

Документ подписан простой электронной подписью
Информация о владельце:

ФИО: Макушев Андрей Евгеньевич

Должность: Ректор

Дата подписания: 04.10.2021 14:15:02

Уникальный программный ключ:

4c46f2d9ddda3fafb9e57683d11e5a4257b6ddfe

09.04.01-УММ-М.1.2.5-ПИС-ЛКО1- Исторический обзор применения искусственного интеллекта. Обзор современных приложений искусственного интеллекта.

Прикладные интеллектуальные системы

Конспект лекции 1. Исторический обзор применения искусственного интеллекта. Обзор современных приложений искусственного интеллекта.

Аннотация: В лекции рассказывается история появления и применения систем искусственного интеллекта. Рассмотрены направления развития и применения современных интеллектуальных систем.

План лекции:

1. Понятие искусственного интеллекта.
2. Три волны искусственного интеллекта
3. Направления ИИ
4. Что достигнуто в ИИ к настоящему времени?
5. Почему вокруг ИИ такой ажиотаж (ожидания)
6. Рынок ИИ
7. Основные международные программы и центры разработки
8. Состояние работ по ИИ в РФ
9. Основные направления работ в области ИИ
10. Тенденции использования систем ИИ в области государственного управления
11. Ключевые вызовы и угрозы развития систем ИИ
12. Задания для самостоятельной работы.

Ключевые слова: системы искусственного интеллекта, интеллектуальные системы, интеллектуальные технологии

СОДЕРЖАНИЕ ЛЕКЦИИ:

1. Понятие искусственного интеллекта

Искусственный интеллект – что это такое? Наиболее сложной сущностью в мире является человек. Стремление познать сложность человека – существовало всегда и проявлялось в разных формах. Сейчас это стало еще более актуально. Развитие компьютерных технологий привело к появлению множества работ, связанных с попытками научиться распознавать и синтезировать человеческую речь, создать системы технического зрения, которые могут опознавать лица людей не хуже, а уже лучше, чем глаза человека, научить автомашины ездить самостоятельно без водителя человека и т.д. Системы, проявляющие поведение, свойственное человеку, называются системами искусственного интеллекта (ИИ). На самом деле изучение этих систем – самостоятельное научное направление, объединяющее многие научные дисциплины.

Согласно одному из определений, искусственный интеллект – это наука и технология, включающая набор средств, позволяющих компьютеру на основании накопленных знаний давать ответы на вопросы и делать на базе этого экспертные выводы, т.е. получать знания, которые в него не закладывались разработчиками. Наука под названием «искусственный интеллект» входит в комплекс компьютерных наук, а создаваемые на ее основе технологии относятся к информационным технологиям. Есть множество других определений, менее устойчивых к критике.

Соответственно, системы ИИ определяют как компьютерные системы, использующие в своей работе технологии ИИ. При этом в большинстве случаев до получения результата неизвестен алгоритм решения задачи.

Системы ИИ условно делятся на два класса – сильный (или общий) ИИ и слабый (или прикладной) ИИ. Определим сильный, или универсальный, искусственный интеллект как ИИ, сравнимый с человеческим, т.е. ИИ, который может учиться, как это делают люди, и не уступает по уровню развития большинству людей, а во многих смыслах даже превосходит их. Есть множество более строгих определений, но для понимания данного определения достаточно.

Все остальные системы, в том числе системы ИИ, которые окружают нас сейчас, называются слабым ИИ, поскольку они могут делать только одно дело, например осуществлять поиск по запросам в Интернете, ставить диагноз по конкретному заболеванию и т.д. Помощь такого ИИ делает жизнь более комфортной, а работу – более производительной. Такие системы в ближайшем будущем будут все больше совершенствоваться, уже сейчас многие конкретные виды работ системы с ИИ делают лучше, чем люди. Следует отметить, что в процессе работы над проектами со слабым ИИ проводится громадная подготовительная работа. Машины учат таким интеллектуальным занятиям, как поиск информации, распознавание речи, обработка естественного языка, распознавание лиц, логический вывод и др. По отдельности это всего лишь мощные инструменты, но они быстро развиваются, год от года продвигая технологии ИИ вперед и приближая создание сильного ИИ.

В связи со слабым ИИ следует упомянуть о так называемом эффекте ИИ (AI Effect). Он заключается в том, что как только с помощью ИИ реально достигается немислимый ранее результат, то такую задачу критики перестают считать задачей ИИ, т.е. де-вальвируют ее значение. Этот эффект сформулирован в формуле Ларри Теслера: «ИИ – это все, что не сделано до сих пор».

Для обеспечения систем ИИ ученым пришлось заняться таким вопросом, как представление знаний, – это позволило создать так называемые экспертные системы (ЭС, системы, которые на основе баз знаний помогают в принятии решений); очень важными стали методы самообучения машин (появились интеллектуальные обучающие системы); попытки повторить работу нервной системы человека. Последнее вылилось в создание искусственных нейронных систем (ИНН). Таким образом, в основе всех исследований по ИИ лежит идея моделирования процессов человеческого мышления с помощью компьютера. Сам ИИ как наука относится к когнитивным наукам, т.е. к наукам, связанным с приобретением (сбором, накоплением, восприятием) знаний.

Предполагается, что ИИ, сравнимый с интеллектом человека, будет иметь неограниченную сферу применения и кардинально изменит наше существование.

В январе 2016 г. основатель Всемирного экономического форума в Давосе Клаус Шваб назвал искусственный интеллект одной из основных движущих сил четвертой промышленной революции. «Эта четвертая промышленная революция идет на нас, как цунами, а ее основной движущей силой являются достижения в области искусственного интеллекта, робототехники, нанотехнологий, “интернета вещей” и других областей науки».

Сейчас происходит качественный переход от вычислительной эры к эре когнитивной (в терминах футурологов, Second Machine Age), когда компьютеры нового типа быстро учатся работать со структурированными, неструктурированными и нечетко структурированными данными, начинают замещать труд людей при решении большого количества когнитивных задач [2].

2. Три волны искусственного интеллекта

ИИ имеет уже более чем полувековую историю, в которой были как пики интереса к нему, так и периоды почти полного его исчезновения.

Первые работы по ИИ относятся к 1950-м годам. Начались они с попыток решения двух задач:

- 1) создание программы для игры в шахматы (в 1954 г. аналитики корпорации RAND А. Ньюэлл, Дж. Шоу и Г. Саймон начали писать программу игры в шахматы. Помочь им вызвались А. Тьюринг и К. Шеннон, а также группа голландских психологов. В 1957 г. шахматная программа (NSS) была написана. В основе ее работы лежали эвристики, т.е. правила выбора решения в отсутствие теоретических оснований [23]);

- 2) создание программ машинного перевода с одного естественного языка на другой. В СССР первые экспериментальные системы перевода с английского и китайских языков разрабатывались в 1954–1957 гг. на ЭВМ БЭСМ-2 в ИТМиВТ под

руководством Л.Н. Королёва. В 1954 г. в корпорации IBM под руководством профессора Леона Достерта перевели на английский язык 60 русских фраз на основе словаря, состоящего из 250 пар слов, и шести правил грамматики. Результаты казались много- обещающими, но проблема оказалась намного сложнее, чем предпо- лагалось. Суть перевода заключалась в обучении компьютеров не только правилам, но и исключениям, а технические средства компьютеров того времени никак не позволяли реализовать эту задачу. Тем не менее по- пытки создания таких систем дали мощный толчок развитию математической лингвистики. К этому же периоду относится и появление первых программ для игры в шахматы.

Третьей областью, которая зародилось в то время, а впоследствии получила большое развитие, стало автоматическое доказательство теорем. В 1960 г. появилась программа, которую называли «Универсальным решателем задач» (GPS), которая позволяла автоматически доказывать теоремы из планиметрии, находить решения алгебраических задач и др. [23].

Среди множества работ по ИИ первой волны следует отметить создание в 1963 г. Джоном Маккарти первого языка для программирования задач ИИ – языка ЛИСП. Появление этого языка открыло функциональное программирование. Заметим, что первые языки высокого уровня (ЯВУ) того времени были процедурными.

Вторая волна ИИ, начавшаяся с конца 60-х годов, была связана с появлением логического программирования (язык Prolog, 1971 г.) и бумом вокруг так называемых экспертных систем (ЭС). Это тоже были зачатки искусственного интеллекта, но в ЭС специалист по управлению знаниями, опрашивая экспертов в предметной области, вручную наполнял базу знаний (БЗ), а машина могла делать логический вывод в рамках того «понимания», которое человек в нее заложил, т.е. полностью отсутствовал такой важный элемент, как самообучение. Кроме того, возникали проблемы с экспертами, которые не делились своими знаниями или переставали это делать, как только понимали, что внедрение ЭС понизит их профессиональный статус, поскольку любой начинающий специалист с помощью ЭС может добиться высоких результатов. Следует отметить, что создание ЭС породило большой интерес к проблеме представления знаний в компьютерных системах. В это время появились семантические сети, системы фреймов, продукционные системы (системы, основанные на правилах) и их комбинации.

Между тем работы над системами машинного перевода текстов продолжались и существенно продвинулись. Развитие таких систем стало особенно актуальной задачей во времена холодной войны, когда в США поступало огромное количество материалов на русском языке, но не хватало человеческих ресурсов для их быстрого перевода. «В 1990-х годах в проекте компании IBM Candide был задействован десятилетний опыт переводов стенограмм заседаний канадского парламента, опубликованных на французском и английском языках, – около 3 млн предложений. Поскольку это официальные документы, их переводы были выполнены с соблюдением чрезвычайно высоких требований. По меркам того времени

количество данных было огромным. Эта технология, получившая известность как “статистический машинный перевод”, превратила задачу перевода в одну большую математическую задачу» [18]. Однако повышение качества машинного перевода на этом застопорилось.

Ко второй волне относится также создание продвинутых программ для игры в шашки и шахматы. Прошли первые чемпионаты мира по игре машин в шахматы между собой. Здесь следует отметить победу на чемпионате мира в 1974 г. советской шахматной программы «Каисса» (М. Донской, А. Арлазаров, А. Битман, А. Усков). Успех «Каиссы» оказался мировой сенсацией, поскольку все предрекали победу американской программе. Как вспоминал М.В. Донской, «Каисса» играла в силу второго шахматного разряда, т.е. до программ, которые обыгрывают гроссмейстеров, было еще далеко.

Системы первой и второй волны получили шуточное название «старый добрый ИИ», или «символьный ИИ». В целом они были основаны на формальной логике, которая хорошо применима для формализуемых задач, типа логических игр, но в ней трудно представить системы реального мира.

Нынешнее возрождение интереса к ИИ – уже третье по счету и отличается от предыдущих как амплитудой, так и охватом, поскольку сейчас для решения задач ИИ имеются как необходимые технические средства, повсеместно распространившиеся беспроводные сети, Интернет, так и далеко продвинувшиеся работы в этой области. Начало третьей волне положила знаменитая победа в матче из шести партий американской программы «Дип Блю» (2,5:3,5) над чемпионом мира по шахматам среди людей Гарри Каспаровым.

В программировании в начале третьей волны появилось так называемое генетическое программирование. Оно позволяет, имитируя процесс мутаций, работающий в биологических системах, решать определенные классы задач, например при поиске оптимальных решений.

В области ИИ действует своя мода на то или иное направление. Очень много внимания одно время уделялось программным агентам. Интеллектуальный программный агент – это активная программная или программно-аппаратная система (например, робот), обладающая автономностью в выполнении той функции (или набора функций), для которой она создана. Основные проблемы: коммуникация интеллектуальных агентов, разработка языков для этой цели, координация поведения агентов, распределение ролей в коалициях агентов, коллективное поведение агентов [23].

Для современного этапа характерно очень быстрое развитие технологий искусственных нейронных сетей (ИНС) – сетей, имитирующих работу биологических нейронов живых существ. Простейшая ИНС состоит из трех слоев искусственных нейронов – на первый слой поступают сигналы из внешнего мира, во внутреннем слое они обрабатываются и передаются в выходной слой, в котором формируется результат. Внутренних, скрытых слоев может быть много.

Развиваются на базе новых технологий также и системы, появившиеся во время первых волн развития ИИ, – системы машинного перевода, достигшие вполне приемлемого качества, ЭС и др. В 2006 г. появилась система Google Переводчик, основанная на Больших данных. Корпус из триллиона слов, выпущенный Google в 2006 г., состоял из разбросанных фрагментов интернет-контента. Он стал «обучающим набором», по которому вычислялась вероятность того, что именно последует за тем или иным английским словом. В систему можно добавлять новые слова, которые появляются в естественном языке, и удалять устаревшие. «Ее переводы точнее, хотя и весьма далеки от совершенства: к середине 2012 г. она охватила более 60 языков, а теперь даже способна принимать голосовой ввод на 14 языках для моментального перевода» [18].

Один из вопросов к современной ИИ состоит в том, завершится ли эта третья волна очередным разочарованием и снижением интереса к ИИ, или в результате нынешних колоссальных усилий мирового научного сообщества будет достигнут необратимый прогресс в данной области.

По мнению автора, ставки на ИИ сейчас очень высоки и это направление будет интенсивно развиваться.

3. Направления ИИ

Научная дисциплина «Искусственный интеллект» – зонтичная. Внутри ИИ делится на множество направлений. Вот основные из них:

- Представление знаний.
- Доказательство теорем.
- Компьютерное зрение.
- Машинное обучение (приобретение знаний, анализ данных и порождение гипотез).
- Робототехника.
- Обработка естественных языков.
- Многоагентные системы.
- Инструментальные средства ИИ.

Каждая из перечисленных ветвей ИИ в свою очередь делится на десятки других направлений, особенно робототехника (несколько десятков направлений).

Как видим, из-за обширности ИИ ученому невозможно охватить все его ветви, требуется специализация.

4. Что достигнуто в ИИк настоящему времени?

1. Громадное количество научных работ по ИИ посвящено компьютерному зрению. Это направление ИИ связано с развитием глубинного обучения (о нем ниже). Впервые компьютеры стали способны выполнять некоторые визуальные задачи классификации лучше, чем люди. Например, заявленная точность

назначения оптимального лечения раковых заболеваний легких у компьютера IBM Watson составляет 90%, т.е. превышает на 40% качество диагностики, проводимой врачами-онкологами.

2. Важным понятием в ИИ является «машинное обучение» (его называют также статистическим обучением). Основу данной технологии в 1959 г. заложил Артур Самюэль, когда предложил работать над обучением компьютеров, не используя определенно запрограммированные алгоритмы. В простейшем смысле программа обучается, когда в ней происходит изменение, позволяющее во второй раз выполнить определенное задание лучше.

Машинное обучение – это технология, в рамках которой создается база обучающих примеров, по которой компьютер или нейросеть настраивается (обучается) и затем может правильно распознавать и классифицировать поступающие новые данные, т.е. это совокупность алгоритмов и методов, позволяющих научить компьютеры делать выводы на основании имеющихся данных. Добавление обучающих примеров позволяет улучшить результаты распознавания. Таким образом происходит как бы самообучение программы. По этой технологии по большой базе фотографий компьютер научили распознавать лица, причем он делает это точнее, чем человек. Настоящий прорыв в обучении машин произошёл в начале 2016 г., когда программа Google AlphaGo сумела обыграть в игру го ее абсолютного чемпиона Ли Седоля. Эта игра является наиболее интеллектуально сложной игрой в мире, намного сложнее шахмат (в го доска 19 × 19 клеток и возможных позиций намного больше, чем в шахматах), в которой для победы необходимо не просто перебирать всевозможные ходы. Добиться победы в го над ее чемпионом позволила технология «глубинного машинного обучения» (deep learning, DL), которая сейчас является самым трендовым направлением развития искусственного интеллекта. Этот термин применяется к искусственным нейронным сетям (ИНС), где используется больше одного скрытого слоя, поэтому формально «глубинный» указывает еще и на более многослойную архитектуру нейронной сети (см. с. 134). Уникальным для глубинного обучения является то, что машина сама находит признаки (ключевые черты чего-либо, по которым легче всего отделить один класс объектов от другого) и структурирует их иерархично: из более простых складываются более сложные. У термина «глубинное обучение» нет формального определения, поскольку он объединяет целую группу различных технологий [29]. Таким образом, компьютер учится на примерах и своем собственном опыте. Программа AlphaGo сначала проанализировала 29,4 млн ходов в 160 тыс. партий профессиональных игроков, а затем две копии программы начали играть одна с другой, добавляя новые партии в обучающую выборку. Сыграв миллионы партий, программа научилась оценивать наиболее выгодное положение камней на доске для достижения победы [12].

Технология глубинного обучения сейчас является неотъемлемой частью исследований в области распознавания речи, изображений, при создании систем управления беспилотными автомобилями, диагностике заболеваний и решении других сложных задач. Развитием технологии глубинного обучения стала реализованная IBM летом 2017 г. технология распределенного глубинного обучения (DDL), позволяющая на порядок сократить время обучения искусственной нейронной

сети.

Следует отметить, чего не может современный ИИ и что отделяет его от общего ИИ:

- отсутствует запоминание ранее приобретенных навыков при обучении новым;

- ИИ не может при обучении новым навыкам опираться на ранее приобретенные, т.е. отсутствует обобщение накопленных знаний и использование их в разных контекстах.

Массовое распространение смартфонов породило широкое использование речевых помощников, в которых реализуются элементы ИИ. Такие приложения помогают пользователю в его повседневной деятельности. Среди них такие известные приложения, как Siri (компания Apple), Cortana (Microsoft), Google Now (Google), Echo (Amazon), «Алиса» (Яндекс) и др., которым уже пользуются десятки миллионов людей. Данные приложения реализуются также на планшетах, ноутбуках и персональных компьютерах. Со временем эти программы станут все интеллектуальнее и незаменимее. Важным направлением работ по ИИ является выявление структуры мозга человека. Такие проекты весьма дорогостоящие, и потому их реализацию могут позволить себе немногие страны и гигантские корпорации.

Анализ структуры мозга – это обратное проектирование, которое предполагает, что сначала нужно разобраться до тонкостей в человеческом мозге, а затем представить то, что мозг делает, в виде аппаратного и программного обеспечения. В итоге ученые надеются создать компьютер, обладающий ИИ человеческого уровня. Несколько громадных проектов (с инвестициями в миллиарды долларов) нацелено на достижение этого результата. Моделированию человеческого мозга посвящен международный проект *Human Brain Project (HBP)*, который ведет команда из швейцарской Федеральной политехнической школы в Лозанне под руководством профессора Генри Маркрама (Henry Markram) и в котором участвует более 100 научных групп со всего мира. Цель проекта – синтезировать все знания, полученные людьми о мозге, в единую полноценную модель мозга внутри суперкомпьютера. Завершение проекта предполагается в 2023 г. Американский проект *Brain Activity Map Project* («Карта активности мозга», 2013 г.) рассчитан на то, за 10 лет американским ученым удастся зафиксировать и картографировать активность каждого нейрона в человеческом мозге. Есть еще около десятка менее емких, но не менее важных проектов по изучению мозга. Весьма интересен проект *Blue Brain*, базирующийся в Швейцарии, в котором ученые изучают работу ансамблей нейронов. Проект *SyNAPSE*, финансируемый DARPA и корпорацией IBM, ставит задачу создания физической копии мозга, воплощенной в виде специальных микросхем с искусственными нейронами. Это направление получило название нейроморфная электроника [1]. Аналогичные проекты развивает Китай. Тем не менее в настоящий момент исчерпывающее моделирование мозга невозможно в силу ограниченных возможностей современных суперкомпьютеров (для этого требуется производительность в десятки эксафлопс, т.е. на 3–4 порядка больше нынешней).

Из других глобальных проектов, в которых предполагается использовать методы ИИ и системную аналитику, следует отметить продолжение проекта «Геном человека», в котором планируется определить ДНК-последовательности всех живых существ планеты. Например, Amazon Third Way разрабатывает проект под названием «Банк кодов Земли» [16].

5. Почему вокруг ИИ такой ажиотаж (ожидания)

1. Проводившаяся более полувека компьютеризация производства и практически всех аспектов деятельности человека породила некоторый тупик, связанный с обработкой и анализом всего растущего объема ежедневно появляющихся данных. По ходу борьбы с ними появились хранилища данных, оперативный анализ данных, облачные вычисления, Большие данные (Big Data). А с другой стороны – новые модные технологии, которые будут генерировать на порядки большие потоки данных, – Интернет вещей, Индустрия 4.0, Общество 5.0 и т.п. Развитие систем ИИ – это стремление перенести высокоуровневую обработку накопленных данных с человека на компьютерные системы, транзисторные и / или нейроморфные. Мировые ИТ-лидеры ведут гонку в создании специализированных процессоров и суперкомпьютеров для обучения нейронных сетей. В идеале их обучение должно происходить в реальном времени, но сейчас это занимает недели.

2. Эра компьютеров как двигателя полупроводниковой индустрии, заканчивается. Сейчас все надежды на то, что таким двигателем станут ИИ и робототехника. Здесь уже сформировались громадные по объемам сегменты: промышленная, сервисная и военная робототехника, беспилотные транспортные средства, медицинская робототехника и др. Однако без систем ИИ полноценное развитие этих направлений невозможно. Таким образом, робототехника сейчас сильно стимулирует исследования в области ИИ. Например, появление беспилотных транспортных средств – это потенциально бизнес в триллионы долларов, поэтому все автомобильные гиганты и другие корпорации вкладывают миллиарды долларов в разработку таких систем.

3. Как в свое время расшифровка генома человека дала экономике США 3% прироста ВВП (каждый вложенный в нее доллар принес 140 долл. прибыли), ожидается, что успехи в создании ИИ дадут громадный прирост бизнесу стран, развивающих исследования в этой области. Поэтому ряд стран, в частности Китай, Южная Корея, а также Евросоюз поставили работы по ИИ в ряд важнейших государственных задач.

4. Более конкретно: ожидается, что ИИ приведет к улучшению процессов управления, развитию отношений производителей с потребителями, оптимизации всех бизнес-процессов, совершенствованию планирования и кадровой работы, превращению продуктов в сервисы и смене бизнес-модели многих бизнесов. Пример – компания Uber.

6. Рынок ИИ

Крайне трудно в наш век неопределенности делать прогнозы. Все перечисленные ниже прогнозы называют сильно завышенные объемы рынка ИИ на ближайшие годы. Рост его, по нашему мнению, будет весьма значительным, но не столь ошеломляющим, как предсказывают аналитические агентства (см. табл.).

Таблица 1. Основные области применения систем искусственного интеллекта (%)

Системы исследований и рекомендаций в сфере управления качеством	10,3
Диагностические и лечебные системы	10,0
Автоматизированные службы поддержки клиентов	9,8
Автоматизированные системы предотвращения угроз	9,8
Системы анализа и расследования мошенничества	9,0
Другое	51,1

Искусственный интеллект стал ключевым технологическим трендом 2016 г., и объем глобальных инвестиций в него превышает 500 млн долл.

Согласно прогнозу IDC, продажи когнитивных систем и систем искусственного интеллекта в 2017 г. в мире вырастут на 59,3% и достигнут 12,5 млрд долл. [31].

По прогнозам международной исследовательской компании Markets and Markets, к 2020 г. рынок ИИ вырастет до 5 млрд долл. за счет применения технологий машинного обучения и распознавания естественного языка в рекламе, розничной торговле, финансах и здравоохранении.

В агентстве Gartner считают, что к 2020 г. около 40% всех взаимодействий с виртуальными голосовыми помощниками будет опираться на данные, обработанные нейронными сетями [15].

Консалтинговая компания Tractica считает, что динамика ИИ будет основываться на шести фундаментальных технологиях: машинное обучение, глубинное обучение, компьютерное зрение, обработка естественного языка, машинная аргументация и сильный ИИ. Хотя в перспективе 10 лет ИИ-технологии повлияют практически на каждый бизнес, основными драйверами рынка станут секторы потребительских продуктов, бизнес-услуг, рекламы и обороны. Tractica предсказывает рост рынка ИИ с 643,7 млн долл. в 2016 г. до 38,8 млрд долл. к 2025 г. [15].

Ключевым драйвером этого рынка является уход всех процессов как в бизнесе, так и в потребительском сегменте в облака, а также рост влияния Интернета, смартфонов, социальных медиа. Акторами этого рынка являются такие крупные корпорации, как NEC, Google, Honeywell, Hitachi и Qualcomm Technologies. Также присутствует множество меньших по размеру игроков, таких как LTU Technologies, Attrasoft, Blippar и SLYCE, и таких вендоров, как Catchoom и Wikitude.

Мировой рынок распознавания речи оценен BCC Research в колоссальные 90,3 млрд долл. в 2015 г. Ожидается, что этот рынок вырастет с 104,4 млрд долл. в 2016 до 184,9 млрд долл. в 2021 г. со средними темпами (CAGR) на уровне 12,1% за период

2016–2021 гг.

Рынок обработки естественного языка (Natural Language Processing, NLP) оценивается Market And Markets в 7,63 млрд долл. в 2016 г. и вырастет до 16,07 млрд долл. к 2021 г., с CAGR на уровне 16,1%. Основными драйверами компания считает возрастающий спрос на более продвинутый уровень пользовательского опыта, рост пользования умными девайсами, рост инвестиций в здравоохранение, растущее применение сетевых и облачных бизнес-приложений и рост M2M-технологий.

ВоА предполагает, что к 2020 г. рынок ИИ-решений будет эквивалентен 153 млрд долл., из которых 83 млрд долл. составят роботы и робототехника и 70 млрд долл. – аналитические решения на основе ИИ.

В результате так называемая «революция роботов», о которой говорят экономисты и аналитики крупнейших банков, позволит мировой экономике повысить производительность на 30% при снижении производственных затрат на рабочую силу от 18 до 33%. В общей сложности на мировом рынке работает порядка 400 компаний, занимающихся производством робототехники [15].

Гонку за искусственным интеллектом китайцы начали не вчера, а как минимум пятилетку назад – Baidu, Alibaba и Tencent создали центры развития ИИ в это время. Результаты значительные – более 8000 патентов в данной области за 2010–2015 гг. Частный бизнес уже давно участвует в этой гонке [19].

Как видно из приведенных оценок, прогнозы аналитических агентств достаточно пестрые, с большим разбросом показателей, но с хорошими перспективами роста во всех секторах ИИ.

7. Основные международные программы и центры разработки

Работами в области ИИ сейчас заняты практически все крупные зарубежные компании, университеты и научные агентства. Существует список из более чем 1000 стартапов, работающих в этой области. Лидерами являются такие известные бренды, как Google, IBM, Microsoft, Amazon, Apple, DARPA и др.

Например, огромный скачок в распознавании речи сделала Microsoft, которая объявила, что ее система распознавания речи теперь так же точна, как распознавание речи живым человеком. Довести систему распознавания речь до такого высокого уровня удалось в том числе с помощью метода, разработанного резидентом «Сколково», компанией «ЦРТ-инновации» (группа «Центр речевых технологий»), который был представлен на международной конференции Interspeech сентябре 2016 г. в Сан-Франциско [15]. При этом корпорации тесно сотрудничают с ведущими университетами. Так, в начале сентября 2017 г. IBM объявила о заключении 10-летнего соглашения о партнерстве с университетом на сумму 250 млн долл. В этих работах будут задействованы более 100 исследователей из обеих организаций, которые будут сотрудничать в продвижении ключевых областей в области ИИ, таких как алгоритмы глубокого обучения, взаимосвязь между машинным обучением и квантовыми вычислениями и применение суперкомпьютера Watson в здравоохранении и кибербезопасности.

Китайские программы развития ИИ базируются на широком финансировании исследований в университетах, государственной поддержке работ отечественных компаний, создании в КНР совместных с западными странами исследовательских и разработческих центров. Отметим, что 2017 г., согласно отчету компании CB Insights, Китай впервые обошел США по объему инвестиций в стартапы, разрабатывающие системы искусственного интеллекта. На долю США пришлось 38% мировых инвестиций в ИИ-стартапы, а на долю Китая – 48% [7].

8. Состояние работ по ИИ в РФ

Россия вступила в третью волну ИИ с существенным запозданием.

Финансированием исследований в области ИИ занимаются Российский фонд фундаментальных исследований (РФФИ), Российский научный фонд (РНФ), однако из-за отсутствия координации работ финансирование исследований и разработок в целом недостаточное и выглядит лоскутным. Наилучшее положение в России сейчас у разработчиков военных роботов.

В РФ в настоящее время разрабатываются проекты по распознаванию речи, распознаванию лиц и распознаванию образов, системы автономного вождения автомобилей и др. приложения ИИ. Крупные российские компании уже используют ИИ для своего бизнеса. С помощью технологий машинного обучения компания МТС прогнозирует желание абонента воспользоваться тем или иным сервисом. Ему делается предложение еще до того, как абонент сам обратится к компании за подключением услуги.

Это экономит время клиента и одновременно повышает уровень продаж [32].

«Сбербанк» совместно с МФТИ разрабатывает проект по использованию технологии ИИ в колл-центре. Задача этого проекта, который называется «Нейроинтеллект iPavlov», – разработка алгоритмов глубинного машинного обучения для создания разговорного ИИ. Данная технология должна снизить нагрузку на сотрудников и повысить качество работы с клиентами [32].

Компания «Яндекс» разрабатывает ИИ-приложения для оптимизации производства в промышленности. Кроме того, компанией представлен метод машинного обучения CatBoost – это наследник метода «Матрикснет», который применяется почти во всех сервисах «Яндекса» [32].

Компания ABBYY использует технологии ИИ в решении ABBYY FlexiCapture, универсальной платформе для интеллектуальной обработки информации. Она определяет тип документа, извлекает из него данные и отправляет их в информационные системы: в CRM, в учетные системы, в системы управления закупками [32].

Приложения ИИ разрабатываются для роботов и БПЛА; так, например, разрабатывается ИИ для семейства беспилотников

«Фрегат», создаваемых группой «Кронштадт». Академическая и вузовская наука проводит отдельные исследования по тематике ИИ, однако масштабные и скоординированные НИОК по ИИ практически полностью отсутствуют.

Наибольшую озабоченность вызывает состояние обучения когнитивным наукам. По многим из них просто нет квалифицированных преподавателей. Программы

обучения катастрофически отстают от потребностей рынка труда.

В целом в стране существует около десятка сильных коллективов, занимающихся ИИ.

Следует отметить разработки Федерального исследовательского центра «Информатика и управление» (ФИЦ ИУ РАН) [23]:

EXACTUS EXPERT – система для семантического поиска и анализа качества научных публикаций;

EXACTUS PATENT – для семантического поиска и анализа патентной информации;

EXACTUS LIKE – для обнаружения близких текстов и вычисления степени семантической близости;

TEXT Appliance – информационно-аналитическая система анализа неструктурированной информации.

Главная проблема в том, что в России очень мало фундаментальных хороших исследований доходят до готового продукта. Существует разрыв между фундаментальной наукой, которая вышла в «лихие 90-е», и реализацией ее наработок в конкретном продукте [17].

9. Основные направления работ в области ИИ

Работы в области ИИ сейчас ведутся во многих странах и развернуты широким фронтом. Они группируются вокруг следующих направлений.

