного сигнала и передача его последующему слою;
- полносвязанно-слоистые – аналогичные по своей структуре слоисто-полносвязанным, но функционирующие другим образом. В таких НС фазы обмена информацией внутри слоя и передачи ее следующему не разделяются. На каждом такте нейроны всех слоев принимают сигналы от нейронов как своего слоя, так и последующих.

Примером сетей с обратными связями могут послужить частично-рекуррентные сети Элмана и Жордана, представленные на рис. 2.

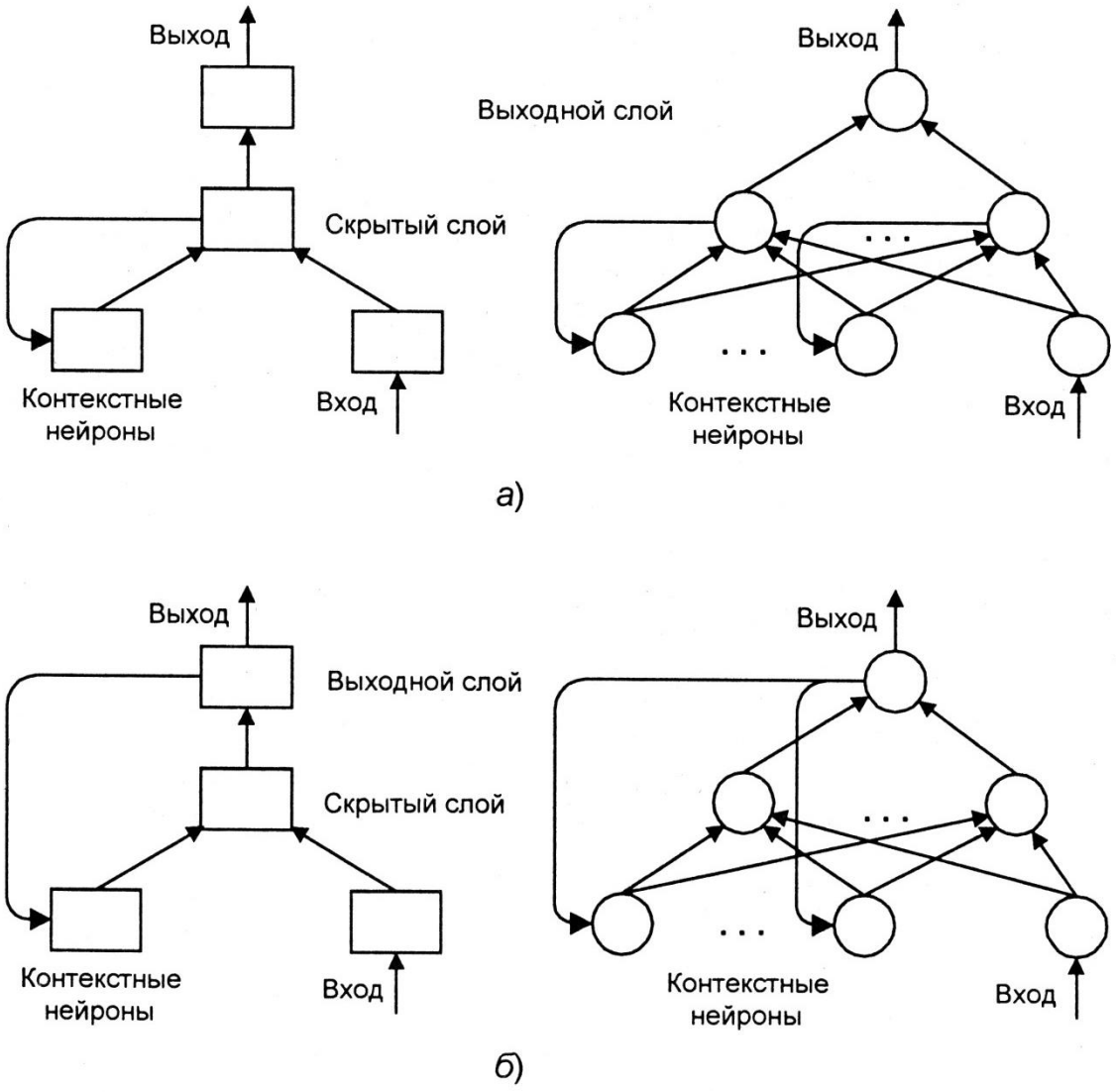


Рис. 2. Сети с обратными связями:
а – Элмана, *б* – Жордана

Наконец, в слабосвязных НС (см. рис. 1, *в*) нейроны располагаются в узлах прямоугольной, гексагональной или октогональной решетки. В таких НС каждый нейрон связан с четырьмя, шестью или восемью своими ближайшими соседями, называемых окрестностями фон Неймана, Голея и Мура соответственно.

По типам структур нейронов известные нейронные сети можно разделить на гомогенные (однородные), состоящие из нейронов одного типа с единой функцией активации, и гетерогенные, типы и функции активации нейронов которой различны.

Существуют бинарные и аналоговые сети. Первые из них оперируют только двоичными сигналами, и выход каждого нейрона может принимать значение либо логического ноля (заторможенное состояние) либо логической единицы (возбужденное состояние).

Еще одна классификация делит нейронные сети на синхронные и асинхронные. В первом случае в каждый момент времени лишь один нейрон меняет свое состояние, во втором – состояние изменяется одновременно у целой группы нейронов, как правило, у всего слоя. Алгоритмически ход времени в нейронных сетях задается итерационным выполнением однотипных действий над нейронами.

Сети можно классифицировать также по числу слоев. Теоретически число слоев и число нейронов в каждом слое может быть произвольным, однако фактически оно ограничено ресурсами компьютера, на котором реализована нейронная сеть. Чем сложнее сеть, тем более сложные задачи она может решать.

Независимо от способа реализации, нейронную сеть можно рассматривать как взвешенный ориентированный граф. Узлы этого графа соответствуют нейронам, а ребра – связям между ними. С каждой связью ассоциирован вес – рациональное число, отображающее оценку возбуждающего или тормозящего сигнала, передаваемого по этой связи на вход нейрона-приемника при возбуждении нейрона-передатчика. Нейронная сеть образуется путем объединения ориентированными взвешенными ребрами выходов нейронов с входами. Граф межнейронных соединений может быть ациклическим, либо произвольным циклическим, что также может служить одним из классификационных признаков нейронной сети.

Принятие некоторого соглашения о тактировании сети (времени срабатывания нейронов) дает аппарат задания алгоритмов посредством нейронных сетей. Разнообразие этих алгоритмов ничем не ограничено, так как можно использовать нейроны с различными функциями активации и состояния, а также двоичными, целочисленными, вещественными и другими

форматами представления значений весов и входов. Это дает возможность описывать в терминах нейронных сетей решение как хорошо формализованных задач из области математики и физики, так и плохо формализуемых задач распознавания, классификации, обобщения и ассоциативного запоминания.

Сети могут быть конструируемыми или обучаемыми. В конструируемой сети число и тип нейронов, граф межнейронных связей, веса входов нейронов определяются при создании сети, исходя из решаемой задачи. Например, при конструировании НС, функционирующей как ассоциативная память, каждая входная последовательность из заранее определенного набора, предназначенного для запоминания сетью, участвует в определении весов входов нейронов сети. После конструирования функционирование сети заключается в следующем. При подаче на входы частичной или ошибочной входной последовательности сеть через какое-то время переходит в одно из устойчивых состояний, предусмотренных при ее конструировании. При этом на входах сети появляется одна из запомненных последовательностей, признаваемая сетью как наиболее близкая к изначально поданной.

В обучаемых сетях графы межнейронных связей и веса входов изменяются при выполнении алгоритма обучения. Алгоритмы обучения сети делятся на наблюдаемые, ненаблюдаемые и смешанные (гибридные). Первые при обучении сравнивают заранее известный выход с получившимся значением. Вторые обучаются, не зная заранее правильных выходных значений, но группируя «близкие» входные векторы так, чтобы они формировали один и тот же выход сети. Ненаблюдаемое обучение используется, в частности, при решении задачи кластеризации. При смешанном алгоритме обучения часть весов определяется при наблюдаемом, а часть – при ненаблюдаемом обучении. Обучение осуществляется путем предъявления примеров, состоящих из наборов входных данных в

совокупности с соответствующими результатами при наблюдаемом обучении и без последних – при ненаблюдаемом.

Как правило, выбор структуры нейронной сети осуществляется в соответствии с особенностями и сложностью задачи. Для решения отдельных типов задач уже существуют оптимальные конфигурации. Если задача не может быть сведена ни к одному из известных типов, то приходится решать сложную проблему синтеза новой конфигурации. При этом необходимо помнить, что возможности сети возрастают с увеличением числа нейронов сети, плотности связей между ними и числом слоев. Однако введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети.

Задачи, решаемые с помощью нейронных сетей

Знания в сети представлены в неявном виде. Нельзя выделить конкретный структурный элемент сети, который представлял бы отдельное правило или сущность предметной области. Знание отражено именно во взвешенных связях между множествами отдельных элементов сети. В данном случае имеют место распределенные знания, которые нельзя представить в виде простого перечисления числовых или символических элементов. По этой причине часто можно встретить утверждение, что НС выполняет несимволическую или субсимволическую [англ. sub symbolic] обработку информации.

В сетях связности знания сохраняются не в декларативном виде, поэтому они не могут быть доступны для интерпретации со стороны какого-либо внешнего процесса. Доступ к знаниям и процесс логического вывода могут быть описаны только в терминах активности сети.

Однако, даже в случае, если узлы представляют сущности предметной области, общая картина активности множеств узлов сети может скрывать понятия достаточно высокого уровня, объединяющие определенные аспекты сущностей, представленных узлами. Например, пусть возбуждены узлы

представляющие слова «гонки», «машина» и «водитель». Таким способом может быть представлено понятие «водитель гоночной машины», а может – сам факт вождения гоночной машины. В любом случае такое представление может расцениваться как субсимволическое, поскольку составляющие его узлы не могут быть сведены к какой-нибудь синтаксической структуре, имеющей однозначный, явно выраженный смысл. Точно так же нельзя выполнить и семантический анализ состояния множеств узлов с помощью какого-либо внешнего набора правил.

Любая нейронная сеть используется в качестве самостоятельной системы представления знаний, которая в практических приложениях выступает, как правило, в качестве одного из компонентов системы управления либо модуля принятия решений, передающих результирующий сигнал на другие элементы, не связанные непосредственно с искусственной нейронной сетью.

Задачи, решаемые нейронными сетями, можно условно разделить на несколько основных групп:

- аппроксимации и интерполяции;
- распознавания и классификации образов;
- сжатия данных;
- прогнозирования;
- идентификации;
- управления;
- ассоциации.

В каждом из указанных приложений нейронная сеть играет роль универсального аппроксиматора функции от нескольких переменных, реализуя нелинейную функцию

$$Y = f(x), \tag{1}$$

где x – входной вектор,

Y – реализация векторной функции нескольких переменных.

Постановки значительного количества задач моделирования, идентификации и обработки сигналов могут быть сведены именно к аппроксимационному представлению.

Для классификации и распознавания образов сеть необходимо обучить важнейшим их признакам, таким, как геометрическое отображение точечной структуры изображения, относительное расположение важнейших элементов образа, компоненты преобразования Фурье и др. В процессе обучения выделяются признаки, отличающие образы друг от друга, создаются классы и эталоны, которые и составляют базу для принятия решений об отнесении образов к соответствующим классам.

При решении задач прогнозирования роль нейронной сети состоит в предсказании будущей реакции системы в зависимости от располагаемой информации о ее предшествующем поведении. Обладая информацией о значениях вектора x в моменты, предшествующие прогнозированию

$x(k\text{-}1)$, $x(k\text{-}2)$, ... , $x(k\text{-}N)$,

сеть вырабатывает решение, каким будет наиболее вероятное значение последовательности $\hat{x}(k)$ в текущий момент k. Для адаптации весовых коэффициентов сети используются фактическая погрешность прогнозирования

$$\varepsilon = x(k) - \hat{x}(k) \qquad (2)$$

и значения этой погрешности в предшествующие моменты времени.

При решении задач идентификации и управления динамическими процессами нейронная сеть, как правило, выполняет несколько функций. Она представляет собой нелинейную модель этого процесса, обеспечивающую выработку соответствующего управляющего воздействия. Сеть также выступает в роли следящей системы, адаптирующейся к изменяющимся условиям окружающей среды. Очень большое значение, особенно при управлении роботами, имеет функция классификации, реализуемая при выработке решения о дальнейшем развитии процесса.

В задачах ассоциации нейронная сеть играет роль ассоциативного запоминающего устройства. Можно выделить ЗУ автоассоциативного типа, с помощью которых определяется корреляция между отдельными компонентами одного и того же входного вектора, и ЗУ гетероассоциативного типа, средствами которых устанавливается корреляция между двумя различными векторами. Если на вход сети подается неструктурированный вектор (например, содержащий искаженные шумом компоненты или вообще не содержащий отдельные компоненты), нейронная сеть сможет восстановить оригинальный и очищенный от шумов вектор и сгенерировать при этом полную версию ассоциированного с ним вектора.

Важнейшее свойство нейронных сетей, свидетельствующее об их огромном потенциале и широких прикладных возможностях, состоит в параллельной обработке информации одновременно всеми нейронами. Благодаря этой способности при большом количестве межнейронных связей достигается значительное ускорение процесса обработки информации. Во многих ситуациях становится возможной обработка сигналов в реальном масштабе времени.

Очень большое количество межнейронных соединений приводит к тому, что сеть становится нечувствительной к ошибкам, возникающим в отдельных контактах. Функции поврежденных соединений принимают на себя другие элементы, в результате в деятельности сети не наблюдаются заметные нарушения. Это свойство используется, в частности, при поиске оптимальной архитектуры нейронной сети путем разрыва отдельных связей.

Другое не менее важное свойство нейронной сети состоит в способности к обучению и к обобщению полученных знаний. Сеть обладает чертами так называемого искусственного интеллекта. Натренированная на ограниченном множестве обучающих выборок, она обобщает накопленную информацию и вырабатывает ожидаемую реакцию применительно к данным, не обрабатывавшимся в процессе обучения. Несмотря на значительное количество уже известных практических приложений искусственных

нейронных сетей, возможности их дальнейшего использования для обработки сигналов не изучены окончательно, и можно высказать предположение, что нейронные сети еще в течение многих лет будут средством развития информационной техники .

В настоящее время известно большое количество программных продуктов, выпускаемых рядом фирм и отдельными исследователями и позволяющих конструировать, обучать и использовать нейронные сети для решения практических задач.

Приведем названия некоторых из них, предназначенных для использования на персональных компьютерах в различных операционных средах:

- NeuroSolutions фирмы NeuroDimension Inc.;
- NeuralWorks Professional II Plus с модулем UDND фирмы NeuralWare Inc.;
- Process Advisor фирмы AlWare Inc.;
- NeuroShell 2 фирмы Ward Systems Group;
- BrainMaker Pro фирмы California Scientific Software.

Тестирование показывает, что все перечисленные пакеты имеют практически одинаковые результаты, а их отличия связаны в основном с возможностями использования различных нейронных структур, критериев оптимизации и алгоритмов обучения сетей, а также с простотой использования и наглядностью представляемой информации.

Эволюция систем инженерии знаний

Инженерия знаний (в классическом смысле) – это раздел искусственного интеллекта, в котором решаются проблемы, связанные с извлечением, приобретением, представлением и манипулированием знаниями.

Инженерия знаний – основа создания экспертных систем и других систем искусственного интеллекта

Разработка технологий искусственного интеллекта, создание прикладных интеллектуальных систем и искусственных сообществ («искусственная жизнь») представляет собой одну из наиболее важных и многообещающих сегодня областей развития информационных коммуникационных технологий.

В данной области происходит интеграция современных сетевых технологий, методов и средств ИИ, включая базы больших и слабоструктурированных данных и знаний, многокомпонентных и многоагентных решателей и систем и средств объектно-ориентированного программирования.

Различные назначения ПС

Различные предметные области

Архитектуры и технологии

Эволюция информационных и лингвистических компонентов прикладных систем осуществляется через призму развития инструментальных средств, методов и технологий ИИ.

Научное направление ИИ зародилось в общем комплексе кибернетических исследований Современное состояние ИИ – искусственный интеллект, искусственная жизнь (в узком смысле этого термина).

Совокупность классических интеллектуальных систем, включающих базы знаний и решатели могут решать сегодня задачи путем рассуждений, связанных с обработкой символов.

Если в системах ИИ реализуются аспекты планирования, прогнозирования на основе знаний, то ИЖ предполагает реализацию механизмов реакций на воздействие среды и локальных взаимодействий. Но так же решаются сложные задачи. Появляется и коллективный интеллект и базовыми дисциплинами могут служить различные области биологических наук (биоинсперированные приложения, эволюционная теория)

Эволюция происходит как в методах, так и в средствах, реализующих эти методы. Рубеж 2000 г. Системы искусственного интеллекта не имеют стимулов к дальнейшему существованию и совершенствованию.

Но развитие других технологий подтолкнуло и развитие в системах инженерии знаний.

Сегодня наибольшее внимание уделяется технологии нейронных сетей, лежащая в основе создания и систем распознавания (текстов, речи, образов, голоса, почерка и т.д.), и беспилотных аппаратов, различных систем прогнозирования, идентификации и т.д.

Выделяют (по Д. Поспелову) два направления в области ИИ: бионическое направление и программно – прагматическое.

Программно – прагматическое: решение сложных логических задач (распознавание, игры, логика, классификация, поиск, синтез и т.д.). В основе информация о языке и мышлении. Иногда в комбинации с бионическими методами.

Бионическое направление: нейробионический, структурно-эвристический, гомеостатический подходы. В основе информации о структурных и функциональных механизмов различных организмов.

Нейробионический – воспроизведение интеллектуальных функций с помощью нейроподобных элементов;

структурно-эвристический – наблюдение за деятельностью субъекта и моделирование его поведения;

гомеостатический – совокупность противобоствующих подсистем, где в процессе функционирования обеспечивается необходимая устойчивость при изменчивости окружающей среды.

Представление знаний – создание методов, позволяющих переходить от знаний в текстовой форме к их аналогам, пригодным для ввода в память ИС. Работа со знаниями лежит в основе развития искусственного интеллекта.

Выделение мягких технологий.

Интеллектуальный решатель C-PRS (Procedural Reasoning System in C), написанный на стандарте ANSI C, используется НАСА, в авиапромышленности, в системах управлении перевозками и мобильными роботами. Он переносим, очень компактен и с помощью кросс-платформных компиляторов допускает встраивание практически в любое оборудование.

Под машинным обучением понимают методы в сфере искусственного интеллекта, применяемые для создания машины, способной обучаться на собственном опыте. Под процессом обучения подразумевают обработку машиной огромных массивов данных и поиск в них закономерностей.

Понятия Machine learning и Data science, несмотря на свою схожесть, все же различаются и справляются каждый со своими задачами. Оба инструмента входят в искусственный интеллект.

Машинное обучение, являющееся одним из разделов ИИ, - алгоритмы, на основании которых компьютер способен делать выводы, не придерживаясь жестко заданных правил. Машина ищет закономерности в сложных задачах с большим количеством параметров, находя более точные ответы, в отличие от мозга человека. Результатом метода является точное прогнозирование.

Наука о способах анализа данных и извлечения из них ценных знаний и информации (data mining). Она сообщается с машинным обучением и наукой

о мышлении, с технологиями взаимодействия с большими объемами данных. Работа Data science позволяет проанализировать данные и отыскать правильный подход для последующей сортировки, обработки, выборки и поиска информации.