1. Глубинное обучение. Здесь усилия направлены на существенное сокращение времени обучения нейросети и уменьшение объема обучающей выборки. В идеале нейросеть должна обучаться в реальном времени.
2. Синтез роботом ответов, исходя из того корпуса знаний, который в него загрузили, в привязке к контексту и последовательности слов. Робот должен научиться соотносить поступающую информацию со своей базой знаний и обучаться [20].
3. Разработка нейроморфных микросхем и компьютеров на их базе. Такие микросхемы уже выпустили корпорации IBM и Intel. Пока они содержат до 4096 искусственных нейронов и до 256 млн синапсов, но направление весьма перспективно.
4. Продолжатся работы по картированию мозга человека и моделированию его работы.
5. Разработка систем распознавания и понимания речи. Многие сервисы используют речевой интерфейс, требующий хорошего распознавания речи. Наряду с этим важно и понимание сказанного пользователем. В этом плане очень важно понимание такими системами контекста сказанного, поскольку контекст – важнейшая часть естественного языка. В этой части исследования также смещаются в направлении разработки систем, способных взаимодействовать с людьми через диалог, а не просто реагировать на стилизованные запросы. Передача поисковым системам и ряду популярных приложений запросов голосом, особенно со смартфонов, уже стала обыденным явлением, и объем таких запросов будет только увеличиваться.

6. Развитие интерфейсов мозг – компьютер. Бурно стартовавшие в этом направлении работы сейчас замедлились, но будут продолжены.
7. Изучение систем группового поведения роботов и взаимодействия роботов и людей в ходе выполнения каких-либо операций. Усилия исследователей направлены на изучение эффективного распределения задач между людьми и машинами. Это направление важно как для военных роботов, так и для сервисных роботов, а также и для производственной сферы.
8. Внедрение ИИ в Интернет вещей (IoT), Индустрию 4.0 и другие современные направления типа цифровой экономики, поскольку обработка громадных объемов данных, генерируемых в таких системах, без ИИ столкнется с большими трудностями.
9. Навигация автономных транспортных средств в среде обитания человека. Это наиболее перспективное с рыночной точки зрения направление должно обеспечить начало широкого внедрения начиная с 2022–2025 гг. беспилотных автомобилей и автомобилей, в которых действия водителя полностью контролируются системой автономного управления.
10. Возможно развитие совершенно нового класса систем типа «человеческий интеллект по требованию», услуги роботов по требованию (RaaS) и т.п.
11. Гиперперсонализация услуг. Так как сейчас пользователь постоянно подключен к сети, то на основе его прошлых данных и модели поведения приложения с ИИ должны будут уметь делать предположения о том, какие сервисы ему следует предложить в данных обстоятельствах, какие решения для него будут оптимальны.
12. Борьба с эффектом «катастрофического забывания». Когда система с ИИ сталкивается с тем, чему она никогда не была обучена, то без переобучения она постоянно будет повторять одни и те же ошибки. Попытка переобучить сегодняшние системы как раз и приводит к феномену, называемому «катастрофическим забыванием», т.е. к ситуации, когда изучение нового предмета нарушает знание ранее изученных вещей [26].
13. Разработка нормативно-правовой базы использования роботов. «Роботы как субъекты права еще законодательно не оформлены ни в одной стране мира, поскольку для этого нужно официально признать, что робот – это не просто вещь, а создание, наделенное пусть и искусственным, но интеллектом, а также собственной волей. Поэтому в мире еще нет законов, которые что-то предписывают либо запрещают роботам напрямую. Перед законодателями лежит отнюдь не простая задача – обеспечить баланс между интересами технического прогресса, безопасности общества и государства, потребностями отдельно взятого человека. В Южной Корее в 2008 г. приняли «Закон о содействии развитию и распространению умных роботов». В феврале 2017 г. Европарламентом принята резолюция Civil Law Rules on Robotics – Нормы гражданского права о робототехнике [25].
14. Развитие таких направлений, как понимание текста, рассуждения, основанные на прецедентах, и др.

По мнению ряда экспертов, текущим разработкам в области ИИ не хватает программно-аппаратных средств физического контакта с окружающим миром, для того чтобы складывалось понимание различия или сходства между виртуальным и реальным мирами. Это направление развивается в робототехнике, например тактильная поверхность рук роботов.

10. Тенденции использования систем ИИ в области государственного управления

1. Наиболее востребован ИИ в силовых структурах, поскольку позволяет полиции анализировать в городах видео с множества камер и разыскивать преступников. Компьютерные программы способны сами формировать документы для прокуратуры и обвинительные заключения.
2. ИИ может оказать серьезное влияние на контроль и управление дорожным движением в городах. В ряде стран такие работы уже проводятся.
3. ИИ для МЧС. Например, управлять работой пожарных, предсказывать развитие пожара.
4. Применение в ИИ в судебной системе. Ассистенты судей.
5. Анализ социальных сетей, чтобы предотвращать возможные антигосударственные акции радикальных группировок.
6. В образовании – отслеживание успехов каждого учащегося, целевая помощь в планировании образования и трудовой карьеры.
7. ИИ в системах поддержки принятия государственных решений в экономике и банковской сфере.
8. Другое.

11. Ключевые вызовы и угрозы развития систем ИИ

Скепсис по поводу безопасного развития искусственного интеллекта (ИИ) подогревается СМИ и подогревался отдельными группами исследователей и бизнесменов, в частности Биллом Гейтсом, Илоном Маском и Стивеном Хокингом. Популярно утверждение со ссылкой на недавний опрос, проведенный Институтом будущего человека (FHI) из Оксфордского университета в Великобритании, который показал, что «Скайнет» как настоящий искусственный интеллект человеческого уровня может возникнуть около 2028 г. Можно упорно спорить, будет ли это 2028 или 2035 г., или же событие перенесется на более поздний период, однако сама такая возможность вызвала ряд опасений, публичных дискуссий и появление исследований, впрочем, весьма гипотетических. Здесь верно следующее утверждение: «...расхождение во взглядах ИИ в основном идет по временной шкале: различаются оценки скорости движения, а не его направления. Мало кто сомневается, что машины будут все больше и больше превосходить нас в плане способностей, характерных именно для людей, или же смогут усиливать их посредством кибернетических технологий» [5].

Опасение первое: Полная зависимость от компьютеров

С каждым днем мир все сильнее и сильнее зависит от компьютеров. Во многих автоматизированных системах решение принимают уже не люди, а компьютеры, – люди просто не успевают это сделать. Миллиарды компьютеров и микроконтроллеров (их уже на порядки больше, чем обычных компьютеров) – например, управляющие производственными процессами, системами связи, вооружениями, системами жизнеобеспечения и т.д. – работают автономно, без участия человека. Развитие систем с ИИ приведет к появлению таких систем в качестве руководителей (менеджеров), которые будут руководить работой сотрудников-людей. Плюсы таких решений очевидны – это рост производительности труда и качества продукции, обеспечение безопасности, повышение комфорта жизни и многое другое. Опасения у экспертов вызывает то, что этот процесс цифровизации продолжается и сейчас речь идет об Интернете вещей, Индустрии 4.0 (цифровой экономике). Таким образом, незаметная передача машинам возможности принимать решения продолжается, но последствия данного процесса из-за масштаба системы совершенно не просчитаны.

Опасение второе: Непредсказуемость

Совершенно очевидно, что технический прогресс приводит к неконтролируемому созданию новых и совершенствованию существующих опасных технологий. Судьба и жизнь множества людей (а может быть, и всего человечества) может зависеть от действий небольших групп ученых и разработчиков. Возник специальный термин для обозначения возможного этапа развития технологий.

Технологическая сингулярность – гипотетический момент, по прошествии которого, по мнению сторонников данной концепции, технический прогресс станет настолько быстрым и сложным, что окажется недоступным человеческому пониманию. Основным представителем этой идеи является Рэй Курцвейл, футуролог и изобретатель, автор книги «Сингулярность близко», а ныне технический директор области машинного обучения и обработки естественного языка Google. При этом Курцвейл весьма идеализированно утверждает, что «интеллект по своей природе не поддается контролю».

Данную концепцию поддерживают далеко не все ученые, утверждая, что развитие технологий происходит по S-кривой и в конце прошлого века началось замедление процесса ускорения.

Переход к принятию решений компьютерами был бы безопасным, но основные опасения вызывает непредсказуемость систем, обладающих самосознанием, а также влияние на них разного рода случайностей.

Это осложняется грядущим неминуемым «интеллектуальным взрывом». Всё дело в потенциальной скорости самоусовершенствования ИИ. Множество разговоров, касающихся ИИ, основано на том, что, обладая способностью к самообучению, ИИ с некоторого момента превращается в искусственный сверхразум, опыта переговоров с которым у человечества нет; и, кроме кнопки выключения электропитания, средств воздействия на него также пока не создано.

Опасение третье: Использование боевых роботов

Ссылки на четыре известных закона робототехники писателя- фантаста Айзека Азимова, которым должны подчиняться системы ИИ, вопреки распространенному мнению, не решают проблему (первоначально этих законов было три, потом добавится четвер- тый «Робот не может нанести вред человечеству»). Уже сейчас понятно, что эти законы не будут реализовываться в каждой ИИ-системе, например в боевых роботах и кибероружии (около 60 стран мира имеют или разрабатывают боевых роботов). Кон- кретные аргументы против автономных боевых роботов заключа- ются, например, в том, что они могут игнорировать «белый флаг», означающий намерение сдаться в плен, не смогут достоверно от- личить мирных граждан от военнослужащих, что может привести к дополнительным человеческим жертвам при использовании та- ких роботов во время боевых действий, и др.

Призыв к тому, чтобы прежде чем развивать технологии, которые рано или поздно приведут к созданию сверхразума, поста- вить вопрос о его отношении к человеку и человечеству, – остается гласом вопиющего в пустыне – слишком большое преимущество получают государства, внедряющие ИИ в робототехнику, системы оружия, медицину, производство и управление. В последнее время в боевых роботах отрабатываются алгоритмы поведения, исполь- зующие ложь и обман. Такие методы помогут им обмануть потен- циальных врагов – людей и других роботов. Однако при этом воз- никают этические проблемы, а попадание подобных систем в руки террористов может привести к катастрофическим последствиям. После того как Китай провозгласил лидерство в ИИ своим национальным приоритетом, между ведущими мировыми держа- вами началась борьба за развитие ИИ.

Опасение четвертое: Социальные риски

Искусственный интеллект несет определенные социальные риски – в первую очередь прогнозируемое исчезновение целого ряда профессий (например, водителей, младшего медперсонала и даже журналистов), разобщение людей, а возможно, даже потерю естественных навыков человека. Развитие техники постоянно идет по пути исключения человека из выполняемых им производствен- ных действий. Процессы эти сложные, потому что с исчезновением старых профессий появляются новые, связанные с новой цифровой экономикой, внедрением и поддержанием новой инфраструктуры. Однако в целом потребности в рабочей силе должны сильно умень- шиться, поскольку машины могут заменить человека в большинстве видов активной (не творческой) деятельности. Интеллектуализа- ция, цифровизация и роботизация могут привести как к появлению в обществе социальной напряженности, так и к внедрению мер для замедления указанных процессов.

Внедрение компьютерных технологий уже привело к изме- нению ментальности людей – появлению так называемой клиппо- вости сознания, компьютерной зависимости, уходу в виртуальную реальность и другим негативным явлениям. В настоящее время трудно оценить глобальные последствия подобных факторов, но

не учитывать их нельзя.

Опасение пятое: Ошибки в системах ИИ

В различных системах, управляемых ИИ, рано или поздно могут проявляться ошибки, допущенные при их создании. Это, в частности, связано с проблемами машинного обучения, поскольку большинство широко используемых методов машинного обучения далеки от совершенства и очень требовательны к данным. Небольшие изменения входных данных могут привести к крупным и странным изменениям в некоторых шаблонах [4]. Поскольку жизни людей все больше доверяются интеллектуальным системам, то проявление ошибок – это только вопрос времени; ошибки будут происходить и при решении ответственных задач, связанных с безопасностью людей. При этом уже сейчас не всегда понятно, каким образом система ИИ выбрала то или иное решение. Для систем с суперинтеллектом такое понимание может оказаться принципиально невозможным, как и исправление ошибочных решений системы.

Экзистенциальные риски

У органического мозга есть химические и метаболические пределы, ограничивающие его размер и вычислительную мощность. Поэтому, по мнению отдельных исследователей, «в долгосрочной эволюционной перспективе люди и все, о чем они когда-либо думали, станут всего лишь примитивной переходной формой, предшествовавшей более глубокому мышлению новой машиноориентированной культуры, простирающейся в отдаленное будущее и далеко за пределы Земли» [5]. Понятно, что подобный сценарий касается достаточно отдаленного будущего при отсутствии развития человеческих рас и крайне нежелателен.

«На теме искусственного интеллекта сейчас слишком много спекулируют, и в этом кроется серьезная проблема. Сфабрикованные успехи люди принимают за чистую монету, после чего реальные технологии уже не соответствуют их гиперожиданиям. Сначала робот София ездит по всему миру, якобы самостоятельно отвечая на любые вопросы, ХОТЯ на самом деле за нее пишут ответы люди, а потом оказывается, что японский Пеппер, подключенный к суперкомпьютеру Watson, не может ответить на простейший запрос покупателя в магазине. Опять же на волне выступлений Маска об угрозах искусственного интеллекта создается впечатление, что умные машины могут захватить нас уже завтра» [20].

Здесь перечислены только основные риски, но список ими не ограничивается.

Когда-то считалось, что одна из основных философских проблем в области ИИ – возможность или невозможность моделирования мышления человека, т.е. вопрос о том, обретет ли искусственный интеллект сознание. Вопрос уже не совсем гипотетический и очевидно имеющий большое значение для прогнозирования ближайшего будущего не только России, но и большей части жителей планеты Земля. Полувековая история развития ИИ все больше убеждает нас в том, что принципиальных препятствий на пути достижения такого результата нет. Скорее

всего, ИИ можно сделать и не на нейросетях. Просто нейросети в некотором плане уже освоенное природой решение.

Вопросы для самоконтроля и текущего контроля

- 1) Дайте определение искусственного интеллекта.
- 2) Опишите особенности каждой волны искусственного интеллекта
- 3) Какие ожидания от ИИ существуют на сегодняшний день?
- 4) В каком состоянии находятся работы по исследованию ИИ в России на сегодняшний день?
- 5) Перечислите основные направления работ в области ИИ.
- 6) Перечислите основные угрозы и проблемы развития систем ИИ, существующие в настоящее время.

Задание на самостоятельную работу

- 1) Практическое задание

Выполнить исследование применяемых в профессиональной деятельности прикладных систем искусственного интеллекта согласно заданию практической работы №1.

- 2) Самостоятельная работа

Исследовать направления развития систем искусственного интеллекта.

Список рекомендуемой литературы¹:

[1] Сидоркина, Ирина Геннадьевна. Системы искусственного интеллекта [Текст] : [учеб. пособие для вузов по направлению 230100 "Информатика и вычисл. техника"] / И. Г. Сидоркина. М.: Кнорус, 2011. - 245 с.

Рассмотрены теоретические аспекты проектирования систем искусственного интеллекта и даны модели представления знаний, изложены материалы информационного, справочного, консультирующего характера по использованию инструментальных средств и технологическим особенностям разработки систем данного класса.

Подробнее: <https://www.labyrinth.ru/books/276217/>

[2] Богомолова, М. А. Экспертные системы (техника и технология проектирования) [Текст] : Методические указания к лабораторным работам / М. А. Богомолова. Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015. - 47 с.

Методические указания являются руководством при выполнении лабораторных занятий, проводимых по курсу «Экспертные системы (техника и технология проектирования)» с магистрами направления 09.04.03 - «Прикладная информатика» в терминальном классе. Лабораторные работы связаны с изучением принципов построения экспертных систем на основе различных моделей представления знаний, и могут быть полезны студентам других специальностей.

Подробнее: <https://www.iprbookshop.ru/71908.html>

[3] Советов, Борис Яковлевич. Представление знаний в информационных системах [Текст] : [учеб. для вузов по направлению подгот. "Информ. системы и технологии"] / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. М.: Академия, 2011.

Учебник создан в соответствии с Федеральным государственным образовательным стандартом по направлению бакалавриата «Информационные системы и технологии». Рассмотрены современное состояние теоретических и прикладных вопросов представления знаний в информационных системах, идеология построения интеллектуальных систем, математический аппарат представления знаний, возможности и пути использования искусственного интеллекта при проектировании информационных систем. Изложены новые аспекты представления знаний на основе искусственных нейронных сетей, расчетно-логических систем, генетических алгоритмов, мультиагентных систем. Приведены примеры практической реализации представления знаний на базе декларативного языка Пролог. Для студентов учреждений высшего профессионального образования. Может быть полезен разработчикам и пользователям информационных систем; преподавателям и научным сотрудникам, сферой интересов которых является интеллектуализация различных предметных областей; менеджерам и руководителям различного ранга, желающим самостоятельно ознакомиться с современным состоянием информационных технологий.

Подробнее: <https://lavkababu.in.com/predstavlenie-znaniy-v-informacionnyh-sistemah-2-e-izd-ster-650564/>

[4] Остроух, А. В. Интеллектуальные информационные системы и технологии [Электронный ресурс] : монография / А. В. Остроух, А. Б. Николаев. Санкт-Петербург: Лань, 2019. - 308 с.

В монографии изложены концептуальные основы и методы представления знаний в интеллектуальных системах. Рассмотрены различные подходы, применяемые при проектировании и разработке интеллектуальных систем и технологий в транспортном комплексе, а также рассмотрены тенденции развития систем искусственного интеллекта.

Монография может быть использована для формирования профессиональной компетентности студентов высших учебных заведений, аспирантов и научных сотрудников обучающихся и ведущих научные исследования в области разработки и практического применения систем искусственного интеллекта по укрупнённой группе направлений подготовки "Информатика и вычислительная техника".

Подробнее: <https://www.labyrinth.ru/books/692861/>

Ссылки на открытые ресурсы (он-лайн-курсы, видеоуроки и т.д.)

[5]

ⁱСписок рекомендуемой литературы может быть более широким по сравнению со списком литературы, указанным в рабочей программе

Прикладные интеллектуальные системы

Конспект лекции 2. Задачи, решаемые прикладными системами искусственного интеллекта (ПСИИ). Классификация ПСИИ. Критерии выбора ПСИИ.

Аннотация: В лекции рассказывается о прикладных задачах, которые решаются в настоящее время системами искусственного интеллекта. Приводится классификация систем искусственного интеллекта. Описаны критерии выбора систем искусственного интеллекта для решения различных задач.

План лекции:

1. Прикладные задачи решаемые ИИ.
2. Классификация интеллектуальных информационных систем.
3. Критерии оценки интеллектуальности систем.
4. Задания для самостоятельной работы.

Ключевые слова: прикладные задачи ИИ, классификация ИИ, критерии оценки ИИ

СОДЕРЖАНИЕ ЛЕКЦИИ:

1. Прикладные задачи решаемые ИИ

Как можно использовать искусственный интеллект на пользу бизнесу? Крупные компании и стартапы часто задаются этим вопросом, и не зря. Искусственный интеллект может стать одним из ключевых факторов развития вашего бизнеса. Как? В этой статье мы расскажем несколько способов.

1.1 Применение искусственного интеллекта в бизнесе

По данным информационного портала DataProt, к 2027 году мировой рынок искусственного интеллекта достигнет 267 миллиардов долларов. При этом уже сейчас 37% компаний применяют ИИ-технологии. Это говорит о том, что многие понимают его ценность, и если вы еще не внедрили ИИ в свой бизнес, пора об этом задуматься.

Технологии искусственного интеллекта полезны для разных сфер бизнеса: ритейл, строительство, IT, образование и т.д. Каждой из них необходимы управление поведением потребителей и клиентов, изучение будущих тенденций рынка и автоматизация рутинных процессов. ИИ помогает упростить эти действия.

Вот несколько сфер, в которых ИИ наиболее востребован:

- **Транспорт.** Беспилотные автомобили, которые “чувствуют” окружающую среду и могут двигаться без вмешательства человека, могут трансформировать транспортную систему. Машины с использованием ИИ анализируют трафик и альтернативные маршруты, сокращая время в пути.
- **Производство.** Высокопроизводительные роботы работают быстрее и выполняют задачи более эффективно, чем люди. Благодаря использованию 3D-технологий и машинного зрения эти машины могут ускорить процесс производства в любой сфере.
- **Здравоохранение.** Приложения с искусственным интеллектом набирают обороты и постепенно превращаются в важного помощника для медицинского персонала. Автономные хирургические роботы, виртуальные помощники медперсонала и автоматическая диагностика изображений — это разработки, благодаря которым ИИ сыграл решающую роль в технологическом прогрессе здравоохранения.
- **Развлечения.** Машинное обучение может предсказывать поведение пользователя и давать рекомендации фильмов, музыки, телешоу и другого контента. Даже реклама персонализирована в зависимости от предпочтений пользователя, что повышает шансы маркетологов на продажу.
- **Спорт.** Предиктивный анализ и автоматизация могут использоваться для принятия бизнес-решений, продажи билетов и определения результатов спортсменов.

1.2 Зачем нужен искусственный интеллект

Искусственный интеллект может стать вашим надежным помощником в бизнесе и улучшить ваши показатели во всех сферах. Вот несколько задач, с которыми вам поможет ИИ:

- **Ценообразование.** Искусственному интеллекту проще изучать статистику и прогнозы, чем людям. Он может быстро обрабатывать огромные массивы информации и предложить наиболее оптимальное распределение цен. Это поможет вам повысить выручку и даже увеличить доходы компании в несколько раз.
- **Безопасность.** Мошенники могут встретиться и среди потребителей. Самообучающиеся нейронные сети анализируют поведение клиентов и видят подозрительные операции, чтобы вы могли вовремя на них отреагировать. Результат — отсутствие финансовых потерь, защищенность системы и доверие пользователей.
- **Маркетинг.** ИИ может изучить ваши предыдущие продажи и исследовать рынок, чтобы составить прогноз. Он анализирует поведение конкурентов, чтобы сравнить ваши правильные и неудачные шаги. Это позволит вам разработать грамотную маркетинговую стратегию, которая с большей вероятностью станет успешной.
- **Скорость.** Иногда очень важно быстро проанализировать информацию и еще быстрее на нее отреагировать. Хороший пример — биржевые операции. К сожалению, обычные алгоритмы не могут сами адаптироваться к новым условиям и данным без предварительного обучения. Искусственный интеллект предоставляет эту возможность и повышает продуктивность вашей работы.
- **Автоматизация.** Какими бы работоспособными ни были ваши сотрудники, они могут в любой момент допустить ошибку. На это влияют десятки факторов, которые вы не можете предугадать и устранить. Искусственному интеллекту это не свойственно: у него нет эмоций и чувств. Есть лишь данные, функции и технологии. Иногда стабильность работы критична, и в этом ИИ выигрывает у человека.

1.3 Области применения искусственного интеллекта

- **Виртуальные помощники.** AI виртуальные помощники

Самая очевидная сфера применения искусственного интеллекта — виртуальные помощники. Siri и Ok Google — не единственные примеры. Чат-боты — тоже продукт искусственного интеллекта, призванный помогать клиентам компаний. Например, чат-бот Олег в приложении банка Тинькофф с помощью распознавания речи общается с клиентами банка через цифровые устройства и выполняет стандартные операции вроде денежных переводов.

Использовать виртуального помощника или нет — выбирать вам, но это один из тех ИИ-инструментов, который будет только глубже проникать в бизнес-процессы и повседневную жизнь современных людей. По статистике Facebook, более 10000 компаний занимаются разработкой чат-ботов. Juniper Research подтверждают

популярность виртуальных помощников: использование чат-ботов в финансовом секторе и медицине способно сэкономить до 20 миллионов долларов в год, а к 2022 году до 8 миллиардов.

- **Контроль и мониторинг данных**

Постоянный контроль и мониторинг инфраструктуры компании — еще один пример области, где применяется искусственный интеллект. Французская энергетическая компания Engie применяет на предприятиях дронов с программами распознавания изображений на основе машинного обучения. Дроны следят за оборудованием и исследуют инфраструктуру, чтобы предотвратить нарушения.

ИИ-системы контроля и мониторинга хорошо подходят и для городской среды.

Наиболее простой пример — распознавание автомобильных номеров — применяется муниципальными организациями. Мы разработали подобный алгоритм для заказчика из Германии. По этому же пути идёт правительство Каталонии, предоставив своему департаменту полиции алгоритмы распознавания лиц и номерных знаков.

Хотите снизить риски износа и повреждения оборудования и создать подушку безопасности для своей компании? В этом помогут системы мониторинга на базе искусственного интеллекта.

- **Автоматизация ручного труда**

Многие боятся, что применение искусственного интеллекта в промышленности вытеснит из этой сферы людей. Страхи в сторону! Технологии помогают автоматизировать разные процессы — от отправки писем до бронирования авиабилетов. ИИ-алгоритмы не только выполняют сложные задачи быстрее людей, но и способны работать 24 часа в сутки. Однако цель высокоинтеллектуальных решений — не заменить людей, а сделать человеческий труд эффективнее.

Так, японская страховая компания Fukoku Mutual Life Insurance установила программу от IBM — Watson Explorer AI. Эта система анализирует данные медполисов по операциям и процедурам, чтобы вычислить размеры выплат. По расчётам представителей Fukoku, внедрение искусственного интеллекта позволит им увеличить производительность на 30%.

- **Предиктивная аналитика**

Люди хотят знать будущее, а компании — тем более. ИИ технологии способны обрабатывать большие массивы данных, выявлять закономерности и строить прогнозы. В одном из наших R&D проектов мы разрабатывали систему рекомендаций для крупной розничной сети супермаркетов. Система анализирует сходство покупателей и товаров, чтобы составлять качественные рекомендации.

Другой пример применения искусственного интеллекта в бизнесе — Expedia, крупнейшая в мире онлайн-платформа по планированию путешествий, от бронирования отелей до аренды транспорта. Компания эффективно использует машинное обучение, чтобы составлять персонализировать процесс планирования поездки.

Искусственный интеллект показывает хорошие результаты в построении прогнозов благодаря способности к самообучению. В отличие от традиционных типов прогнозирования, предиктивная аналитика легко адаптируется к изменениям поведения — когда поступают новые данные, она становится лучше.

- **Анализ неструктурированных данных**

С широким распространением мобильных устройств мы ежедневно создаем массу цифрового неструктурированного контента: сообщения в мессенджерах, письма, фото и видео. Алгоритмы ИИ помогают компаниям разобраться с этим “богатством” и структурировать данные так, чтобы впоследствии их проанализировать. Подобный принцип заложен в основе Siri — спонтанная разговорная речь, проходя через алгоритм программы, становится структурированной и подвергается дальнейшей обработке.

В анализе неструктурированных данных заложен огромный потенциал для производственных и ресурсодобывающих предприятий, которые накапливают смешанную информацию годами. Такой анализ способен облегчить работу и самих R&D инженеров — сэкономить время на сортировку и организацию данных перед тем, как оценить их и проследить важные взаимосвязи.

- **Робототехника**

Искусственный интеллект — шанс делегировать роботам утомительные и трудоемкие задачи. Бизнес уже сейчас следует этой тенденции. Например, онлайн-ритейлер Ocado разработал систему компьютерного зрения и сеть роботов, чтобы заменить процесс сканирования баркодов на своих торговых складах. Это поможет ускорить поиск и выдачу нужных товаров.

Хирургические и сельскохозяйственные роботы уже успели зарекомендовать себя, но роботизация других направлений только набирает обороты. В любом случае в ближайшее десятилетие рынок роботов и искусственного интеллекта будет активно расти. Почему бы не воспользоваться его возможностями сегодня, чтобы громко заявить о себе завтра?

1.4 Внедрение искусственного интеллекта

Чтобы начать внедрение искусственного интеллекта в бизнес, важно прежде всего собрать нужные данные, например, информацию о продажах за последние годы. Она может включать контактные данные клиентов, суммы сделок и приобретенные товары или услуги. Возможно, вы уже храните эту информацию в структурированном виде. В других случаях вам придется потратить время на ее систематизацию, чтобы затем ввести в программу для обработки данных.

Следующий шаг — разработка алгоритма искусственного интеллекта, который самостоятельно обучается. Здесь вам нужны хорошие специалисты, которые смогут научить систему необходимым действиям. Хорошие новости: на рынке уже достаточно

много готовых решений, которые помогут вам настроить ИИ быстрее.

Когда вы получаете нужную информацию от системы искусственного интеллекта, вам нужно перестроить технические или бизнес-процессы, на которые влияет ИИ.

Несомненно, здесь потребуется участие не только машин, но и людей. Однако в итоге ИИ поможет вам оптимизировать расходы и увеличить продажи.

Перспективные направления развития ИИ

Основная область применения искусственного интеллекта — это процессы, в которых можно отследить алгоритм действий человека и повторить его. Однако технологии еще не развиты до такой степени, чтобы делать вместо нас абсолютно все. Для чего сегодня еще не применяется искусственный интеллект?

Крупные технологические компании уже способны решать многие задачи по созданию компьютерного зрения, модулей управления движением, понимания речи, организации и предоставления доступа к информации с помощью компьютерного обучения.

Несмотря на это, современные компании не обладают достаточно надежными технологиями, которые могут воспроизводить точность человеческих глаз или зон мозга, ответственных за речь. В будущем и это может стать реальностью.

ИИ открывает новые возможности для решения экологических проблем планеты.

Основные риски в этой области связаны с безопасностью технологий и контролем за ними. Также необходимо учитывать этические вопросы и социально-экономические последствия применения ИИ. Несмотря на это, новые технологии действительно могут помочь людям следить за состоянием растений и животных и даже влиять на климатические условия.

Медицина — еще одна отрасль, в которой применение ИИ пока не может быть полноценным. Люди не настолько доверяют машинам, чтобы позволить им ставить диагнозы вместо квалифицированных специалистов. Однако уже сейчас эти самые машины оказывают врачам огромную помощь в изучении результатов анализов и постановке диагноза. Вероятно, в будущем власти у машин в этой сфере станет только больше.

2. Классификация интеллектуальных информационных систем

Существует большое множество интеллектуальных информационных систем (ИИС). Однако общепринятого единого определения интеллектуальной информационной системы нет.

Интеллектуальной информационной системой называют автоматизированную информационную систему, основанную на знаниях, или комплекс программных, лингвистических и логико-математических средств для реализации основной задачи – осуществления поддержки деятельности человека и поиска информации в режиме продвинутого диалога на естественном языке.

Кроме того, информационно-вычислительными системами с интеллектуальной поддержкой для решения сложных задач называют те системы, в которых логическая обработка информации превалирует над вычислительной.

Таким образом, любая информационная система, решающая интеллектуальную задачу или использующая методы искусственного интеллекта, относится к интеллектуальным.

Для интеллектуальных информационных систем характерны следующие признаки:

- развитые коммуникативные способности;
- умение решать сложные плохо формализуемые задачи;
- способность к самообучению;
- адаптивность.

Коммуникативные способности ИИС характеризуют способ взаимодействия (интерфейса) конечного пользователя с системой, в частности возможность формулирования произвольного запроса в диалоге с ИИС на языке, максимально приближенном к естественному.

Сложные плохо формализуемые задачи – это задачи, которые требуют построения оригинального алгоритма решения в зависимости от конкретной ситуации, для которой могут быть характерны неопределенность и динамичность исходных данных и знаний.

Способность к самообучению – это возможность автоматического извлечения знаний для решения задач из накопленного опыта конкретных ситуаций.

Адаптивность – способность к развитию системы в соответствии с объективными изменениями модели проблемной области.

В соответствии с перечисленными признаками ИИС делятся на (данная классификация – одна из возможных) (рис. 1):

- системы с коммутативными способностями (с интеллектуальным интерфейсом);
- экспертные системы (системы для решения сложных задач);
- самообучающиеся системы (системы, способные к самообучению);
- адаптивные системы (адаптивные информационные системы).



Рис. 1. Классификация интеллектуальных информационных систем по типам систем

Интеллектуальные базы данных отличаются от обычных баз данных возможностью выборки по запросу необходимой информации, которая может явно не храниться, а выводиться из имеющейся в базе данных.

Естественно-языковой интерфейс предполагает трансляцию естественно-языковых конструкций на внутримашинный уровень представления знаний. Для этого необходимо решать задачи морфологического, синтаксического и семантического анализа и синтеза высказываний на естественном языке. Так, морфологический анализ предполагает распознавание и проверку правильности написания слов по словарям, синтаксический контроль – разложение входных сообщений на отдельные компоненты (определение структуры) с проверкой соответствия грамматическим правилам внутреннего представления знаний и выявления недостающих частей и, наконец, семантический анализ – установление смысловой правильности синтаксических конструкций. Синтез высказываний решает обратную задачу преобразования внутреннего представления информации в естественно-языковое. Естественно-языковой интерфейс используется для:

- доступа к интеллектуальным базам данных;
- контекстного поиска документальной текстовой информации;
- голосового ввода команд в системах управления;
- машинного перевода с иностранных языков.

Гипертекстовые системы предназначены для реализации поиска по ключевым словам в базах текстовой информации. Интеллектуальные гипертекстовые системы

отличаются возможностью более сложной семантической организации ключевых слов, которая отражает различные смысловые отношения терминов. Таким образом, механизм поиска работает прежде всего с базой знаний ключевых слов, а уже затем непосредственно с текстом. В более широком плане сказанное распространяется и на поиск мультимедийной информации, включающей, помимо текстовой, и цифровую информацию.

Системы контекстной помощи можно рассматривать как частный случай интеллектуальных гипертекстовых и естественно-языковых систем. В отличие от обычных систем помощи, навязывающих пользователю схему поиска требуемой информации, в системах контекстной помощи пользователь описывает проблему (ситуацию), а система с помощью дополнительного диалога ее конкретизирует и сама выполняет поиск относящихся к ситуации рекомендаций. Такие системы относятся к классу систем распространения знаний (Knowledge Publishing) и создаются как приложение к системам документации (например, технической документации по эксплуатации товаров).

Системы когнитивной графики позволяют осуществлять интерфейс пользователя с ИИС с помощью графических образов, которые генерируются в соответствии с происходящими событиями. Такие системы используются в мониторинге и управлении оперативными процессами. Графические образы в наглядном и интегрированном виде описывают множество параметров изучаемой ситуации. Например, состояние сложного управляемого объекта отображается в виде человеческого лица, на котором каждая черта отвечает за какой-либо параметр, а общее выражение лица дает интегрированную характеристику ситуации. Системы когнитивной графики широко используются также в обучающих и тренажерных системах на основе использования принципов виртуальной реальности, когда графические образы моделируют ситуации, в которых обучаемому необходимо принимать решения и выполнять определенные действия.

Экспертные системы предназначены для решения задач на основе накапливаемой базы знаний, отражающей опыт работы экспертов в рассматриваемой проблемной области.

Многоагентные системы. Для таких динамических систем характерна интеграция в базе знаний нескольких разнородных источников знаний, обменивающихся между собой получаемыми результатами на динамической основе.

Для многоагентных систем характерны следующие особенности:

- 1) проведение альтернативных рассуждений на основе использования различных источников знаний с механизмом устранения противоречий;
- 2) распределенное решение проблем, которые разбиваются на параллельно решаемые подпроблемы, соответствующие самостоятельным источникам знаний;
- 3) применение множества стратегий работы механизма вывода заключений в зависимости от типа решаемой проблемы;
- 4) обработка больших массивов данных, содержащихся в базе данных;

5) использование различных математических моделей и внешних процедур, хранимых в базе моделей;

6) способность прерывания решения задач в связи с необходимостью получения дополнительных данных и знаний от пользователей, моделей, параллельно решаемых подпроблем.

В основе *самообучающихся систем* лежат методы автоматической классификации примеров ситуаций реальной практики.

Характерными признаками самообучающихся систем являются:

- самообучающиеся системы «с учителем», когда для каждого примера задается в явном виде значение признака его принадлежности некоторому классу ситуаций (классообразующего признака);

- самообучающиеся системы «без учителя», когда по степени близости значений признаков классификации система сама выделяет классы ситуаций.

Индуктивные системы используют обобщение примеров по принципу от частного к общему. Процесс классификации примеров осуществляется следующим образом:

1. Выбирается признак классификации из множества заданных (либо последовательно, либо по какому-либо правилу, например в соответствии с максимальным числом получаемых подмножеств примеров).

2. По значению выбранного признака множество примеров разбивается на подмножества.

3. Выполняется проверка, принадлежит ли каждое образовавшееся подмножество примеров одному подклассу.

4. Если какое-то подмножество примеров принадлежит одному подклассу, то есть у всех примеров подмножества совпадает значение классообразующего признака, то процесс классификации заканчивается (при этом остальные признаки классификации не рассматриваются).

5. Для подмножеств примеров с несовпадающим значением классообразующего признака процесс классификации продолжается начиная с пункта 1 (каждое подмножество примеров становится классифицируемым множеством).

Нейронные сети представляют собой устройства параллельных вычислений, состоящие из множества взаимодействующих простых процессов. Каждый процессор такой сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. В экспертных системах, основанных на *прецедентах* (аналогиях), база знаний содержит описания не обобщенных ситуаций, а собственно сами ситуации или прецеденты.

Поиск решения проблемы в экспертных системах, основанных на прецедентах, сводится к поиску по аналогии (то есть абдуктивный вывод от частного к частному).

В отличие от интеллектуальной базы данных, *информационное хранилище* представляет собой хранилище извлеченной значимой информации из оперативной базы данных, которое предназначено для оперативного ситуационного анализа данных (реализации OLAP-технологии).

Типичными задачами оперативного ситуационного анализа являются:

- определение профиля потребителей конкретных объектов хранения;
- предсказание изменений объектов хранения во времени;
- анализ зависимостей признаков ситуаций (корреляционный анализ).

Адаптивная информационная система – это информационная система, которая изменяет свою структуру в соответствии с изменением модели проблемной области.

При этом:

- 1) адаптивная информационная система должна в каждый момент времени адекватно поддерживать организацию бизнес-процессов;
- 2) адаптивная информационная система должна проводить адаптацию всякий раз, как возникает потребность в реорганизации бизнес-процессов;
- 3) реконструкция информационной системы должна проводиться быстро и с минимальными затратами.

Ядром адаптивной информационной системы является постоянно развиваемая модель проблемной области (предприятия), поддерживаемая в специальной базе знаний – репозитории. На основе ядра осуществляется генерация или конфигурация программного обеспечения. Таким образом, проектирование и адаптация ИС сводится, прежде всего, к построению модели проблемной области и ее своевременной корректировке.

Так как нет общепринятого определения, четкую единую классификацию интеллектуальных информационных систем дать затруднительно.

Если рассматривать интеллектуальные информационные системы с точки зрения решаемой задачи, то можно выделить системы управления и справочные системы, системы компьютерной лингвистики, системы распознавания, игровые системы и системы создания интеллектуальных информационных систем (рис. 2).

При этом системы могут решать не одну, а несколько задач или в процессе решения одной задачи решать и ряд других. Например, при обучении иностранному языку система может решать задачи распознавания речи обучаемого, тестировать, отвечать на вопросы, переводить тексты с одного языка на другой и поддерживать естественно-языковой интерфейс работы.

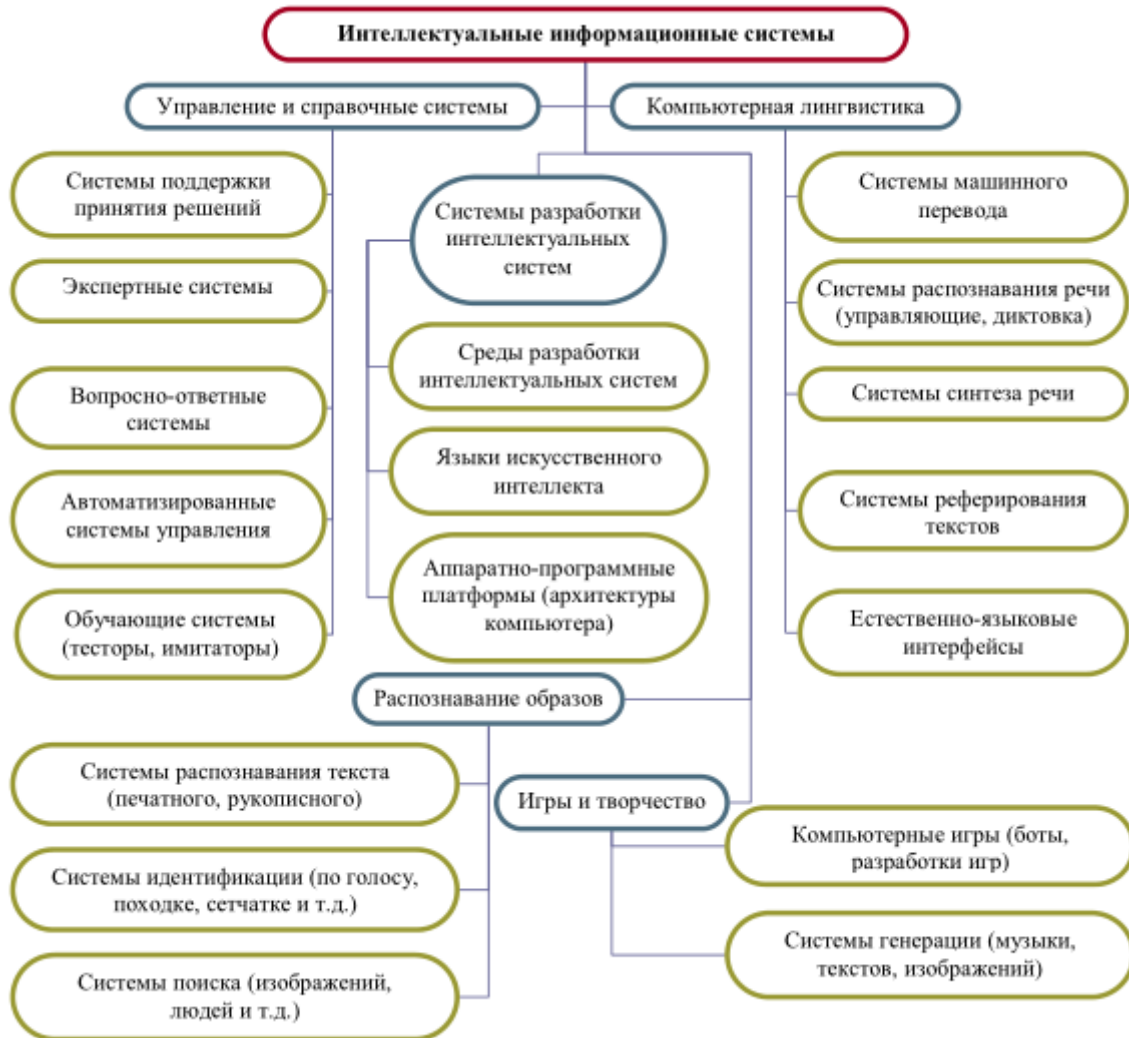


Рис. 2. Классификация интеллектуальных информационных систем по решаемым задачам

Если классифицировать интеллектуальные информационные системы по критерию «используемые методы», то они делятся на жесткие, мягкие и гибридные (рис. 3). *Мягкие вычисления* – это сложная компьютерная методология, основанная на нечеткой логике, генетических вычислениях, нейрокомпьютинге и вероятностных вычислениях. *Жесткие вычисления* – традиционные компьютерные вычисления (не мягкие). *Гибридные системы* – системы, использующие более чем одну компьютерную технологию (в случае интеллектуальных систем – технологии искусственного интеллекта).



Рис. 3. Классификация интеллектуальных информационных систем по методам

Возможны и другие классификации, например выделяют системы общего назначения и специализированные системы (рис. 4).



Рис. 4. Классификация интеллектуальных систем по назначению

Кроме того, эта схема отражает еще один вариант классификации по методам: системы, использующие методы представления знаний, самоорганизующиеся системы и системы, созданные с помощью эвристического программирования. Также в этой классификации системы генерации музыки отнесены к системам общения. К интеллектуальным системам *общего назначения* относятся системы, которые не только исполняют заданные процедуры, но на основе метапроцедур поиска генерируют и исполняют процедуры решения новых конкретных задач. *Специализированные* интеллектуальные системы выполняют решение фиксированного набора задач, predetermined при проектировании системы.

Отсутствие четкой классификации также объясняется многообразием интеллектуальных задач и интеллектуальных методов, кроме того, искусственный интеллект – активно развивающаяся наука, в которой новые прикладные области осваиваются ежедневно.

3. Критерии оценки интеллектуальности систем.

Разработка систем интеллектуального функционирования стала модным брендом многих научных исследований. Интеллектуальные системы появились в медицине и на транспорте, в промышленности и сфере услуг, теории математических доказательств и лингвистике, многих иных сферах практической деятельности людей. Под девизом интеллектуальности часто подаются системы, этим свойством не обладающие, а порой и просто некачественно выполненные работы. Произвольное толкование основных категорий исследуемой предметной области (машинный и искусственный интеллект, интеллектуальное функционирование, интеллектуальные и/или интеллектуализированные системы, виртуальная интеллектуальность и т.д.) приводит в лучшем случае к взаимному непониманию разработчиков и пользователей систем, в худшем – к серьезным последствиям экономического, производственного, техногенного характера. В этой связи актуальными проблемами для исследования являются:

- уточнение и систематизация категориального аппарата исследования;
- разработка методов оценки степени интеллектуальности технических и организационно-технологических систем.

На наш взгляд, время систематизации категориального аппарата исследования систем интеллектуального функционирования еще не наступило. Теория интеллектуальных систем находится только на стадии накопления и осмысления материала. Критическая точка, после которой происходят изменения количества поступившей информации в новое качество, не пройдена. Поэтому по первой проблеме позволим себе лишь некоторые рассуждения для дальнейшего обсуждения поставленной проблемы. Вот некоторые аспекты.

Создавая искусственные интеллектуальные системы, мы сравниваем их с человеком. Существующие представления обыденной жизни априори предполагают, что человек обладает интеллектом и он (интеллект субъекта) есть эталон для разработчиков искусственного интеллекта. Это в общем неверное представление. Действительно, одни военные начальники (менеджеры, финансисты, воспитатели, врачи и т. д.) более успешны, другие менее. Вследствие чего? Можно назвать тому ряд причин: природный дар, профессионализм, сложившиеся обстоятельства и т. д. А можно просто сказать, что интеллект успешных выше интеллекта посредственных руководителей и профессионалов, а интеллект последних выше интеллекта неудачников. То есть интеллект человека (как и машины) может изменяться от нуля до единицы. Так с чем и с кем сравнивать искусственный интеллект?

- 1) Как выделить в вышеуказанных свойствах личности интеллектуальную составляющую? Она может иметь различную природу и находиться в каждой из

названных свойств личности. Что важно для решения задачи развития интеллектуально функционирующих технических комплексов? Даже в том, что одни обучаются быстрее, другие медленнее, уже проявляется интеллект субъекта (и машины, если она также обучается в процессе функционирования).

2) Кроме того, интеллект интеллекту рознь: творения музыканта и поэта, шах-матиста и математика, решения руководителя организации и профессионала, бизнесмена и предпринимателя существенно различаются, но, конечно, отражают их интеллектуальное развитие, которое должно быть измерено и оценено. Очевидно, что разные системы оценки на различных объектах будут давать несовпадающие результаты. Мы это уже наблюдаем, оценивая, например, интеллект политиков, артистов, бизнесменов и других значимых персон мира. Многие из них дают довольно скромные значения показателя IQ, но заподозрить их в недостаточно развитой интеллектуальности не поворачивается язык. Достижения этих людей говорят сами за себя. Вопрос в том, что и как мы измеряем.

3) В рамках нашей работы следует говорить о машинном интеллекте [1] и не ввязываться в дискуссию о соответствии искусственного интеллекта естественному. То есть предлагается не гнаться за соответствием создаваемого машинного интеллекта интеллекту человека, а обеспечить соответствующее качество работ. Равно тому, как самолеты, выполняя свою функцию полета, не стремятся махать крыльями, но прекрасно справляются со своей миссией.

4) Целесообразно также ввести понятия врожденной и приобретенной интеллектуальности. Об этом можно говорить уже и на уровне человека. Известно, что мужской и женский типы организмов обладают разными стилями мышления. Мужчина, как правило, опирается на логику (приобретая знания в процессе жизни), женщина – на интуицию (врожденное свойство). Аналогично этому интеллект можно вложить в технико-технологическую систему до начала ее эксплуатации, а можно обучать машинный блок принятия решений в процессе выполнения работы. Как сравнить эти два типа интеллекта?

5) Интеллект должен моделироваться вектором, компоненты которого отражают различные его проявления. То есть интегральная характеристика системы при всей ее важности не может служить исчерпывающим показателем.

Таким образом, следует, анализируя синтезируемые системы, говорить не о наличии или отсутствии у них интеллекта, а оценивать степень интеллектуальности функционирования по многим критериям. Признать, что интеллектуальность искусственных систем (как и интеллект человека) зависит от сферы деятельности, внутренних и внешних условий функционирования (разве мы не знаем случаев, когда разум покидал вполне благоразумного человека), а также изменялся от 0 (полное отсутствие интеллекта в данный момент, в конкретной задаче, в рассматриваемой сфере!) до 1. Единица означает абсолютное признание за лицом, принимающим решение (человек или машина), интеллектуальности функционирования.

Рассмотрим два характерных подхода к оценке степени интеллектуальности. Метод А.А. Красилов ориентирован на оценку интеллектуальности компьютерных

систем [2]. Его система оценки разрабатывалась на экспертной основе в целях оценки интеллектуальности авторского программного комплекса «Интеллсист».

Целями такой оценки было заявлено:

- понимание пригодности программы для применения;
- постановка новых задач перед развитием программного комплекса;
- поиск погрешностей существующего комплекса с позиций пользователя;
- проведение позитивной критики на программу.

С точки зрения автора, подобная оценка должна проводиться по следующим семи номинациям:

- 1) наглядности или внешней интеллектуальности;
 - 2) осмысленности действий или внутренней интеллектуальности, оснащенности системы языками общения;
 - 3) концептуальности или базовым основам программного комплекса;
 - 4) использованию БД, взаимосвязи данных, их целостности и защищенности;
 - 5) наличию логического вывода или обоснованного вывода ответов на запросы пользователей;
 - 6) программной пригодности или настраиваемости и адаптируемости создаваемой программы;
 - 7) системной пригодности или общественной жизни программного комплекса.
- Совокупная оценка по всем номинациям, по данному методу, каждого программ-

ного комплекса является средством для сравнения интеллектуальности программ между собой. Согласно гипотезе авторов она может быть получена как *среднеарифметическая* оценка по результатам измерения группой экспертов всех указанных выше номинаций.

Слабыми местами этого подхода для эксплуатации и дальнейшего распространения являются:

- установление перечня номинаций. Для оценки интеллектуальности системы

этот перечень должен обладать свойствами полноты и универсальности (кстати, автор, ограничившись оценкой компьютерных систем, и не претендовал на выполнение этого требования);

- разработка механизма оценки значений экспертом номинаций.
- Высказывания

эксперта, как правило, субъективны и выражаются качественными категориями. Нужны адаптированные к задаче методы оперирования с лингвистическими переменными, технологии шкалирования (далее – масштабирования, нормирования) данных и способы снижения субъективности мнения человека;

- формирование команды экспертов. Здесь появляется целый спектр проблем:

как отобрать и оценить самих экспертов, каков механизм и критерии их совместного функционирования;

– механизм окончательного вывода, интегрально описывающего интеллектуальность системы. Автор анализируемого метода предлагает среднеарифметическое усреднение частных результатов по номинациям.

Следующий анализируемый нами метод оценки интеллектуальности описали В.А. Литвинов и И.Н. Оксанич [3]. В нем рассматривается процесс оценки уровня интеллектуальности программно-технической системы на основе построения эргономической модели процесса взаимодействия «пользователь – компьютер» и анализа интеллектуальной нагрузки на пользователя. Авторы в своем подходе к количественной оценке интеллекта программно-технической системы используют показатель «степень снижения интеллектуальной нагрузки на человека» независимо от ее структуры и внутренних свойств. Сущность подхода заключается в использовании оценок, основанных на:

–сравнении характеристик рассматриваемой системы с набором базовых показателей некоторой идеализированной интеллектуальной системы (ИИС). Результатом такого сравнения должно быть значение абсолютного коэффициента интеллектуальности $AAIQ$;

–определении степени соответствия значений выбранных проблемно-ориентированных показателей интеллектуальности, характеризующих системы заданного класса, заданным (желаемым) значениям. В результате такого сравнения оценивается относительный коэффициент интеллектуальности $RAIQ$.

Согласно данному методу интеллектуальная нагрузка на пользователя при решении некоторой задачи определяется как сумма количества выполняемых операций, помноженная на интеллектуальную трудоемкость этой операции. Этот подход имеет недостатки, аналогичные предыдущему. Прежде всего, это отсутствие структуры факторов, влияющих на оценку интеллектуальности, базирование на экспертных оценках при неизвестной схеме экспертного опроса, отсутствие системы предварительной обработки экспертных данных. И, главное, интегральный критерий вновь рассчитывается как сумма эффектов, а не как произведение.

Последнее замечание особенно важно для данного исследования. Поэтому прокомментируем его подробнее. Общеизвестно, что эффективность любой экономической системы (ее результативность) оценивается производственной функцией, которая мультипликативно сворачивает эффекты факторов влияния: капитала, человеческого ресурса, информации, предпринимательской способности и т.д. Это позволяет учитывать синергетический эффект взаимодействия факторов. Мы имеем аналогичную ситуацию.

Действительно, указанные автором метода номинации связаны между собой не союзом «или», а союзом «и». Например: интеллектуальная система должна одновременно быть «и» наглядной (номинация 1), «и» содержать базы данных (номинация 4), «и» быть настраиваемой и адаптируемой (номинация 6), «и» т.д. Логическая связка «и» в алгебре логики соответствует произведению факторов, поэтому усреднение значений номинаций следует проводить не по принципу среднего арифметического, а на основе среднего гео-

метрического. Если обозначить значения номинаций соответственно x_i , где $i = 1, \dots, 7$, то интеллектуальность системы будем оценивать соотношением:

$$J = \prod_i^7 x_i^{m_i}. \quad (1)$$

В соотношении (1) показатель степени m_i – вес i -фактора интеллектуальности.

Очевидно, что значение показателя интеллектуальности системы J существенным образом зависит от выбранной совокупности S базовых признаков x_i . Изменение ее состава равносильно изменению точки зрения (проекции «взгляда») на систему. Поэтому J величина относительна, и важно указывать, по какой совокупности признаков оценивалась интеллектуальность исследуемой системы. То есть:

$$J = J(S). \quad (2)$$

Вычисление интеллектуальности компьютерных систем актуально по целому ряду причин:

- неинтеллектуальные компьютерные системы не обеспечивают в настоящее время необходимого качества работ (скорость, точность, безопасность);
- интеллектуальные системы способны адаптироваться к изменениям внешней среды;
- интеллектуальные системы расширяют спектр решаемых народнохозяйственных задач.

Для успешного продвижения теории, формулирования единой и стандартизированной концепции разработки, эффективного внедрения интеллектуальных систем на практике требуется объединение усилий всех людей, занятых решением данного вопроса.

Вопросы для самоконтроля и текущего контроля

- 1) Перечислите направления применения современных интеллектуальных систем.
- 2) Дайте определение интеллектуальной информационной системы.
- 3) К каким интеллектуальным системам относится система, использующая генетические вычисления и базы данных?
- 4) Какие системы являются системами общего назначения?
- 5) Какие системы относятся к самоорганизующимся системам?
- 6) Какие системы относятся к системам компьютерной лингвистики?
- 7) Перечислите критерии оценки интеллектуальных систем.

Задание на самостоятельную работу

- 1) Практическое задание

Выполнить исследование применяемых в профессиональной деятельности прикладных систем искусственного интеллекта согласно заданию практической работы №1.

- 2) Самостоятельная работа

Исследовать примеры современных систем искусственного интеллекта.

Список рекомендуемой литературы¹:

[1] Сидоркина, Ирина Геннадьевна. Системы искусственного интеллекта [Текст] : [учеб. пособие для вузов по направлению 230100 "Информатика и вычисл. техника"] / И. Г. Сидоркина. М.: Кнорус, 2011. - 245 с.

Рассмотрены теоретические аспекты проектирования систем искусственного интеллекта и даны модели представления знаний, изложены материалы информационного, справочного, консультирующего характера по использованию инструментальных средств и технологическим особенностям разработки систем данного класса.

Подробнее: <https://www.labyrinth.ru/books/276217/>

[2] Богомолова, М. А. Экспертные системы (техника и технология проектирования) [Текст] : Методические указания к лабораторным работам / М. А. Богомолова. Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015. - 47 с.

Методические указания являются руководством при выполнении лабораторных занятий, проводимых по курсу «Экспертные системы (техника и технология проектирования)» с магистрами направления 09.04.03 - «Прикладная информатика» в терминальном классе. Лабораторные работы связаны с изучением принципов построения экспертных систем на основе различных моделей представления знаний, и могут быть полезны студентам других специальностей.

Подробнее: <https://www.iprbookshop.ru/71908.html>

[3] Советов, Борис Яковлевич. Представление знаний в информационных системах [Текст] : [учеб. для вузов по направлению подгот. "Информ. системы и технологии"] / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. М.: Академия, 2011.

Учебник создан в соответствии с Федеральным государственным образовательным стандартом по направлению бакалавриата «Информационные системы и технологии». Рассмотрены современное состояние теоретических и прикладных вопросов представления знаний в информационных системах, идеология построения интеллектуальных систем, математический аппарат представления знаний, возможности и пути использования искусственного интеллекта при проектировании информационных систем. Изложены новые аспекты представления знаний на основе искусственных нейронных сетей, расчетно-логических систем, генетических алгоритмов, мультиагентных систем. Приведены примеры практической реализации представления знаний на базе декларативного языка Пролог. Для студентов учреждений высшего профессионального образования. Может быть полезен разработчикам и пользователям информационных систем; преподавателям и научным сотрудникам, сферой интересов которых является интеллектуализация различных предметных областей; менеджерам и руководителям различного ранга, желающим самостоятельно ознакомиться с современным состоянием информационных технологий.

Подробнее: <https://lavkababuin.com/predstavlenie-znaniy-v-informacionnyh-sistemah-2-e-izd-ster-650564/>

[4] Остроух, А. В. Интеллектуальные информационные системы и технологии [Электронный ресурс] : монография / А. В. Остроух, А. Б. Николаев. Санкт-Петербург: Лань, 2019. - 308 с.

В монографии изложены концептуальные основы и методы представления знаний в интеллектуальных системах. Рассмотрены различные подходы, применяемые при проектировании и разработке интеллектуальных систем и технологий в транспортном комплексе, а также рассмотрены тенденции развития систем искусственного интеллекта.

Монография может быть использована для формирования профессиональной компетентности студентов высших учебных заведений, аспирантов и научных сотрудников обучающихся и ведущих научные исследования в области разработки и практического применения систем искусственного интеллекта по укрупнённой группе направлений подготовки "Информатика и вычислительная техника".

Подробнее: <https://www.labyrinth.ru/books/692861/>

Ссылки на открытые ресурсы (он-лайн-курсы, видеоуроки и т.д.)

[5]

ⁱСписок рекомендуемой литературы может быть более широким по сравнению со списком литературы, указанным в рабочей программе

Прикладные интеллектуальные системы

Конспект лекции 3. Системы основанных на знаниях. Понятие инженерии знаний. Этапы получения знаний о предметной области. Методы получения знаний о предметной области

Аннотация: В лекции рассказывается о понятии систем основанных на знаниях и понятии инженерии знаний. Рассмотрены этапы и методы получения знаний о предметной области в инженерии знаний.

План лекции:

1. Системы основанных на знаниях.
2. Понятие инженерии знаний.
3. Этапы получения знаний о предметной области.
4. Методы получения знаний о предметной области
5. Задания для самостоятельной работы.

Ключевые слова: системы основанные на знаниях, инженерия знаний, знания о предметной области, Методы получения знаний о предметной области

СОДЕРЖАНИЕ ЛЕКЦИИ:

1. Системы основанных на знаниях.

На начальном этапе исследований по искусственному интеллекту возникло всеобщее убеждение, что за интеллектуальным поведением человека скрываются его знания об окружающем мире. Речь идет о знаниях, которыми обладают специалисты профессионалы.

Данные – отдельные факты, характеризующие объекты, процессы и явления в предметной области, а также их свойства.

Знания основаны на данных, полученных эмпирическим путем. Это результат мыслительной деятельности человека, направленной на обобщение опыта, полученного в результате практической деятельности.

Знания – это закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области.

Знания – выводы, сделанные на основе данных. Они более компактны, чем данные. Знания можно сформулировать в виде правил, т.е. изложить в форму.

Особенности знаний, которые отличают их от данных.

Интерпретируемость. Данные, помещенные в ЭВМ, могут содержательно интерпретироваться лишь соответствующей программой. В отрыве от программы, данные не несут никакой содержательной информации. Знания отличаются тем, что в них возможность содержательной интерпретации всегда присутствует.

Наличие классифицирующих отношений. Несмотря на разнообразие форм хранения данных, ни одна из них не обеспечивает возможности компактного описания всех связей между различными типами данных. При переходе к знаниям между отдельными единицами знаний можно установить такие отношения, как “элемент-множество”, “тип-подтип”, “ситуация-подситуация”, отражающие характер их взаимосвязи. Это позволяет записать и хранить отдельно информацию, одинаковую для всех элементов множества. При необходимости эту информацию можно автоматически передать описанию любого элемента множества. Такой процесс передачи называют “наследованием” информации.