К примеру, существуют сведения о финансовых тратах предприятия и сведения контрагентов, связанные между собой только посредством времени и даты проведения операций и промежуточных банковских данных. Глубокий машинный анализ промежуточных данных позволяет определить наиболее затратного контрагента.

Т.е. сегодня искусственный интеллект проник во все сферы жизнедеятельности

Используется конвергентный или гибридный подход.

"Нечеловеческая цивилизация: Общество программ"

Появилось понятие «Искусственная жизнь»

ЛЕВЕНЧУК: Мы как раз не дошли ни до матрицы, ни до сети. Ничего этого пока нет. Мы рассматриваем одну единственную возможность, как вы сказали, тварь, которая создает тварь, за мелким исключением. В качестве этой твари является она сама, то есть мы имеем какую-то программу, которая обладает какими-то интеллектуальными навыками. И эти навыки эта программа использует для того, чтобы себя улучшить и стать умнее, потому что если есть какая-то программа, достаточно умная, то так же, как и человек, первое, что она соображает, что ей надо учиться, ей надо становиться лучше, мощнее и эффективнее. И она делает этот маленький-маленький шаг и становится чуть-чуть лучше, умнее и эффективнее. Только делает она это в отличие от человека, которому довольно много времени надо, для того чтобы стать чуть-чуть лучше, быстрее и эффективнее, делает она это довольно-таки быстро. И следующий шаг, который делает эта уже более умная, более мощная программа – она становится еще чуть-чуть умнее, еще чуть-чуть быстрее. И этот цикл начинает крутиться быстрее, быстрее, быстрее и быстрее.

Направление Искусственная жизнь (ИЖ) – понятие связано с трактовкой интеллектуального поведения, адаптации, саморганизации, как результат локальных взаимодействий различных прикладных систем.

СФЕРА ПРИМЕНЕНИЯ СИСТЕМ ИСКУСТВЕННОГО ИНТЕЛЕКТА

К сфере ИИ относятся те области, где:

1. Нет точного метода, и для решения используют **эвристики.**

Эвристики *– способы нахождения гипотез для решения задач, на основе успешного опыта и интуиции;*

2. Информация часто задается в символьном виде, а не в цифровом;

3. Есть возможность выбора, но нет алгоритма и используются различные процедуры вывода и поиска.

Системы ИИ работают в следующих областях:

1. Семантический анализ и обработка естественно-языковой информации при осуществлении

- машинного поиска естественно-языковых документов в базах данных;
- машинного перевода;
- автоматического реферирования;
- автоматической классификации документов;
- генерации (синтезе) текста;
- генерация (синтезе) речи.

2. Распознавание образов и анализ изображений.

Модель распознавания образов состоит из 3х частей:

- датчика, который воспринимает воздействие внешней среды (например, акустические сигналы) и преобразует его к виду, удобному для машинной обработки;
- устройства выделения признаков (рецептора), которое производит отбор информации из входных данных, предположительно относящейся к делу;
- классификатора, относящего отобранные данные к одному из нескольких классов.
- Наиболее развиты системы распознавания зрительных образов.

3. Системы, моделирующие поведение или Системы планирования действий (например, манипуляций робота)

4. Системы решения отдельных интеллектуальных задач :

- программы компьютерного доказательства теорем;
- игровые программы.

Системы решения задач основаны на методах поиска решений с помощью проб и ошибок. Задачи решаются посредствам поиска в пространстве возможных решений.

5. Экспертные системы, т.е. системы основанные на знаниях о заданной проблемной области, в которой знания слабо структурированы, решаются сложные **NP-задачи**, осуществляется взаимодействие с естественным языком на основе рассуждений и комментирования действий с целью обучения пользователя или при самообучении системы.

NP-задачи- задачи, где логическая обработка информации превалирует над вычислительной, задачи, которые могут не сойтись при конечном количестве итераций, а также задачи, для которых может не существовать точного решения

6. Извлечение полезных данных (Data Mining, DM-системы)

DM-системы – это системы ИИ, которые

- производят целенаправленный поиск и отбор нужных сведений из больших массивов данных,
- выявляют корреляцию между различными атрибутами элементов данных в реляционных БД,
- оценивают вероятности появления тех или иных ситуаций (например, динамики продаж, тенденций на финансовом рынке и т.д.).

7. Программные продукты на основе нейронных сетей.

8. Использование генетических алгоритмов.

9. Управление знаниями.

ЭКСПЕРТНЫЕ СИСТЕМЫ
и их особенности

Экспертная система (ЭС) - система, предназначенная для решения плохо формализуемых задач;

- для задач, у которых отсутствует алгоритм решения;
- для задач, алгоритм решения которых не известен; или обладает достаточно большой размерностью и используемых для работы в агрессивных средах.

Это система, **в которой заложен опыт специалистов** в заданной предметной области, **представленный знаниями**, которые **сформулированы на естественном языке** или легко взаимодействуют с естественным языком **и реализованы с помощью правил**.

Экспертная система разрабатывается в том случае, если ее разработка,
- необходима,
- оправдана и неоценима.

Назначение экспертных систем

- **1) Интерпретация** – процесс определения смысла данных, результаты которого должны быть согласованными и корректными.
- **2) Планирование** – заранее намеченный порядок, последовательность осуществления какой либо программы, работы, проведения мероприятий.
- **3) Прогнозирование** – обоснованное описание последовательности событий, с возможностью обнаружения новых факторов.
- **4) Мониторинг** - непрерывное оповещение о состоянии системы, приложения или процесса.
- **5) Проектирование** – процесс создания новой информации об объекте, системе (имеется возможность исключения профессионала из процесса проектирования).
- **6) Диагностика** – процесс распознавания состояния на основе имеющихся факторов
- **7) Обучение** - обучение пользователя, а так же самообучение системы, как на этапе приобретения знаний, так и в процессе работы ЭС (пополнение базы знаний (БЗ) ЭС новыми цепочками вывода).

Различают два вида экспертных систем:

1. **Статические.**

Оценивается информация на основе постоянных и неизменных данных в базе данных и правил в базе знаний (знания могут добавляться, но не в процессе функционирования ЭС).

2. Динамические.

Ориентированы на работу с постоянно меняющейся информацией в информационной базе и в базе знаний системы (компонентом базы знаний является база данных).

Однако: Современные ЭС могут сочетать оба класса, реализованных в виде соответствующих режимов работы.

(Например, в зависимости от вида работы, архитектура экспертной системы будет просто включать различные функциональные элементы)

Архитектура экспертной системы

- **1- Рабочая область памяти** – виртуальный блок
- **2- Блок логического вывода** - осуществляется сопоставление данных и знаний, но только тех, которые требуются в заданном промежутке времени
- **3- База знаний** - правила логического вывода, не меняющиеся в заданном промежутке времени (для статических систем), представлены в установленных для системы формализмах (фреймах, семантических сетях, предикатах первого порядка).

!!! БЕЗ ДАННЫХ ЗНАНИЙ НЕ БЫВАЕТ!

- **4- База данных** - содержит постоянные, не меняющиеся в заданном промежутке времени данные системы для активизации знаний из базы знаний;
- **5- Блок приобретения знаний** - предназначен для работы с экспертом (когнитологом) при пополнении и приобретении знаний;
- **6- Блок рекомендаций и комментариев**;
- **7- Интерфейс пользователя** - для взаимодействия с пользователем, отображает состояние рабочей памяти;
- **8- Блок проверки знаний на корректность и противоречивость**.
- **9- Автоматизированный (А) и интерактивный (И) режим работы информационного приложения (ИП)**;
- **10- Прикладное информационное приложение** (ИП) системы
- Для динамических ЭС характерно наличие **датчиков взаимодействия с внешней средой**

Представление знаний в ЭС

Разработка ЭС считается скорее искусством, чем наукой.

В общем виде процесс разработки ЭС можно разделить на шесть этапов:

- Выбор проблемы.
- Разработка прототипа ЭС.
- Доработка до промышленной ЭС.
- Оценка ЭС.
- Стыковка ЭС.
- Поддержка ЭС.

Последовательность этапов не является фиксированной. Каждый последующий этап может принести новые идеи, которые повлияют на предыдущие этапы и приведут к их переработке.

Поэтому:

- 1. Что представлять в ЭС?
- 2 Как представлять знания?

Что представлять в экспертной системе

Современные экспертные системы широко используются для тиражирования опыта ведущих специалистов

Традиционно знания существуют в двух видах - коллективный опыт и личный опыт.

Рис. 1. Предметная область, не пригодная для создания экспертной системы

Рис. 2. Предметная область, пригодная для создания экспертной системы

- Если большая часть знаний в предметной области представлена в виде коллективного опыта (например, высшая математика), эта предметная область не нуждается в экспертных системах (рис.1)

- Если в предметной области большая часть знаний является личным опытом специалистов высокого уровня (экспертов), если эти знания по каким-либо причинам слабо структурированы, такая предметная область скорее всего нуждается в экспертной системе (рис. 2)

Разновидности ЭС

- **Для специалистов, чей профессиональный уровень не слишком высок**.

В базах знаний таких ЭС хранятся знания, полученные от экспертов, используемые всякий раз, когда в этом возникнет необходимость

- **Для специалистов высокой квалификации**, выполняя для них значительную часть рутинных операций и просмотр больших массивов информации. Особенностью является наличие в них системы объяснений, повышающей консультационную силу ЭС.

Режимы работы ЭС

1. Режим приобретения знаний.

Общение с ЭС осуществляет (через посредничество инженера по знаниям).

В этом режиме эксперт, используя компонент приобретения знаний, наполняет систему знаниями, которые позволяют ЭС в режиме решения самостоятельно (без эксперта) решать задачи из проблемной области.

Эксперт описывает проблемную область в **виде совокупности данных и правил.**

Данные определяют объекты, их характеристики и значения, существующие в области экспертизы.

Правила определяют способы манипулирования с данными, характерные для рассматриваемой области

Таким образом, в случае ЭС разработку программ осуществляет **не программист,** а эксперт (с помощью ЭС), не владеющий программированием.

Режимы работы ЭС

2. Режим консультации.

Общение с ЭС осуществляет конечный пользователь, которого интересует результат и (или) способ его получения.

Пользователь может не быть специалистом в данной проблемной области

(в этом случае он обращается к ЭС за результатом, не умея получить его сам), или

Пользователь может быть специалистом (в этом случае он может сам получить результат, но обращается к ЭС с целью ускорить процесс получения результата, либо возложить на ЭС рутинную работу).

В режиме **консультации данные о задаче пользователя** после обработки их диалоговым компонентом **поступают в рабочую память**. **Решатель** на основе входных данных из рабочей памяти, общих данных о проблемной области и правил из БЗ **формирует решение задачи**.

ЭИС при решении задачи не только исполняет предписанную последовательность операции, но и предварительно формирует ее

. Если реакция системы не понятна пользователю, то он может потребовать объяснения: "Почему система задает тот или иной вопрос?", "Как ответ, собираемый системой, получен?".

Компьютерная программа может считаться экспертной системой если:

- Программа обладает знаниями.
- Знания сконцентрированы на определенной предметной области.
- Из этих знаний должны непосредственно вытекать решения проблем.

Таким образом:

- ЭС можно назвать систему искусственного интеллекта, в которой в частично формализованном виде накапливаются знания экспертов - специалистов в соответствующей предметной области и имеются правила использования этих знаний для решения конкретной задачи.
- ЭС дает советы, проводит анализ, выполняет классификацию, дает консультации, … ставит диагнозы.
- ЭС решает задачи в узкой предметной области, обычно требующей проведения экспертизы человеком – специалистом.

Отличия ЭС от других программных продуктов

- Использование не только данных, но и знаний, а также специального механизма вывода решений и новых знаний на основе имеющихся.
- Моделирование ни сколько физической (или иной природы) предметной области, сколько механизма мышления человека применительно к решению задач в этой предметной области.
- ЭС помимо вычислений формируют определенные соображения и выводы, основываясь на тех знаниях, которыми система располагает.
- При решении задач в основном используются **эвристические и приближенные методы**, которые не всегда гарантируют успех.

Отличие ЭС от других систем искусственного интеллекта:

- ЭС явно выражают практическую и коммерческую направленность, имеют дело с предметами реального мира, тогда как другие системы с абстрактными предметами.
- Основная характеристика ЭС – это производительность, то есть скорость получения результата и его достоверность (надежность). В исследовательских системах ИИ не предъявляются требования к производительности и надежности. Для ЭС решение должно быть не хуже, чем то, которое нашел бы специалист.
- ЭС должна обладать способностью объяснить свои решения и доказать их обоснованность.
- Накопление высококачественных знаний экспертов.
- Возможность использования этих знаний неспециалистами или специалистами более низкой квалификации, решающими типовые задачи.

Требования, предъявляемые к ЭС с точки зрения пользователя:

- **накопление и получение знаний** в предметной области, выраженных в явной форме, чтобы они были доступными при решении конкретных задач;
- **адаптация** к уровню подготовленности пользователя;
- **запоминание** промежуточных и окончательных результатов решения задач и способов их получения;
- **возможность общения** с системой на подмножестве профессионального языка пользователя;
- возможность **инициализации** диалога с самой системой (желательно, но не обязательно;
- возможность создания и ведения **базы знаний и базы данных**, описывающих конкретную предметную область;

Требования, предъявляемые к ЭС с точки зрения пользователя

- **скорость** принятия решений;
- возможность **принятия решения на реальной информации**, отображающей текущее состояние исследуемого объекта, в том числе с учетом его «зашумленности» и неполноты;
- возможность **решения узкого класса задач** в конкретной предметной области и самообучение с учетом имеющейся информации об объекте;
- возможность **объяснения** принятых решений;
- наличие **«доски объявлений»**, на которую помещаются планируемые в системе действия и представляются результаты принятых решений;
- возможность **интерпретации** разных заданий пользователя с учетом имеющейся в базе знаний информации;
- желательность **проверки** задания пользователя, фактов и правил базы знаний на логическую непротиворечивость.

Базовые функции экспертных систем

- ***Приобретение знаний*** – это передача потенциального опыта решений проблемы от некоторого источника знаний и преобразование его в вид, который позволяет использовать эти знания в программе.

Процесс *приобретения знаний* является очень медленным и представляет самое узкое место в технологии разработки ЭС

Базовые функции экспертных систем

- ***Представление знаний*** – это средство отыскания методов формального описания больших массивов полезной информации, с целью их последующей обработки с помощью символических вычислений.

Формальное описание означает упорядочивание в рамках какого-либо языка, обладающего четко формализованным синтаксисом построения выражений и такого же уровня семантикой, увязывающей смысл выражения с его формой.

Базовые функции экспертных систем

- ***Управление процессом поиска решений***.
- ***Разъяснения принятого решения*** *п*редоставляет пользователю информацию о том, как интеллектуальная система получила выданное пользователю решение.

Используя маршрут, который сохранился в памяти системы от процесса поиска решения, интеллектуальная система формирует пользователю объяснение на профессиональном естественном языке, позволяющее ему представить все принципиальные шаги решения

Формализация знаний о проблемной области. Онтологический подход к представлению знаний.

Таксономическая классификационная схема.

- Таксономия является структурой из трех компонентов:

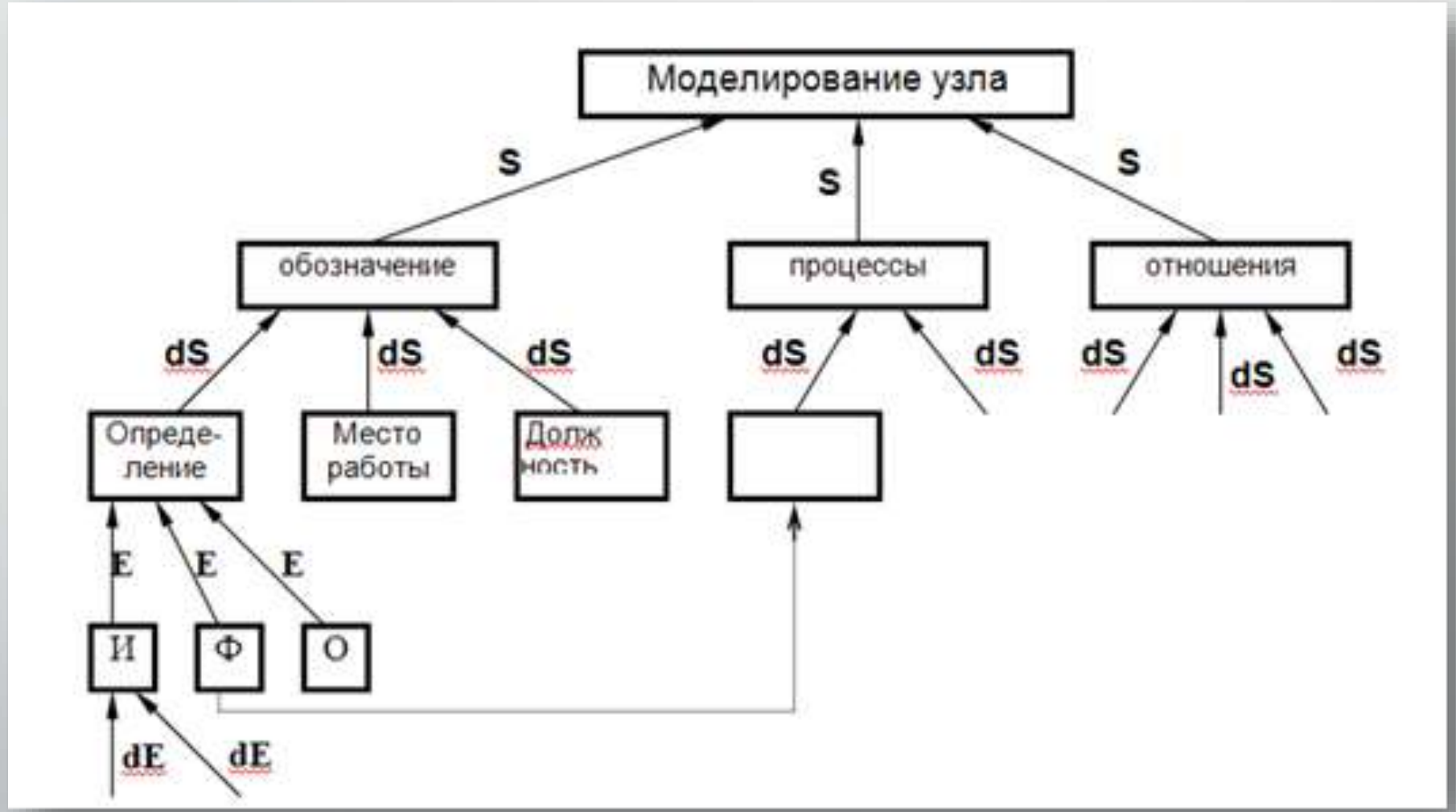
  1)элементы;

  2)дуги;

  3)подсети / узлы.

- Таксономия используется для упорядочивания информации в ИнС. Она имеет вид иерархического дерева, ветвями которого являются обобщенные понятия, соответствующие рассматриваемой проблемной области. В корне таксономической схемы лежит основное понятие, описывающее проблемную область.

Таксономия


Моделирование узла
S
S
S
обозначение
процессы
отношения
dS
dS
dS
dS
dS
dS
dS
dS
dS
Опреде-
ление
Место
работы
Долж
ность
E
E
E
И
Ф
О
dE
dE

Основные виды отношений, используемых в системе ИИ, также могут быть представлены в виде таксономии:

- **каузальные**: а)причина и следствие (прямая цепочка логических рассуждений);
  б)следствие и причина (обратная цепочка логических рассуждений);
- **логико-арифметические**: а)арифметические; б)логические;
- **теоретико-множественные**: а)отношение принадлежности; б)классификационные:
- характеристические (присущие всем БД):
  а) атрибутивные: 1)иметь свойство; 2)быть свойством;
  б) идентифицирующие: 1)быть именем; 2)иметь имя;
- **квантифицирующие**: а)иметь количественное значение; б)иметь лингвистическое значение;
- **динамические**: а)изменить положение; б)изменить ориентацию (на 90°);
- **временные**: а)быть раньше; б)быть позже;
- **пространственные**: а)быть расположенным; б)расстояние между;
- **инструментальные**: а)выполняться посредством; б)выполняться с помощью.