Наличие ситуативных связей. Эти связи определяют ситуативную совместимость отдельных событий или фактов, хранимых или вводимых в память, а также отношения, как одновременность, расположение в одной области пространства, нахождение в

состоянии механического взаимодействия. Ситуативные связи помогают строить процедуры анализа знаний на совместимость, противоречивость, которые трудно реализовать при хранении традиционных массивов данных.

Традиционно знания классифицируются по категориям:

поверхностные – знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области. (Кнопка звонка).

глубинные – абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов.

Помимо этой классификации, знания делят на процедурные и декларативные.

Декларативные знания - знания, которые записаны в памяти интеллектуальной системы так, что они непосредственно доступны для использования после обращения к соответствующему полю памяти. Обычно декларативных знаний используются для представления информации о свойствах и фактах предметной области. По форме представления декларативные знания противопоставляются процедурным знаниям.

Процедурные знания - знания, хранящиеся в памяти интеллектуальной системы в виде описаний процедур, с помощью которых их можно получить. Обычно процедурные знания используются для представления информации о способах решения задач в проблемной области, а также различные инструкции, методики и т.п.

Интеллектуальная система, основанная на знаниях, представляет собой такую систему, в которой с помощью логического вывода знания применяются к решению поставленных задач.

- Экспертные системы отличаются от традиционных вычислительных пакетов программ тем, как они организованы. С традиционных позиций программы представляет собой процедуру и данные.

Общее представление о работе экспертной системы показано на рисунке 6.2.

Экспертные системы содержат в базе данных три различных модуля:

- факты о признаках конкретной ситуации
 - правила относящиеся к области экспертизы
- правила работы с «экспертными правилами» или «машина вывода»

Схема работы системы основанной на знаниях:

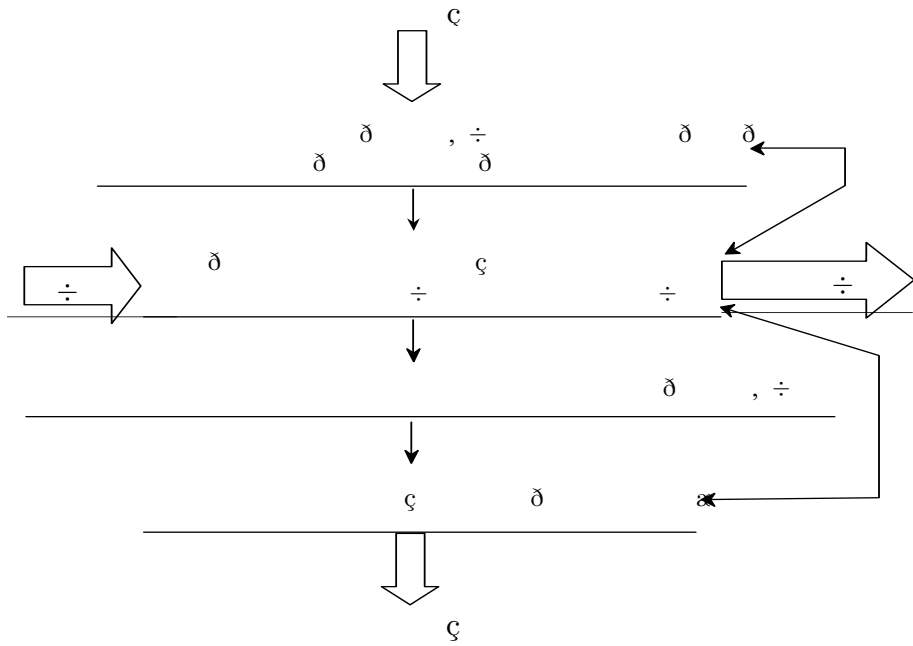


Рисунок 1 – Схема работы СОЗ



Рисунок 2 – Свойства СОЗ



Рисунок 3 – Преимущества СОЗ

Области применения систем, основанных на знаниях, весьма разнообразны:

- бизнес;
- производство;
- военные приложения;
- медицина;
- социология;
- геология;
- космос;
- сельское хозяйство;
- управление;
- юриспруденция;
- и т.д.

2. Понятие инженерии знаний.

Инженерия знаний (англ. knowledge engineering) — входит в область наук об искусственном интеллекте, связана с разработкой экспертных систем и баз знаний. Относится ко всем техническим, научным и социальным аспектам, связанным с построением, поддержкой и применением систем, использующих знания. Изучает методы и средства извлечения, представления, структурирования и использования знаний до программной реализации компонентов системы.

Инженерия знаний применяется в менеджменте знаний для организации сбора, накопления, хранения и использования знаний организации в стратегии управления знаниями, ориентированную на кодификацию знаний.

Инженерия знаний (ИЗ) была определена Фейгенбаумом и МакКордак в 1983 году как «ИЗ — раздел (дисциплина) инженерии, направленный на внедрение знаний в компьютерные системы для решения сложных задач, обычно требующих богатого человеческого опыта».

В настоящее время это также предполагает создание и обслуживание подобных систем (Кендэл, 2007). Это также тесно сопрягается с разработкой программного обеспечения и используется во многих информационных исследованиях, например таких, как исследования искусственного интеллекта, включая базы знаний, сбор данных, экспертные системы, системы поддержки принятия решений и географические информационные системы. ИЗ связана с математической логикой, также используемой в разных научных дисциплинах, например, в социологии, где целью исследований понимание социального поведения людей, анализ логики взаимоотношения людей в обществе.

Инженерия знаний представляет собой совокупность моделей, методов и технических приемов, нацеленных на создание систем, которые предназначены для решения проблем с использованием знаний. Фактически инженерия знаний – это теория, методология и технология, которые охватывают методы добычи, анализа, представления и обработки знаний экспертов.

Представление знаний, их обработка и использование, рассматриваемые применительно к конкретной прикладной области, являются предметом инженерии знаний.

Инженерия знаний включает применение единой стратегии, которая может быть реализована с помощью технических средств на этапе сбора и обработки данных. Она тесно связана и находит применение в системе автоматизированного проектирования (CAD). Инжиниринг, основанный на знаниях, объединяет по принципам объективной ориентации и правилам с CAD и другими традиционными инструментами инженерного программного обеспечения.

На высоком уровне, процесс инженерии знаний состоит из двух:

1. Извлечение знаний – преобразование «сырых знаний» в организованные.
2. Внедрение знаний – преобразование организованных знаний в реализованные.

Предметная область - сфера человеческой деятельности, выделенная и описанная согласно установленным критериям. В описываемое понятие должны входить сведения об ее элементах, явлениях, отношениях и процессах, отражающих различные аспекты этой деятельности. В описании предметной области должны присутствовать характеристики возможных воздействий окружающей среды на элементы и явления предметной области, а также обратные воздействия этих элементов и явлений на среду.

Проблемная область - комплексное понятие, включающее предметную область, решаемые задачи, цели, возможные стратегии и эвристики. Предметную область можно определить как объект или, например, производственную систему со всем комплексом понятий и знаний о ее функционировании. При исследовании проблемной области необходимы знания о задачах, решаемых в производственной системе, и стоящих перед ней целях.

При исследовании экономических систем и решаемых ими задач с целью формализации знаний в БЗ и работе необходимо учитывать специфику таких систем. Экономическим системам присуща динамичность функционирования, частая смена ситуаций, обновление больших массивов измерительных и других данных, характеризующих состояние объекта. Они часто функционируют в условиях полной определенности из-за действия случайных возмущающих факторов.

Автоматизация извлечения знаний и запись их в БЗ. Неавтоматизированный сбор знаний специалистов трудоемкий процесс. В развитых интеллектуальных системах предусматриваются вспомогательные средства для приобретения знаний.

3. Этапы получения знаний о предметной области.

Представление знаний заключается в последовательной реализации следующих этапов.

На первом этапе осуществляется описание проблемной области с помощью таксономий или онтологий [1].

Затем необходимо определить факты, правила и отношения для заданной проблемной области. Под фактами понимается описания значений данных, правила служат для интеграции фактов в каузальной форме в целях получения решения, а отношения представляют собой совокупность семантических зависимостей, используемую для интерпретации правил в стратегиях поиска решений.

На следующем этапе, на основании определенных выше фактов, правил и отношений осуществляется идентификация знаний, причем:

- факты и правила составляют главные компоненты знания;
- факты, правила и отношения определяют стратегию поиска решения;
- декларативные и процедурные знания выражаются в виде цепочки логических рассуждений в стратегиях поиска решения и лежат в основе организации логического вывода.

Очередной этап связан с разработкой моделей представления знаний на основе одного из известных подходов:

- фреймы;
- семантические сети;
- исчисление предикатов первого порядка;
- продукции;
- комбинированные модели на основе правил продукции.

Процедура лингвистической обработки проблемной информации реализует еще один этап представления знаний и заключается в формулировании предложений (совокупности фактов) и утверждений (совокупности предложений).

Следующим этапом является организация логического вывода, определяемого в виде результата совокупного применения стратегий и цепочек логических рассуждений. Цепочки логических рассуждений представляют собой интерпретацию фактов предметной области, причем может быть использована как прямая, так и обратная цепочка логического вывода. На основе цепочек логических рассуждений строится соответствующая стратегия поиска решения.

На заключительном этапе осуществляется оценка неопределенности в системах ИИ, которая может быть реализована на основе:

- коэффициентов неопределенности;
- вероятностных характеристик событий;
- факторов логической необходимости и логической достаточности событий;
- формулы Байеса для оценки вероятности событий;
- лингвистической переменной;
- переменной неопределенности.

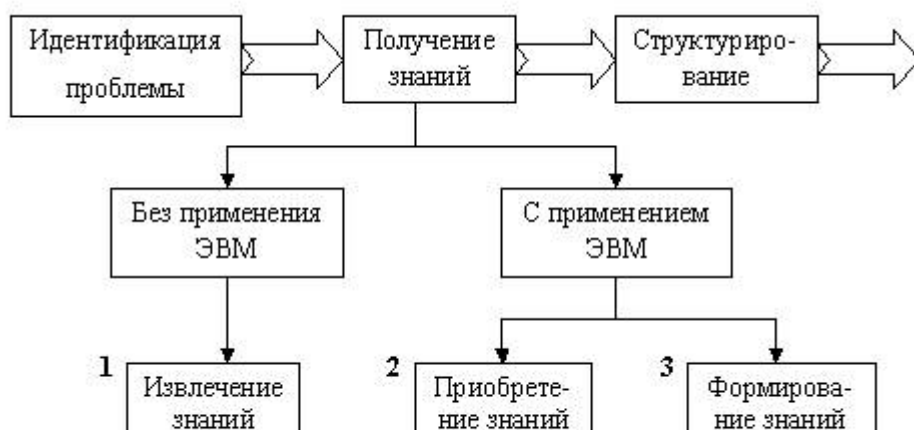
Реализация ИнС предполагает выбор инструментальных средств. При этом уже определены характеристики задачи, решаемой проектируемой экспертной системой, имеется структура, пространство поиска и пространство формируемых решений, требуется обеспечить необходимые функциональные возможности инструментального комплекса или среды разработки.

Для реализации ЭС используются как традиционные языки программирования, языки обработки списков, логического программирования, процедурные языки, так программные оболочки (среды) для создания экспертных систем, экспертные системы и системы приобретения знаний.

4. Методы получения знаний о предметной области

Существует несколько стратегий получения знаний. Наиболее распространенные:

- извлечение;
- приобретение;
- обнаружение (формирование).



Термин извлечение знаний касается непосредственного живого контакта инженера по знаниям и источника знаний. Авторы склонны использовать этот термин как более емкий и более точно выражающий смысл процедуры переноса компетентности эксперта через инженера по знаниям в базу знаний экспертной системы. Извлечение знаний - это процедура взаимодействия инженера по знаниям с источником знаний, в результате которой становятся явными процесс рассуждений специалистов-экспертов при принятии решения и структура их представлений о предметной области.

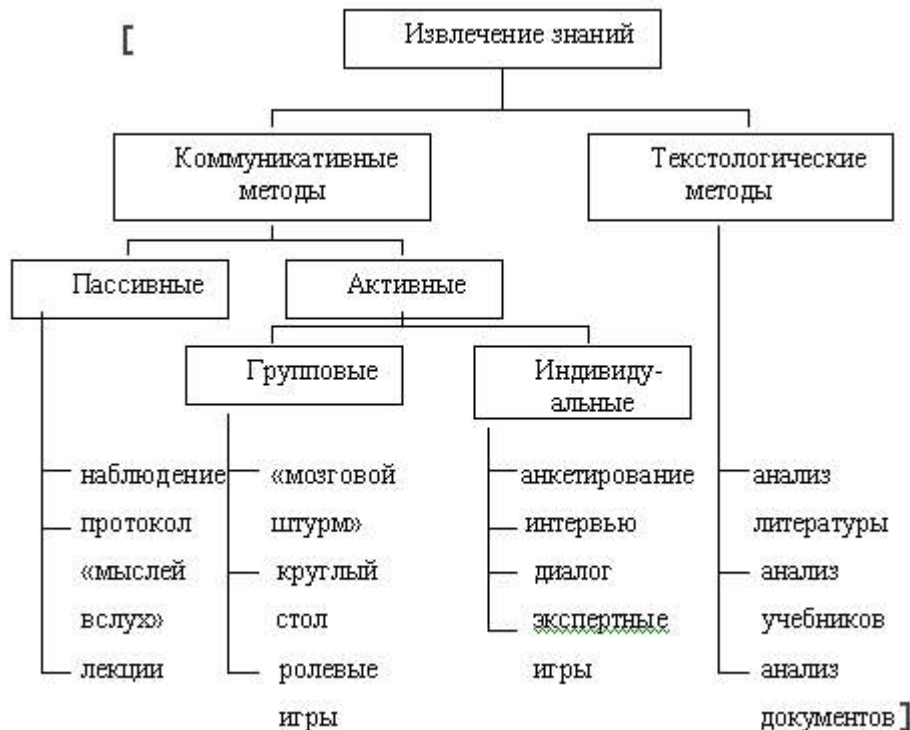
Приобретение знаний для системы, основанной на знаниях, осуществляется у эксперта в заданной прикладной области, когнитологом (инженером по знаниям).

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.

Вопросы для самоконтроля и текущего контроля

- 1) Дайте определение системы искусственного интеллекта.
- 2) В каких случаях оправдано создание системы искусственного интеллекта?
- 3) Охарактеризуйте понятие «знание».
- 4) В чем заключаются различия между декларативными и процедурными знаниями?

Задание на самостоятельную работу

- 1) Практическое задание

Выполнить создание базы знаний по предметной области, определенной номером варианта практической работы №2.

- 2) Самостоятельная работа

Исследовать примеры современных систем основанных на знаниях.

Список рекомендуемой литературы¹:

[1] Сидоркина, Ирина Геннадьевна. Системы искусственного интеллекта [Текст] : [учеб. пособие для вузов по направлению 230100 "Информатика и вычисл. техника"] / И. Г. Сидоркина. М.: Кнорус, 2011. - 245 с.

Рассмотрены теоретические аспекты проектирования систем искусственного интеллекта и даны модели представления знаний, изложены материалы информационного, справочного, консультирующего характера по использованию инструментальных средств и технологическим особенностям разработки систем данного класса.

Подробнее: <https://www.labyrinth.ru/books/276217/>

[2] Богомолова, М. А. Экспертные системы (техника и технология проектирования) [Текст] : Методические указания к лабораторным работам / М. А. Богомолова. Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015. - 47 с.

Методические указания являются руководством при выполнении лабораторных занятий, проводимых по курсу «Экспертные системы (техника и технология проектирования)» с магистрами направления 09.04.03 - «Прикладная информатика» в терминальном классе. Лабораторные работы связаны с изучением принципов построения экспертных систем на основе различных моделей представления знаний, и могут быть полезны студентам других специальностей.

Подробнее: <https://www.iprbookshop.ru/71908.html>

[3] Советов, Борис Яковлевич. Представление знаний в информационных системах [Текст] : [учеб. для вузов по направлению подгот. "Информ. системы и технологии"] / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. М.: Академия, 2011.

Учебник создан в соответствии с Федеральным государственным образовательным стандартом по направлению бакалавриата «Информационные системы и технологии». Рассмотрены современное состояние теоретических и прикладных вопросов представления знаний в информационных системах, идеология построения интеллектуальных систем, математический аппарат представления знаний,

возможности и пути использования искусственного интеллекта при проектировании информационных систем. Изложены новые аспекты представления знаний на основе искусственных нейронных сетей, расчетно-логических систем, генетических алгоритмов, мультиагентных систем. Приведены примеры практической реализации представления знаний на базе декларативного языка Пролог. Для студентов учреждений высшего профессионального образования. Может быть полезен разработчикам и пользователям информационных систем; преподавателям и научным сотрудникам, сферой интересов которых является интеллектуализация различных предметных областей; менеджерам и руководителям различного ранга, желающим самостоятельно ознакомиться с современным состоянием информационных технологий.

Подробнее: <https://lavkababuin.com/predstavlenie-znaniy-v-informacionnyh-sistemah-2-e-izd-ster-650564/>

[4] *Остроух, А. В. Интеллектуальные информационные системы и технологии [Электронный ресурс] : монография / А. В. Остроух, А. Б. Николаев. Санкт-Петербург: Лань, 2019. - 308 с.*

В монографии изложены концептуальные основы и методы представления знаний в интеллектуальных системах. Рассмотрены различные подходы, применяемые при проектировании и разработке интеллектуальных систем и технологий в транспортном комплексе, а также рассмотрены тенденции развития систем искусственного интеллекта.

Монография может быть использована для формирования профессиональной компетентности студентов высших учебных заведений, аспирантов и научных сотрудников обучающихся и ведущих научные исследования в области разработки и практического применения систем искусственного интеллекта по укрупнённой группе направлений подготовки "Информатика и вычислительная техника".

Подробнее: <https://www.labyrinth.ru/books/692861/>

Ссылки на открытые ресурсы (он-лайн-курсы, видеоуроки и т.д.)

[5]

ⁱСписок рекомендуемой литературы может быть более широким по сравнению со списком литературы, указанным в рабочей программе

Прикладные интеллектуальные системы

Конспект лекции 4. Инструмент Microsoft Analysis Services в процессе анализа данных

Аннотация: В лекции рассказывается о понятии систем основанных на знаниях и понятии инженерии знаний. Рассмотрены этапы и методы получения знаний о предметной области в инженерии знаний.

План лекции:

1. Службы SQL Server Analysis Services.
2. Понятие инженерии знаний.
3. Этапы получения знаний о предметной области.
4. Методы получения знаний о предметной области
5. Задания для самостоятельной работы.

Ключевые слова: архитектура служб Analysis Services, средство разработки Business Intelligence Development Studio (BIDS), Big Data, анализ данных, извлечение данных, извлечение знаний.

СОДЕРЖАНИЕ ЛЕКЦИИ:

1. Службы SQL Server Analysis Services

Службы SQL Server Analysis Services (службы SSAS) являются группой сервисов, которые используются для управления данными, хранящимися в хранилищах или киосках данных. Посредством служб SSAS данные из хранилища данных можно организовать в многомерные кубы (см. главу 21) с агрегированными данными, чтобы получить возможность создавать сложные запросы и подробные отчеты. Основными особенностями служб SSAS являются следующие:

- ♦ удобство использования;
- ♦ поддержка различных архитектур;
- ♦ поддержка некоторых интерфейсов API.

Службы SSAS предоставляют мастеров почти для каждой задачи, выполняющейся в процессе проектирования и реализации хранилища данных. Например, мастер указания источников данных *Data Source Wizard* позволяет задать один или несколько источников данных, а мастер кубов *Cube Wizard* применяется для создания многомерных кубов для хранения агрегированных данных. Кроме этого, простота использования обеспечивается средством *Business Intelligence Development Studio (BIDS)*, с помощью которого можно разрабатывать базы данных и другие объекты хранилищ данных. Этому способствует то, что средство BIDS предоставляет один общий интерфейс для разработки проектов SSAS, а также SQL Server Integration Services (SSIS) и SQL Server Reporting Services (SSRS).

В отличие от большинства других систем хранилищ данных, службы SSAS позволяют использовать ту архитектуру, которая наиболее отвечает имеющимся требованиям. В частности, можно выбирать между тремя архитектурами хранилищ для данных (MOLAP, ROLAP и HOLAP).

Службы SSAS предоставляют много различных интерфейсов API для извлечения и доставки данных. Одним из этих интерфейсов является OLE DB для взаимодействия с OLAP, который позволяет обращаться к данным кубов SSAS.

1.1 Терминология служб SSAS

Далее приводятся наиболее важные термины служб SSAS

Куб (cube) — это многомерная структура, содержащая все или часть данных из хранилища данных. Хотя термин "куб" предполагает три измерения, многомерный куб обычно может иметь намного больше измерений. Каждый куб содержит все другие элементы, перечисленные в предыдущем списке.

Измерение (dimension) — это набор логически связанных атрибутов (хранящихся вместе в таблице измерения), которые подробно описывают меры (хранящиеся в таблице фактов). Например, Time, Product и Customer будут типичными измерениями, используемыми во многих приложениях бизнес-аналитики. (Эти три измерения входят в базу данных AdventureWorksDW и используются в примере в следующем разделе для

демонстрации создания и обработки многомерного куба, применяя средство BIDS.) Каждое измерение содержит дискретные значения, называемые *членами измерения* (member). Например, членами измерения Product могут быть Computers, Disks и CPUs. Каждый член может быть вычисляемым, что означает, что его значение вычисляется в процессе выполнения, используя выражение, которое задается при определении члена. (Поскольку вычисляемые члены не сохраняются на диске, то можно добавлять новых членов, не увеличивая объем соответствующего куба.)

Иерархии (hierarchy) задают группирование множественных членов в каждом измерении. Они применяются для уточнения запросов, связанных с анализом данных. *Ячейки* (cell) являются частями многомерного куба, которые определяются координатами (x, y и z для трехмерного куба). Это означает, что ячейка является набором, содержащим членов из каждого измерения. Рассмотрим, например, трехмерный куб из главы 21 (см. рис. 21.3), который представляет продажи легковых автомобилей для одного региона за один квартал. К этому кубу принадлежат, среди прочих, ячейки со следующими координатами:

- ◆ 1-й квартал, Южная Америка, Falcon;
- ◆ 3-й квартал, Азия, Eagle.

Иерархии определяются на основе их уровней. Иными словами, *уровни* (level) описывают иерархию от самого высшего уровня данных (наиболее обобщенной) до самого низшего (наиболее подробной). В следующем списке приводятся возможные уровни иерархии для измерения Time:

- ◆ Quarter (Q1, Q2, Q3, Q4);
- ◆ Month (January, February, ...);
- ◆ Day (Day1, Day2, ...).

Как уже упоминалось в главе 21, *меры* (measure) — это числовые значения, такие как цена или количество, которые присутствуют в таблице фактов, но не используются в ее первичном ключе. *Группа мер* (measure group) — это набор мер, которые совместно создают логическую единицу для целей деловой деятельности. Каждая группа мер создается на лету, используя соответствующую информацию из метаданных.

Для куба можно задать одну или несколько секций. *Секции* (partition) используются службами SSAS для управления и хранения данных и агрегированных данных для группы мер куба. Каждая группа мер имеет, по крайней мере, одну секцию, которая создается при определении этой группы мер. Секции являются мощным и гибким средством управления больших кубов.

1.2 Разработка многомерного куба, используя средство BIDS

Основным компонентом служб SSAS является средство управления Business Intelligence Development Studio (средство BIDS), которое предоставляет единую платформу для разработки разных приложений бизнес-аналитики. Средство BIDS основано на Visual Studio и поддерживает интегрированную платформу для разработчиков систем бизнес-аналитики.

С помощью средства BIDS можно не только создавать и управлять кубами, но так же разрабатывать возможности для служб отчетов SQL Server Reporting Services (SSRS) и интеграционных служб SQL Server Integration Services (SSIS). (Службы SSRS рассмотрены в главе 24, но предмет служб SSIS выходит за рамки тематики этой книги.)

Процесс создания и обработки многомерного куба посредством средства BIDS состоит из следующих шагов:

1. Создание проекта бизнес-аналитики.
2. Определение источников данных.
3. Создание представлений источников данных.
4. Создание куба.
5. Проектирование агрегирования для хранилища.
6. Обработка куба.
7. Просмотр куба.

Все эти шаги подробно описываются в последующих разделах.

1.3 Создание проекта бизнес-аналитики

Первым шагом в разработке приложения бизнес-аналитики будет создание нового проекта в BIDS. Чтобы запустить средство BIDS, выполните последовательность команд меню Пуск | Все программы | Microsoft SQL Server 2012 | SQL Server Business Intelligence Development Studio. Далее выберите File | New Project. В панели Installed Templates диалогового окна New Project (рис. 22.1) разверните узел Business Intelligence и выберите в правой панели окна пункт Analysis Services Multidimensional and Data Mining Project.

В поля Name и Location введите имя и соответственно место расположения проекта. Для этого примера назовем проект BI_Project (рис. 22.1). Затем, чтобы создать новый проект, нажмите кнопку ОК.

Новый проект всегда создается в новом решении (solution). Следовательно, решение является самой большой единицей управления в средстве BIDS и содержит один или несколько проектов. (В данном примере имя решения такое же, как и имя проекта.)

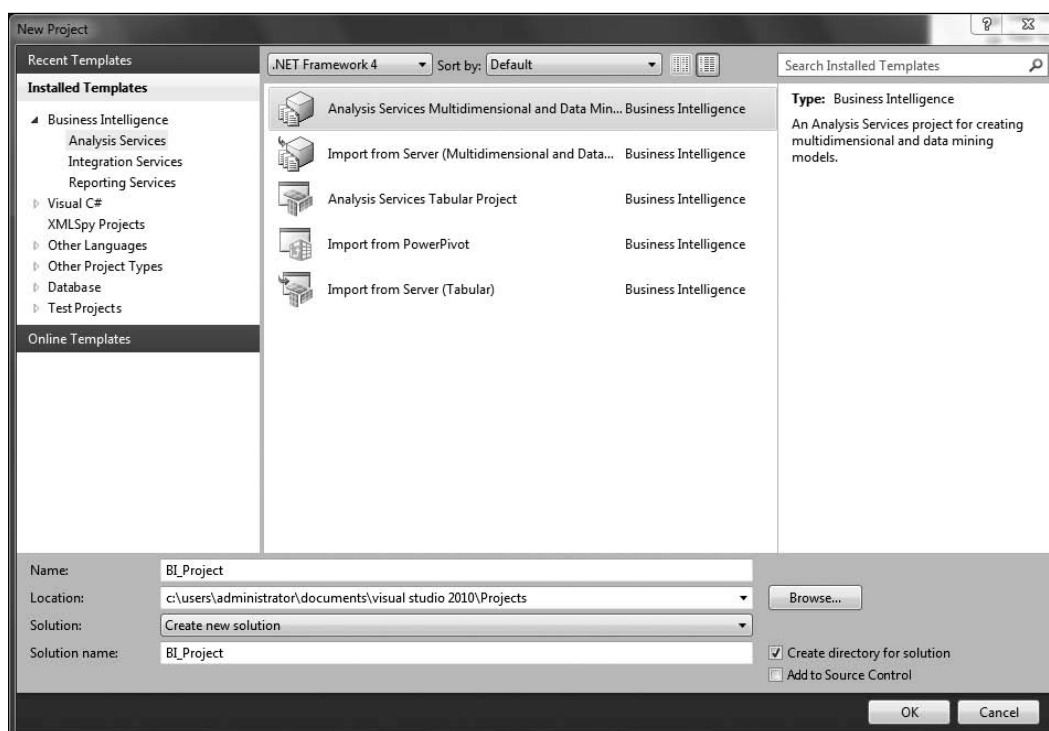


Рис. 1. Диалоговое окно New Project средства BIDS

1.4 Определение источников данных

Чтобы задать источники данных, в панели Solution Explorer щелкните правой кнопкой папку Data Sources и в появившемся контекстном меню выберите пункт New Data Source. Откроется мастер источников данных Data Source Wizard, который будет предоставлять инструкции по созданию источника данных. (В данном примере в качестве источника данных используется база данных примера SQL Server с именем AdventureWorksDW.)

Сначала на странице Select How to Define the Connection установите переключатель Create a Data Source Based on an Existing or New Connection и нажмите кнопку New. В открывшемся диалоговом окне Connection Manager (рис. 22.2) выберите в раскрывающемся списке Provider опцию Native OLE DB\SQL Server Native Client 11.0, а в раскрывающемся списке Server name выберите имя требуемого сервера баз данных. (Выбор поставщика Native OLE DB\SQL Server Native Client 11.0 позволяет подключаться к существующей базе данных экземпляра сервера Database Engine.) В этом же диалоговом окне установите переключатель Use Windows Authentication; здесь же установите переключатель Select or enter a database name и в раскрывающемся списке под ним выберите базу данных AdventureWorksDW. Проверьте соединение с базой данных, нажав кнопку Test Connection, и, при положительном результате проверки, нажмите кнопку OK. (Страница Select How to Define the Connection отображается только при первом соединении с базой данных.)

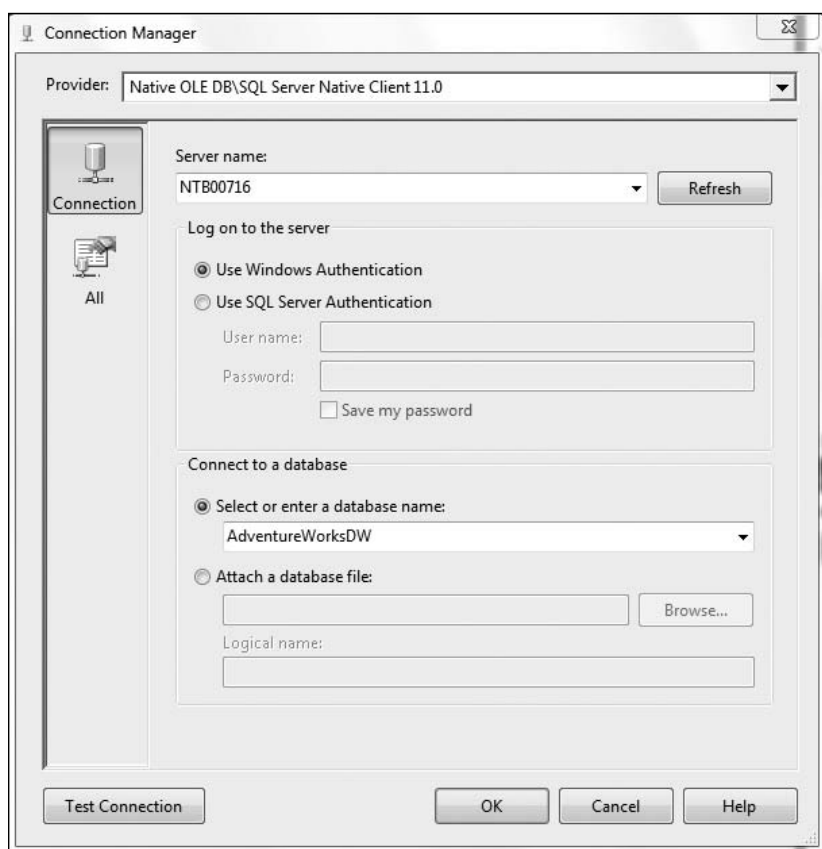


Рис. 2. Диалоговое окно Connection Manager

Откроется следующая страница мастера, Impersonation Information, параметры которой определяют учетную запись пользователя, применяемую службами SSAS при соединении с источником данных, используя проверку подлинности Windows. Выбор определенной настройки зависит от способа использования этого источника данных. Установите переключатель Use a Specific Windows User Name and Password, введите в соответствующие поля свои имя пользователя и пароль и нажмите кнопку Next.

На следующей странице мастера, Completing the Wizard, присвойте новому источнику данных имя (для этого примера это будет BI_Source) и нажмите кнопку Finish. Созданный источник данных отобразится в папке Data Sources панели Solution Explorer.

После определения источника данных нужно указать точные данные, которые требуется выбрать из него. В нашем примере это означает, что нужно выбрать таблицы из базы данных AdventureWorksDW, которые будут использоваться для создания куба. Для этого нужно указать представления источника данных; этот процесс рассматривается в следующем разделе.

1.5 Создание представлений источников данных

Чтобы создать представления источников данных, в панели Solution Explorer щелкните правой кнопкой папку Data Sources Views и в контекстном меню выберите пункт New Data Source View. Откроется мастер Data Source View Wizard, который будет предоставлять инструкции по всем шагам, требуемым для создания представления источника данных. (В этом примере создается представление с именем BI_View, которое основывается на таблицах Customer и Project.)

Сначала на странице Select a Data Source выберите существующий реляционный источник данных (для этого примера выберите BI_Source) и нажмите кнопку Next. На следующей странице мастера — Select Tables and Views — добавьте таблицы, входящие в куб как таблицы измерений или как таблицы фактов. Чтобы добавить таблицу, в панели Available objects выберите ее имя и нажмите кнопку >, чтобы переместить эту таблицу в панель Included Objects. Для этого примера выберите в базе данных AdventureWorksDW таблицы для клиентов и продуктов (Dim Customer и Dim Product соответственно). Эти таблицы будут использоваться для создания куба измерений. Они создают набор таблиц измерения для схемы типа "звезда".

Далее, на этой же странице мастера нужно указать одну или несколько таблиц фактов, которые соответствуют выбранным таблицам измерений. (Одна таблица фактов вместе с соответствующими таблицами измерений создает схему типа "звезда".)

Для этого внизу панели Included objects нажмите кнопку Add Related Tables. Таким образом мы даем задание системе найти таблицы, которые связаны с таблицами DimCustomer и DimProduct. (Чтобы найти связанные таблицы, система просматривает все зависимости "первичный ключ/внешний ключ", которые существуют в базе данных.)

Система обнаружит несколько таблиц фактов и поместит их в панель Included objects.

Для создания схемы типа "звезда" из всех этих таблиц требуется только одна:

FactInternetSales. Кроме соответствующих таблиц фактов, система также выполняет поиск и добавляет другие таблицы, созданные отдельно для уровня иерархии соответствующего измерения. Одной из таких таблиц, которую следует оставить, является таблица DimProductSubcategory, которая воплощает уровень иерархии Subcategory измерения Product. Также следует оставить таблицу Dim Date, т. к. измерение Time почти всегда является частью куба.

Таким образом, для нашей схемы типа "звезда" требуются следующие пять таблиц (как показано на рис. 22.3):

- ◆ FactInternetSales;
- ◆ DimCustomer;
- ◆ DimDate;
- ◆ DimProduct;
- ◆ DimProductSubcategory.

Удалите все другие выбранные системой таблицы из правой панели, выбрав ненужную таблицу и нажав кнопку <

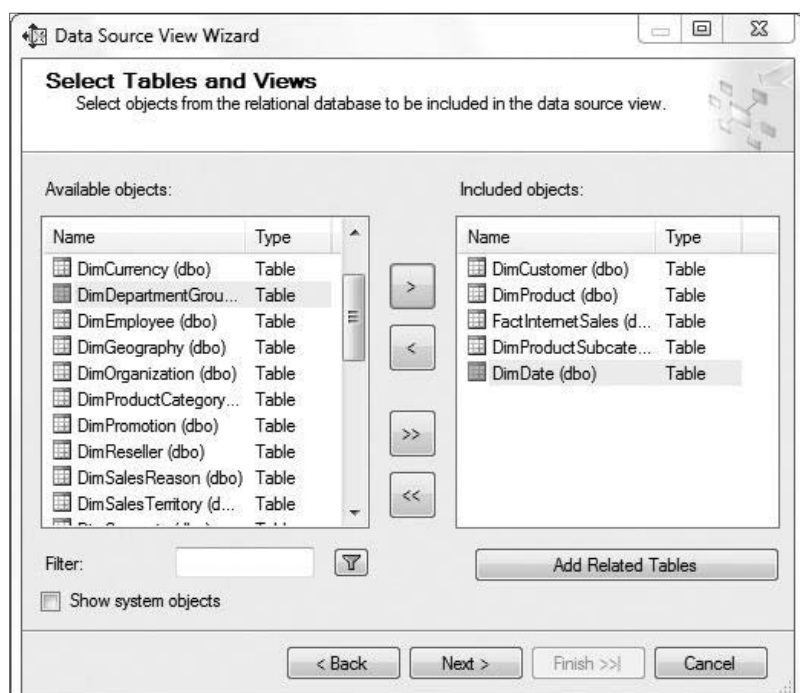


Рис. 3. Страница **Select Tables and Views** мастера **Data Source View Wizard**

Завершив отбор таблиц, нажмите кнопку **Next**. На следующей странице мастера, **Completing the Wizard**, укажите имя для нового представления источника данных (для нашего примера это будет **BI_View**) и нажмите кнопку **Finish**.

После нажатия кнопки **Next** и небольшого времени обработки в окне **Data Source View Designer** будет отображено графическое представление выбранных таблиц в определенной нами схеме данных, как это показано на рис. 22.4.

(Инструмент **Data Source View Designer** используется для отображения графического представления схем данных.)

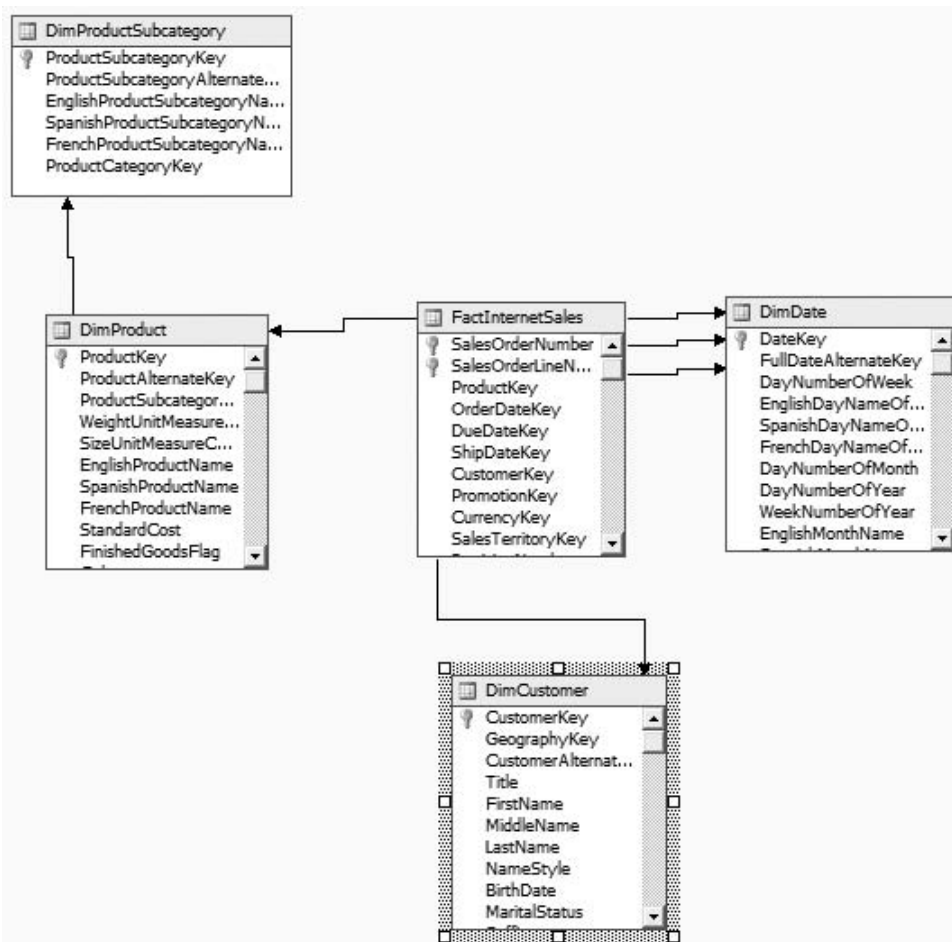


Рис. 4. Отображение выбранных таблиц в окне Data Source View Designer

Инструмент Data Source View Designer предоставляет несколько полезных функций. Для перемещения по объектам представления источника данных нужно навести указатель мыши на значок скрещенных двунаправленных стрелок в правой нижней части окна. Когда форма указателя мыши сменится на подобную этому значку, нажмите и удерживайте левую кнопку мыши. В правом нижнем углу панели просмотра схемы откроется небольшое окно навигации, в котором можно перемещаться в любую часть диаграммы. Эта функциональность особенно полезна для диаграмм с большим количеством объектов. Альтернативно, перемещаться по диаграмме можно с помощью обычных полос прокрутки (горизонтальной и вертикальной), что более подходит для диаграмм небольшого размера. Для просмотра данных таблицы щелкните требуемую таблицу правой кнопкой мыши и в контекстном меню выберите пункт **Explore Data**. Содержимое таблицы отобразится в отдельном окне.

Можно также создавать именованные запросы, которые сохраняются на постоянной основе, что позволяет обращаться к ним как к любой таблице. Чтобы создать именованный запрос, выберите пункт меню **Data Source View**, а затем пункт **New Named Query**. Откроется диалоговое окно **Create Named Query**, в котором можно создавать любые запросы для выбранных таблиц.

1.6 Создание куба

Прежде чем создавать куб, нужно задать один или несколько источников данных и создать представление источника данных, как было описано ранее в предшествующей

щих разделах. После этого можно приступить к созданию куба.

Для этого в обозревателе решений щелкните правой кнопкой мыши папку Cubes проекта BI_Project и в контекстном меню выберите пункт New Cube. Откроется страница приветствия мастера кубов. Нажмите кнопку Next. На странице Select Creation Method выберите переключатель Use Existing Tables, поскольку представление источника данных уже существует и его можно использовать для создания куба. Нажмите кнопку Next.

На странице Select Measure Group Tables выбираются меры из таблиц фактов. Выберите здесь таблицу Fact Internet Sales и нажмите кнопку Next. Мастер отберет все возможные меры из указанной таблицы фактов и отобразит их на странице Select Measures. Установите только флажок столбца Total Product Costs таблицы Fact Internet Sales только для одной меры (рис. 22.5) и нажмите кнопку Next.

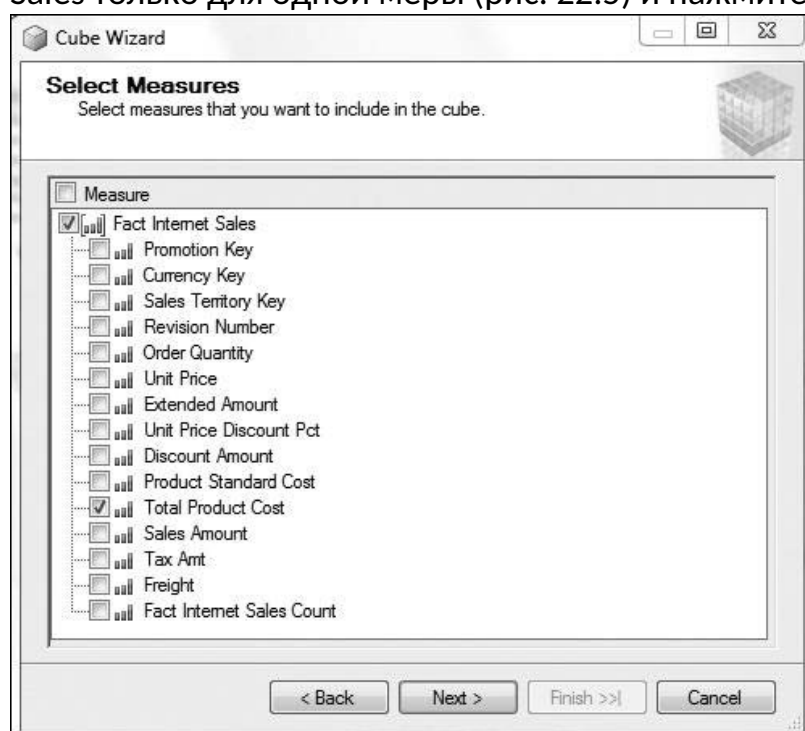


Рис. 5. Таблица мастера Select Measures

На странице Select New Dimensions выберите все три измерения (Dim Date, Dim Product и Dim Customer), которые нужно создать на основе имеющихся таблиц, и нажмите кнопку Next. На последней странице мастера Completing the Wizard отображается сводная информация по всем выбранным мерам и измерениям. Введите имя создаваемого куба (в данном примере BI_Cube) и, чтобы создать этот куб, нажмите кнопку Finish.

1.7 Проектирование агрегирования для хранилища

Как упоминалось в главе 21, базовые данные из таблицы фактов можно суммировать наперед и сохранять результаты в постоянных таблицах. Этот процесс называется *агрегированием* и может существенно повысить скорость выполнения запросов, поскольку сканирование миллионов строк по ходу выполнения запроса для вычисления агрегатного значения может занять очень длительное время.

Существует прямая взаимосвязь между требованиями дискового пространства для хранения агрегированных значений и процентом всех возможных агрегирований, которые вычисляются и сохраняются. Создание всех возможных агрегированных значений куба и сохранение их на диске позволяет получить наилучшее время выполнения всех запросов. Недостатком этого подхода являются значительные требования процессорных ресурсов для вычисления агрегированных значений и дискового пространства для их хранения.

С другой стороны, если не вычислять наперед и не сохранять никаких агрегаций, не будет требоваться дополнительного дискового пространства, однако выполнение запросов, содержащих агрегатные функции, будет медленным, вследствие необходимости вычисления каждого агрегатного значения в ходе выполнения запроса. Для разработки оптимальных агрегаций службы SSAS предоставляют мастер Aggregation Design Wizard. Чтобы запустить этот мастер, сначала нужно запустить средство Cube Designer. (Средство Cube Designer применяется для редактирования разных свойств существующих кубов, включая группы мер и отдельные меры, измерения куба и взаимосвязи между измерениями.) Для этого щелкните правой кнопкой требуемый куб в обозревателе решений и в контекстном меню выберите пункт Open или View Designer. В открывшемся окне средства Cube Designer откройте вкладку Aggregations. В отображенной таблице (Fact Internet Sales) щелкните правой кнопкой ячейку в столбце Aggregations и в контекстном меню выберите пункт Design Aggregations. Будет запущен мастер Aggregation Design Wizard.

На первой странице мастера, Review Aggregation Usage, можно просмотреть и откорректировать параметры агрегирования. В частности, можно включить или отключить атрибуты, отображенные на этой странице. Оставьте параметры как они есть и нажмите кнопку Next.

На следующей странице, Specify Object Counts, нужно указать количество членов в каждом атрибуте. Прежде чем мастер может начать создать и сохранять выбранные агрегации, для каждого выбранного объекта куба требуется предоставить число значений или число секций. Эту задачу можно выполнить с помощью мастера, нажав кнопку Count. На рис. 22.6 показана страница Specify Object Counts с результатами этих вычислений мастера. Нажмите кнопку Next.

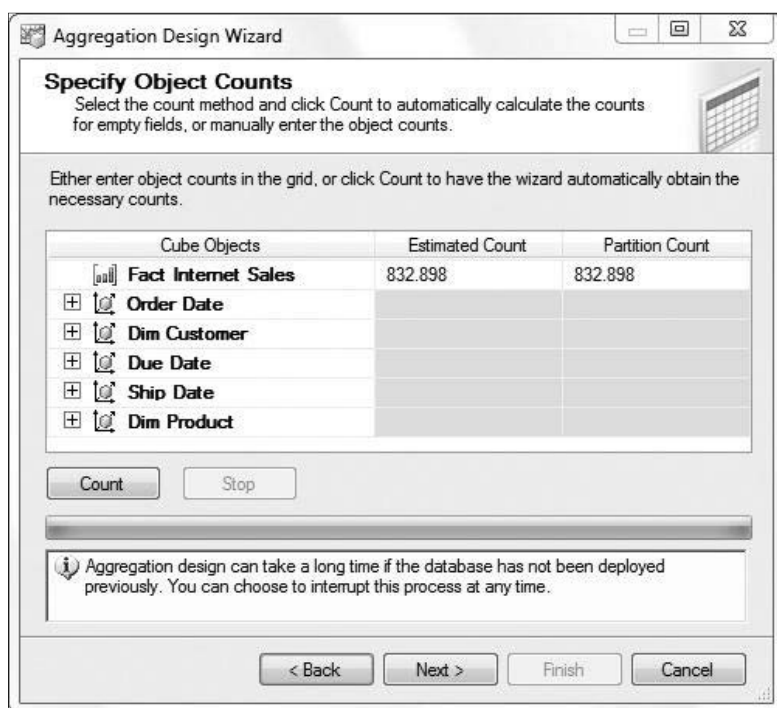


Рис.6. Страница мастера Specify Object Counts после подсчета мастером объектов. На предпоследней странице, Set Aggregation Options, для указания точки, до которой нужно проектировать агрегации (или не проектировать их вообще), выберите одну из четырех следующих опций:

- ◆ Estimated storage reaches_MB (предполагаемый объем хранилища достигнет_Мбайт) — задает максимальный объем дискового хранилища, который следует использовать для вычисленных наперед агрегаций. Чем больше это значение, тем большее количество вычисленных наперед агрегаций будет создано;
- ◆ Performance gain reaches_% (прирост производительности достигнет_%) — задает прирост производительности, которую требуется получить. Чем больше процентное значение вычисленных наперед агрегаций, тем лучше будет производительность запросов;
- ◆ I click Stop (я нажму кнопку Stop) — позволяет пользователю остановить процесс проектирования в любой момент по своему желанию;

Do not design aggregation (0%) (не проектировать агрегирование) — указывает не создавать вычисляемые наперед агрегации.

На рис. 7 показан результат выбора второй опции, для которой было установлено значение 80%.

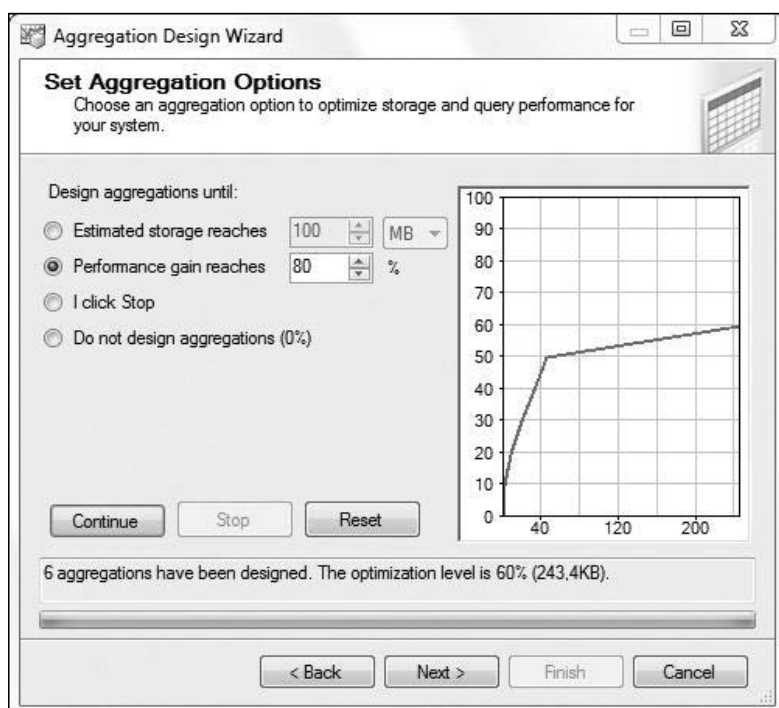


Рис. 7. Результат выбора проектирования агрегаций для получения 80% прироста производительности

Система создала шесть агрегаций и использует 243,4 Кбайт для их хранения. Чтобы перейти к странице *Completing the Wizard*, нажмите кнопку *Next*. На этой странице можно выбрать метод применения полученных агрегаций: немедленно (переключатель *Deploy and process now*) или позже (переключатель *Save the aggregations but do not process them*). Выберите второй переключатель и нажмите кнопку *Finish*.

1.8 Обработка куба

Если в предшествующем разделе вы выбрали рекомендованную опцию *Save the aggregations but do not process them*, то теперь нужно выполнить обработку куба. Куб требуется обрабатывать после его создания и после каждого его изменения. Если куб содержит большой объем данных и вычисляемых наперед агрегаций, его обработка может занимать очень длительный промежуток времени. Чтобы начать обработку куба, в папке *Cubes* обозревателя решений щелкните правой кнопкой мыши требуемый куб и в контекстном меню выберите пункт *Process*. Система начинает обработку куба, отображая ход выполнения этого процесса (рис. 8).

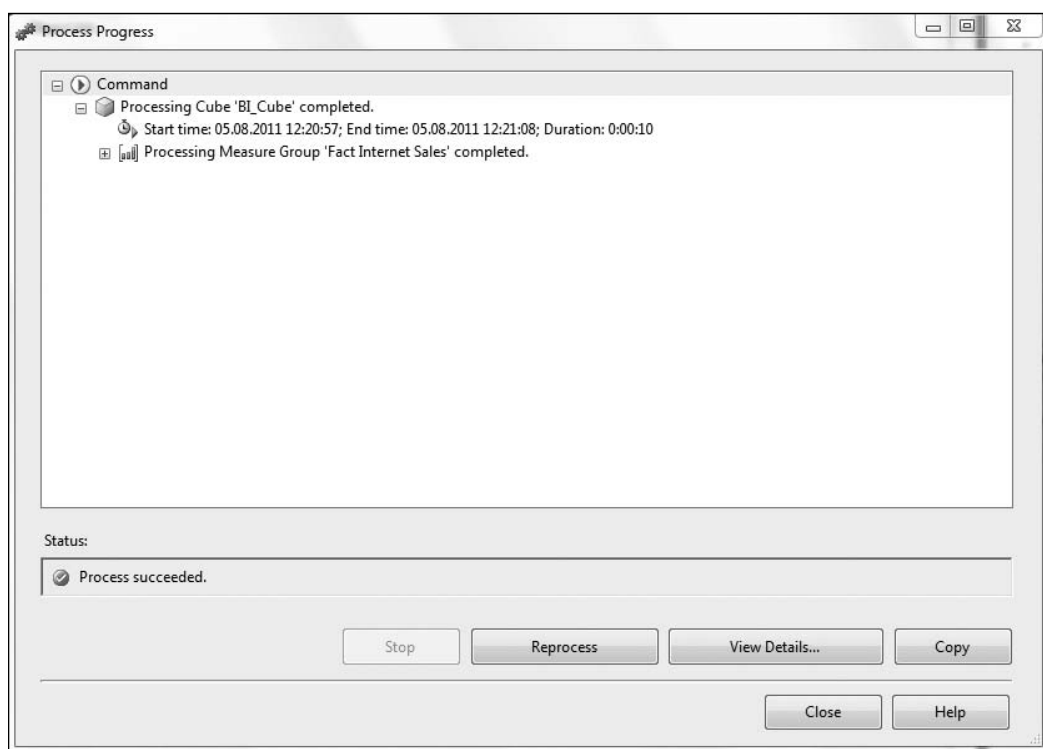


Рис. 8. Окно обработки куба с индикатором хода выполнения

1.9 Просмотр куба

Для просмотра куба, щелкните правой кнопкой мыши требуемый куб и в контекстном меню выберите опцию Browse. Откроется окно просмотра куба Browse. Здесь в запрос можно вставить любое измерение, щелкнув правой кнопкой имя требуемого измерения в левой панели и выбрав в контекстном меню опцию Add to Query. Таким же образом к запросу добавляются и меры. (Меры рекомендуется добавлять первыми.) На рис. 22.9 показано табличное представление общей стоимости продуктов для продаж типа Internet Sales для разных клиентов и продуктов.

Для вычисления значений мер для определенных измерений и их иерархий применяется другой подход. Предположим, например, что нам нужно предоставить клиентам с идентификаторами 11008 и 11741 информацию об общей стоимости всех

Product Key	Customer Key	Total Product Cost
1	21768	15375076569
2	21768	15140459117
3	21768	15360315461
4	21768	15792196904
5	21768	15041739727
6	21768	15506999608
7	21768	15145385055
8	21768	15236820640
9	21768	15664781579
10	21768	15607907113
11	21768	15637123233
12	21768	15673286113
13	21768	15943888060
14	21768	15764363325
15	21768	15780568093
16	21768	15184801044
17	21768	15677256663
18	21768	15214147584
19	21768	15651406355
20	21768	15510412068
21	21768	15396149402
22	21768	15504415049
23	21768	15466773605
24	21768	14956504459
25	21768	15362250773
26	21768	15676839216

Рис. 9. Общая стоимость продаж типа Internet Sales для разных клиентов и продуктов

продуктов, которые они заказали в период между 1/03/2006 и 31/08/2006. Для этого мы сначала перетаскиваем меру Total Product Cost с левой панели в панель редактирования, а затем в расположенной выше панели выбираем значения, чтобы ограничить условия для каждого измерения (рис. 22.10).

Сначала в столбце Dimension выбираем таблицу Dim Customer, а в столбце Hierarchy первичный ключ этой таблицы: Customer Key. Затем в столбце Operator выбираем Equal, а в столбце Filter Expression — значения 11008 и 11741.

Таким же образом задаются условия для таблицы измерений Dim Product с единственной разницей, что вместо указания только двух значений, выбирается опция All с помощью установки соответствующего флажка в столбце Filter Expression. (Корень каждого измерения представляется значением All.) Наконец, для таблицы измерения Dim Date выбирается столбец Order Date с соответствующим ключом. В этом случае в столбце Operator выбирается значение Range (Inclusive), а в столбце Filter Expression — начальная и конечная дата требуемого периода. Общая сумма продаж по заданным фильтрам составляет 2171.2942, как это можно видеть на рис. 10.

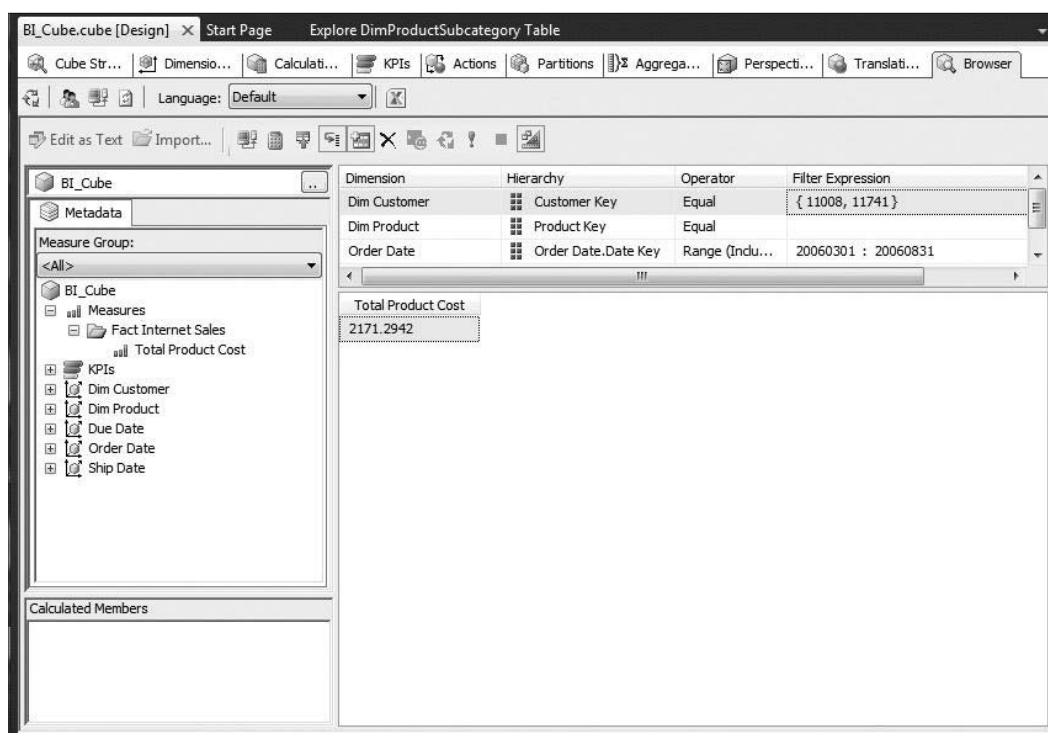


Рис. 10. Вычисление общей стоимости продаж продуктов определенным клиентам за определенный период времени

1.10 Извлечение и доставка данных

Теперь, когда мы знаем, как создавать и просматривать куб с помощью средства BIDS, можем перейти к рассмотрению того, как извлекать из куба данные и доставлять их пользователям. Основной целью средства Development Studio является разработка проектов бизнес-аналитики, а не извлечение данных и доставка их пользователям. Для этой задачи существует много других интерфейсов, включая следующие:

- ◆ надстройка PowerPivot for Excel;
- ◆ язык запросов Multidimensional Expressions (MDX);
- ◆ среда Management Studio;
- ◆ инструментальный набор OLE DB for OLAP;
- ◆ инструменты сторонних разработчиков;
- ◆ поставщик данных ADOMD.NET.

Надстройка PowerPivot for Excel и язык запросов MDX рассматриваются каждый отдельно в последующих двух разделах, а остальные средства, из ранее приведенного списка, рассмотрены здесь вкратце. Этим двум средствам уделено большое внимание, чем остальным по той причине, что надстройка PowerPivot for Excel является самым важным интерфейсом для конечных пользователей, а язык запросов MDX используется во многих SSAS-решениях сторонних разработчиков.

Для просмотра куба в среде Management Studio запустите этот инструмент и подключитесь к серверу служб SSAS, на котором развернут требуемый куб. В обозревателе объектов разверните папку Database, а в ней разверните папку Cube, которая содержит все кубы, созданные для данной базы данных. Щелкните правой кнопкой мыши требуемый куб и, в появившемся контекстном меню, выберите пункт

Browse. Интерфейс для просмотра кубов точно такой же, как и для их разработки. Это можно видеть на рис. 22.9 и 22.10. Поэтому просмотр данных куба можно осуществлять таким же способом, как описано ранее в разд. "Просмотр куба" в этой главе.