Онтологический подход к представлению проблемной информации.

- **Онтология** – это формальное описание понятий предметной области и отношений между ними в рассматриваемой предметной области, свойств каждого понятия, описывающих различные атрибуты понятия, а также ограничений, наложенных на слоты.
- Онтология вместе с набором индивидуальных экземпляров классов образует базу знаний. В центре большинства онтологий находятся классы. Классы описывают понятия предметной области.
- Онтологии в сети варьируются от больших таксономий, категоризирующих веб-сайты (как на сайте Yahoo!), до категоризаций продаваемых товаров и их характеристик (как на сайте Amazon.com).

Цели разработки онтологий.

Онтологии разрабатываются в целях:

- совместного использования людьми или программными агентами для общего понимания структуры информации;
- обеспечения возможности повторного использования знаний в предметной области;
- возможности явных допущений в предметной области;
- отделения знаний в предметной области от оперативных знаний;
- анализа знаний в предметной области.

При необходимости создать большую онтологию, можно интегрировать несколько уже существующих онтологий.

Многоуровневая схема отношений между онтологиями

При необходимости создать большую онтологию, можно интегрировать несколько уже существующих онтологий

Фундаментальные правила разработки онтологии.

Перечислим фундаментальные правила разработки онтологий.

- 1. Не существует единственно правильного способа моделирования предметной области – всегда найдется жизнеспособная альтернатива. Лучшее решение почти всегда зависит от предполагаемого приложения и ожидаемых расширений.
- 2. Разработка онтологии – это обязательно итеративный процесс.
- 3. Понятия в онтологии должны быть близки к объектам (физическим или логическим) и отношениям в интересующей вас предметной области.

Определение области и масштаба онтологии.

Разработку онтологии следует начинать с определения ее области и масштаба. При этом необходимо ответить на следующие основные вопросы.

- Какую область будет охватывать онтология?
- Для чего будет использоваться данная онтологию?
- На какие типы вопросов должна давать ответы информация в онтологии?
- Кто будет использовать и поддерживать онтологию?

Ответы на эти вопросы могут измениться во время процесса проектирования онтологии, но в любой заданный момент времени они помогают ограничить масштаб модели.

Одним из способов определения масштаба онтологии является составление списка вопросов, на которые должна ответить база знаний, основанная на онтологии, т.е. так называемые вопросы для проверки компетентности.

Определение классов и их иерархии.

Существует несколько возможных подходов для разработки иерархии классов.

Процесс нисходящей разработки начинается с определения самых общих понятий предметной области с последующей конкретизацией понятий.

Процесс восходящей разработки начинается с определения самых конкретных классов, листьев иерархии, с последующей группировкой этих классов в более общие понятия.

Определение свойств классов – слотов.

В онтологии слотами могут стать несколько типов свойств объектов:

- «внутренние» свойства;
- «внешние» свойства;
- части, если объект имеет структуру (как физические, так и абстрактные);
- отношения с другими индивидными концептами (между отдельными членами класса и другими элементами).

Определение фацетов слотов.

Слоты могут иметь различные фацеты, которые описывают тип значения, разрешенные значения, число значений (мощность) и другие свойства значений, которые может принимать слот.

Мощность слота определяет, сколько значений может принимать слот.

Фацет типа значения описывает, какие типы значений можно ввести в слот. Вот список наиболее общих типов значений.

- **Строка** – самый простой тип значения, который используется в таких слотах, как название: значением является простая строка.
- Слот типа число содержит числовые значения. Иногда используются более конкретные типы, например, в формате с плавающей запятой (float) или в целочисленной форме (integer).
- **Булевы слоты** – это простые флаги «да – нет».
- Нумерованные слоты определяют список конкретных разрешенных значений слота. Например, в инструментальной среде Protégé [24] нумерованные слоты имеют тип «символ».
- **Слоты-экземпляры** позволяют определить отношения между индивидными концептами, а также задают список разрешенных классов, т.е. классы, экземпляры которых можно использовать.

Домен слота и диапазон значений слота.

Основные правила определения домена слота и диапазона значений слота схожи друг с другом:

- при определении домена или диапазона значений слота необходимо найти наиболее общие классы или класс, которые могут быть соответственно доменом или диапазоном значений слотов;
- с другой стороны, нет необходимости определять слишком общий домен и диапазон значений. Все классы в домене слота должны быть описаны слотом, а экземпляры всех классов в диапазоне значений слота должны являться потенциальными заполнителями слота;
- нельзя выбирать слишком общий класс для диапазона значений. Т.е. надо выбирать класс, охватывающий все заполнители, а не устанавливать диапазон значений как THING (самый общий класс в онтологии).

Создание экземпляров.

. Для определения отдельного экземпляра класса требуется выбрать класс, создать отдельный экземпляр этого класса, а затем ввести значения слотов.

Как уже отмечалось выше, для любой предметной области не существует единственно правильной иерархии классов. Ее конкретная реализация зависит от возможных способов применения онтологии, уровня детализации, необходимого для приложения, личных предпочтений, а также от требований совместимости с другими моделями. Тем не менее, существует ряд руководящих принципов, которые необходимо учитывать при разработке иерархии классов. После определения значительного количества новых классов полезно остановиться и проверить возникающую иерархию на соответствие этим принципам.

Пример онтологии «Уровни материи».

НЕЖИВАЯ ПРИРОДА	ЖИВАЯ ПРИРОДА	ОБЩЕСТВО
Атомы	Доклеточные формы	Индивид
Молекулы	Одноклеточные формы	Социальные группы
Планеты	Многоклеточный организм	Человечество в целом
Галактика	Популяции	
	Биосфера	

Ответы на контрольные вопросы.

1)Что характеризует понятия концептуализация и формализация при определении проблемной области задачи?

Концептуализация − описание объектов, свойств, рассуждений;

Формализация − представление информации в заданных структурах;

На этапе ***концептуализации*** определяются следующие особенности задачи: типы доступных данных; исходные и выводимые данные, подзадачи общей задачи; применяемые стратегии и гипотезы; виды взаимосвязей между объектами ПО, типы используемых отношений (иерархия, причина — следствие, часть — целое и т.п.); процессы, применяемые в ходе решения; состав знаний, используемых при решении задачи; *типы ограничений*, накладываемых на процессы, которые применены в ходе решения; состав знаний, используемых для обоснования решений.

На этапе формализации все ключевые понятия и отношения (структуры знаний), выражаются на некотором формальном языке, предложенном (выбранном) инженером по знаниям.

2) Опишите структуру таксономической классификационной схемы.

Она имеет вид иерархического дерева, ветвями которого являются обобщенные понятия, соответствующие рассматриваемой проблемной области. В корне таксономической схемы лежит основное понятие, описывающее проблемную область. Основное понятие таксономической схемы не должно иметь предшествующих понятий.

Таксономия является структурой из трех компонентов: 1)элементы; 2)дуги; 3)подсети / узлы.

3) Охарактеризуйте понятие и определите назначение онтологии предметной области.

Онтология – это формальное описание понятий предметной области и отношений между ними в рассматриваемой предметной области, свойств каждого понятия, описывающих различные атрибуты понятия, а также ограничений, наложенных на слоты.

Разрабатываемые в настоящее время стандартные онтологии, могут использоваться экспертами по предметным областям для совместного использования и аннотирования информации в своей области.

Онтология предметной области Другое название – онтология домена. Ее назначение аналогично назначению онтологии верхнего уровня, но область интереса ограничена одной предметной областью (так называемым доменом), например судостроением, авиацией, эксплуатацией корабля и т.д. Онтология предметной области обобщает понятия, использующиеся в некоторых задачах домена, абстрагируясь от самих задач (так, онтология корабля не зависит от особенностей конкретных видов судов).

4) Какова роль онтологий при поиске в сети Интернет?

Онтологии разрабатываются и могут быть использованы при решении различных задач, в том числе для совместного применения людьми или программными агентами, для возможности накопления и повторного использования знаний в предметной области, для создания моделей и программ, оперирующих онтологиями. В работе для решения задачи повышения эффективности поиска в сети Интернет предлагается строить *порталы знаний*, каждый из которых предоставляет доступ к ресурсам сети Интернет определенной тематики. Основу таких порталов знаний составляют онтологии, содержащие описание структуры и типологии соответствующих сетевых ресурсов.

5)Для чего предназначены модели знаний?

Способ представления моделей знаний оказывает существенное влияние на характеристики и свойства разрабатываемой системы. Поэтому представление знаний является одной из наиболее важных проблем, характерных для систем, основанных на знаниях.

6)Охарактеризуйте известные модели представления знаний.

Принято выделять следующие модели представления знаний: фреймы, семантические сети, исчисление предикатов первого порядка, продукции и комбинированные модели.

Организация принятия решений в экспертных системах

Организация логического вывода в ЭС

- Механизм применения правил – это управляющая структура, реализующая стратегии принятия решений. Порядок интерпретации фактов в правилах определяется последовательностью цепочек логического вывода.
-
- На практике используются два вида цепочек логических рассуждений или их комбинация:
- 1) прямое доказательство или прямая цепочка логического вывода
- Например:
- если (Голубь - птица), то (У голубя есть крылья)
- если (У голубя есть крылья), то (Голубь умеет летать)
- 2) обратная цепочка логического вывода или доказательство от противного
- Например:
- если (Голубь умеет летать), то (У голубя есть крылья);
- если (У голубя есть крылья), то (Голубь - птица).

- Принципиальным отличием двух вариантов цепочек логического вывода является форма приложения правил и использования фактов предметной области.

Правила

- Правила, лежат в основе реализации механизма организации логического вывода. Механизм принятия решения на основе логического вывода в экспертных системах называется интерпретатором (рис. 1).

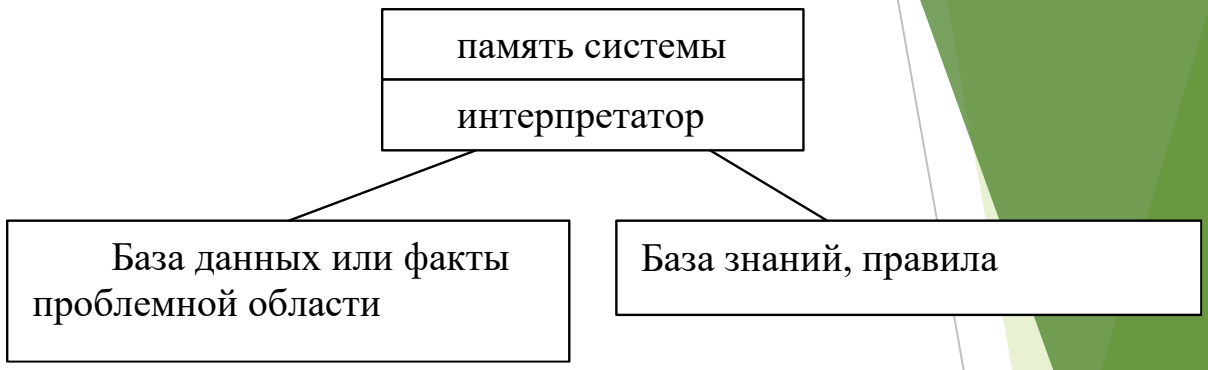


Рисунок 1 – Механизм принятия решения

- Организация вывода или принятия решения выполняется в четыре этапа: выборка, сопоставление, разрешение конфликтов и выполнение (рис. 2).

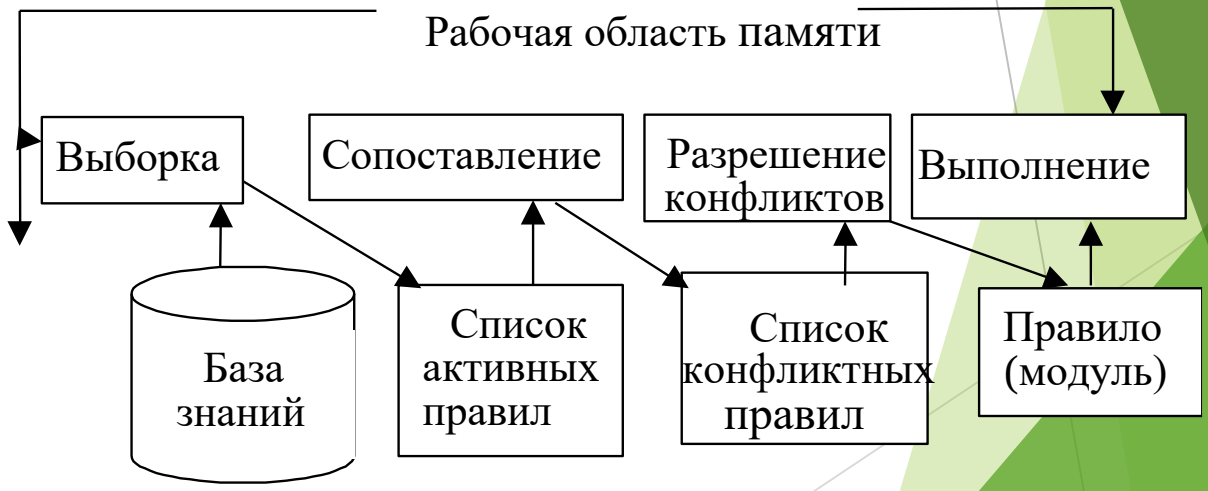


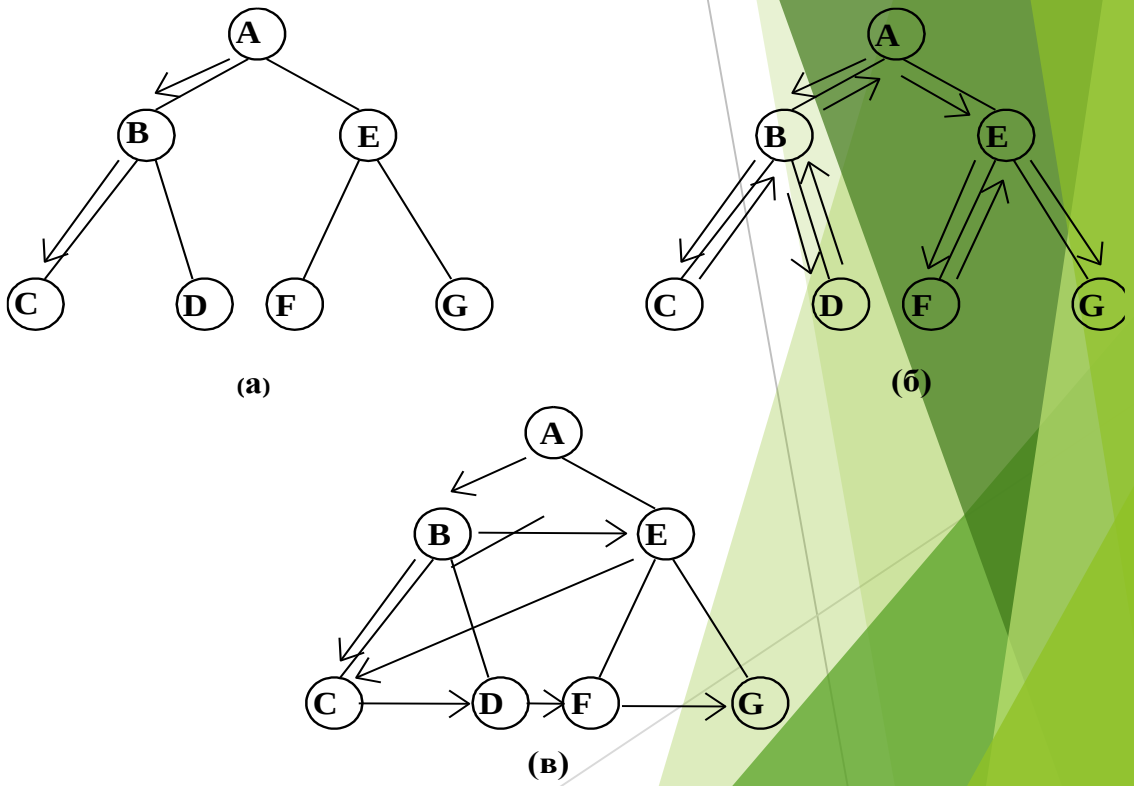
Рисунок 2 – Механизм вывода

Поиск решений

- Основными компонентами организации логического вывода в экспертных системах являются цепочки вывода и управляющая структура, реализующая стратегию поиска.
- Стратегия поиска решений – это задание метаправил по поиску решений, использование специфических эвристик, механизм, позволяющий усовершенствовать метод поиска решений.
- Организация логического вывода в экспертной системе, опираясь на управляющую структуру, использует собственный механизм вывода, называемый интерпретатором. Работа интерпретатора осуществляется на основе данных, описывающих ситуацию и знания в виде правил, объединение которых предполагают получение решения.
- Правила используются в экспертных системах, основанных на правилах. В ЭС, основанных на сопоставлении с образцом, их роль играют так называемые модули.

Управляющая структура

- Механизм приложения правил-продукций и правил-доказательств называется **управляющей структурой**. Эта структура определяет способ поиска фактов предметной области и, в зависимости от их наличия или отсутствия, – применение того или иного правила. Механизм работы с правилами реализуется метауправляющей структурой. Поиск фактов в классической управляющей структуре осуществляется путем реализации одной из стратегий поиска: вглубь или вширь (рис. а, б и рис. в соответственно, где A, B, C, … – факты, участвующие в правилах).



Управляющая структура

- Глубинная стратегия является исчерпывающей, раскрывает все возможные значения пары, появившейся в списке слотов. Новые правила добавляются начало списка в соответствии с приоритетами.

- Широтная стратегия также является исчерпывающей, раскрывает все возможные значения нового факта. Новые правила добавляются к концу списка в соответствии с их приоритетами. Первым применяется правило, которое может производить новые пункты в списке слотов. Если не пусты оба списка – список правил и список активных слотов (агенда), – то приоритет отдается очередному элементу списка.

- Стратегия выборочной оценки является наиболее эффективной, т.к. выбирается одно самое успешное направление рассуждения. Как только одно правило из списка стало истинным, все остальные правила исключаются из него.

- В соответствии со стратегией оценки лучших новые правила перемешиваются с уже имеющимися в списке в соответствии с приоритетами. Первым применяется правило, которое может производить новые пункты в списке слотов. Если не пусты оба списка, то приоритет отдается очередному элементу списка слотов.

Технологии принятия решений в системах с базами знаний

- Технологии принятия решений классифицируются в соответствии со следующими подходами:
- поверхностный подход подразумевает использование правил общего характера, релевантных предметной области и полученных только от эксперта в виде эвристик. Поиск решения ведется индуктивным методом;
- при структурном подходе правила общего и частного видов принятия решений используются на основе поиска по дереву решения или с помощью специального механизма (стратегии принятия решений);
- глубинный подход подразумевает процедуру систематизации проблемной области с учетом модели представления знаний. Правила общего и частного видов формируются исходя из используемой модели, состояния предметной области и рекомендации экспертов.
- совмещенный или комбинированный подход сочетает в себе все перечисленные выше методы.