Инструментальный набор OLE DB for OLAP является промышленным стандартом для обработки многомерных данных, предлагаемый компанией Microsoft. Этот набор инструментальных средств и интерфейсов расширяет возможности OLE DB, предоставляя доступ к многомерным складам данных. Инструментальный набор OLE DB for OLAP позволяет выполнять анализ данных посредством интерактивного обращения к разнообразным возможным представлениям данных. Многие независимые разработчики программного обеспечения используют спецификацию OLE DB for OLAP для реализации разных интерфейсов, позволяющих пользователям обращаться к кубам, созданным посредством служб SSAS. Кроме этого, используя OLE DB for OLAP, разработчики могут реализовывать приложения OLAP, которые могут одинаково обращаться как к реляционным, так и нереляционным данным, хранящимся в различных источниках информации, независимо от их расположения или типа.

Поставщик данных ADOMD.NET (ActiveX Data Objects Multidimensional) предназначен для взаимодействия со службами SSAS. С помощью этого интерфейса можно обращаться к данным в многомерном кубе и манипулировать ими, что предоставляет возможность разработки приложений OLAP на основе веб-технологий. Этот интерфейс использует протокол XML for Analysis для взаимодействия с источниками аналитических данных. Команды обычно отправляются на языке MDX. С помощью ADOMD.NET можно также просматривать метаданные и работать с ними.

11. Обращение к данным посредством PowerPivot for Excel

Надстройка PowerPivot for Excel позволяет анализировать данные, используя наиболее популярное средство для этой цели — Microsoft Excel. Это удобный для пользователя способ выполнять анализ данных, используя такие функциональности, как PivotTable, PivotChart представления и срезы (slices).

Прежде чем приступить к изучению, как работать с этим инструментом, давайте рассмотрим его преимущества, которые включают, среди прочих, следующие:

- ◆ знакомые инструменты и возможности Excel для представления данных;
- ◆ наборы данных очень большого объема можно загружать практически из любого источника;
- ◆ наличие новых аналитических возможностей, таких как Data Analysis Expressions (DAX).

Как мы вскоре увидим, те же самые источники данных, которые используются для служб SSAS, можно использовать почти таким же образом и для PowerPivot. (В рассматриваемых примерах будет использоваться куб наподобие куба, созданного в предшествующем разделе, чтобы научиться доставлять данные из куба. Далее этот куб будет использоваться в других упражнениях.)

Новый язык надстройки PowerPivot — Data Analysis Expressions (DAX) — позволяет определять специализированные вычисления в таблицах PowerPivot и Excel

PivotTables. Язык DAX содержит некоторые функции, которые используются в формулах Excel, и дополнительные операции для работы с реляционными данными.

1.12 Работа с PowerPivot for Excel

Первым шагом в работе с PowerPivot for Excel будет импортировать данные с одного или нескольких источников в таблицу Excel. Запустите Excel 2010 и выберите вкладку PowerPivot. В ленте вкладки PowerPivot значок PowerPivot Window. Откроется окно PowerPivot for Excel. Наша задача заключается в том, чтобы создать куб, подобный тому, который мы создали посредством служб SSAS. Поэтому создайте и обработайте куб, следуя шагам, описанным в предшествующем разделе, и применяя схему таблиц, показанную на рис. 22.11.

Далее приводится список таблиц, которые нужно выбрать (см. рис. 22.11):

- ◆ FactInternetSales
- ◆ FactResellerSales
- ◆ DimCustomer
- ◆ DimDate
- ◆ DimProduct
- ◆ DimProductSubcategory

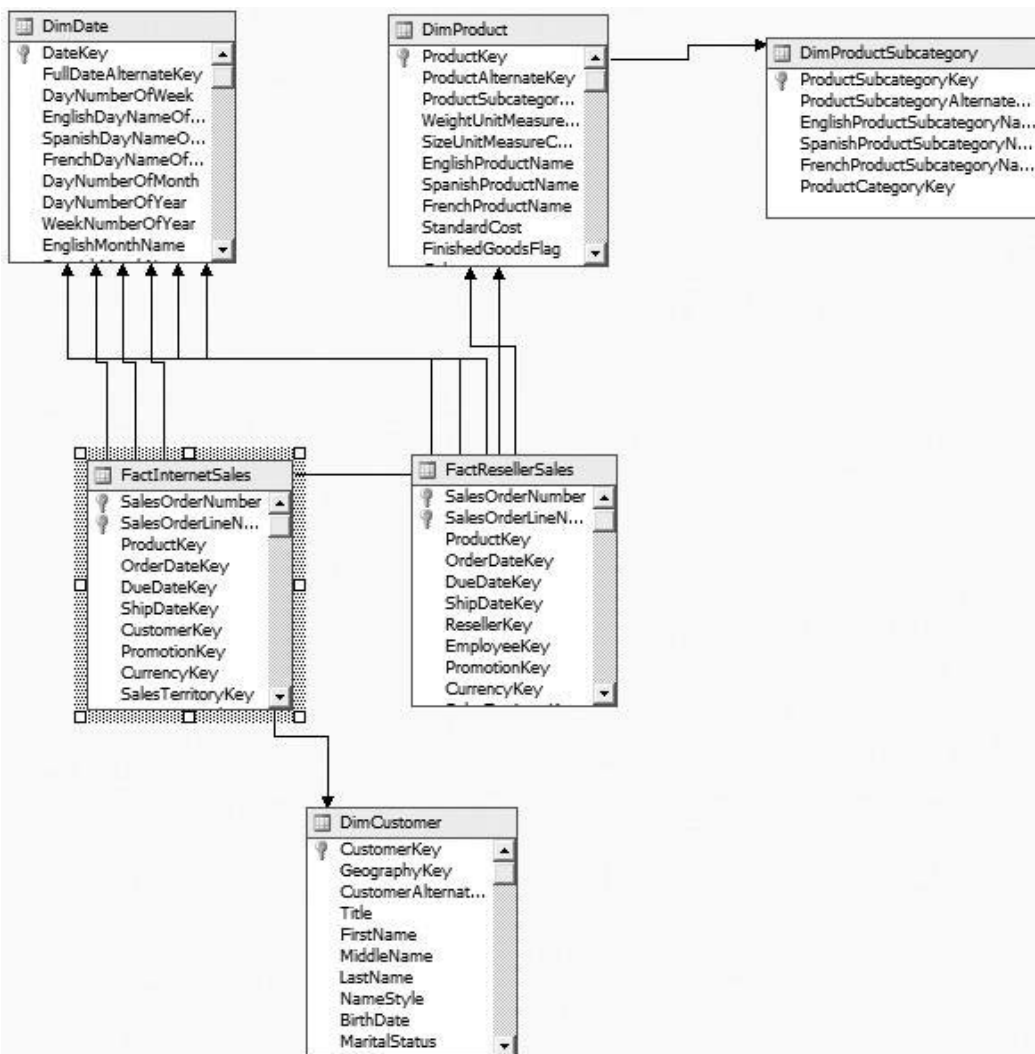


Рис. 11. Таблицы (в окне Data Source View Designer), которые нужно выбрать для создания куба

Чтобы выбрать эти таблицы, используйте тот же самый источник данных (BI_Source), что и для проекта BI_Cube, и создайте новое представление источника данных. Затем создайте с помощью мастера Cube Wizard новый куб и назовите его BI_Cube2. На странице мастера Select Measure Group Tables выберите следующие меры с обеих таблиц фактов (FactInternetSales и FactResellerSales): Sales Amount, Total Product Costs и Freight.

Следующей задачей после создания куба будет задача подключения к нему. Для этого в разделе Get External Data ленты окна PowerPivot for Excel нажмите значок From Other Sources. Откроется мастер Table Import Wizard на странице Connect to a Data Source (рис. 12). На ней выберите опцию Microsoft Analysis Services и нажмите кнопку Next.

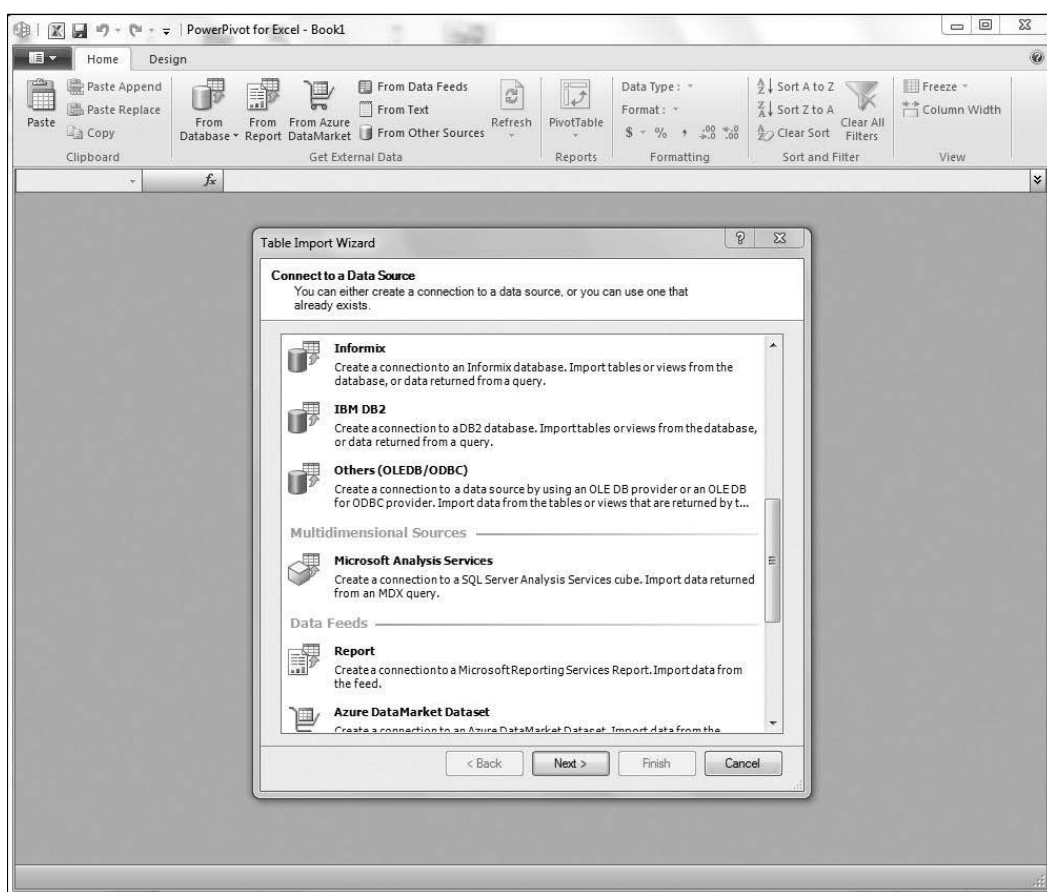


Рис. 12. Мастер импорта таблиц, страница Connect to a Data Source

На следующей странице мастера, Connect to Microsoft SQL Server Analysis Services (рис. 13), нужно указать информацию о кубе, к которому нужно подключиться. В текстовое поле Friendly connection name введите имя соединения (например, PowerPivot_Project), в поле Server or File Name введите имя сервера, а также выберите способ проверки подлинности для входа на сервер.

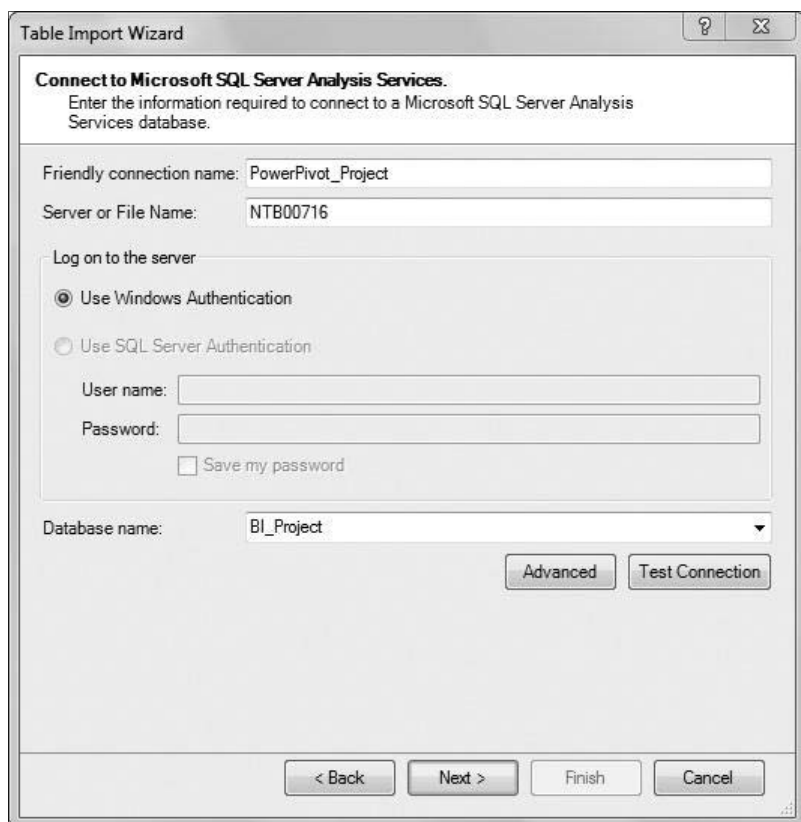


Рис. 13. Страница мастера Connect to Microsoft SQL Server Analysis Services

Наконец, в поле Database name введите имя существующего куба (в данном случае это будет BI_Cube2). (Из списка имен существующих кубов можно также выбрать имя куба, который предоставляется для конкретного сервера.) Нажмите кнопку Next.

В следующем окне Choose How to Import the Data установите переключатель Write a query that will specify the data to import и нажмите кнопку Next. В следующем окне Specify SQL Query для выборки данных для импортирования из источника данных имеется возможность задать запрос MDX. Этот запрос можно ввести в поле SQL Statement вручную или вставить, скопировав его, а затем нажать кнопку Validate, чтобы проверить его правильность. Альтернативно, этот запрос можно создать с помощью графического средства Query Designer. (Создание запросов на языке MDX рассматривается в следующем разделе.)

Чтобы создать запрос графически, нажмите кнопку Design. Откроется окно графического разработчика запросов мастера Table Import Wizard (рис. 14).

Здесь можно выбрать требуемые для вычислений меры и поля из существующих таблиц фактов и таблиц измерений соответственно. Перетащите из таблицы фактов Fact Reseller Sales меру Sales Amount, а из таблиц измерений Due Date и Dim Product меры Data Key и Product Key соответственно. Нажмите кнопку OK.

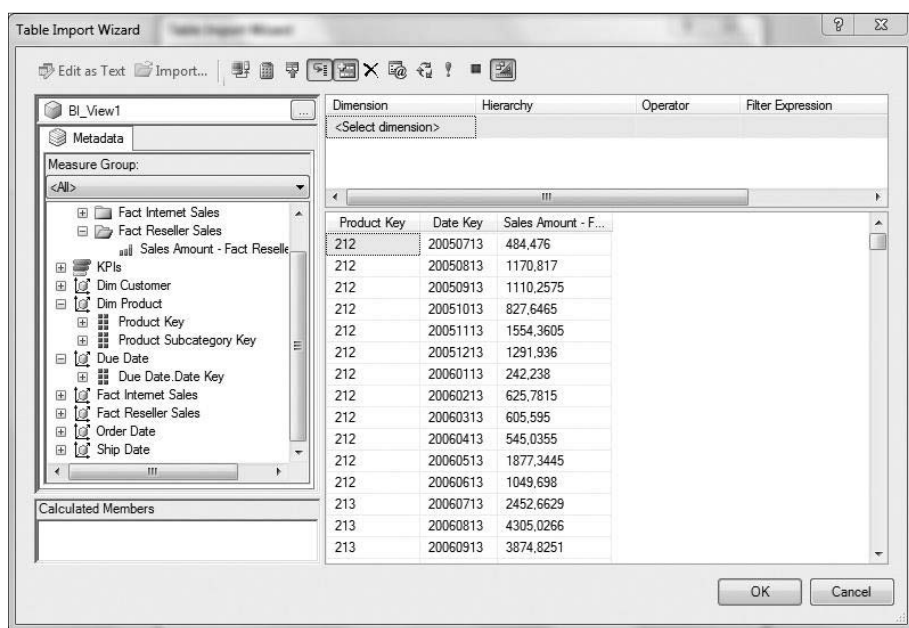


Рис. 14. Графический разработчик запросов мастера Table Import Wizard

Product Key	Date Key	Sales Amount - Fact Reseller Sales
214	20070813	4982.1980999999996
217	20070813	5660.0138999999999
222	20070813	6190.9030999999998
225	20070813	2143.8597000000000
231	20070813	6993.7259999999998
234	20070813	11563.2268999999998
237	20070813	4432.7732999999998
240	20070813	46380.5999999999999
243	20070813	39509.4000000000000
246	20070813	15460.1999999999999
255	20070813	10318.9320000000003
258	20070813	3439.6439999999999
281	20070813	2630.3159999999998
287	20070813	19626.2040000000002
290	20070813	40116.3000000000003
295	20070813	22104.8999999999998
298	20070813	35629.4399999999995
306	20070813	27531.84
309	20070813	45847.1999999999999
353	20070813	112751.514
355	20070813	132824.99549999999
357	20070813	114729.07349999999
359	20070813	206764.82910000000
361	20070813	126769.51009999999
363	20070813	95012.58600000000
372	20070813	57174.38999999999
374	20070813	105552.71999999999

Рис. 15. Выбранные запросом данные вставляются в таблицу PowerPivot for Excel

Система возвратится назад к окну Specify SQL Query, но в котором уже будет отображен текст созданного запроса. Чтобы начать импорт данных, нажмите кнопку Finish. По завершению процесса импорта в поле сведений сообщается о его успешном (или неуспешном) выполнении.

Нажатие кнопки Close закрывает окно импорта данных и помещает выбранные запросом данные в таблицу PowerPivot for Excel (рис. 15).

Теперь эти данные можно представлять, используя одну из форм презентации Excel. В этом примере мы рассмотрим, как создавать перекрестные табличные данные (сводные таблицы) для представления данных в Excel. В разделе Reports ленты окна PivotTable for Excel нажмите значок PivotTable. Откроется диалоговое окно Create PivotTable. Установите в нем переключатель New Worksheet и нажмите кнопку ОК. В окне Excel появится новый лист (рис. 22.16).

В правой панели, которая называется PowerPivot Field List, можно выбирать столбцы для представления в листе. (Если панель PowerPivot Field List не отображается, то нажмите на ленте значок Field List, чтобы отобразить ее.) В этом примере выбраны (установлены флажки) мера и два столбца из таблицы измерений. Теперь данные можно представлять в листе, перетаскивая и помещая их в область Row Labels, Column Labels или Values. На рис. 22.16 ключевой столбец таблицы

	A	B	C	D	E	F
13	220			1	1	1
14	221					
15	222					
16	223			1	1	1
17	224					
18	225					
19	229			1	1	1
20	230					
21	231					
22	232			1	1	1
23	233					
24	234					
25	235			1	1	1
26	236					
27	237					
28	238					
29	239					
30	240					
31	241					
32	242					
33	243					
34	245					
35	246					
36	253			1	1	1
37	254					
38	255					
39	256					
40	257					
41	258					
42	262			1	1	1
43	263					
44	264			1	1	1
45	265					
46	266					

Рис. 16. Представление данных в листе Excel

измерения Dim Product помещен в область Row Labels, а ключевой столбец таблицы измерений Dim Date помещен в область Column Labels. (Мера помещена в область Values.)

1.12. Обращение к данным посредством многомерных выражений

Язык Multidimensional Expressions (MDX) применяется для обращения к многомерным данным, хранящимся в кубах OLAP. (С помощью этого языка можно также создавать кубы.) В языке MDX инструкция SELECT задает результирующий набор, который содержит подмножество многомерных данных, выбранных из куба. Чтобы задать результирующий набор, запрос MDX должен содержать перечисленную да-

информацию.

- ◆ Одну или несколько осей, применяемых для указания результирующего набора. В запросе MDX можно указать до 128 таких осей. Первая ось задается предложением ON COLUMNS, а вторая — предложением ON ROWS. В случае использования больше чем двух осей, применяется альтернативный синтаксис: первая ось указывается предложением ON AXIS(0), вторая — предложением ON AXIS(1) и т. д.
- ◆ Набор членов или записей, включаемых в каждую ось запроса MDX. Это указывается в списке SELECT.
- ◆ Имя куба, который устанавливает контекст запроса MDX, указывается в предложении FROM запроса.
- ◆ Набор чисел или записей для включения в ось среза, которые указываются в предложении WHERE (см. примеры 22.1 и 22.2).

Синтаксис языка объясняется на примере 22.1. Запросы MDX можно выполнять непосредственно в среде Management Studio. Для разработки и выполнения запросов и сценариев на языке MDX применяется редактор запросов MDX Query Editor. Сценарий вводится или копируется в панель редактирования запроса, а для его выполнения нажимается клавиша <F5> или кнопка Execute на панели инструментов редактора запросов.

Пример 1. Выборка общей стоимости продуктов для каждого клиента к оплате на 1 марта 2007 г.

```
SELECT [Measures].MEMBERS ON COLUMNS,
[Dim Customer].[Customer Key].MEMBERS ON ROWS FROM BI_Cube
WHERE ([Due Date].[Date Key].[20070301])
```

В примере 22.1 осуществляется выборка данных из куба BI_Cube. Список SELECT первой оси запроса выбирает всех членов измерения Measures. Иными словами, он выбирает значения столбца Total Product Costs, т. к. это единственная мера в этом кубе. Вторая ось запроса отображает всех членов столбца Customer Key измерения Dim Customer.

В предложении FROM указывается, что источником данных в этом случае является куб BI_Cube. Предложение WHERE выполняет срез по измерению Due Data в соответствии с ключевыми значениями, используя одно значение даты — 2007/03/01.

В примере 22.2 показан еще один запрос MDX.

Пример 2. Вычисление общей стоимости продукта номер 7 для клиента номер 11741 к уплате в марте 2007 г.

```
SELECT [Measures].MEMBERS ON COLUMNS
```


FROM BI_Cube

WHERE ({[Due Date].[Date Key].[20070301]:[Due Date].[Date key].[20070331]}, [Dim Customer].[Customer Key].[11741],
[Dim Product].[Product Key].[7])

Список SELECT в примере 22.2 содержит только членов измерения Measures. Поэтому запрос отображает значение столбца Total Product Costs. Предложение WHERE в примере 22.2 более сложное, чем в примере 22.1. Во-первых, в нем выполняется три операции среза, которые разделяются запятыми. Для среза используется только один член измерения Customer и один член измерения Product, тогда как из измерения Due Date вырезаются даты с 2007/03/01 по 2007/03/31. (Как можно видеть в запросе для указания диапазона дат используется знак двоеточия ":").

1.13 Безопасность служб Analysis Services SQL Server

Аспекты безопасности служб SQL Server Analysis Services такие же, как и аспекты безопасности компонента Database Engine. Это означает, что службы SSAS поддерживают те же самые возможности — авторизацию и проверку подлинности, — что и компонент Database Engine, но в более сжатом виде.

В процессе авторизации определяются пользователи, которые имеют санкционированный доступ к службам SSAS. Этот вопрос тесно связан с авторизацией в операционной системе. Иными словами, функции авторизации пользователей служб SSAS основаны на правах доступа, предоставляемых пользователю операционной системой Windows.

Количество пользователей, которые могут выполнять функции администрирования служб SSAS, можно ограничить. Можно также указывать конечных пользователей, которые могут обращаться к данным и определять типы операций, которые они могут выполнять. Кроме этого, можно также управлять доступом пользователей к разным уровням данным, таким как уровень куба, измерения и ячейки куба. Такое управление реализуется посредством применения ролей.

Как уже упоминалось, роль базы данных определяет группу пользователей базы данных, которые могут обращаться к определенным объектам базы данных. Каждая роль определяется на уровне базы данных служб SSAS, а затем назначается кубам. После этого членство в этой роли присваивается отдельным пользователям и другим ролям.

Чтобы создать роль, щелкните правой кнопкой мыши папку Roles в панели обозревателя решений и выберите в контекстном меню пункт New Role. В открывшемся окне New Role на странице General измените имя по умолчанию роли на требуемое имя роли. В этом же окне (рис. 22.17) на его разных вкладках можно настроить разные параметры этой роли.

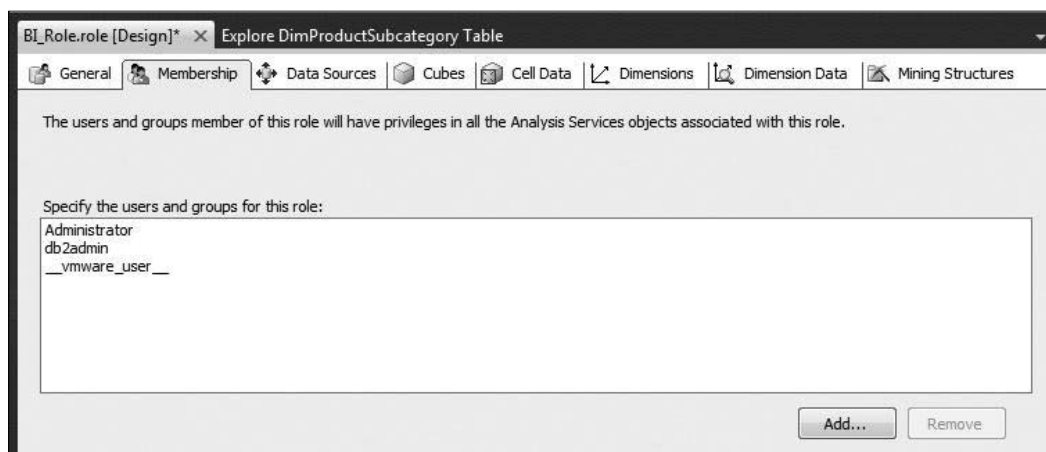


Рис. 17. Окно для создания новой роли для служб SSAS и настройки ее параметров

На вкладке Membership можно указать всех пользователей, которым присвоено членство в этой роли. (На рис. 22.17 показано три пользователя, добавленных в членство роли BI_Role.) На вкладках Data Sources и Cubes указываются источники данных, т. е. кубы, которые могут использовать члены роли. На вкладке Cell Data можно присвоить разрешения для определенной ячейки. На вкладках Dimensions и Dimensions Data указываются измерения (данные измерений), к которым члены роли могут иметь доступ.

Вопросы для самоконтроля и текущего контроля

- 1) Опишите назначение и возможности SQL Server Analysis Service.
- 2) Дайте определение кубу в SQL Server Analysis Service.
- 3) Дайте определение Business Intelligence Development Studio.
- 4) Объясните, для чего применяется мастер источников данных.
- 5) Объясните, для чего применяется мастер кубов.
- 6) Перечислите задачи Aggregation Design Wizard.

Задание на самостоятельную работу

- 1) Практическое задание

Выполнить создание базы знаний по предметной области, определенной номером варианта практической работы №2.

- 2) Самостоятельная работа

Используя среду Management Studio создать куб и провести анализ данных.

Список рекомендуемой литературы¹:

[1] Сидоркина, Ирина Геннадьевна. Системы искусственного интеллекта [Текст] : [учеб. пособие для вузов по направлению 230100 "Информатика и вычисл. техника"] / И. Г. Сидоркина. М.: Кнорус, 2011. - 245 с.

Рассмотрены теоретические аспекты проектирования систем искусственного интеллекта и даны модели представления знаний, изложены материалы информационного, справочного, консультирующего характера по использованию инструментальных средств и технологическим особенностям разработки систем данного класса.

Подробнее: <https://www.labyrinth.ru/books/276217/>

[2] Богомолова, М. А. Экспертные системы (техника и технология проектирования) [Текст] : Методические указания к лабораторным работам / М. А. Богомолова. Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015. - 47 с.

Методические указания являются руководством при выполнении лабораторных занятий, проводимых по курсу «Экспертные системы (техника и технология проектирования)» с магистрами направления 09.04.03 - «Прикладная информатика» в терминальном классе. Лабораторные работы связаны с изучением принципов построения экспертных систем на основе различных моделей представления знаний, и могут быть полезны студентам других специальностей.

Подробнее: <https://www.iprbookshop.ru/71908.html>

[3] Советов, Борис Яковлевич. Представление знаний в информационных системах [Текст] : [учеб. для вузов по направлению подгот. "Информ. системы и технологии"] / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. М.: Академия, 2011.

Учебник создан в соответствии с Федеральным государственным образовательным стандартом по направлению бакалавриата «Информационные системы и технологии». Рассмотрены современное состояние теоретических и прикладных вопросов представления знаний в информационных системах, идеология построения

интеллектуальных систем, математический аппарат представления знаний, возможности и пути использования искусственного интеллекта при проектировании информационных систем. Изложены новые аспекты представления знаний на основе искусственных нейронных сетей, расчетно-логических систем, генетических алгоритмов, мультиагентных систем. Приведены примеры практической реализации представления знаний на базе декларативного языка Пролог. Для студентов учреждений высшего профессионального образования. Может быть полезен разработчикам и пользователям информационных систем; преподавателям и научным сотрудникам, сферой интересов которых является интеллектуализация различных предметных областей; менеджерам и руководителям различного ранга, желающим самостоятельно ознакомиться с современным состоянием информационных технологий.

Подробнее: <https://lavkababuin.com/predstavlenie-znaniy-v-informacionnyh-sistemah-2-e-izd-ster-650564/>

[4] Остроух, А. В. Интеллектуальные информационные системы и технологии [Электронный ресурс] : монография / А. В. Остроух, А. Б. Николаев. Санкт-Петербург: Лань, 2019. - 308 с.

В монографии изложены концептуальные основы и методы представления знаний в интеллектуальных системах. Рассмотрены различные подходы, применяемые при проектировании и разработке интеллектуальных систем и технологий в транспортном комплексе, а также рассмотрены тенденции развития систем искусственного интеллекта.

Монография может быть использована для формирования профессиональной компетентности студентов высших учебных заведений, аспирантов и научных сотрудников обучающихся и ведущих научные исследования в области разработки и практического применения систем искусственного интеллекта по укрупнённой группе направлений подготовки "Информатика и вычислительная техника".

Подробнее: <https://www.labirint.ru/books/692861/>

Ссылки на открытые ресурсы (он-лайн-курсы, видеоуроки и т.д.)

[5]

ⁱСписок рекомендуемой литературы может быть более широким по сравнению со списком литературы, указанным в рабочей программе

Прикладные интеллектуальные системы

Конспект лекции 5. Экспертные системы. Классификация экспертных систем. Архитектура экспертных систем. Применение экспертных систем в профессиональной деятельности

Аннотация: В лекции рассказывается о понятии экспертных систем, их классификации и архитектуре, приводятся примеры применения экспертных систем в профессиональной деятельности.

План лекции:

1. Экспертные системы.
2. Классификация экспертных систем.
3. Архитектура экспертных систем.
4. Применение экспертных систем в профессиональной деятельности.
5. Задания для самостоятельной работы.

Ключевые слова: экспертная система, классификация экспертных систем. Архитектура экспертных систем.