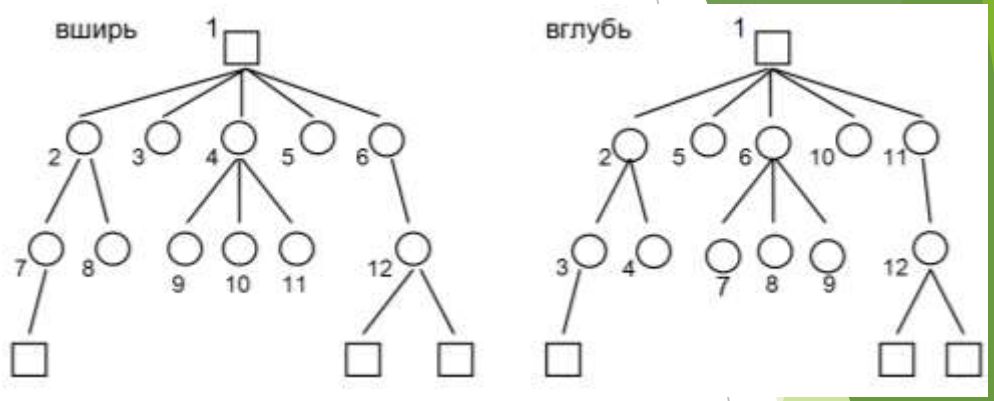
Методы поиска, реализованные в ЭС

- Методы поиска можно классифицировать следующим образом:
- 1. по определению предметной области;
  - по размерности пространства,
  - по количеству пространств и определению места и времени,
  - по моделям, описывающим предметную область,
  - по совокупности моделей,
  - по определению неопределенности, то есть точности задания данных, размытости представления информации и т.д.;
- 2. по представлению результатов:
  - по количеству представленных результатов (один, несколько, все),
  - полноте представления информации и результата.

Далее рассмотрим некоторые известные методы.

Методы поиска, реализованные в ЭС

- **Слепой поиск**. Стратегия слепого списка использует стратегии поиска «вглубь» и поиска «вширь» в совокупности с прямой и обратной цепочкой логического вывода. Пример приведен на рисунке, конечные вершины графа – результаты поиска.



Эвристический метод поиска.. При увеличении пространства поиска происходит рост объема памяти и времени решения. Стремление сократить время поиска привело к созданию эвристических методов поиска, т.е. методов, использующих лишь некоторую информацию о предметной области. При этом методе просматриваются только пути, приводящие к цели с наибольшей вероятностью. Для сокращения пространства поиска используются различные критерии, например, мера «перспективности» вершины.

Методы поиска, реализованные в ЭС

- **Метод редукции** описывается с помощью и/или-графа. Реализация задачи подразумевает ее разбиение на совокупность подзадач, каждая из которых представляется дугой графа. Каждая дуга графа имеет свое назначение. Различают дизъюнктивные ветви (дуги «или») и конъюнктивные дуги (дуги «и»).
- При выполнении подзадач с помощью дуги «или» должна быть выполнена хотя бы одна из подзадач текущей задачи.

При реализации дуги «и» должны быть выполнены все подзадачи. Конъюнктивные дуги на графе объединяются специальным значком в виде дужки (рис.). При поиске результата на «и/или»-графе обычно применяются стратегии поиска«вширь» и «вглубь».

Методы поиска, реализованные в ЭС

- **Метод поиска с помощью генерации и проверки.** В процессе реализации процедуры поиска на каждом очередном шаге генерируется возможное решение и проверяется: ведет ли оно к результату или является результирующим. При генерации текущего возможного решения (состояния, подзадачи) возникает проблема распределения знаний между генератором и устройством проверки. При реализации слепого или эвристического поиска используются минимальные знания об области, достаточные для генерации возможных решений, а устройство проверки определяет, не является ли очередное решение целевым.

- **При методе поиска в иерархических пространствах** в целях разрешения проблемы увеличения времени из-за увеличения пространства поиска общее пространство поиска разбивается на ряд иерархически связанных подпространств, поиск в которых осуществляется в первую очередь в соответствии со специальным алгоритмом. Методы поиска в иерархических пространствах делятся на поиск в факторизованном

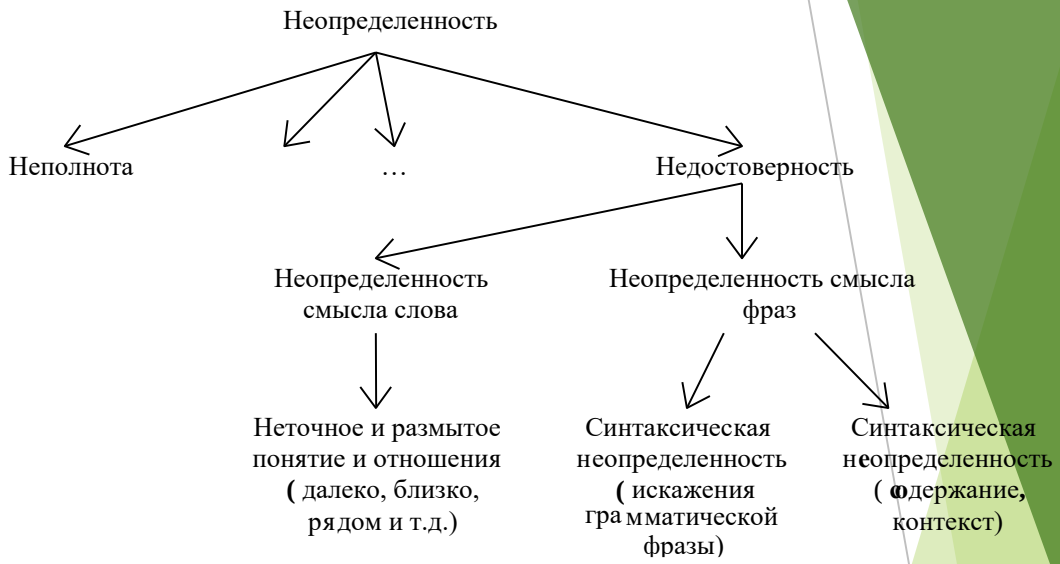
Использование процедур

- В традиционных языках программирования при вызове модуля или процедуры используется имя процедуры. При разработке систем искусственного интеллекта выбор модуля осуществляется на основе текущего состояния проблемной области.
- Совокупность правил, реализованных в управляющей структуре в соответствии с алгоритмом на основе исходной структуры данных, выполняется процедурами различного вида:
  - семантического и синтаксического контроля;
  - моделирования;
  - обработки и изменения структур данных и сред их сопровождения;
  - поиска решений.

- Экспертная система должна выполнять процедуры, позволяющие реализовать процессы таким же образом, как это делает человек при использовании собственного опыта экспертных знаний и рассуждений. Важной особенностью является способность вырабатывать грамотные решения, опираясь на явно неполный или противоречивый набор данных. Таким образом, в полноценной ЭС должны быть реализованы процедуры обработки неопределенностей.

Представление неопределенности в информационных приложениях с базами знаний

- Неопределенность можно определить как степень соответствия процесса или состояния характеристикам реального мира.
- Для исключения семантической неопределенности слов используются вероятностные характеристики и аппарат нечетких множеств.



Неопределенность смысла фраз идентифицируется с использованием лингвистического анализатора текста, причем отдельно анализируется синтаксическая неопределенность и семантическая.
Оценка семантической составляющей неопределенности может быть получена с помощью вероятностных показателей различного рода, коэффициентов уверенности и мощности правил в методе редукции, введения интервалов значений и вероятностей попадания в заданный интервал, использованием формулы Байеса, использованием лингвистических переменных, использованием переменных неопределенности и др.

Организация принятия решений в экспертных системах

Организация логического вывода в ЭС

- Механизм применения правил – это управляющая структура, реализующая стратегии принятия решений. Порядок интерпретации фактов в правилах определяется последовательностью цепочек логического вывода.
-
- На практике используются два вида цепочек логических рассуждений или их комбинация:
- 1) прямое доказательство или прямая цепочка логического вывода
- Например:
- если (Голубь - птица), то (У голубя есть крылья)
- если (У голубя есть крылья), то (Голубь умеет летать)
- 2) обратная цепочка логического вывода или доказательство от противного
- Например:
- если (Голубь умеет летать), то (У голубя есть крылья);
- если (У голубя есть крылья), то (Голубь - птица).

- Принципиальным отличием двух вариантов цепочек логического вывода является форма приложения правил и использования фактов предметной области.

Правила

- Правила, лежат в основе реализации механизма организации логического вывода. Механизм принятия решения на основе логического вывода в экспертных системах называется интерпретатором (рис. 1).

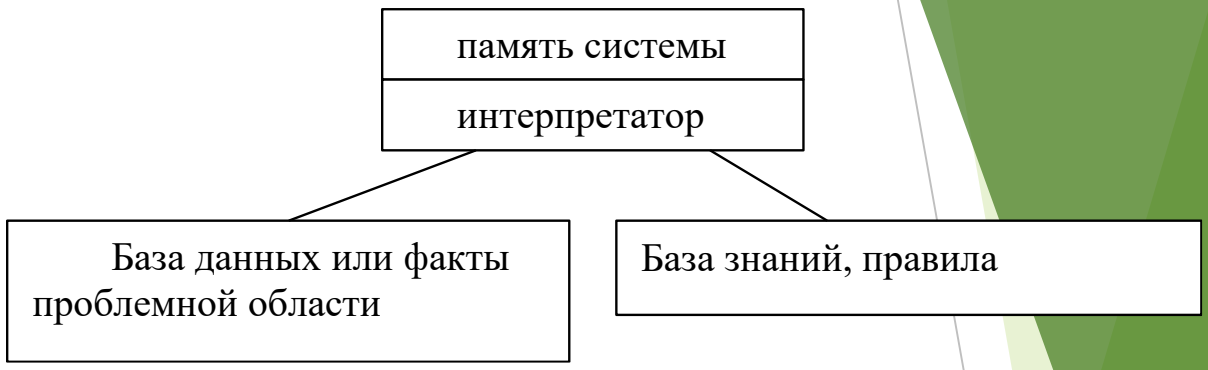


Рисунок 1 – Механизм принятия решения

- Организация вывода или принятия решения выполняется в четыре этапа: выборка, сопоставление, разрешение конфликтов и выполнение (рис. 2).

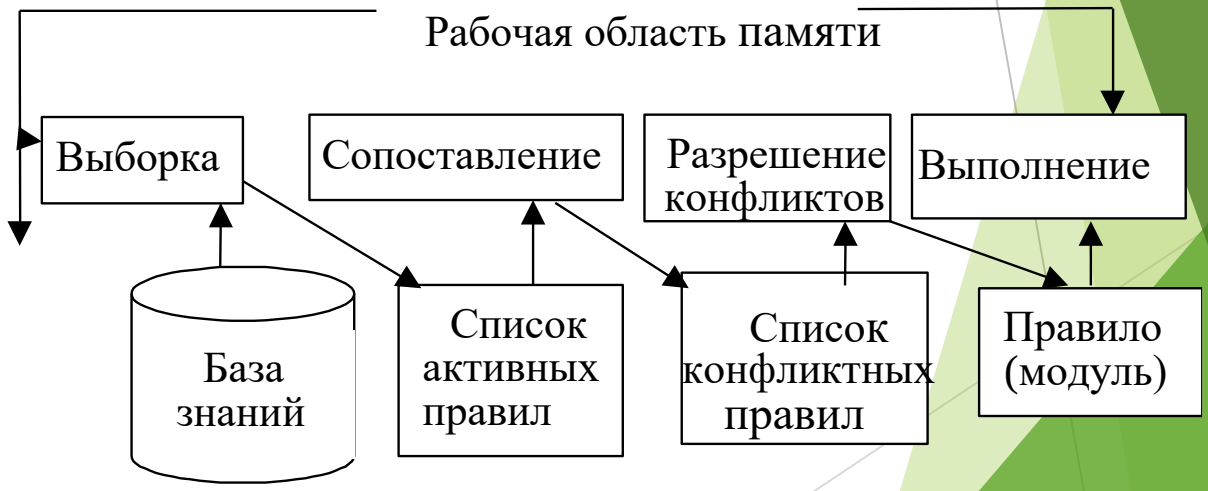


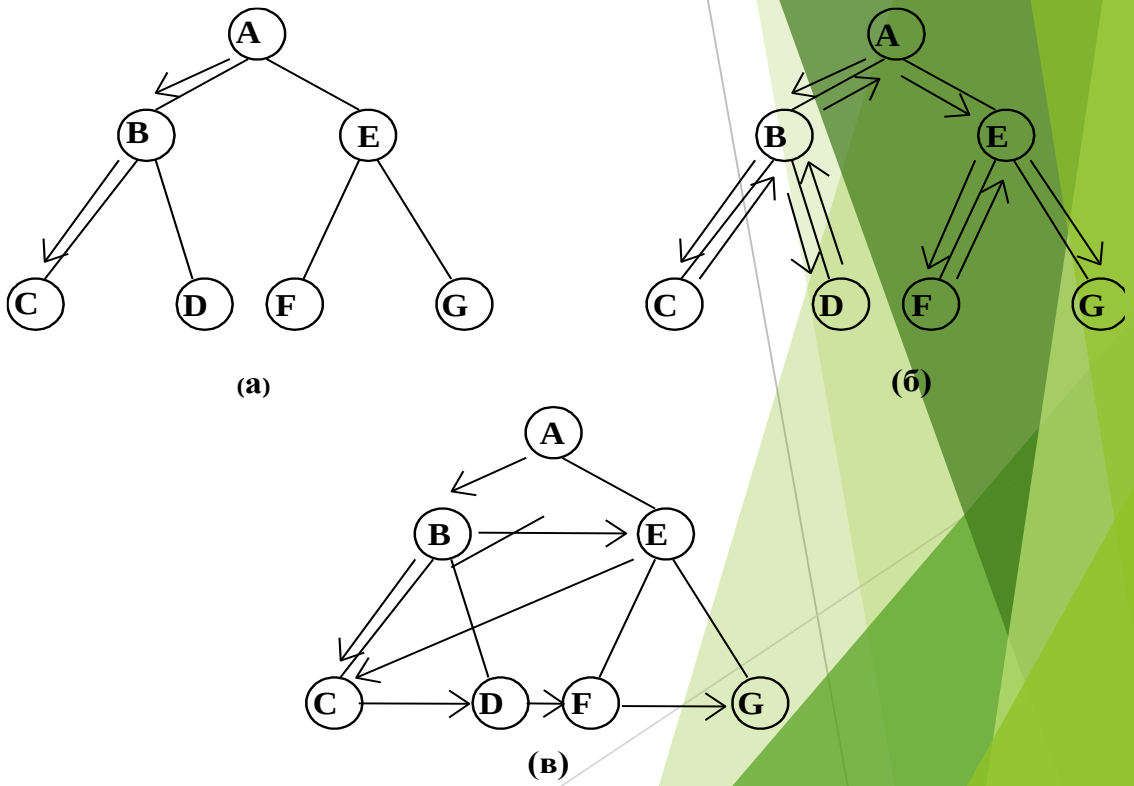
Рисунок 2 – Механизм вывода

Поиск решений

- Основными компонентами организации логического вывода в экспертных системах являются цепочки вывода и управляющая структура, реализующая стратегию поиска.
- Стратегия поиска решений – это задание метаправил по поиску решений, использование специфических эвристик, механизм, позволяющий усовершенствовать метод поиска решений.
- Организация логического вывода в экспертной системе, опираясь на управляющую структуру, использует собственный механизм вывода, называемый интерпретатором. Работа интерпретатора осуществляется на основе данных, описывающих ситуацию и знания в виде правил, объединение которых предполагают получение решения.
- Правила используются в экспертных системах, основанных на правилах. В ЭС, основанных на сопоставлении с образцом, их роль играют так называемые модули.

Управляющая структура

- Механизм приложения правил-продукций и правил-доказательств называется **управляющей структурой**. Эта структура определяет способ поиска фактов предметной области и, в зависимости от их наличия или отсутствия, – применение того или иного правила. Механизм работы с правилами реализуется метауправляющей структурой. Поиск фактов в классической управляющей структуре осуществляется путем реализации одной из стратегий поиска: вглубь или вширь (рис. а, б и рис. в соответственно, где A, B, C, … – факты, участвующие в правилах).



Управляющая структура

- Глубинная стратегия является исчерпывающей, раскрывает все возможные значения пары, появившейся в списке слотов. Новые правила добавляются начало списка в соответствии с приоритетами.

- Широтная стратегия также является исчерпывающей, раскрывает все возможные значения нового факта. Новые правила добавляются к концу списка в соответствии с их приоритетами. Первым применяется правило, которое может производить новые пункты в списке слотов. Если не пусты оба списка – список правил и список активных слотов (агенда), – то приоритет отдается очередному элементу списка.

- Стратегия выборочной оценки является наиболее эффективной, т.к. выбирается одно самое успешное направление рассуждения. Как только одно правило из списка стало истинным, все остальные правила исключаются из него.

- В соответствии со стратегией оценки лучших новые правила перемешиваются с уже имеющимися в списке в соответствии с приоритетами. Первым применяется правило, которое может производить новые пункты в списке слотов. Если не пусты оба списка, то приоритет отдается очередному элементу списка слотов.

Технологии принятия решений в системах с базами знаний

- Технологии принятия решений классифицируются в соответствии со следующими подходами:
- поверхностный подход подразумевает использование правил общего характера, релевантных предметной области и полученных только от эксперта в виде эвристик. Поиск решения ведется индуктивным методом;
- при структурном подходе правила общего и частного видов принятия решений используются на основе поиска по дереву решения или с помощью специального механизма (стратегии принятия решений);
- глубинный подход подразумевает процедуру систематизации проблемной области с учетом модели представления знаний. Правила общего и частного видов формируются исходя из используемой модели, состояния предметной области и рекомендации экспертов.
- совмещенный или комбинированный подход сочетает в себе все перечисленные выше методы.

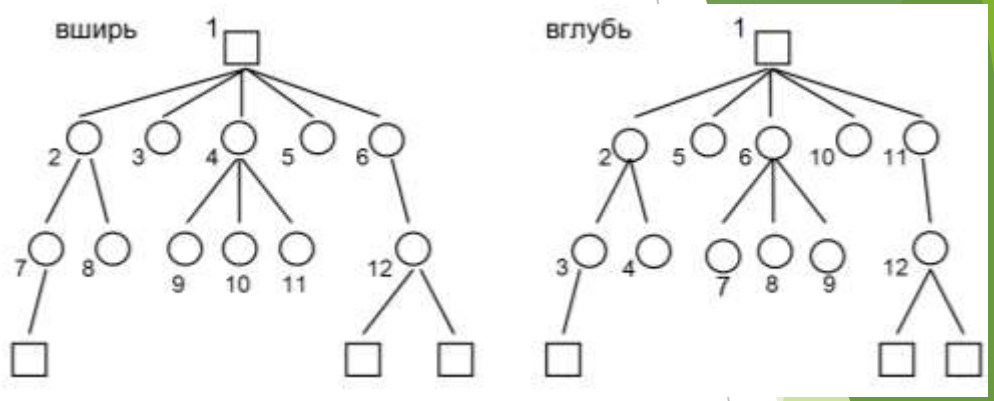
Методы поиска, реализованные в ЭС

- Методы поиска можно классифицировать следующим образом:
- 1. по определению предметной области;
  - по размерности пространства,
  - по количеству пространств и определению места и времени,
  - по моделям, описывающим предметную область,
  - по совокупности моделей,
  - по определению неопределенности, то есть точности задания данных, размытости представления информации и т.д.;
- 2. по представлению результатов:
  - по количеству представленных результатов (один, несколько, все),
  - полноте представления информации и результата.

Далее рассмотрим некоторые известные методы.

Методы поиска, реализованные в ЭС

- **Слепой поиск**. Стратегия слепого списка использует стратегии поиска «вглубь» и поиска «вширь» в совокупности с прямой и обратной цепочкой логического вывода. Пример приведен на рисунке, конечные вершины графа – результаты поиска.