СОДЕРЖАНИЕ ЛЕКЦИИ:

1. Экспертные системы

В основе экспертной системы находится обширный запас знаний о конкретной проблемной области. В большинстве случаев их ограничивают правила, которые позволяют делать заключения на основе исходных данных или предположений. Такой подход к конструированию систем, опирающийся на использование знаний заменяет традиционное соотношение

Данные+Алгоритм=Программа

на новую архитектуру, основу которой составляют база знаний и машина логического вывода, так что мы имеем

Знания+Вывод = Система.

Экспертная система рассматривается как результат создания в компьютере основанной на знаниях компоненты, соответствующей навыку эксперта, в такой форме, которая позволит системе дать разумный совет или принять разумное решение о функции обработки данных. Желательной дополнительной характеристикой является способность системы по требованию объяснить свою линию рассуждений в виде, непосредственно понятном тому, кто задал вопрос. Достижение таких свойств обеспечивается методом программирования и использованием правил.

Основные характеристики экспертных систем:

- экспертная система ограничена определенной сферой экспертизы
- она способна рассуждать при сомнительных данных
- она способна объяснять цепочку рассуждений понятным способом
- факты и механизмы вывода четко отделены друг от друга
- она строится так, чтобы имелась возможность постепенно наращивать систему
- чаще всего она основана на использовании правил
- на выходе она дает совет
- она экономически выгодна.

Экспертная система содержит базу знаний, машину вывода, модуль извлечения знаний, систему объяснения (интерфейс).

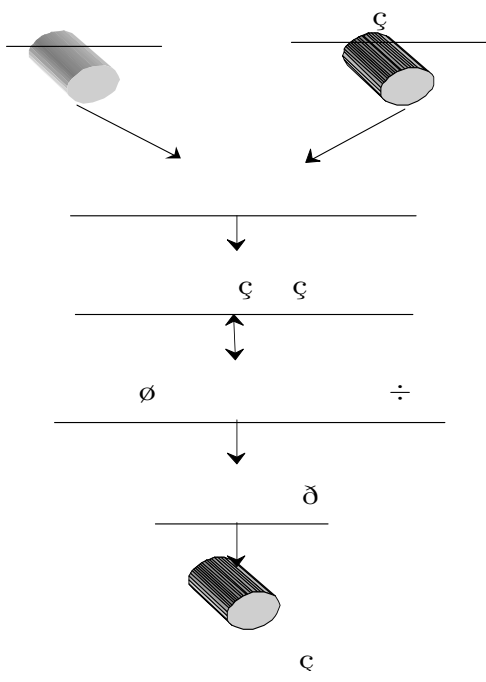


Рисунок 1 – Состав ЭС

База знаний содержит факты и правила. Факты представляют собой краткосрочную информацию в том плане, что они могут изменяться, например, в ходе консультации. Правила представляют собой долговечную информацию о том, как порождать новые факты или гипотезы из того, что известно. Такой подход отличается от обычной методики использования базы данных тем, что факты в базе данных обычно пассивны: они либо есть, либо их нет. База знаний активно пытается восполнить недостающую информацию.

Правила продукций являются предпочтительным средством отображения неформальных знаний. Такие знания имеют формат ЕСЛИ-ТО, но эти знания не воплощены в какую-то программу, а представляют собой данные для высокоуровневого интерпретатора, а именно машины ввода. Правила продукций не единственный способ представления знаний. Для этой цели использовались семантические сети, исчисление предикатов.

Большинство экспертных систем «стремится», как правило, использовать личностные знания эксперта и чаще всего формализуют их (без усиления логических возможностей современных информационных систем) в выбранной системе правил, образующих некоторую эвристику. Поэтому основной трудностью создания экспертных систем становится получение знаний от эксперта, так как этот процесс зависит от многих существенных факторов: предметной области, методики получения знаний, квалификации эксперта и его способности выделять собственные знания. Экспертные системы применяются в случаях, когда тип решаемой задачи обладает хотя бы одним из следующих аспектов:

- диагностика
- нет установившейся теории
- мало специалистов

- данные «зашумлены».

Диагностическая задача связана с областью, в которой имеется множество возможных ответов, и трудность состоит в том, что необходимо выбрать из них один верный, или, по крайней мере, отбросить заведомо неверные.

Область, где нет твердо установившейся теории, содержит слишком много переменных величин, затрудняющих создание полной и цельной теории, где искусные практики опираются на свои познания и интуицию.

Область с малым числом специалистов, связана с тем, что экономически выгодно автоматизировать навыки на которые высок спрос.

2. Классификация экспертных систем.

Рассмотрим различные способы классификации ЭС.

По назначению ЭС делятся на:

- ЭС общего назначения.
- Специализированные ЭС:
 - проблемно-ориентированные для задач диагностики, проектирования, прогнозирования
 - предметно-ориентированные для специфических задач, например, контроля ситуаций на атомных электростанциях.

По степени зависимости от внешней среды выделяют:

- Статические ЭС, не зависящие от внешней среды.
- Динамические, учитывающие динамику внешней среды и предназначенные для решения задач в реальном времени. Время реакции в таких системах может задаваться в миллисекундах, и эти системы реализуются, как правило, на языке C++.

По типу использования различают:

- Изолированные ЭС.
- ЭС на входе/выходе других систем.
- Гибридные ЭС или, иначе говоря, ЭС интегрированные с базами данных и другими программными продуктами (приложениями).

По сложности решаемых задач различают:

- Простые ЭС - до 1000 простых правил.
- Средние ЭС - от 1000 до 10000 структурированных правил.
- Сложные ЭС - более 10000 структурированных правил.

По стадии создания выделяют:

- Исследовательский образец ЭС, разработанный за 1-2 месяца с минимальной БЗ.
- Демонстрационный образец ЭС, разработанный за 2-4 месяца, например, на языке типа LISP, PROLOG, CLIPS
- Промышленный образец ЭС, разработанный за 4-8 месяцев, например, на языке типа CLIPS с полной БЗ.

- Коммерческий образец ЭС, разработанный за 1,5-2 года, например, на языке типа C++, Java с полной БЗ.

Общепринятая классификация экспертных систем отсутствует, однако наиболее часто экспертные системы различают по назначению, предметной области, методам представления знаний, динамичности и сложности.



По назначению классификацию экспертных систем можно провести следующим образом:

диагностика состояния систем, в том числе мониторинг (непрерывное отслеживание текущего состояния);
прогнозирование развития систем на основе моделирования прошлого и настоящего;
планирование и разработка мероприятий в организационном и технологическом управлении;
проектирование или выработка четких предписаний по построению объектов, удовлетворяющих поставленным требованиям;
автоматическое управление (регулирование);
обучение пользователей и др.

По предметной области наибольшее количество экспертных систем используется в военном деле, геологии, инженерном деле, информатике, космической технике, математике, медицине, метеорологии, промышленности, сельском хозяйстве,

управлении процессами, физике, филологии, химии, электронике, юриспруденции.

Классификация экспертных систем по методам представления знаний делит их на традиционные и гибридные. Традиционные экспертные системы используют, в основном, эмпирические модели представления знаний и исчисление предикатов первого порядка. Гибридные экспертные системы используют все доступные методы, в том числе оптимизационные алгоритмы и концепции баз данных.

По степени сложности экспертные системы делят на поверхностные и глубинные. Поверхностные экспертные системы представляют знания в виде правил «ЕСЛИ-ТО». Условием выводимости решения является безобрывность цепочки правил. Глубинные экспертные системы обладают способностью при обрыве цепочки правил определять (на основе метазнаний) какие действия следует предпринять для продолжения решения задачи. Кроме того, к сложным относятся предметные области в которых текст записи одного правила на естественном языке занимает более 1/3 страницы.

Классификация экспертных систем по динамичности делит экспертные системы на статические и динамические. Предметная область называется статической, если описывающие ее исходные данные не изменяются во времени. Статичность области означает неизменность описывающих ее исходных данных. При этом производные данные (выводимые из исходных) могут и появляться заново, и изменяться (не изменяя, однако, исходных данных).

Если исходные данные, описывающие предметную область, изменяются за время решения задачи, то предметную область называют динамической. В архитектуру динамической экспертной системы, по сравнению со статической, вводятся два компонента:

подсистема моделирования внешнего мира;

подсистема связи с внешним окружением.

Последняя осуществляет связи с внешним миром через систему датчиков и контроллеров. Кроме того, традиционные компоненты статической экспертной системы (база знаний и механизм логического вывода) претерпевают существенные изменения, чтобы отразить временную логику происходящих в реальном мире событий.

3. Архитектура экспертных систем.

Если компьютерной программе предстоит выполнить задачу эксперта, то она нуждается в большом объеме знаний, позволяющих решать сложные проблемы, подобно тому, как это делает человек. Система должна быть тщательно организована. В общем случае знания разделяются на три типа:

- Фактические (декларативные) знания. Этот вид знаний представляет собой информацию о конкретном случае, обычно собираемую посредством диалога с пользователем, который указывает какие факты следует считать справедливыми в настоящее время. Важно то, как представлена эта информация, поскольку сама структура представления также информативна. Структуру представления следует выбирать в зависимости от содержания знаний.
- Процедурные знания. Эти знания обычно собираются заранее путём опроса специалиста в данной предметной области и составляют ядро базы знаний. Они используются также в блоке рассуждения системы, позволяя потом выводить следствия. Процедурные знания дают возможность при необходимости генерировать и факты. Таким образом, фактические и процедурные знания тесно переплетаются. Кроме того, в ходе работы системы приходится принимать решения, какие из тех правил следует использовать.
- Управляющие знания. Системе должен быть предусмотрен некоторый набор стратегий, чтобы можно было рассматривать альтернативные возможности в ходе работы, переход при неудаче от одной стратегии к другой.

Системы продукций представляют собой конкретный метод организации программ в виде совокупностей трех групп, указанных выше. Каждая группа рассматривается отдельно, поэтому такую систему легко понять и с ней легко работать.

- 1) База данных (БД)
- 2) Список порождающих правил (ПП)
- 3) Метод выбора, какое порождающее правило следует применить при данном состоянии базы данных

Каждое порождающее правило (продукция) имеет форму ЕСЛИ (условие), ТО (действие) или, возможно, форму ЕСЛИ (условие), ТО (действие 1), В ПРОТИВНОМ СЛУЧАЕ (действие2). При этом процедуры сопоставления с образцом определяет, является ли данное правило применимым вообще.

Условная часть продукции сопоставляется с имеющейся базой данных. Если процесс сопоставления заканчивается успешно, то всем переменным, входящим в образец, то есть переменным из условной части некоторого порождающего правила, придаются определенные значения из БД, которые обеспечивают сопоставимость. В дальнейшем эти значения будут использоваться в части, соответствующей действию того же самого порождающего правила. Эта часть может быть произвольной процедурой, которая запускается только при условии успешного сопоставления и если данная продукция была выбрана управляющим механизмом. Все изменения могут быть осуществлены посредством двух элементарных функций - добавления и удаления. В начале работы системы запускается некоторое простое правило, которое обеспечивает сбор информации о рассматриваемом случае и вносит ее в базу данных. В дальнейшем система действует по простому принципу: «выбери продукцию - примени соответствующие действия». Централизованная база данных является сосредоточием активности системы, причем программист должен внести в систему потенциально весьма сложные стратегии управления, которые позволят избежать

столь естественной неупорядоченности в работе системы, поведут ее по верному пути к успешному решению поставленной задачи. Самый распространенный способ как «запуска» системы продукций, так и «управления» ею состоит в использовании механизмов, названных «демонами».

Существует три возможные разновидности демонов, а именно «если-добавлено», «если-удалено» и «если необходимо». Демоны запускаются в работу путем добавления их в базу данных. Как правило, демоны вида «если-добавлено» будут использоваться для проведения прямых логических дедукций, осуществляемых автоматически, если стали известны некоторые дополнительные факты. Демоны вида «если-удалено» запускаются функцией удаления. Эти демоны представляют по смыслу полную противоположность демонам «если-добавлено», но работают в значительной степени точно таким же образом. Как правило, демоны «если-удалено» используются для удаления из БД тех явных дедукций, которые были сделаны демонами «если-добавлено», как только те вещи, которые вызывали эти дедукции, были удалены из БД (что необходимо для поддержания согласования БД). Демоны «если-необходимо» используются для проведения дедукции в обратном направлении.

Большую часть БД составляют правила, которые запускаются путем сопоставления с соответствующими признаками, характеризующими данную задачу, и которые могут добавляться, модифицироваться или изыматься пользователем. Базу данных такого специального типа обычно называют базой знаний.

Для взаимодействия пользователя с экспертной системой существует три режима, в отличие от единственного режима взаимодействия (получение ответов на поставленные задачи), типичного для более привычных вычислительных систем:

- получение решений поставленных задач - пользователь как потребитель
- усовершенствование или наращивание знаний системы - пользователь как учитель
- извлечение пользы для человека из базы знаний - пользователь как ученик

Архитектура экспертной системы включает в себя следующие блоки:

- рабочую память;
- блок логического вывода, осуществляющий сопоставление данных и знаний, которые требуются лишь на заданном промежутке времени;
- базу знаний, содержащую правила логического вывода, не меняющиеся на заданном промежутке времени и представленные в установленных для системы формализмах (фреймах, семантических сетях, предикатах первого порядка и др.);
- базу данных, содержащую постоянные, не меняющиеся на заданном промежутке времени данные, необходимые для работы БЗ;
- блок приобретения знаний, предназначенный для работы с экспертом и когнитологом;
- блок рекомендаций и комментариев;
- интерфейс пользователя, предназначенный для взаимодействия с пользователем и отображающий состояние рабочей памяти на основе рекомендаций и комментариев;

- блок проверки на корректность и противоречивость, выполняющий согласование вновь поступающих в базу правил с уже имеющимися в ней.

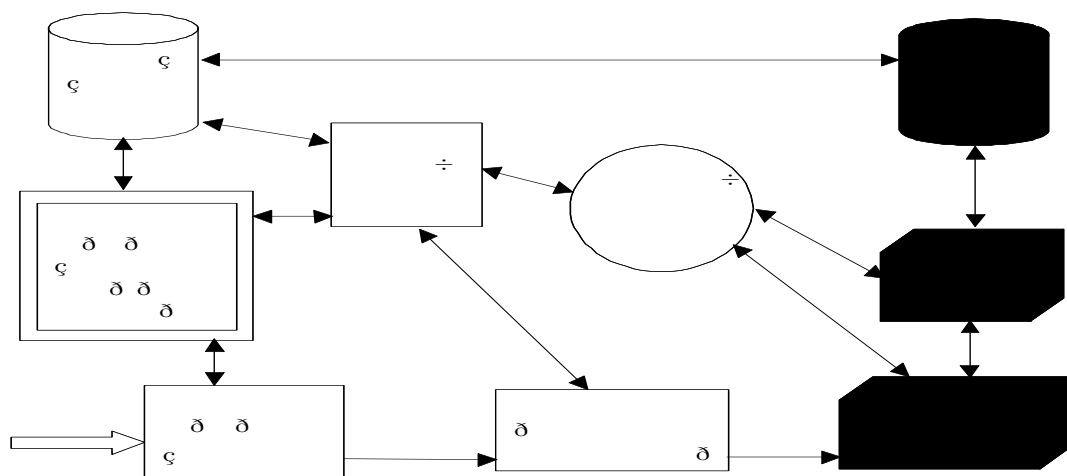


Рис. 2. Архитектура экспертной системы

4. Применение экспертных систем в профессиональной деятельности

Точкой отсчета для работ по созданию экспертных систем можно считать 1965 г., когда ученые из Станфордского научно-исследовательского института Эдвард Фейгенбаум и Брюс Бучанан вместе с нобелевским лауреатом Джошуа Ледербергом приступили к созданию компьютерной системы, предназначенной для определения молекулярной структуры химических соединений. При построении этой системы - она получила название Dendral - они в первую очередь создали программу, основанную на простой, но вместе с тем достаточно мощной системе Аристотелевой логики. С ее помощью формулировалась серия вопросов типа «если - то», которые описывали правила атомных связей. Создатели системы Dendral, используя знания, накопленные в процессе ее совершенствования, вскоре разработали новую систему - Meta-Dendral. Последняя стала вспомогательным инструментом при построении других экспертных систем. Среди многочисленных «потомков» программы Dendral следует упомянуть: MYCIN - для диагностики серьезных бактериальных инфекций и помощи в выборе антибиотиков, PUFF - для диагностики легочных заболеваний, ONCOCIN - для назначения лечения при раковых заболеваниях, а также системы MOLGEN и GENESIS, оказывающих помощь специалистам в области молекулярной биологии при планировании экспериментов по изучению ДНК и клонированию. С 70-х годов ЭС стали ведущим направлением в области искусственного интеллекта. В этот период было создано множество разнообразных экспертных и диагностических систем, большая часть которых действует и сегодня. Самыми известными из них являются MYCIN и PROSPECTOR, предназначенная для геологической разведки месторождений полезных ископаемых. В 80-х годах в мире начался принципиально новый этап развития интеллектуальных технологий - эра интеллектуальных систем - консультантов, которые предлагали варианты решений, обосновывали их, способны

были к обучению и, следовательно, к развитию, общались с человеком на привычном для него, хотя и ограниченном, естественном языке. Знания стали товаром. Носителей систем знаний называли экспертами. Человечество получило возможность сохранять и накапливать базы знаний отдельных специалистов (или групп специалистов) в определенной области. Знания стало возможным собирать, тиражировать, проектировать, сделать доступными для всех заинтересованных в нем людей. Появилась новая профессия – «инженер по знаниям» или «инженер-когнитолог». При разработке последующих экспертных систем были учтены особенности и недостатки PROSPECTOR и MYCIN. Благодаря этому такие диагностические медицинские системы, как INTERNIST и CASNET, основанные на ассоциативном и казуальном (от англ. casual - случайный) подходах, приобрели более мощные механизмы вывода. В 90-е года усложнение систем связи и решаемых задач потребовало качественно нового уровня «интеллектуальности» обеспечивающих программных систем, таких систем, как защита от несанкционированного доступа, информационная безопасность ресурсов, защита от нападений, смысловой анализ и поиск информации в сетях и т.п. И новой парадигмой создания перспективных систем защиты всех видов стали интеллектуальные системы. Именно они позволяют создавать гибкие среды, в рамках которых обеспечивается решение всех необходимых задач. Это новое направление получило название мультиагентных систем. Каждый агент имеет свою систему целеполагания и мотивации, свою область действий и ответственности, а взаимодействие между ними обеспечивается метаинтеллектом. В рамках такого осмысления традиционные методы, алгоритмы и программы становятся элементарными «кирпичиками», из которых строятся затем алгоритмы и решения возникающих задач. Таким образом, моделируется некоторое виртуальное сообщество интеллектуальных агентов-систем, которые автономны, активны, вступают в различные «социальные» отношения – кооперации и сотрудничества (дружбы), конкуренции, соревнования, вражды и т.п. Этот «социальный» аспект решения задач - фундаментальная особенность концептуальной новизны передовых информационных технологий, искусственных (виртуальных) организаций, виртуального общества. Уже сегодня мультиагентные системы находят широчайшее применение для: распределенного решения сложных задач, совмещенного проектирования изделий, построения виртуальных предприятий, моделирования больших производственных систем и электронной торговли, электронной разработки сложных компьютерных систем, управления системами знаний и информации и т.п. В XXI веке количество экспертных систем исчисляется тысячами и десятками тысяч. В развитых зарубежных странах сотни фирм занимаются их разработкой и внедрением в различные сферы жизни. . В настоящее время ведутся разработки самостоятельно обучаемых экспертных систем. В качестве современных ЭС можно назвать быстродействующую систему OMEGAMON (фирма Candle, с 2004 г. IBM) для отслеживания состояния корпоративной информационной сети и G2 (фирма Gensym) - коммерческую экспертную систему для работы с динамическими объектами. Они служат для принятия решения за считанные

секунды с момента наступления внештатных или критических ситуаций. Для G2 также характерно распараллеливание процессов рассуждений.

Современные технологии бизнеса характеризуются высокой динамичностью, связанной с постоянно изменяющимися потребностями рынка, ориентацией производства товаров и услуг на индивидуальные потребности заказчиков и клиентов, непрерывным совершенствованием технических возможностей и сильной конкуренцией. В этих условиях требуется разработка информационных систем, которые, с одной стороны, должны оперативно поддерживать функционирование существующих бизнес-процессов, а, с другой стороны, давать прогнозы на долговременной основе об эффективности организации бизнес-процессов и рекомендации по их реорганизации. Первой цели соответствуют динамические экспертные системы оперативного управления бизнес-процессами, второй цели – интеллектуальные системы моделирования бизнес-процессов. В том и другом случае под бизнес-процессом будем понимать взаимосвязанную совокупность материальных, информационных, финансовых потоков или рабочих потоков (workflow), проходящих через взаимодействующие подразделения предприятия и направленных на удовлетворение потребностей клиента (изготовление товара или оказание услуги). К динамическим бизнес-процессам на предприятии относят: а) управление заказами; б) управление запасами; в) оперативно-календарное планирование и управление производством. Организация перечисленных бизнес-процессов определяется целями и задачами предприятия и зависит от конкретных видов выпускаемой продукции и оказываемых услуг. Вместе с тем, перечисленные бизнес-процессы в современных системах управления сильно взаимосвязаны.

Типичными задачами, которые решаются динамическими экспертными системами оперативного управления бизнес-процессами, являются [3]: Мониторинг бизнес-процессов и оперативное информирование лиц, принимающих решение, об отклонениях; Упреждающая диагностика, прогнозирование отклонений в параметрах операций бизнес-процессов; Динамическое распределение ресурсов в соответствии с изменяющейся обстановкой; Планирование действий, диспетчирование и составление сетевых графиков работ; Моделирование последствий принимаемых решений по изменению процессов. В основные задачи интеллектуальных систем динамического моделирования для реинжиниринга (реорганизации) бизнес-процессов входят: Определение оптимальной последовательности выполняемых операций, которая приводит к сокращению длительности цикла изготовления и продажи товаров и услуг, обслуживания клиентов. Следствие оптимизации – повышение оборачиваемости капитала и рост всех экономических показателей фирмы. Оптимизация использования ресурсов в различных бизнес-процессах, которая приводит к минимизации издержек производства и обращения. Построение адаптивных бизнес-процессов, нацеленных на быструю адаптацию к изменениям потребностей конечных потребителей продукции, производственных технологий, поведения конкурентов на рынке и, как следствие, повышение качества обслуживания клиентов в условиях динамичности внешней среды. Отработка

рациональных схем взаимодействия с партнерами, сочетания бизнес-процессов, которые оптимизируют финансовые потоки, обеспечение равномерности поступления и использования денежных средств. Для обоих классов интеллектуальных систем характерны общие особенности реализации: Объектно-ориентированный характер модели проблемной области; Динамическое создание и уничтожение временных (рабочих) объектов; Динамическое поведение как постоянных объектов (ресурсов), так и временных (рабочих) объектов; Многоальтернативность выполнения бизнес-процесса в зависимости от возникающих событий; Анализ и обработка временных характеристик бизнес-процессов.

Вопросы для самоконтроля и текущего контроля

- 1) Что входит в состав экспертной системы?
- 2) Кто такой инженер по знаниям?
- 3) Перечислите классификацию экспертных систем по типу решаемых задач.
- 4) Что такое интегрированная экспертная система?
- 5) Что такое "стыковка экспертной системы"?
- 6) Что происходит при извлечении знаний?

Задание на самостоятельную работу

- 1) Практическое задание

Выполнить проектирование структуры экспертной системы решающей задачу предметной области, определенной номером варианта практической работы №2.

- 2) Самостоятельная работа

Исследовать примеры современных экспертных систем.

Список рекомендуемой литературы¹:

[1] Сидоркина, Ирина Геннадьевна. Системы искусственного интеллекта [Текст] : [учеб. пособие для вузов по направлению 230100 "Информатика и вычисл. техника"] / И. Г. Сидоркина. М.: Кнорус, 2011. - 245 с.

Рассмотрены теоретические аспекты проектирования систем искусственного интеллекта и даны модели представления знаний, изложены материалы информационного, справочного, консультирующего характера по использованию инструментальных средств и технологическим особенностям разработки систем данного класса.

Подробнее: <https://www.labyrinth.ru/books/276217/>

[2] Богомолова, М. А. Экспертные системы (техника и технология проектирования) [Текст] : Методические указания к лабораторным работам / М. А. Богомолова. Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015. - 47 с.

Методические указания являются руководством при выполнении лабораторных занятий, проводимых по курсу «Экспертные системы (техника и технология проектирования)» с магистрами направления 09.04.03 - «Прикладная информатика» в терминальном классе. Лабораторные работы связаны с изучением принципов построения экспертных систем на основе различных моделей представления знаний, и могут быть полезны студентам других специальностей.

Подробнее: <https://www.iprbookshop.ru/71908.html>

[3] Советов, Борис Яковлевич. Представление знаний в информационных системах [Текст] : [учеб. для вузов по направлению подгот. "Информ. системы и технологии"] / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. М.: Академия, 2011.

Учебник создан в соответствии с Федеральным государственным образовательным стандартом по направлению бакалавриата «Информационные системы и технологии». Рассмотрены современное состояние теоретических и прикладных вопросов

представления знаний в информационных системах, идеология построения интеллектуальных систем, математический аппарат представления знаний, возможности и пути использования искусственного интеллекта при проектировании информационных систем. Изложены новые аспекты представления знаний на основе искусственных нейронных сетей, расчетно-логических систем, генетических алгоритмов, мультиагентных систем. Приведены примеры практической реализации представления знаний на базе декларативного языка Пролог. Для студентов учреждений высшего профессионального образования. Может быть полезен разработчикам и пользователям информационных систем; преподавателям и научным сотрудникам, сферой интересов которых является интеллектуализация различных предметных областей; менеджерам и руководителям различного ранга, желающим самостоятельно ознакомиться с современным состоянием информационных технологий.

Подробнее: <https://lavkababuin.com/predstavlenie-znaniy-v-informacionnyh-sistemah-2-e-izd-ster-650564/>

[4] Остроух, А. В. Интеллектуальные информационные системы и технологии [Электронный ресурс] : монография / А. В. Остроух, А. Б. Николаев. Санкт-Петербург: Лань, 2019. - 308 с.

В монографии изложены концептуальные основы и методы представления знаний в интеллектуальных системах. Рассмотрены различные подходы, применяемые при проектировании и разработке интеллектуальных систем и технологий в транспортном комплексе, а также рассмотрены тенденции развития систем искусственного интеллекта.

Монография может быть использована для формирования профессиональной компетентности студентов высших учебных заведений, аспирантов и научных сотрудников обучающихся и ведущих научные исследования в области разработки и практического применения систем искусственного интеллекта по укрупнённой группе направлений подготовки "Информатика и вычислительная техника".

Подробнее: <https://www.labyrinth.ru/books/692861/>

Ссылки на открытые ресурсы (он-лайн-курсы, видеоуроки и т.д.)

[5]

ⁱСписок рекомендуемой литературы может быть более широким по сравнению со списком литературы, указанным в рабочей программе

Прикладные интеллектуальные системы

Конспект лекции 6. Этапы и методы создания и применения экспертных систем. Оболочки экспертных систем. Использование оболочек экспертных систем при создании конкретных экспертных систем.

Аннотация: В лекции рассказывается об этапах и методах создания и применения экспертных систем. Описано назначение оболочек экспертных систем и их использование при создании экспертных систем.

План лекции:

1. Этапы разработки экспертных систем
2. Инструментальные средства проектирования и разработки экспертных систем.
Оболочки экспертных систем
3. Задания для самостоятельной работы.

Ключевые слова: экспертная система, оболочки экспертных систем, G2, CLIPS.

СОДЕРЖАНИЕ ЛЕКЦИИ:

1. Этапы разработки экспертных систем

В настоящее время сложилась определенная технология разработки ЭС, которая включает следующие шесть этапов (рис.1):

- идентификация,
- концептуализация,
- формализация,
- выполнение,
- тестирование
- опытная эксплуатация.

На этапе **идентификации** определяются задачи, которые подлежат решению, выявляются цели разработки, определяются эксперты и типы пользователей. Идентификация задачи заключается в составлении неформального (вербального) описания, в котором указываются: общие характеристики задачи; подзадачи, выделяемые внутри данной задачи; ключевые понятия (объекты), их входные (выходные) данные; предположительный вид решения, а также знания, относящиеся к решаемой задаче.

Результатом данного этапа является ответ на вопрос, что надо сделать и какие ресурсы необходимо задействовать (идентификация задачи, определение участников процесса проектирования и их роли, выявление ресурсов и целей). На выходе этапа – требования к экспертной системе.

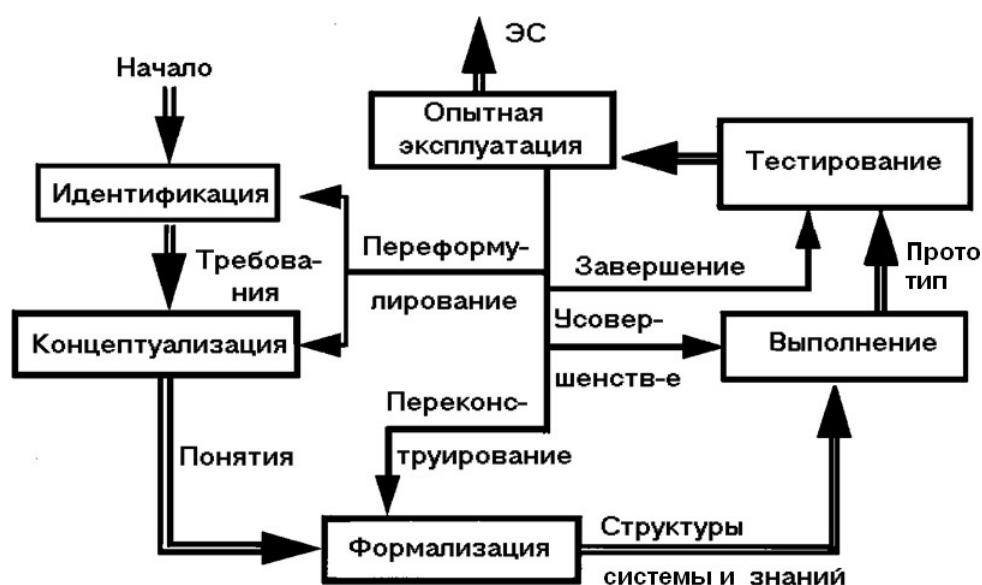


Рис.1 Этапы разработки ЭС.

На этапе **концептуализации** проводится содержательный анализ проблемной области и решаемой задачи, выявляются используемые понятия и их взаимосвязи, определяются методы решения задачи.

Этот этап завершается созданием модели предметной области, включающей основные концепты и отношения. На этапе концептуализации определяются следующие особенности задачи: типы доступных данных; исходные и выводимые данные, подзадачи общей задачи; используемые стратегии и гипотезы; виды взаимосвязей между объектами, типы используемых отношений (иерархия, причина — следствие, часть — целое и т.п.); процессы, используемые в ходе решения; состав знаний, используемых при решении задачи; типы ограничений, накладываемых на процессы, используемые в ходе решения; состав знаний, используемых для обоснования решений. На выходе — понятия и их взаимосвязи.