Эвристический метод поиска.. При увеличении пространства поиска происходит рост объема памяти и времени решения. Стремление сократить время поиска привело к созданию эвристических методов поиска, т.е. методов, использующих лишь некоторую информацию о предметной области. При этом методе просматриваются только пути, приводящие к цели с наибольшей вероятностью. Для сокращения пространства поиска используются различные критерии, например, мера «перспективности» вершины.

Методы поиска, реализованные в ЭС

- **Метод редукции** описывается с помощью и/или-графа. Реализация задачи подразумевает ее разбиение на совокупность подзадач, каждая из которых представляется дугой графа. Каждая дуга графа имеет свое назначение. Различают дизъюнктивные ветви (дуги «или») и конъюнктивные дуги (дуги «и»).
- При выполнении подзадач с помощью дуги «или» должна быть выполнена хотя бы одна из подзадач текущей задачи.

При реализации дуги «и» должны быть выполнены все подзадачи. Конъюнктивные дуги на графе объединяются специальным значком в виде дужки (рис.). При поиске результата на «и/или»-графе обычно применяются стратегии поиска«вширь» и «вглубь».

Методы поиска, реализованные в ЭС

- **Метод поиска с помощью генерации и проверки.** В процессе реализации процедуры поиска на каждом очередном шаге генерируется возможное решение и проверяется: ведет ли оно к результату или является результирующим. При генерации текущего возможного решения (состояния, подзадачи) возникает проблема распределения знаний между генератором и устройством проверки. При реализации слепого или эвристического поиска используются минимальные знания об области, достаточные для генерации возможных решений, а устройство проверки определяет, не является ли очередное решение целевым.

- **При методе поиска в иерархических пространствах** в целях разрешения проблемы увеличения времени из-за увеличения пространства поиска общее пространство поиска разбивается на ряд иерархически связанных подпространств, поиск в которых осуществляется в первую очередь в соответствии со специальным алгоритмом. Методы поиска в иерархических пространствах делятся на поиск в факторизованном

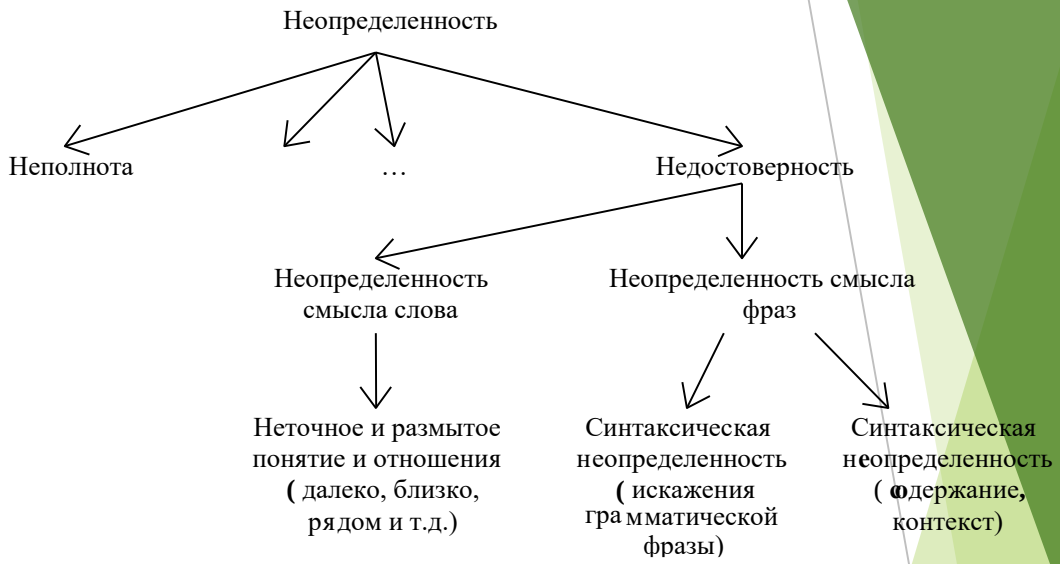
Использование процедур

- В традиционных языках программирования при вызове модуля или процедуры используется имя процедуры. При разработке систем искусственного интеллекта выбор модуля осуществляется на основе текущего состояния проблемной области.
- Совокупность правил, реализованных в управляющей структуре в соответствии с алгоритмом на основе исходной структуры данных, выполняется процедурами различного вида:
  - семантического и синтаксического контроля;
  - моделирования;
  - обработки и изменения структур данных и сред их сопровождения;
  - поиска решений.

- Экспертная система должна выполнять процедуры, позволяющие реализовать процессы таким же образом, как это делает человек при использовании собственного опыта экспертных знаний и рассуждений. Важной особенностью является способность вырабатывать грамотные решения, опираясь на явно неполный или противоречивый набор данных. Таким образом, в полноценной ЭС должны быть реализованы процедуры обработки неопределенностей.

Представление неопределенности в информационных приложениях с базами знаний

- Неопределенность можно определить как степень соответствия процесса или состояния характеристикам реального мира.
- Для исключения семантической неопределенности слов используются вероятностные характеристики и аппарат нечетких множеств.



Неопределенность смысла фраз идентифицируется с использованием лингвистического анализатора текста, причем отдельно анализируется синтаксическая неопределенность и семантическая.
Оценка семантической составляющей неопределенности может быть получена с помощью вероятностных показателей различного рода, коэффициентов уверенности и мощности правил в методе редукции, введения интервалов значений и вероятностей попадания в заданный интервал, использованием формулы Байеса, использованием лингвистических переменных, использованием переменных неопределенности и др.

Инструментальная оболочка разработки экспертных систем CLIPS

CLIPS (C Language Integrated Production System)

1.Общие сведения о CLIPS

Первая версия инструментальной оболочки CLIPS была разработана весной 1985 г. Особое внимание было уделено созданию инструмента, совместимого с известными системами искусственного интеллекта. Таким образом, синтаксис CLIPS очень близок к синтаксису подмножества С. CLIPS является одним из распространенных инструментальных средств разработки экспертных систем. В оболочке реализуется продукционная модель знаний. В основе разработки - три компонента: список фактов, база знаний в виде правил, блок логического вывода. Система полностью реализована на С. Кроме процедурной, поддерживает и объектно-ориентированную парадигму программирования. Аспекты объектно-ориентированного программирования в пособии не рассматриваются. Из-за мобильности, расширяемости, широты возможностей CLIPS получила широкое распространение и в правительстве, промышленности, и университетах. Всем желающим доступен как сам CLIPS последней версии, так и его исходные коды. CLIPS состоит из интерактивной среды – ЭО со своим способом представления знаний, гибкого языка и нескольких вспомогательных инструментов.

2.Представление знаний

- В системе CLIPS одной из основных форм представления информации являются *факты*. Каждый *факт* представляет фрагмент информации, который был помещен в текущий список фактов, называемый fact-list. Факт представляет собой основную единицу данных, используемую правилами.
- Одним из основных методов представления знаний в CLIPS являются правила. Правила используются для представления эвристик, определяющих ряд действий, которые необходимо выполнить в определенной ситуации. Разработчик экспертной системы определяет совокупность правил, которые используются совместно для решения проблемы. Правило состоит из двух частей: *антицедента* (условия), который является аналогом условия в if-then операторе и записывается *слева,* и *консеквента* (заключения), который является аналогом then части этого оператора и записывается *справа.*

3.Логический вывод

CLIPS включает язык представления порождающих правил и язык описания процедур. Основными компонентами языка описания правил являются база фактов (fact base) и база правил (rule base). На них возлагаются следующие функции:

- база фактов представляет собой исходное состояние проблемы;
- база правил содержит операторы, которые преобразуют состояние проблемы, приводя его к решению.

Машина логического вывода CLIPS сопоставляет эти факты и правила и выясняет, какие из правил можно активизировать. Это выполняется циклически, причем каждый цикл состоит из трех шагов:

1. сопоставление фактов и правил;
2. выбор правила, подлежащего активизации;
3. выполнение действий, предписанных правилом.

Такой трехшаговый циклический процесс иногда называют «циклом распознавание – действие»

4.Программирование в CLIPS

- 4.1.Основные элементы программирования.

  CLIPS предоставляет три основных элемента для написания программ:

  - простые типы данных;
  - функции для манипулирования данными;
  - конструкции для пополнения базы знаний.

Простые типы данных

- Для представления информации в CLIPS предусмотрено восемь простых типов данных: float, integer, symbol, string, external-address, instance-name и instance-address.
- Для представления числовой информации используются типы float и integer, символьной - symbol и string.

  При записи числа могут использоваться только цифры (0-9), десятичная точка (.), знак (+) или (-) и (e) при экспоненциальном представлении. Число сохраняется либо как целое, либо как действительное. Любое число, состоящее только из цифр, перед которыми может стоять знак, сохраняется как целое.

Функции

- Под функцией в CLIPS понимается фрагмент исполняемого кода, с которым связано уникальное имя и который возвращает полезное значение или имеет полезный побочный эффект (например, вывод информации на экран).
- Существует несколько типов функций. Пользовательские и системные функции - это фрагменты кода, написанные на внешних языках (например, на C) и связанные со средой CLIPS. Системными называются те функции, которые были определены изначально внутри среды CLIPS.
- Хотя CLIPS и не ориентирована на вычислительные операции, в ней предусмотрен ряд стандартных арифметических и математических функций.

Функции

- Конструкция deffunction позволяет пользователю определять новые функции непосредственно в среде CLIPS с использованием синтаксиса CLIPS. Функции, определенные таким образом, выглядят и работают подобно остальным функциям, однако они выполняются не напрямую, а интерпретируются средой CLIPS.
- Вызовы функций в CLIPS имеют префиксную форму: аргументы функции могут стоять только после ее названия. Вызов функции начинается с открывающейся скобки, за которой следует имя функции, затем идут аргументы, каждый из которых отделен одним или несколькими пробелами. Аргументами функции могут быть данные простых типов, переменные или вызовы других функций. В конце вызова ставится закрывающаяся скобка.

Конструкции

- В CLIPS существует несколько описывающих конструкций:
  defmodule, defrule, deffacts, deftemplate, def global, deffunction, defclass, definstances, defmessage-handler, defgeneric.
- При записи все они заключаются в скобки. Определение конструкции отличается от вызова функции главным образом по производимому эффекту. Обычно вызов функции оставляет состояние среды CLIPS без изменений (за рядом исключений, когда речь идет о функциях сброса, очистки, открытия файла и т.п.). Определение конструкции, напротив, в точности направлено на изменение состояния среды путем внесения изменений в базу знаний CLIPS. В отличие от функций конструкции никогда не возвращают значений.
- Все конструкции (за исключением defglobal) позволяют размещать комментарии сразу вслед за именем конструкции. Кроме того, комментарии могут вставляться в код CLIPS при помощи точки с запятой (;). Все. что следует за (;) до конца строки, будет игнорироваться CLIPS. Если (;) стоит первым символом в строке, то вся строка считается комментарием.

4.2.Факты

- Факты являются одной из основных форм представления информации в системе CLIPS. Каждый *факт* представляет фрагмент информации, который был помещен в текущий список фактов, называемый fact-list. Факт представляет собой основную единицу данных, используемую правилами.
- Количество фактов в списке и объем информации, который может быть сохранен в факте, ограничивается только размером памяти компьютера. Если при добавлении нового факта к списку обнаруживается, что он полностью совпадает с одним из уже включенных в список фактов, то эта операция игнорируется (хотя такое поведение можно изменить).

4.2.Факты

- Факт может описываться *индексом* или *адресом.* Всякий раз, когда факт добавляется (изменяется), ему присваивается уникальный целочисленный индекс. Индексы фактов начинаются с нуля и для каждого нового или измененного факта увеличиваются на единицу. Каждый раз после выполнения команд reset и clear выделение индексов начинается с нуля. Факт также может задаваться при помощи адреса. Адрес факта может быть получен путем сохранения возвращаемого значения команд, которые возвращают в качестве результата адрес факта (таких как assert, modify и duplicate), или путем связывания переменной с адресом факта в левой части правила (см. далее).
- *Идентификатор* факта - это короткая запись для отображения факта на экране. Она состоит из символа f и записанного через тире индекса факта. Например, запись f-10 служит для обозначения факта с индексом 10. Существует два формата представления фактов: позиционный и непозиционный.

Позиционные факты

- Позиционные факты состоят из выражения символьного типа, за которым следует последовательность (возможно, пустая) из полей, разделенных пробелами. Вся запись заключается в скобки. Обычно первое поле определяет "отношение", которое применяется к оставшимся полям. Например:

  (the pump is on)

  (altitude is 10000 feet)

  (grocery_list bread milk eggs)

- Поля в позиционных фактах могут быть любого простого типа (за исключением первого поля, которое всегда должно быть типа symbol), на порядок полей также не накладывается никаких ограничений. Следующие символьные выражения зарезервированы и не должны использоваться как первое поле любого факта (позиционного или нет): *test, and, or, not, declare, logical, object, exists u forall.*

Непозиционные факты

- Для того чтобы обратиться к информации, содержащейся в позиционном факте, пользователь должен знать, не только какие данные содержатся в факте, но и то, в каком поле они хранятся. Непозиционные (шаблонные) факты дают возможность пользователю абстрагироваться от структуры факта, задавая имена каждому из полей факта. Для задания шаблона, который затем может использоваться при доступе к полям по именам, используется конструкция deftemplate. Эта конструкция подобна структуре или записи в языках программирования С и Паскаль.
- Конструкция deftemplate позволяет наряду с определением *именованных полей,* или *слотов,* вводить имя шаблона. В отличие от позиционных фактов слоты шаблонного факта могут быть ограничены по типу, значению, числовому диапазону. Кроме того, для любого слота можно определить значения по умолчанию. Слот состоит из открывающейся скобки, за которой следует имя слота, полей (могут отсутствовать) и закрывающейся скобки. Заметим, что слоты не могут использоваться в позиционных фактах, так же как позиционные поля не могут использоваться в шаблонных фактах.

Манипулирование фактами

- Факты могут добавляться к списку фактов (с помощью команды assert), удаляться из него (с помощью команды retract), изменяться (с помощью команды modify) и дублироваться (с помощью команды duplicate) самим пользователем или программой. Например:

  (assert (light green))
- Кроме того, конструкция deffacts позволяет определить множество исходных или априорных знаний в виде набора фактов. Например:

  (deffacts walk "Some facts about, walking"
  (status walking)
  (walk-sign walk))
- Когда производится сброс состояния среды CLIPS (с помощью команды reset), все факты, описанные в конструкции deffacts, добавляются к списку фактов. Кроме того, по этой команде в список фактов заносится *исходный факт* (initial-fact). Этот факт включается в список фактов всегда с идентификатором f-0. Его назначение будет рассмотрено в следующем пункте.

4.3.Правила

- Одним из основных методов представления знаний в CLIPS являются правила. Правила используются для представления эвристик, определяющих ряд действий, которые необходимо выполнить в определенной ситуации. Разработчик экспертной системы определяет совокупность правил, которые используются совместно для решения проблемы. Правило состоит из двух частей: *антицедента* (условия), который является аналогом условия в if-then операторе и записывается *слева,* и *консеквента* (заключения), который является аналогом then части этого оператора и записывается *справа.*
- Правая часть правила представляет собой совокупность действий, которые должны быть выполнены, если правило применимо. Действия, описанные в применимых правилах, выполняются тогда, когда блок вывода CLIPS получает команду начать выполнение применимых правил. Если существует множество применимых правил, то для того, чтобы выбрать правило, действия которого должны быть выполнены, блок вывода использует *стратегию разрешения конфликтов.*

4.3.Правила

- Левая часть правила представляет собой ряд *условий (условных элементов*), которые должны выполняться, чтобы правило было применимо. В CLIPS принято считать, что условие выполняется, если соответствующий ему факт присутствует в списке фактов. Одним из типов условных элементов может быть *образец.* Образцы состоят из набора ограничений, которые используются для. описания того, какие факты удовлетворяют условию, определяемому образцом. Процесс сопоставления фактов и образцов выполняется блоком вывода CLIPS, который автоматически сопоставляет образцы, исходя из текущего состояния списка фактов, и определяет, какие из правил являются применимым». Если все условия правила выполняются. то оно *активируется* и помещается в *список активированных правил.*
- Если левая часть правила пуста, то для его активации необходимо наличие в списке фактов исходного факта (initial- fact). Такие безусловные правила часто используются для того, чтобы инициировать работу программы. Поэтому перед запуском таких программ необходимо произвести сброс состояния среды CLIPS.

4.3.Переменные

- Как и в других языках программирования, в CLIPS для хранения значений используются переменные. В отличие от фактов, которые являются статическими, или неизменными, содержание переменной динамично и изменяется по мере того, как изменяется присвоенное ей значение.
- Идентификатор переменной всегда начинается с вопросительного знака, за которым следует ее имя. В общем случае формат переменной выглядит следующим образом:
- ?<variable-name>
- Примеры переменных:
- ?x ?sensor ?noun ?color
- Перед использованием переменной ей необходимо присвоить значение. Все переменные, кроме глобальных, считаются локальными и могут использоваться только в рамках описания конструкции. К этим локальным переменным можно обращаться внутри описания, но они не определены вне него.
- Для определения глобальных переменных, которые видны всюду в среде CLIPS, используется конструкция defglobal.

4.4.Дополнительные средства

- CLIPS предоставляет ряд дополнительных средств, необходимых при написании программ. Основными из них являются: ограничения на значения полей; оператор проверки условия test; использование функций в правилах; использование процедурных знаний.

5.Интерфейс CLIPS

- Оболочка ЭС CLIPS может работать в нескольких режимах:

  1. Интерактивно, с использованием простого текстового интерфейса командной строки;

  2. Интерактивно, с использованием GUI-интерфейса;

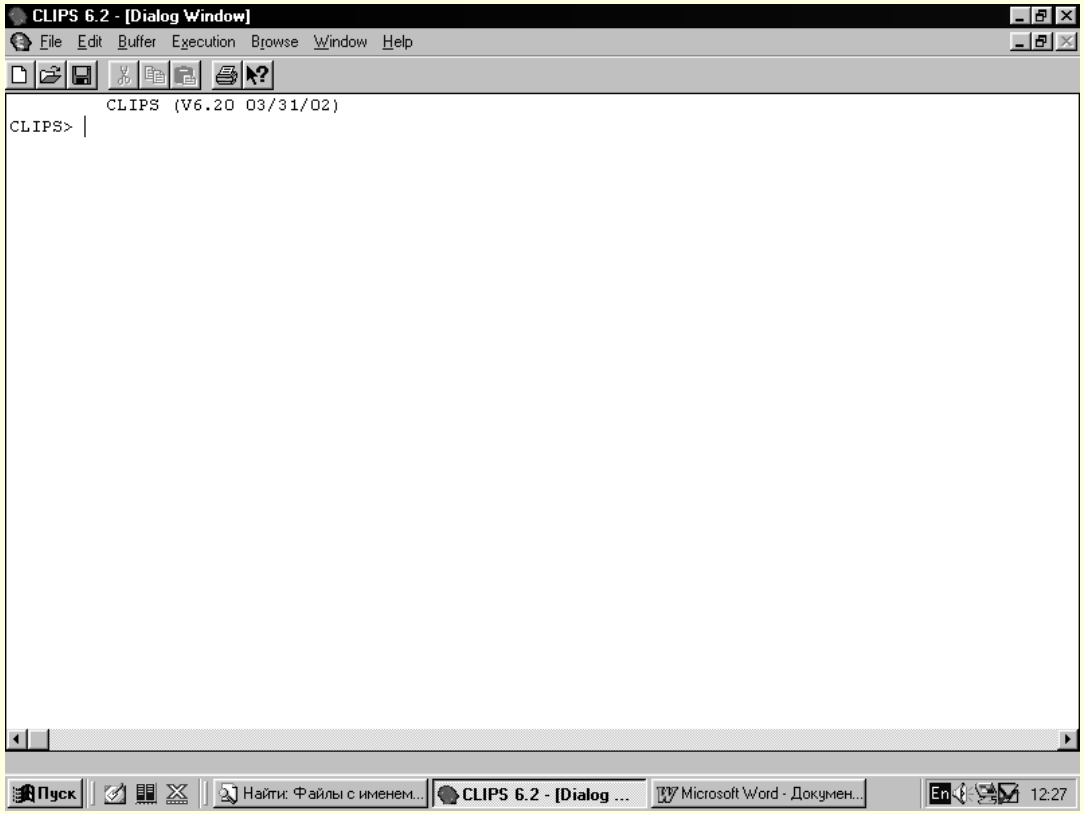
  3. Как ЭС, интегрированная в другие приложения.

5.1.Интерфейс командной строки

- Основным методом взаимодействия пользователя с CLIPS является ввод команд с командной строки CLIPS. После появления на экране подсказки
- “CLIPS>
- Командами могут быть вызовы функций, конструкции, глобальные переменные или константы. Если ввести вызов функции, вычисляется значение этой функции и на экран выводится результат. Как уже отмечалось, вызовы функций в CLIPS имеют префиксную форму, т.е. аргументы функции могут стоять только после ее названия. Если ввести определение, то будет создана конструкция соответствующего типа. В ответ на ввод глобальной переменной на экран будет выведено ее значение. За вводом константы последует вывод ее на экран.
- Стандартная процедура использования интерфейса командной строки выглядит следующим образом:
- 1) создать и редактировать базу знаний при помощи любого текстового редактора;
- 2) сохранить базу знаний в одном или нескольких текстовых файлах:
- 3) выйти из редактора и запустить CLIPS;
- 4) загрузить базу знаний в CLIPS.

6.Запуск CLIPS

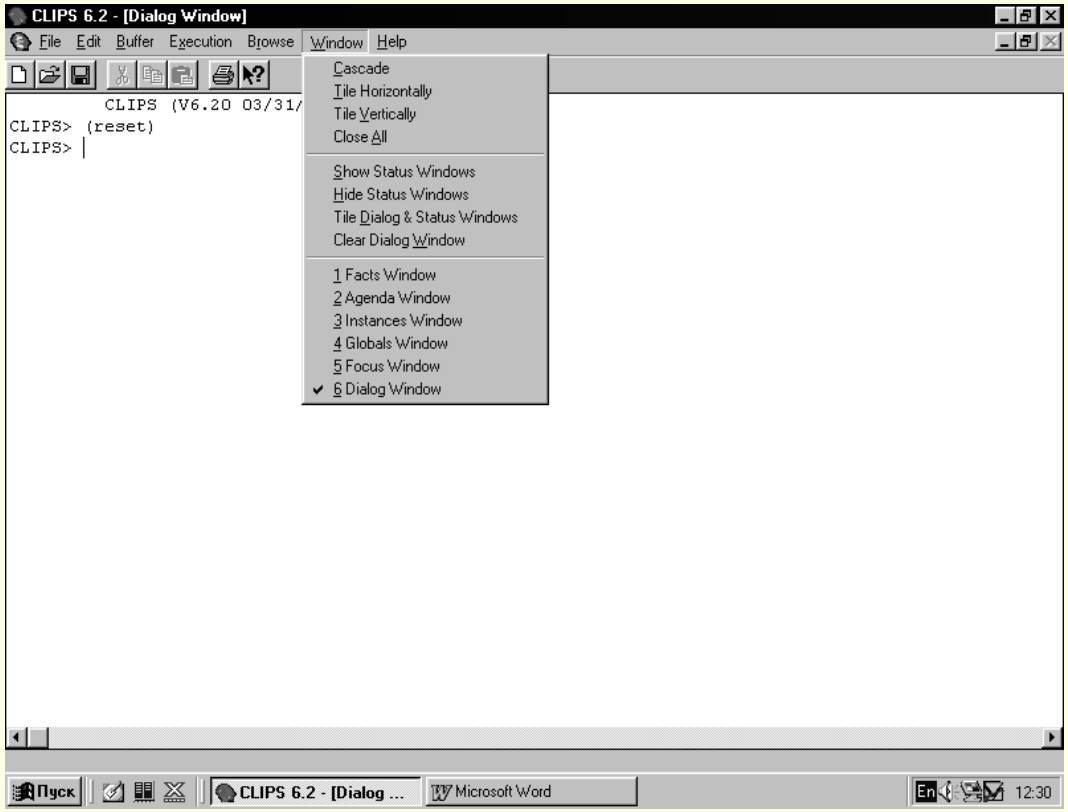
- Система CLIPS 6.1 реализована в виде исполняемого файла clipswin.exe, предназначенного для работы в операционных системах Windows 95 и Windows NT 4.0. После запуска на экране появится диалоговое окно, представленное на рис.



6.Запуск CLIPS

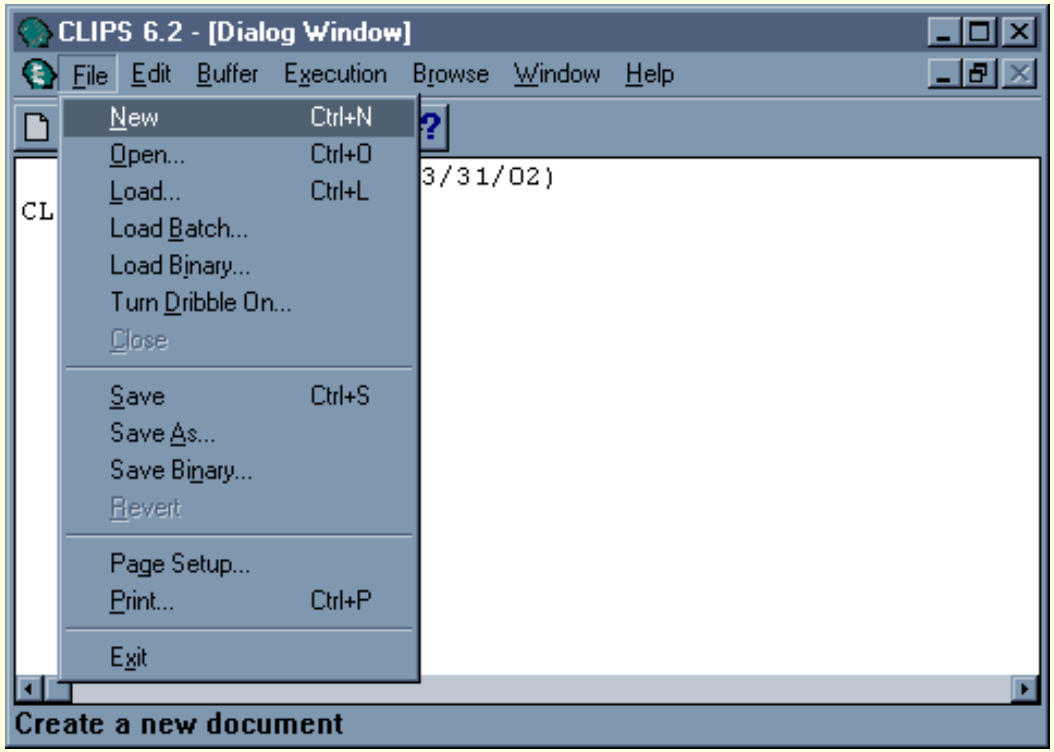
- Для того чтобы иметь возможность наблюдать за всеми изменениями, происходящими в состоянии CLIPS, выполните команду All Above (рис.).

  Данная команда открывает все окна. Окно Facts содержит факты из списка фактов. Окно Agenda содержит все правила из списка активных правил.

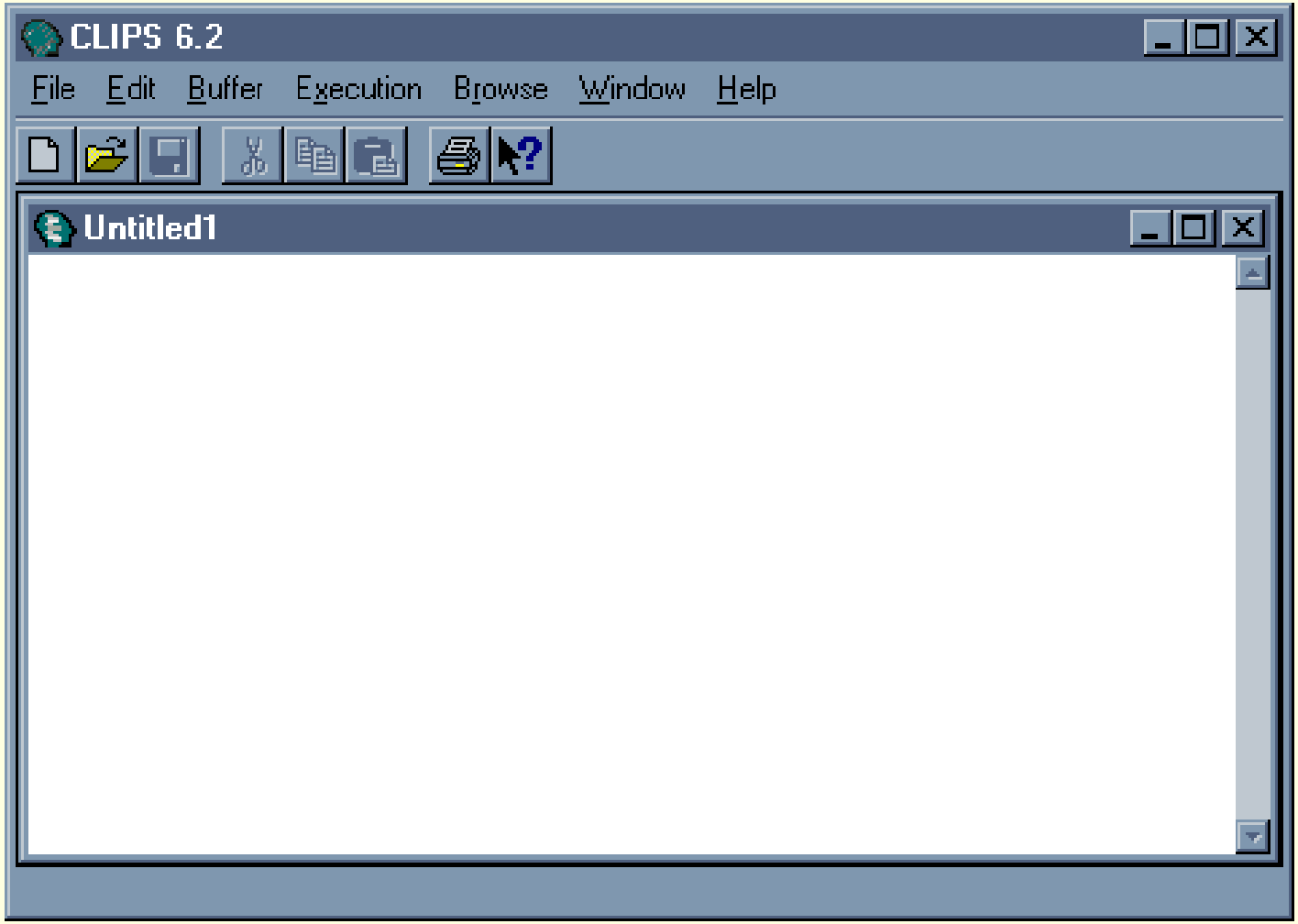


7.Ввод программы

- Ввести программу в CLIPS можно непосредственно из диалогового окна, появившегося после запуска. Но в этом случае все написанные правила после закрытия CLIPS будут потеряны. Чтобы этого не происходило, необходимо' сохранить текст программы в каком-либо файле. Для редактирования файлов в CLIPS имеется встроенный редактор. Запустить редактор можно, как показано на рис.

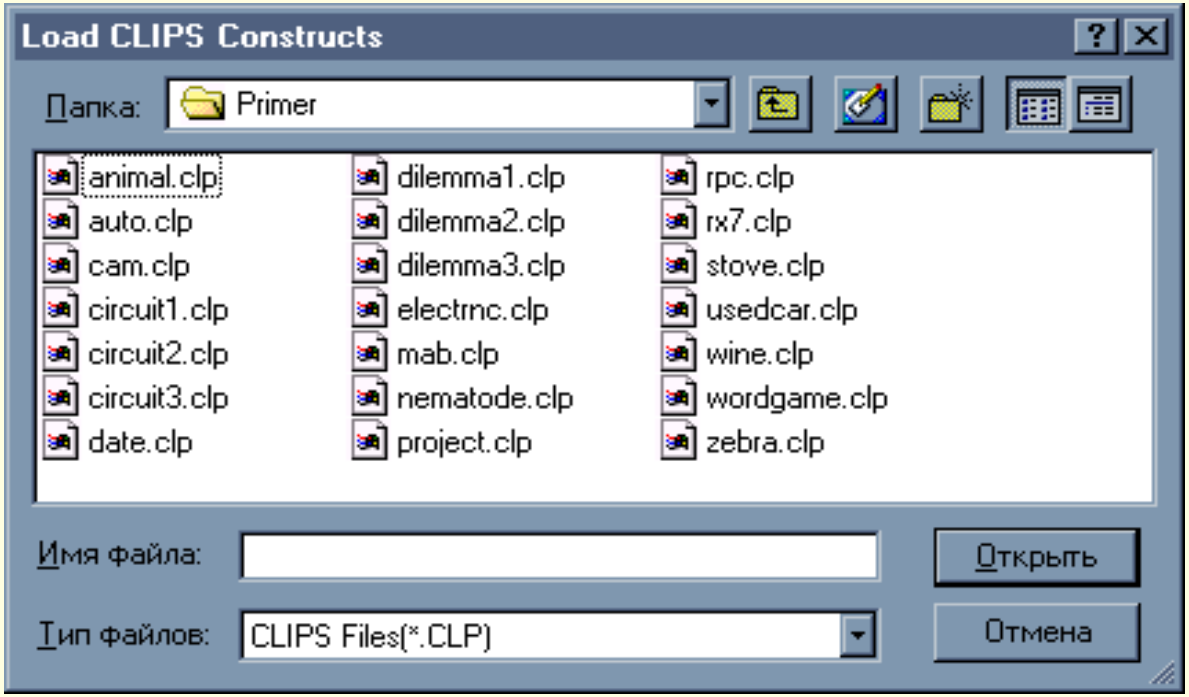


Внешний вид редактора



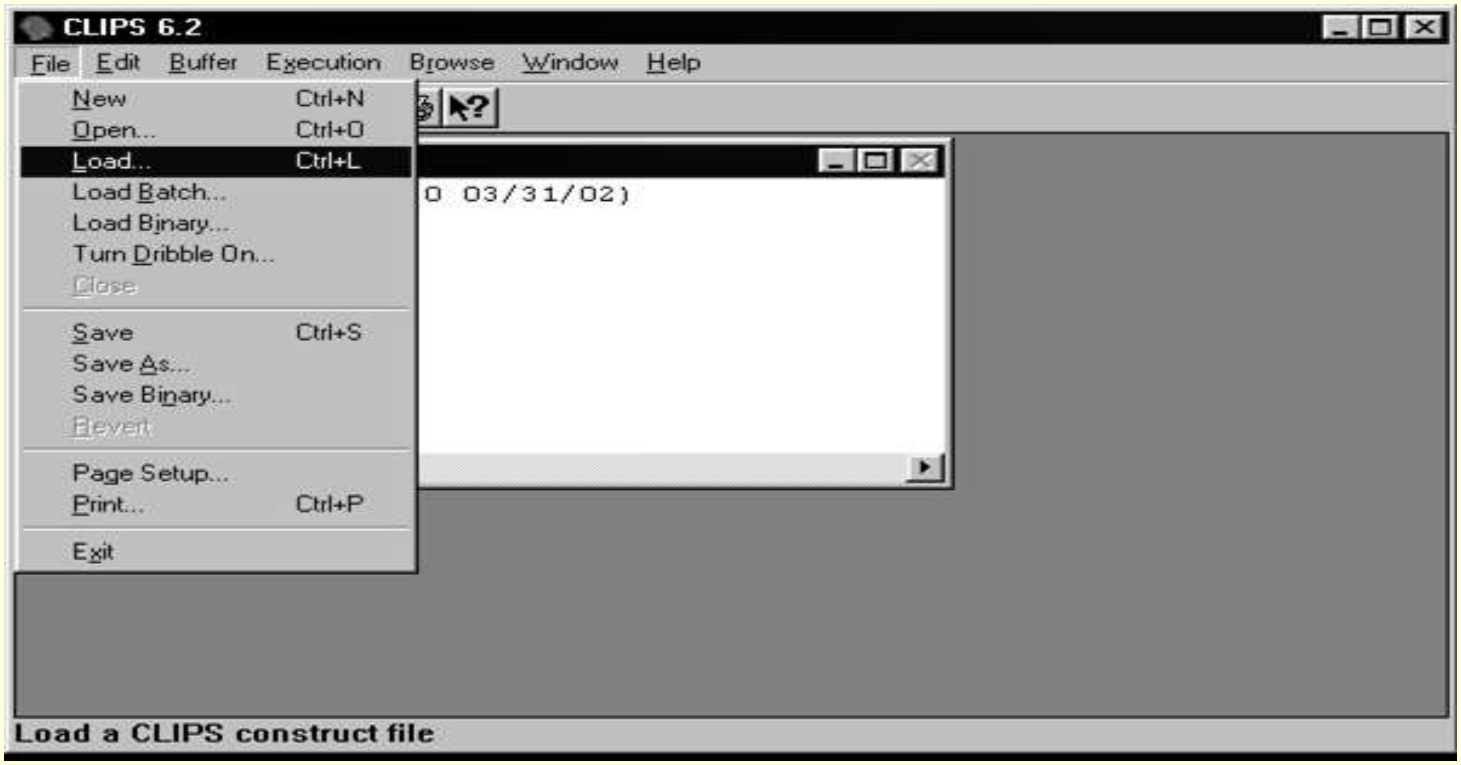
Выбор файла с конструкциями.

- В появившемся затем диалоговом окне необходимо выбрать этот файл и нажать кнопку Open. Выбор файла с конструкциями.



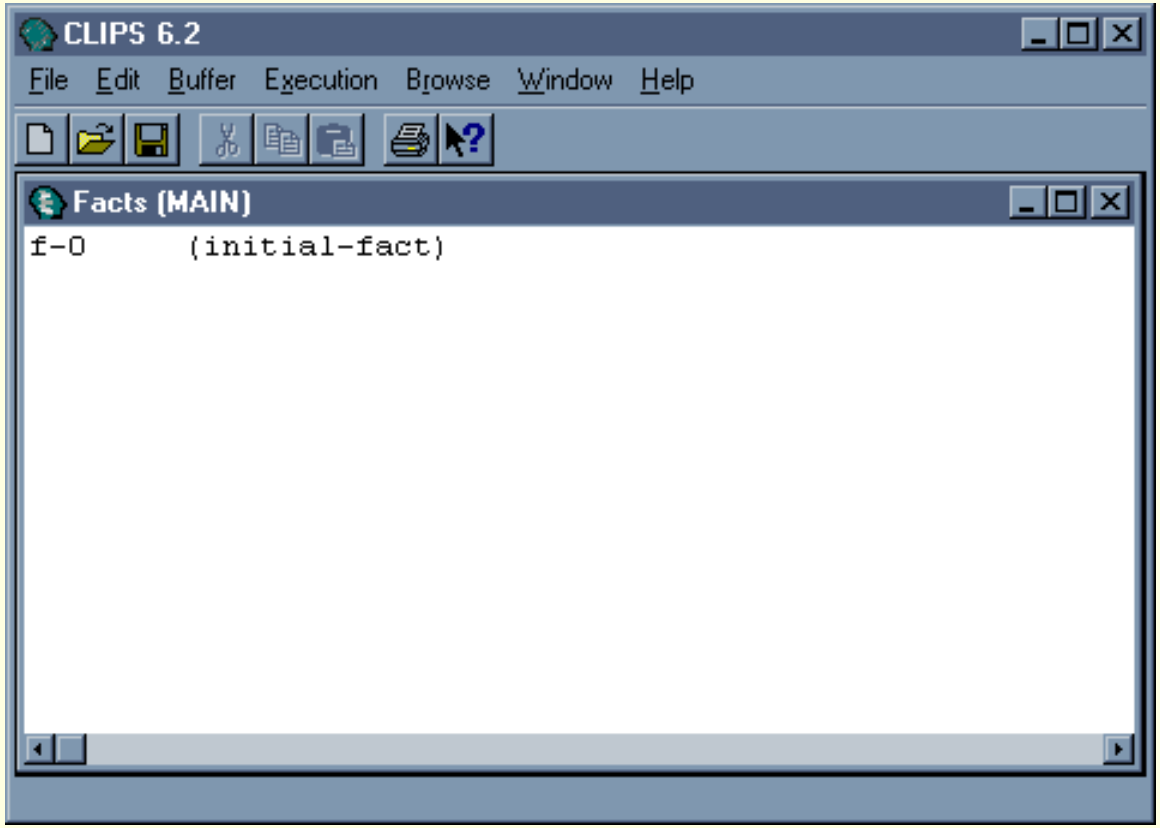
Загрузка и запуск программы

- Чтобы загрузить в базу знаний CLIPS содержимое файла wordgame.clp, нужно воспользоваться пунктом Load Constructs. Загрузка конструкций из файла.



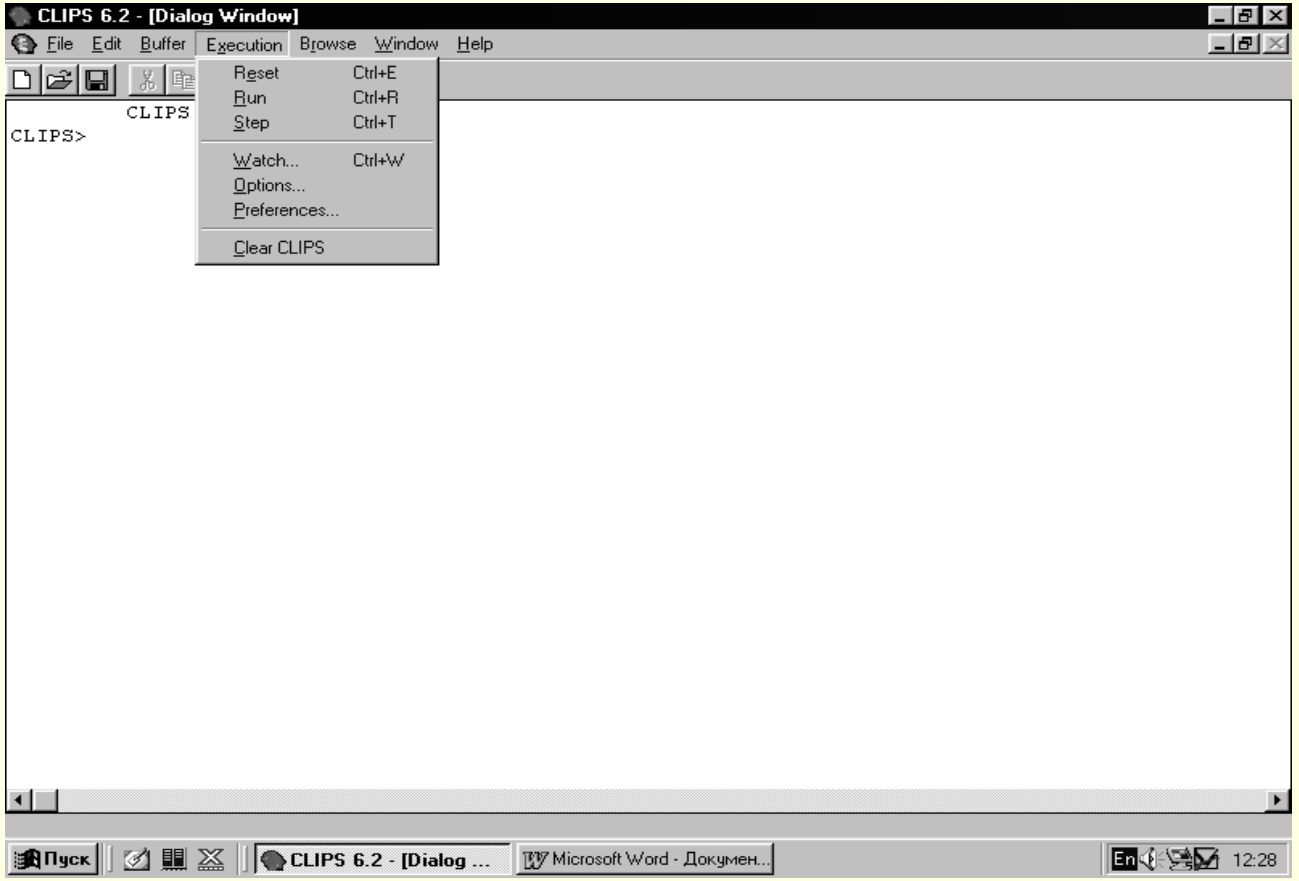
Установка начального факта

- Данная команда удаляет существующие факты из списка фактов. Включает в список фактов исходный факт (initial-fact). Включает в список фактов все Факты, описанные в конструкциях (deffacts). После выполнения команды окно Facts – факты из списка фактов – будет выглядеть следующим образом.



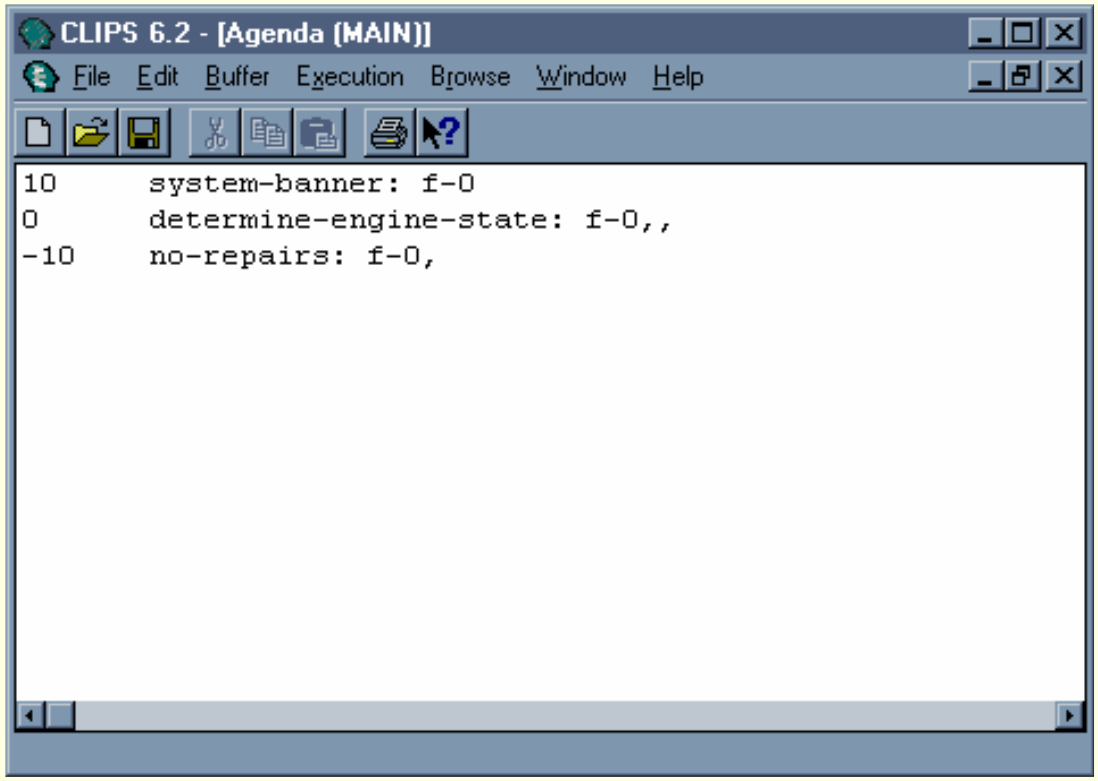
Очистка CLIPS

- В появившемся затем диалоговом окне необходимо выбрать этот файл и нажать кнопку Open. Выбор файла с конструкциями.

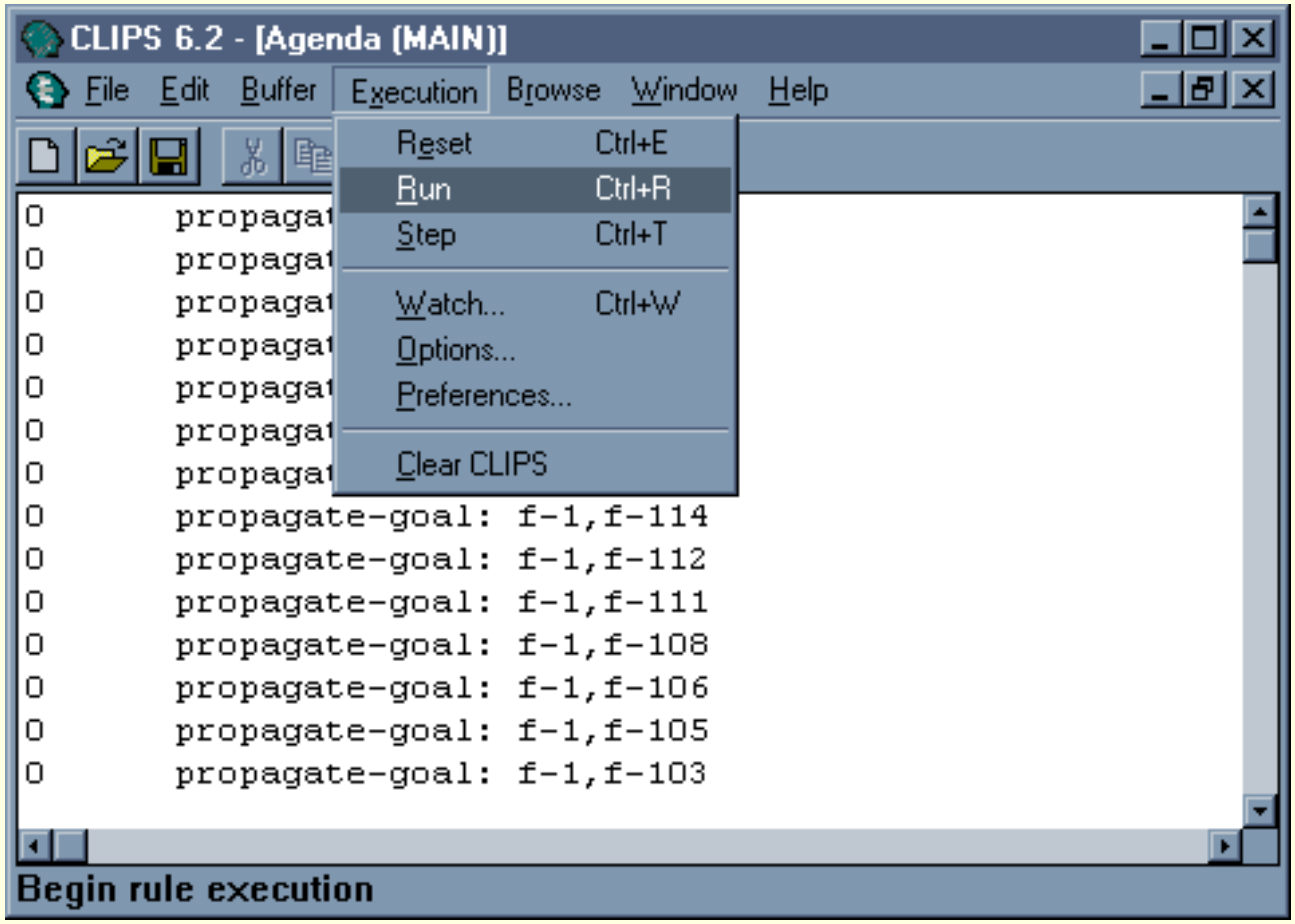


Список активных правил.

- Данный факт активизирует правило, не содержащее условий, и это правило будет помещено в список активных правил - Agenda. В этом окне показано, что в списке активных правил есть правило с именем startup и это правило было активизировано фактом с идентификатором f-0.

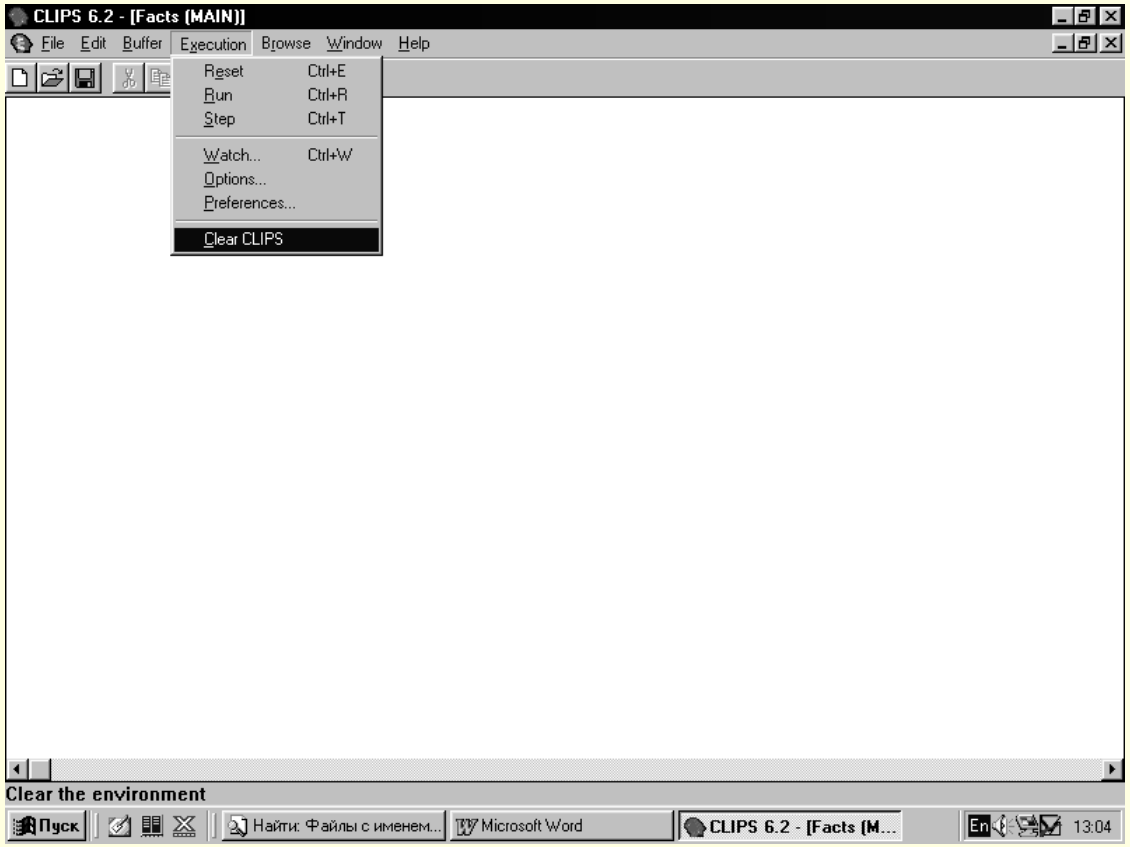


Запуск программы.



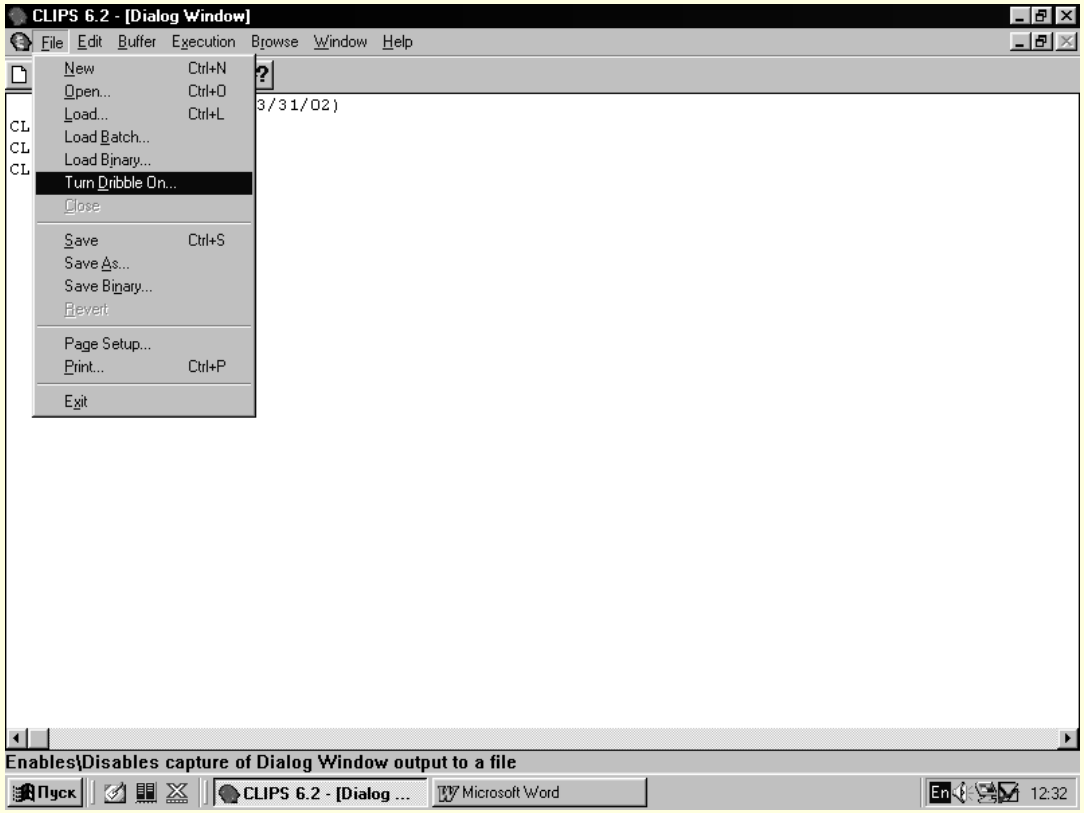
Очистка CLIPS

- Если после выполнения программы необходимо очистить базу знаний CLIPS, а также убрать все факты из списка фактов, т.е. привести CLIPS в начальное состояние, то необходимо воспользоваться командой (clear)



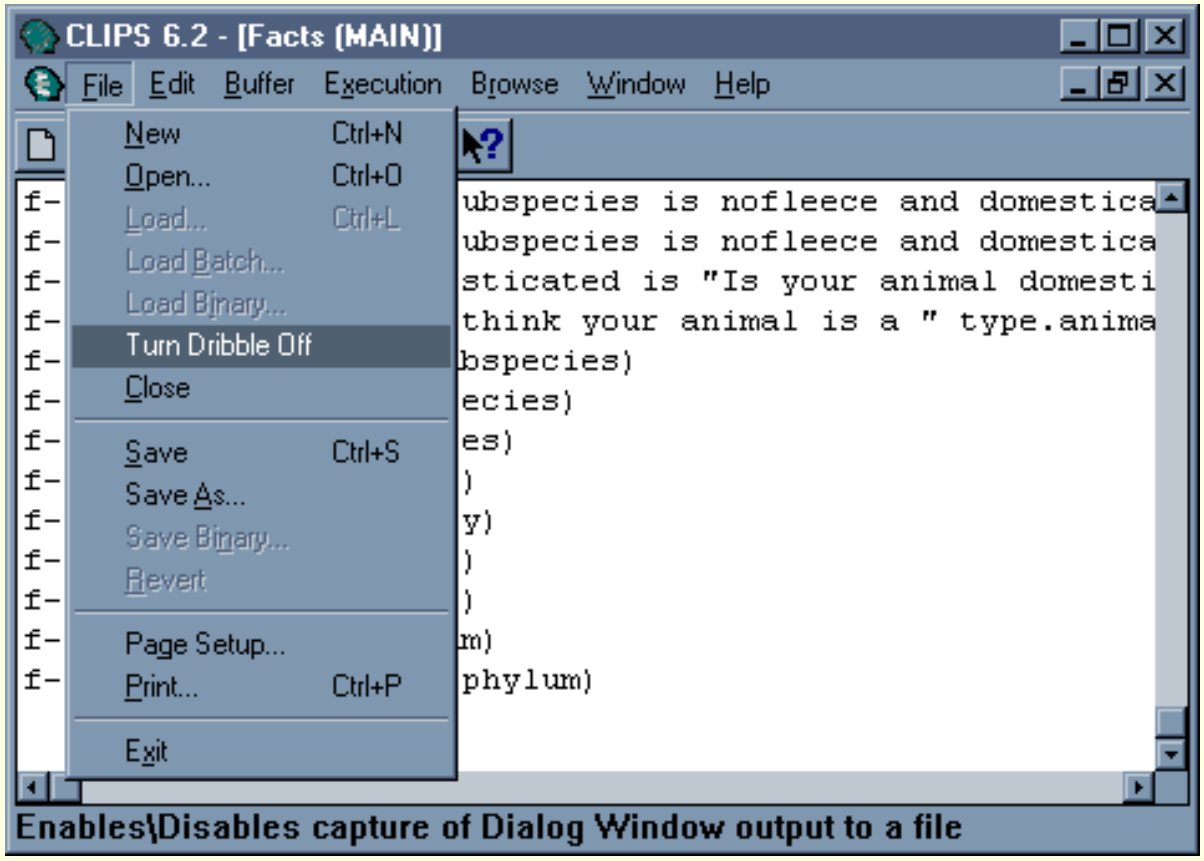
Сохранение протокола работы

- Для сохранения протокола работы программы, а также полученного ответа в текстовом файле необходимо сразу после запуска CLIPS выполнить команду Turn Dribble On.



Остановка записи протокола работы

- После получения ответа, перед очисткой CLIPS выполните команду Turn Dribble Off. По этой команде файл, в который записывается содержимое главного диалогового окна CLIPS, будет закрыт.



Инструментальные средства логического программирования.

3.1.Язык логического программирования Пролог .

Пролог – это декларативный язык, в среде которого необходимо точно и грамотно на логическом языке описать условие задач, а ее выполнение – это результат работы некоторого рутинного процесса, выполняемого интерпретатором. Фрагменты Пролог-программы иногда называют базами знаний. Далее рассмотрение языка осуществляется для версии Турбо Пролог.

Пролог реализует управляющую структуру в виде обратной цепочки логического вывода, то есть доказательство от противного. Этим частично исключается монотонность.

Составляющие Пролог-программы.

В самом общем виде Пролог-программа может быть представлена как совокупность двух основных разделов.

Первый раздел содержит постановку задачи, т.е. описание ее условия в виде набора исходных данных, представленных в форме БД.

Второй раздел описывает алгоритм, т.е. порядок выполнения программы, и представляет собой совокупность утверждений (формул), содержащих факты, правила и вопросы, необходимые для достижения поставленной цели.

Таким образом, в структуре программы выделяются следующие виды утверждений: факты, вопросы (запросы) и правила.

Простейший вид утверждений Пролог-программы – это факт. Он констатирует наличие отношений между объектами или переменными. Отношение в программе называется предикатом.

Вопросы в Пролог-программе имеют тот же формат, что и факты. Их можно различить по контексту, расположению на экране в режиме отладки программы или по наличию знака «?» в начале или в конце утверждения (в зависимости от диалекта интерпретатора). Семантически вопрос в Пролог-программе определяет наличие того или иного отношения между объектами.

Пример вопроса в Пролог-программе.

Ищет(“Надя”, “улицу”).

Ищет(“Надя”, “дом”).

? Ищет(“Надя”, X).

X = “дом”.

Правила предназначены для вывода логического следствия на основе имеющейся БД.

Пример правила в Пролог - программе.

Дано: “все ученики получают оценку”, “Иванов - отличник”.

Первую фразу можно формализовать следующим образом: «для всех X, X получают оценку, если X – ученик».

Запишем исходные данные в терминах Пролог-программы:

- estimation (X) :- schooler (X).
- schooler (Ivanov).
- ?- estimation(Ivanov).
- в результате получим ответ: yes.

3.2.Основные разделы программы.

Описание данных присутствует в двух основных разделах Пролог-программы, один из которых называется predicates (отношения), а другой – clauses (утверждения). В разделе predicates описываются предикаты, используемые в БД, с описанием их типа. Для определения основной цели программы служит ключевое слово goal. В раздел clauses заносятся факты и правила, известные априорно. О содержимом этого раздела можно говорить как о данных, необходимых для работы программы.

Пример.

```
predicates
        dog (symbol)
        a toy (symbol)
goal
        some (goals).
clauses
        dog (2).
        a toy (bunny).
```

Заполнение разделов программы.

Такие разделы Пролог-программы как trace, project (название проекта) и include (внешний исходный файл) заполняется по желанию.

Раздел domains, объявляющий новые типы области данных, может быть заполнен по желанию или необходимости.

Пример.

```
domains
    dog = symbol
    a toy = symbol
```

Раздел database объявляет предикаты, которые по желанию можно добавлять или удалять.

Пример.

```
database
    work (dog, a toy)
```

Раздел global predicates или predicates объявляет области определения каждого предиката или глобального предиката в обязательном порядке.

Логические термы.

Совокупность синтаксических структур в Пролог-программе называется логическими термами. Термы могут иметь простую или сложную структуры.

Константы в Пролог-программе могут быть числом или атомом. Атом представляет собой последовательность строчных и прописных символов, цифр и специальных знаков, которая обязательно начинается со строчной буквы. Если атом в программе начинается с прописной буквы, то его заключают в апострофы (двойные кавычки).

Пример.

name (dog, «Рекс")

Основными называются термы, не содержащие переменных, т.е. факты. Термы, содержащие переменные называются неосновными.

Сложный терм является в Пролог-программе или функтором, или оператором. Определение сложного терма в программе содержит имя, представляющее собой атом, и совокупность аргументов, которые в свою очередь также могут быть простыми и сложными термами.

Переменные в Пролог-программе.

Переменные в Пролог-программе являются не местом в памяти, а реальным объектом. Место действия одной переменной – одна резольвента. При осуществлении механизма возврата значения всех переменных в резольвенте, через которые осуществляется возврат, очищаются.

Переменная в Пролог-программе представляет собой последовательность знаков подчеркивания, строчных и прописных букв, которая всегда начинается с прописной буквы или со знака подчеркивания.

Наличие символа «__» означает использование фиктивной переменной. В рассмотренном примере с ее помощью реализован квантор общности.

В разделе данных domains объявлены типы данных пользователя («тип фигуры» и «выражение для площади»), а также указано, что «тип фигуры» эквивалентен стандартному символьному типу (symbol), а тип «выражение для площади» – стандартному строковому типу (string).

3.3.Рекурсивные вычисления в Пролог-программе.

Рекурсивное описание правила содержит в своем теле ссылку на заголовок этого же правила. Возможны следующие три варианта рекурсивных правил:

- правая рекурсия −*pr*1( ) :- *pr*11( ), *pr*12( ), ... , *pr*1N( ), *pr*1( ).
- левая рекурсия − *pr*1( ) :- *pr*1(1), *pr*21( ), *pr*22( ), ... , *pr*2M( ).
- обобщенная рекурсия −
  *pr*1( ) :- *pr*11( ), *pr*12( ), ... , *pr*1N( ), *pr*1( ), *pr*21( ), *pr*22( ), … , *pr*2M( ).

Для исключения зацикливания во время выполнения рекурсивного правила необходимо предусмотреть условия завершения рекурсии, которое обычно реализуется одним из двух способов:

- заданием в программе альтернативного правила или факта *pr*1( ), не содержащего рекурсии (выход происходит при успешном выполнении этого правила);
- формированием условия выхода одним из предикатов *pr*11( ), *pr*12( )…, - выход происходит если в процессе выполнения правилах хотя бы один из предикатов завершается неуспехом.

Пример. Возведение в степень x^n

```
class predicates
power: (real, integer, real) procedure(i,i,o).
clauses
power (_,0,1):-!
power (X,N,R):-M=N-1, power (X,M,R1), R=R1*N.
run():-console::init(), power(srdio::read(),stdio::read(),R),stdio::write(R).
```

3.4. Процесс реализации вывода.

Абстрактный интерпретатор Пролог выполняет вычисления, получая вопрос *P* и последовательность резольвент *Q*. Если *P* выводится из *Q*, то результат «yes», если не выводится, то – «no». Возможен также и третий вариант, когда результат непредсказуем.

В процессе реализации логического вывода в Пролог-программе используются следующие механизмы:

- унификации;
- сопоставления (согласования, совпадения);
- прямой трассировки;
- возврата.

Сопоставление – это процесс проверки двух термов на их соответствие друг другу. Термы сопоставимы в двух случаях: если они идентичны, либо, если они становятся идентичными при конкретизации своих переменных.

Если термы тождественны (идентичны), то процесс терпит успех, и неуспех – в противном случае.

Механизм возврата.

Процесс согласования конъюнкций подцелевых утверждений в теле резольвенты осуществляется слева направо:

(голова) цель :– (тело) подцель_1, подцель_2, ... , подцель_N.

Каждая подцель согласуется с фактом, хранящимся в БД, или уже доказанным утверждением. Однако может возникнуть ситуация, когда одна из подцелей не может быть согласована. При этом осуществляется сдвиг влево на одну подцель предшествующую и выполняется её сопоставление с новыми значениями из БД, и так до тех пор, пока не будет найдена сопоставимая подцель. Если она найдена, то процесс продолжается дальше слева направо до тех пор, пока не будет доказано целевое утверждение.

Если подцелей больше нет и факты в БД исчерпаны, то доказательства заканчиваются неудачей. При осуществлении механизма возврата значения всех переменных очищаются.

Пример механизма возврата.

```
predicates
        short (symbol)
        middle (symbol)
        long (symbol)
        strong (symbol)
        powerful (symbol)
goal
        powerful human.
clauses
        short (Dima).
        short (Maksim).
        middle (Nadya).
        middle (Sasha).
        big (Kostya).
        big (Lesha).
        strong (Denis).
        powerful (human):- middle (human), strong (human).
        powerful (human):- big (human).
```

Шаги выполнения программы на языке Пролог.

Итак, рассмотрим самый общий алгоритм работы Пролог-программы.

1. Нахождение подходящего правила для текущей цели (или подцели).

2. Передача и согласование параметров цели и правила.

3. Определение в зависимости от исхода шага 2 текущей (новой) цели или возврат к предыдущей цели с отменой значений для переменных, полученных в результате выполнения шага 2 для последней цели.

4. Создание точек ветвления программы с помощью предикатов repeat, fail и «!».

5. Инициализация переменных процедуры и восстановление их значений при возврате.

3.5. Предикаты.

Системные предикаты.

В программе на языке Пролог используются следующие группы системных предикатов:

• предикаты типа;

• арифметические и логические предикаты;

• предикаты ввода-вывода;

• предикаты, влияющие на ход выполнения программы (отсечения, вызов пользователя);

• металогические предикаты (работа с термами, классификация термов);

• предикаты управления и связи с операционной системой;

• предикаты отладки программы.

Предикаты типа позволяют задать новую переменную определенного типа, а также проверить принадлежность переменной к тому или иному типу.

Арифметические предикаты.

В Пролог-программе справедливы и используются все виды арифметических операций:

1)+ − сложение; 2) - − вычитание; 3) * − умножение; 4) / − деление;

5) exp − возведение в степень.

Кроме этого имеются другие встроенные математические операции:

1) mod− модуль, остаток от целочисленного деления; 2) div− частное от деления;

3) sin− синус; 4) cos− косинус; 5) ln − натуральный логарифм.

Предикат унификации.

Термин унификация [лат. unio − единство + facere − делать] − означает приведение чего-либо к единой норме, форме, единообразию. Унификация в программе на языке Пролог – это операция приведения к единому значению. Следует особо отметить, что предикат «=» не является оператором присваивания, хотя и может выполнять его функцию в частном случае.

Пример.

```
domains
        i = integer
predicates
        t(i,i)
clauses
        t(1,2).
```

Предикаты отношений.

В Пролог-программе (версия Турбо-Пролог 2.0) используются следующие предикаты отношений:

< – меньше;

> – больше;

<= – меньше или равно;

>= – больше или равно;

= – равно;

>< или <> – не равно.

Предикат ввода-вывода.

Предикат ввода read используется в Пролог-программе в следующих нотациях:

- readln (var) – ввод строковой переменной;
- readint (var) – ввод целой переменной;
- readreal (var)– ввод вещественной переменной;
- readchar (var) – ввод символьной переменной,

Предикат форматного вывода: writef (format, arg1, ... , argN)

Предикат закрытия окна имеет вид: removewindow (W, Z), где, W – номер окна;

Пример предиката ввода-вывода.

Пример.

```
predicates
        run (char)
goal
        run (X).
clauses
        run (X) :- makewindow (1, 2, 3, "Привет", 0, 0, 25, 80),
        write ("Здравствуйте!"),
        readchar (X),
        removewindow.
```

Предикаты управления работой программы

В процессе работы Пролог-программы осуществляется:

- нахождение правила для текущей цели (подцели);
- конкретизация переменных данного правила по аксиомам;
- возврат, если сопоставимых аксиом нет;
- ветвление программы;
- означивание переменных и дальнейшее сопоставление правил с аксиомами.

Предикат цикла.

Предикат repeat предназначен для организации циклических операций, он является недетерминированным и задается в программе следующим образом:

```
predicates
        nondeterm repeat
clauses
        repeat :- repeat.
        repeat.
```

Предикаты отсечения.

Для реализации ветвления программы используются два предиката отсечений – fail и cut. Предикат cut, чаще обозначаемый в программе знаком «!», предназначен для исключения возможного перебора всех вариантов решений.

Пример.

```
PREDICATES
играет(имя, вид_сп) спис_спортс
CLAUSES
играет("Саша", теннис).
играет("Аня", волейбол).
играет("Олег", футбол).
играет("Коля", теннис).
играет("Саша", футбол).
играет("Андрей", теннис).
спис_спортс: - играет(X, теннис), !, играет(Y, теннис), X<>Y, write(X, "-", Y), nl, fail.
GOAL
write("Пары теннисистов"),
nl, спис_спортс.
```

3.6. Списковые структуры.

Пролог работает с такими структурами данных как бинарные деревья и структурные списки, причем списки являются частным случаем бинарных деревьев.

Для выполнения операций, связанных с обработкой текстов, в таких языках программирования как LISP необходимо наличие операции следования. В Пролог-программе процедура следования задаётся в виде списковой структуры данных.

Список – представляет собой упорядоченный набор объектов (эле-ментов списка), следующих друг за другом. Элементы списка должны принадлежать к одному и тому же доменному типу. В теле программы список задается с помощью символа «*».

Список в Пролог-программе – это совокупность термов, разделенных запятыми и заключенных квадратные в скобки.

Для повышения наглядности программ в Прологе предусматриваются специальные средства списковой нотации, позволяющие представлять списки не только в традиционном виде [элемент_1, элемент_2, ... , элемент_N], но и в бинарной форме: [H | T] , где H (Head) – это голова списка, T (Tail) – хвост или окончание списка, а знак «|» служит разделителем.

Пример поиска элемента в списке.

["Лена", "Петр", "Олег", "Сергей"];

[1, 2, 3, 6, 9, 3, 4];

[3.2, 4.6, 1.1, 2.64, 100.2];

[yesterday, today, tomorrow];

[element01] – список из одного элемента;

[ ] – пустой список;

G = graf ([a, b, c, d], [r(a, b), r(b, c), r(b, d), r(d, c)]) – описание графа в виде совокупности списков его вершин и дуг (ребер).

3.7. Вызов внешних функций из Пролог-программы и интерфейс с программами на других языках программирования.

Вызов внешних функций из Пролог-программы продемонстрируем на примере реализации интерфейса с программой, написанной на языке программирования Borland C.

Компиляция программы из среды Turbo-Prolog.

Основная программа на языке Турбо-Пролог [3] компилируется из среды в OBJ-файл. Во всех глобальных именах должен стоять предварительный символ подчеркивания. Описание внешней функции, которая будет вызвана из основной программы, имеет вид:

_имя_функции() – language c.

Получение исполнимого файла

Первый этап процесса получения исполнимого файла представляет собой компиляцию исходного текста программы, отличающуюся от работы интерпретатора тем, что в ее результате создается объектный файл. Для рассматриваемого примера − это файлы Пролога hello_pr.obj и hello_pr.sym, а также файл на языке Borland C − hello_c.obj. Компилятор выполняет преобразование исходной программы в программу на языке машинных команд.

Следующий этап − это компоновка, т.е. собственно создание исполнимого файла, заключающийся в объединении в общем коде всех необходимых объектных файлов и подстыковке требуемых библиотек. В рассматриваемом примере объектные модули на языках Пролог и Borland C включаются в исполнимый файл outfile.exe.

В дистрибутиве Пролог роль компоновщика выполняет специальная программа связи или так называемая TLINK-программа.

Вызов TLINK-программы из командной строки ИС Турбо-Пролог для рассматриваемого примера имеет следующий вид:

Tlink init.obj hello_pr.obj hello_c.obj hello_pr.sym, outfile,, prolog

где init.obj – стандартный файл Турбо-Пролог,

hello_pr.obj, hello_pr.sym – файлы, получаемые после компиляции hello_pr.pro,

outfile.exe – результирующий исполнимый файл.

3.9. Диалекты и языки, используемые для задач искусственного интеллекта.

За более чем полувековую историю проблемы искусственного интеллекта были разработаны и используются в настоящее время следующие диалекты языка Пролог: C-Prolog, Quintus Prolog, SKW-Prolog, Arity Prolog, SWI-Prolog и др.

Для обработки проблемных областей, связанных с математикой и описываемых с помощью формул и логических выражений, предназначен язык FORLOG.

Кроме того, хорошим языком программирования задач ИИ зарекомендовал себя язык LISP (LISt Processing), разработанный для символьных вычислений и модифицированный применительно к потребностям разработки интеллектуальных систем. LISP – функциональный язык, синтаксис и семантика которого обусловлены теорией рекурсивных вычислений.

В языке LISP осуществляется обработка списков, аргументами которыми могут быть объекты, их свойства и функции. В свою очередь каждую функцию можно представить списком, среди аргументов которого имеется другая функция. Работа формализма ориентирована на обработку функций. Управляющая структура языка LISP находит среди аргументов списка описание состояния объекта или ситуации и выполняет процедуру, включенную в этот список. Если процедура содержит список, то он сопоставляется с текущей ситуацией. Управляющая структура языка LISP ориентирована на сопоставление с образцом, а модель – на исчисление предикатов.