На этапе **формализации** выбираются инструментальные средства разработки, определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, разрабатывается программная оболочка (прототип), моделируется работа системы, оценивается адекватность системы.

Выходом этапа формализации является описание того, как рассматриваемая задача может быть представлена в выбранном или разработанном формализме. Сюда относится указание способов представления знаний (фреймы, сценарии, семантические сети и т.д.) и определение способов манипулирования этими знаниями (логический вывод, аналитическая модель, статистическая модель и др.) и интерпретации знаний. В конце этапа имеем структуру системы.

На этапе **выполнения** осуществляется создание одного или нескольких прототипов ЭС, решающих требуемые задачи, наполнение экспертом базы знаний. В связи с тем, что основой ЭС являются знания, данный этап является наиболее важным и наиболее трудоемким этапом разработки ЭС. Процесс приобретения знаний разделяют на извлечение знаний из эксперта, формализацию знаний, обеспечивающую эффективную работу системы, и представление знаний в виде, понятном ЭС. Процесс приобретения знаний осуществляется инженером по знаниям на основе анализа деятельности эксперта по решению реальных задач. На выходе этапа — программная реализация всех структурных элементов экспертной системы.

Этап **тестирования**. В ходе данного этапа производится оценка выбранного способа представления знаний в ЭС в целом. Для этого инженер по знаниям совместно с экспертом подбирает тестовые примеры, обеспечивающие проверку возможностей разработанной ЭС. При подготовке тестовых примеров следует классифицировать их по подпроблемам предметной области, выделяя стандартные случаи, определяя границы трудных ситуаций и т.п. Степень соответствия решения тестового примера полученного экспертной системой и известного решения характеризует качество разработки системы. Главным в оценке работы системы является полнота и безошибочность работы правил вывода.

На этом этапе **опытной эксплуатации** проверяется пригодность ЭС для конечного пользователя. Пригодность ЭС для пользователя определяется в основном удобством работы с ней и ее полезностью. Под полезностью ЭС понимается ее способность в ходе диалога определять потребности пользователя, выявлять и устранять причины неудач в работе, а также удовлетворять указанные потребности пользователя (решать поставленные задачи). В свою очередь, удобство работы с ЭС подразумевает естественность взаимодействия с ней (общение в привычном, не утомляющем пользователя виде), гибкость ЭС (способность системы настраиваться на различных пользователей, а также учитывать изменения в квалификации одного и того же пользователя) и устойчивость системы к ошибкам (способность не выходить из строя при ошибочных действиях неопытного пользователя).

В ходе разработки ЭС почти всегда осуществляется ее модификация. Выделяют следующие виды модификации системы: переформулирование понятий и требований, переконструирование представления знаний в системе и усовершенствование прототипа.

1.1. Коллектив разработчиков ЭС.

Под коллективом разработчиков понимается группа специалистов, ответственных за создание ЭС. В разработке ЭС участвуют представители следующих специальностей.

Эксперт определяет знания (данные и правила), характеризующие проблемную область, обеспечивает полноту и правильность введенных в ЭС знаний.

Инженер по знаниям помогает эксперту выявить и структурировать знания, необходимые для работы ЭС; осуществляет выбор того инструментального средства, которое наиболее подходит для данной проблемной области, и определяет способ представления знаний. Руководит коллективом.

Программист(ы) осуществляет программную реализацию ЭС, сопрягает все основные компоненты ЭС.

Пользователь осуществляет проверку удобства работы с ЭС на заключительных этапах разработки. Конечно же его нельзя отнести к специалистам, однако его роль достаточно важна.

Таким образом, минимальный состав коллектива включает четыре человека; реально же он разрастается до 8- 10 человек. Численное увеличение коллектива разработчиков происходит по следующим причинам: необходимость учета мнения нескольких пользователей, помощи нескольких экспертов; потребность как в проблемных, так и системных программистах. На Западе в этот коллектив дополнительно традиционно включают менеджера и одного технического помощника. При отсутствии профессионального менеджера руководителем, участвующим во всех стадиях разработки, является инженер по знаниям, поэтому к его квалификации предъявляются самые высокие требования. В целом уровень и численность группы зависят от характеристик поставленной задачи.

Для обеспечения эффективности работы любой творческой группы, в том числе и группы разработчиков ЭС, необходимо возникновение атмосферы взаимопонимания и доверия, которое, в свою очередь, обусловлено психологической совместимостью членов группы; следовательно, при формировании группы должны учитываться психологические свойства участников.

Приведем два аспекта характеристик членов коллектива разработчиков ЭС: 1 — психофизиологический, 2 — профессиональный.

Эксперт.

1. *Психологические качества.* Эксперт — чрезвычайно важная фигура в группе. В конечном счете, его подготовка определяет уровень компетенции базы знаний.

Желательные качества:

- а) доброжелательность;
- б) готовность поделиться своим опытом;
- в) умение объяснить (педагогические навыки);
- г) заинтересованность (моральная, а лучше еще и материальная) в успешности разработки.

Возраст эксперта обычно почтенный, что необходимо учитывать всем членам группы. Часто встает вопрос о количестве экспертов. Поскольку проблема совмещения подчас противоречивых знаний остается открытой, обычно с каждым из экспертов работают индивидуально, иногда создавая альтернативные базы.

2. *Профессиональные качества.* Помимо безусловно высокого профессионализма в выбранной предметной области, желательно знакомство эксперта с популярной литературой по искусственному интеллекту и экспертным системам для того, чтобы эффективнее прошел этап извлечения знаний.

Инженер по знаниям.

1. *Психологические качества.* Существуют такие профессии и виды деятельности, для которых природные качества личности (направленность, способности, темперамент) могут иметь характер абсолютного показания или противопоказания к занятиям. По-видимому, инженерия знаний принадлежит к таким профессиям. По различным оценкам, это одна из самых малочисленных, высокооплачиваемых и дефицитных в мире специальностей. Попытаемся дать наброски к портрету инженера по знаниям (без претензии на полноту и точность определений).

Пол. Психологи утверждают, что мужчины более склонны к широкому охвату явлений и в среднем у них выше аналитичность, чрезвычайно полезная инженеру по знаниям, которому надо иметь развитое логическое мышление и умение оперировать сложными формальными структурами.

Кроме того, при общении с экспертами, которые в большинстве своем настроены скептически по отношению к будущей ЭС, инженер по знаниям – мужчина вызывает более высокую мотивацию успешности со стороны эксперта-женщины. С другой стороны, известно, что у женщин выше наблюдательность к отдельным деталям объектов. Так что пол не является окончательным показанием или проти-

вопоказанием к данной профессии.

Интеллект. Это понятие вызывает самые бурные споры психологов; существует до 50 определений интеллекта, но с прагматической точки зрения очевидно, что специалист в области искусственного интеллекта должен стремиться к максимальным оценкам по тестам как вербального, так и невербального интеллекта.

Стиль общения. Инженер по знаниям «задает тон» в общении с экспертом, он ведет диалог, и от него в конечном счете зависит его продуктивность. Можно выделить два стиля общения: деловой (или жесткий) и дружеский (или мягкий, деликатный). Нам кажется, что дружеский будет заведомо более успешным, так как снижает «эффект фасада» у эксперта, раскрепощает его. Деликатность, внимательность, интеллигентность, ненавязчивость, скромность, умение слушать и задавать вопросы, хорошая коммуникабельность и в то же время уверенность в себе — вот рекомендуемый стиль общения. Безусловно, что это дар и искусство одновременно, однако занятия по психологическому тренингу могут дать полезные навыки.

Портрет инженера по знаниям можно было бы дополнить другими характеристиками — широтой взглядов и интересов, артистичностью, чувством юмора, обаянием и т. д.

2. Профессиональные качества. При определении профессиональных требований к аналитику следует учитывать, что ему необходимы различные навыки и умения для грамотного и эффективного проведения процессов извлечения, концептуализации и формализации знаний.

Инженер по знаниям имеет дело со всеми формами знаний Z1 (знания в памяти) Z2 (знания в книгах) Z3 (поле знаний) Z4 (модель знаний) Z5 (база знаний).

Работа на уровне Z1 требует от инженера по знаниям знакомства с элементами когнитивной психологии и способами репрезентации понятий и процессов в памяти человека, с двумя основными механизмами мышления – логическим и ассоциативным, с такими способами активизации мышления как игры, мозговой штурм и др., с различными моделями рассуждений.

Изучение и анализ текстов на уровне Z2 подразумевает широкую общенаучную подготовку инженера; знакомство с методами реферирования и аннотирования текстов; владение навыками быстрого чтения, а также текстологическими методами извлечения знаний.

Разработка поля знаний на уровне Z3 требует квалифицированного знакомства с методологией представления знаний, системным анализом, теорией познания, аппаратом многомерного шкалирования, кластерным и факторным анализом.

Разработка формализованного описания Z4 предусматривает предварительное изучение аппарата математической логики и современных языков представления знаний. Модель знаний разрабатывается на основании результатов глубокого анализа инструментальных средств

разработки ЭС и имеющихся «оболочек». Кроме того, инженеру по знаниям необходимо владеть методологией разработки ЭС, включая методы быстрого прототипирования.

И наконец, реализация базы знаний Z5, в которой инженер по знаниям участвует вместе с программистом, подразумевает овладение практическими навыками работы

на ЭВМ и, возможно, одним из языков программирования. Так как инженеров по знаниям «выращивают» из программистов, уровень Z5 обычно не вызывает затруднения, особенно если разработка ведется на традиционных языках типа С или Паскаль. Специализированные языки искусственного интеллекта Лисп и Пролог требуют некоторой перестройки архаично-алгоритмического мышления.

Программист.

1. *Психологические качества.* Известно, что программисты обладают самой низкой потребностью в общении среди представителей разных профессий. Однако при разработке ЭС необходим тесный контакт членов группы, поэтому желательны следующие его качества:

- а) общительность;
- б) способность отказаться от традиционных навыков и освоить новые методы;
- в) интерес к разработке.

2. *Профессиональные качества.* Поскольку современные ЭС — сложнейшие и дорогостоящие программные комплексы, программисты должны иметь опыт и навыки разработки программ. Обязательно знакомство с основными структурами представления знаний и механизмами вывода, состоянием отечественного и мирового рынка программных продуктов для разработки ЭС и диалоговых интерфейсов.

Пользователь.

1. *Психологические качества.* К пользователю предъявляются самые слабые требования, поскольку его не выбирают. Он является в некотором роде заказчиком системы. Желательные качества:

- а) дружелюбие;
- б) умение объяснить, что же он хочет от системы;
- в) отсутствие психологического барьера к применению вычислительной техники;
- г) интерес к новому.

От пользователя зависит, будет ли применяться разрабатываемая ЭС. Замечено, что наиболее ярко качества в) и г) проявляются в молодом возрасте, поэтому иногда такие пользователи охотнее применяют ЭС, не испытывая при этом комплекса неполноценности оттого, что ЭВМ им что-то подсказывает.

2. *Профессиональные качества.* Необходимо, чтобы пользователь имел некоторый базовый уровень квалификации, который позволит ему правильно истолковать рекомендации ЭС. Кроме того, должна быть полная совместимость в терминологии интерфейса к ЭС с той, которая привычна и удобна для пользователя. Обычно требования к квалификации пользователя не очень велики, иначе он переходит в разряд экспертов и совершенно не нуждается в ЭС.

Таким образом, успешность выбора и подготовки коллектива разработчиков ЭС определяет эффективность и продолжительность всего процесса разработки.

1.2. Основы разработки экспертной системы Постановка задачи.

Прежде всего, поставим задачу, для решения которой будет разрабатываться экспертная система. Подходящей задачей, при решении которой можно использовать обратную цепочку рассуждений, может быть задача, вытекающая из следующей ситуации: к директору крупной технической фирмы пришел человек, желающий устроиться на работу. Директор располагает сведениями о потребностях фирмы в специалистах и общем положении дел в фирме. Ему нужно решить, какую должность в фирме может занять посетитель. Для этого необходимо задать посетителю такие вопросы, ответы на которые дадут возможность сделать правильный выбор должности.

На первый взгляд задача не очень сложная, но на решение директора влияет много факторов. Допустим, претендент работает в данной области недавно, но уже сделал важное открытие или он закончил учебное заведение с посредственными оценками, но несколько лет работал по специальности. В данной ситуации люди ведут себя по-разному, и хотя для того, чтобы получить работу необходимо, удовлетворять определенным критериям, в биографии претендента могут быть самые различные факты, анализ которых поможет подобрать для него соответствующую должность. Поскольку в задаче надо выбрать один из нескольких возможных вариантов (должностей), для её решения можно воспользоваться обратной цепочкой рассуждений.

Таким образом, необходимо разработать экспертную систему, которая определит подходящую должность посетителю. Экспертная система будет содержать экспертные знания директора и заменит его при решении описанной задачи. Такой системой может пользоваться как сам посетитель, так и менее квалифицированный (не эксперт - директор) сотрудник фирмы.

Итак, задача поставлена. Теперь нужно наглядно ее представить. Для описания подобных задач обычно используются диаграммы, которые называются деревьями решений. Деревья решений дают необходимую наглядность и позволяют проследить ход рассуждений.

Разработка дерева решений.

Дерево решений – это ориентированный граф, вершинами которого являются условия и выводы, а дугами результат выполнения (проверки) условий.

Диаграммы называются деревьями решений потому, что, подобно настоящему дереву, имеют ветви. Ветви деревьев решений заканчиваются логическими выводами. Для рассматриваемого примера вывод заключается в том, предложит ли директор должность поступающему на работу, и если да, то какую. Многие задачи сложны, и их непросто представить (или для их решения не собираются использовать экспертную систему). Дерево решений помогает преодолеть эти трудности.

На рис.2 показано дерево решений для рассматриваемого примера приема на работу. Видно, что диаграмма состоит из кружков и прямоугольников, которые называются вершинами. Каждой вершине присваивается номер. На вершины можно ссылаться по этим номерам. Номера вершин можно выбрать произвольно, т.к они и служат только

для удобства идентификации, за исключением первой вершины. Линии, соединяющие вершины, называются дугами. Совокупность вершин и дуг называется ветвями. Кружки, содержащие вопросы, называются вершинами условий. Прямоугольники содержат логические выводы. Линии (стрелки) показывают направление диаграммы. Подписи возле линий это ответы на вопрос, содержащийся в вершине условия. Вершины условий могут иметь сразу по несколько выходящих линий (стрелок), связывающих их с другими вершинами. В этом случае каждая линия (стрелка) должна быть четко определена. Не может быть две линии, у которых подписи одинаковые, например, подпись «Да». Выбор выходящей из вершины ветви определяется проверкой условия (вопроса), содержащегося в вершине. В программе под каждую вершину отводится переменная, а затем ей присваивается значение (ответ посетителя). Можно сказать, что вершины содержат переменные, а пути - это условия, в соответствии с которыми переменным присваиваются значения. В дереве решений могут быть локальные (частные) выводы или цели. Для рассматриваемого примера локальным выводом может быть содержащийся в прямоугольнике 3 ответ на вопрос, будет ли посетителю предложена должность. Однако эта вершина имеет и исходящие ветви, и, следовательно, через неё может проходить путь к следующему логическому выводу. В последнем случае, поскольку исходящая ветвь не содержит условия и она только одна, говорят, что вершина содержит локальный вывод для другой цели. Локальный вывод - это также составляющая условной части (ЕСЛИ) правила.

Повторять шаг 2 до тех пор, пока не будут исчерпаны все вершины условия, расположенные в обратном направлении стрелок от зафиксированной вершины вывода, или не встретится вершина локального вывода. Если встретилась вершина локального вывода, то её надо зафиксировать и прекратить выполнение шага 2. Каждая вершина условия (кружок), составляющая путь,

- это одна из переменных части ЕСЛИ правила. Эти вершины объединяются логическим оператором **И**

Выбранный на шаге 1 логический вывод перенести в часть ТО правила.

Пример создание правила. В качестве примера рассмотрим путь 6, 4, 5, 3. Создание правила начинается с вывода (вершина 6) и дерево решения просматривается в обратную сторону. Просмотр данной ветви (пути) заканчивается на вершине 3, которая является локальным выводом. Если бы вершины 3 не было в дереве решений, то путь закончился бы на вершине 1.

Применив полученный путь, запишем правило в следующем виде:

ЕСЛИ посетитель, возможно, будет принят на работу = да

И средний балл за время учебы $\geq 3,5$

И посетитель имеет изобретения = да,

ТО предложенная должность = научный сотрудник.

Для каждой вершины логического вывода определяется путь и записывается правило. В рассматриваемом дереве решений имеется 6 вершин логического вывода (прямоугольников), в том числе и локальный вывод в вершине 3, значит будет составлено 6 правил.

Длинную фразу — посетитель, возможно, будет принят на работу || можно заменить переменной, принимающей значения — да|| или — нет||. Список имен переменных, текст, который они заменяют, и номера вершин пути сводят в таблицу, (табл.1).

Использование переменных вместо полного текста упрощает формирование и запись правил.

Таблица 1 Таблица имён переменных

Имя переменной	Условия	Вершина(ы)
DEGREE	Посетитель имеет ученое звание?	1
QUALIFY	Посетитель, возможно, будет принят на работу	3
PATENT	Посетитель имеет изобретения?	4
EXPERIENCE	Стаж работы по специальности?	7

GRADE	Средний балл за время учебы?	5
POSITION	Предложенная должность	2,6,8,9,10

Используя имена переменных из табл.1, вышеприведенное правило можно запишется в следующем виде:

ЕСЛИ QUALIFY = да **И** GRADE $\geq$ 3,5 **И** PATENT = да,
ТО POSITION = научный сотрудник

В табл.2 приведены все правила для дерева решений, показанного на рис.6.1. Правила соответствуют всем шести путям, ведущим к шести возможным целям дерева решений. Правила желательно пронумеровать. Совокупность правил является формализованными знаниями (в рассматриваемом примере – знаниями руководителя) и представляет собой базу знаний.

Таблица 2 База знаний

№	Правило	Путь
10	ЕСЛИ DEGREE = <i>НЕТ</i> , ТО POSITION= <i>ОТКАЗАТЬ</i>	2, 1
20	ЕСЛИ DEGREE = <i>ДА</i> , ТО QUALIFY= <i>ДА</i>	3, 1
30	ЕСЛИ QUALIFY = <i>ДА</i> И GRADE $\geq$ 3,5 И PATENT = <i>ДА</i> , ТО POSITION = <i>НАУЧНЫЙ СОТРУДНИК</i>	6,4,5,3
40	ЕСЛИ QUALIFY = <i>ДА</i> И GRADE $\geq$ 3,5 И PATENT = <i>НЕТ</i> , ТО POSITION = <i>ИНЖЕНЕР КОНСТРУКТОР</i>	8,4,5,3
50	ЕСЛИ QUALIFY = <i>ДА</i> И AVERAGE $\leq$ 3,5 И EXPERIENCE $<$ 2, ТО POSITION= <i>ОТКАЗАТЬ</i>	9,7,5,3
60	ЕСЛИ QUALIFY = <i>ДА</i> И AVERAGE $\leq$ 3,5 И EXPERIENCE $>$ 2, ТО POSITION = <i>ИНЖЕНЕР ПО ЭКСПЛУАТАЦИИ</i>	10,7,5, 3

Таким образом, дерево решений позволяет просто и наглядно представить ход рассуждений эксперта при решении задачи и формировать правила для базы знаний, а без базы знаний экспертную систему не построить. Аналогично можно построить

базу знаний для своей проблемной области или решаемой задаче.

Структуры данных экспертной системы.

При создании экспертной системы для упрощения ответа на вопросы и решения поставленной задачи в систему включается ряд полезных таблиц или структур данных. Структуры данных нужны для работы с базой знаний. После определения метода решения выбранного круга задач можно приступить к разработке системы.

Список логических выводов - это структура данных, содержащая упорядоченный список возможных логических выводов.

Список состоит из номера правила, логического вывода, связанного с этим правилом, и условий, которые формируют вывод. На каждое правило базы знаний в списке приходится одна запись. Создание записи списка поясним на примере правила 10. Часть ТО правила 10 содержит переменную POSITION, т.е. переменная POSITION связана с логическим выводом правила 10.

Список логических выводов используется исключительно для поиска вывода по номеру правила. Когда условия части ЕСЛИ истинны, вызывается часть ТО правила, ей присваивается значение. Например, если надо узнать, будет ли посетителю предложена работа, в списке ищется переменная POSITION. Она содержится в первой же записи, т.е. в правиле 10:

ЕСЛИ DEGREE=НЕТ,
ТО POSITION=НЕТ

Посетитель не будет принят на работу, если переменная DEGREE имеет значение НЕТ. Если же переменная DEGREE имеет значение ДА, тогда обращаться к части ТО правила нельзя, поскольку не выполняется условие части ЕСЛИ (DEGREE=ДА). Поэтому надо продолжить поиск правила, содержащего в части ТО переменную POSITION (в данном случае правило 30).

На рис.3 приведён полностью сформированный список логических выводов для всех правил базы знаний.

10 POSITION
20 QUALIFY
30 POSITION
40 POSITION
50 POSITION
60 POSITION

Рис. 3. Список логических выводов.

Список считается сформированным, когда логический вывод каждого правила помещён в запись с номером, совпадающим с номером правила.

Список переменных – это перечень имен переменных для всех условных частей правил базы знаний и признак их инициализации.

Признак инициализации показывает, присвоено ли переменной значение. Независимо от того, в скольких условиях встречается переменная, в список переменных она включается всего один раз. В этот список также нельзя включать переменные из списка логических выводов, поскольку их значения определяются с помощью правил. Например, правило 20 использует переменную QUALIFY. Список переменных приведён в таблице3.

Таблица3. Список имен переменных

Имя	Признак	Значение
DEGREE	I	НЕТ
PATENT	NI	
EXPEREIENCE	NI	
GRADE	NI	

Первоначально предполагается, что переменным значения еще не присвоены и признак инициализации для всех переменных равен NI. По мере того как полученная от посетителя информация передается системе и переменным присваиваются значения, признак инициализации меняется на I. С этого момента, в каком бы правиле в условной части не встретилась переменная, она будет считаться проинициализированной, имеющей какое либо значение и ее можно использовать для работы с любыми правилами.

Таким образом, до того, как правило включается в работу, все переменные, входящие в его условную часть, должны быть проинициализированы.

Список переменных условия – это перечень всех переменных для всех условных частей всех правил базы знаний.

Условная часть правила (ЕСЛИ) может содержать несколько переменных. Под каждое правило выделяется одинаковое число позиций в списке переменных условия. Минимальное число позиций равно числу переменных условия самого «длинного» правила. Можно добавить еще одну-две позиции «про запас» на случай доработки базы правил.

На рис.3.10 показан список переменных условия для шести правил рассматриваемой базы знаний. Для простоты программирования предполагается, что каждое правило не может содержать больше четырех переменных условия (т.к. самое длинное правило, например №40, содержит три переменных условия). Четвертая позиция добавлена «про запас».

1	DEGREE
2	
3	
4	
5	DEGREE
6	
7	
8	-----
9	
	QUALIF Y
10	GRADE
11	
	PATEN T
12	
13	
	QUALIF Y
14	-----
15	GRADE
	PATEN T
16	
17	
	QUALIF Y
18	GRADE
19	
	EXPERI ENCE
20	
21	
	QUALIF Y
22	GRADE
23	EXPERIENCE
24	

Рис.4. Список переменных условия.

Слева от имен переменных даны числа (1-24), указывающие индекс элемента массива (по четыре на правило), в который помещается имя соответствующей переменной. Незанятые элементы массива, отведенные правилу, остаются пустыми. В принципе можно запрограммировать любое число переменных для каждого правила. Однако при отведении места под переменные условия лучше для каждого правила резервировать одинаковое число элементов массива. Это упростит вычисление индекса первого элемента, отведенного правилу в списке. Его можно вычислить с помощью простой формулы:

$N^{\circ} = 4 * (\text{номер правила} / 10 - 1) + 1$.

Например, переменные правила 50 будут размещаться, начиная с 17-го элемента массива: $4 * (50/10 - 1) + 1 = 17$. $N^{\circ} = 17$.

Теперь посмотрим, каким образом три описанные структуры данных соотносятся с мыслительной деятельностью человека в процессе обратной цепочки рассуждений. Прежде всего, человек просматривает все возможные пути, способные привести к решению задачи (список логических выводов). Затем он выделяет условия, составляющие эти пути (список переменных и список переменных условия). Такие структуры данных позволяют быстро обрабатывать информацию, не повторяя одни и те же шаги по несколько раз, потому что значения переменных можно использовать в определенной ситуации для различных логических выводов. Если же при разговоре с человеком, устраивающимся на работу, у директора нет не только компьютера, но даже карандаша и бумаги, ему придется много раз переспрашивать, ведь сразу просто невозможно запомнить. Конечно, в конце концов, он примет решение, но затратит много сил и времени.

2. Инструментальные средства проектирования и разработки экспертных систем.

Оболочки экспертных систем

Инструментальные средства для проектирования и разработки экспертных систем можно классифицировать по следующим основным параметрам:

- уровень используемого языка;
- парадигмы программирования и механизмы реализации;
- способ представления знаний;
- механизмы вывода и моделирования;
- средства приобретения знаний.

2.1 Уровень используемого языка

Мощность и универсальность языка программирования определяет трудоемкость разработки ЭС.

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

инструментальных средств).

2. Специальные языки программирования (например, язык LISP, ориентированный на обработку списков; язык логическо-го программирования PROLOG; язык рекурсивных функций PE-ФАЛ, OCAML, Haskell и т.д.). Их недостатком является слабая приспособленность к объединению с программами, написанными на языках традиционного программирования.
3. Инструментальные средства, содержащие многие, но не все компоненты ЭС (например, система OPS 5, которая поддерживает производственный подход к представлению знаний; языки KRL и FRL, используемые для разработки ЭС с фреймовым представлением знаний). Такое программное обеспечение предназначено для разработчиков, владеющих технологиями программирования и умеющих интегрировать разнородные компоненты в программный комплекс.
4. Оболочки ЭС общего назначения, содержащие все программные компоненты, но не имеющие знаний о конкретных предметных средах. Средства этого типа и последующего не требуют от разработчика приложения знания программирования. Примерами являются ЭКО, Leonardo, Nexpert Object, Карпа, EXSYS, GURU, ART, KEE и др. В последнее время все реже употребляется термин «оболочка», его заменяют более широким термином «среда разработки». Если хотят подчеркнуть, что средство используется не только на стадии разработки приложения, но и на стадиях использования и сопровождения, то употребляют термин «полная среда» (complete environment). Для поддержания всего цикла создания и сопровождения программ используются интегрированные инструментальные системы, например, KEATS, VITAL. Основными компонентами системы KEATS являются: ACQUIST – средства фрагментирования текстовых источников знаний, позволяющие разбивать текст или протокол беседы с экспертом на множество взаимосвязанных, аннотированных фрагментов и создавать понятия (концепты); FLIK – язык представления знаний средствами фреймовой модели; GIS – графический интерфейс, используемый для создания гипертекстов и концептуальных моделей, а также для проектирования фреймовых систем; ERI – интерпретатор правил, реализующий процедуры прямого и обратного вывода; TRI – инструмент визуализации логическо-го вывода, демонстрирующий последовательность выполнения правил; Tables – интерфейс манипулирования таблицами, используемыми для хранения знаний в БЗ; CS – язык описания и распространения ограничений; TMS – система поддержания истинности. При использовании инструментария данного типа могут возникнуть следующие трудности:
 - а) управляющие стратегии, заложенные в механизм вывода, могут не соответствовать методам решения, которые использует эксперт, взаимодействующий с данной системой, что может привести к неэффективным, а возможно, и неправильным решениям;
 - б) способ представления знаний, используемый в инструментарии, мало подходит для описания знаний конкретной предметной области.

Большая часть этих трудностей разрешена в проблемно/предметно-ориентированных средствах разработки ИИС.

5. Проблемно/предметно-ориентированные оболочки и среды (не требуют знания программирования):

- проблемно-ориентированные средства – предназначены для решения задач определенного класса (задачи поиска, управления, планирования, прогнозирования и др.) и содержат соответствующие этому классу альтернативные функциональные модули;
- предметно-ориентированные средства – включают знания о типах предметных областей, что сокращает время разработки БЗ.

При использовании оболочек и сред разработчик приложения полностью освобождается от программирования, его основные трудозатраты связаны с формированием базы знаний.

2.2 Парадигмы программирования и механизмы реализации

Способы реализации механизма исполняемых утверждений часто называют парадигмами программирования. К основным парадигмам относят следующие:

- процедурное программирование;
- программирование, ориентированное на данные;
- программирование, ориентированное на правила;
- объектно-ориентированное программирование.

Парадигма процедурного программирования является самой распространенной среди существующих языков программирования (например, С и Паскаль). В процедурной парадигме активная роль отводится процедурам, а не данным; причем любая процедура активизируется вызовом. Подобные способы задания поведения удобны для описаний последовательности действий одного процесса или нескольких взаимосвязанных процессов.

При использовании программирования, ориентированного на данные, активная роль принадлежит данным, а не процедурам. Здесь со структурами активных данных связывают некоторые действия (процедуры), которые активизируются тогда, когда осуществляется обращение к этим данным.

В парадигме, ориентированной на правила, поведение определяется множеством правил вида «условие-действие». Условие задает образ данных, при возникновении которого действие правила может быть выполнено. Правила в данной парадигме играют такую же роль, как и операторы в процедурной парадигме. Однако если в процедурной парадигме поведение задается последовательностью операторов, не зависящей от значений обрабатываемых данных, то в парадигме, ориентированной на правила, поведение не задается заранее предписанной последовательностью правил, а формируется на основе значений данных, которые в текущий момент обрабатываются программой. Подход, ориентированный на правила, удобен для описания поведения, гибко и разнообразно реагирующего на большое многообразие

состояний данных.

Парадигма объектного программирования в отличие от процедурной парадигмы не разделяет программу на процедуры и данные. Здесь программа организуется вокруг сущностей, называемых объектами, которые включают локальные процедуры (методы) и локальные данные (переменные). Поведение (функционирование) в этой парадигме организуется путем пересылки сообщений между объектами. Объект, получив сообщение, осуществляет его локальную интерпретацию, основываясь на локальных процедурах и данных. Такой подход позволяет описывать сложные системы наиболее естественным образом. Он особенно удобен для интегрированных ЭС.

2.3 Способ представления знаний

Наличие многих способов представления знаний вызвано стремлением представить различные типы проблемных сред с наибольшей эффективностью. Обычно способ представления знаний в ЭС характеризуют моделью представления знаний.

Типичными моделями представления знаний являются правила (продукции), фреймы (или объекты), семантические сети, логические формулы. Инструментальные средства, имеющие в своем составе более одной модели представления знаний, называют гибридными. Большинство современных средств, как правило, использует объектно-ориентированную парадигму, объединенную с парадигмой, ориентированной на правила.

2.4 Механизмы вывода и моделирования

В статических ЭС единственным активным агентом, изменяющим информацию, является механизм вывода экспертной системы. В динамических ЭС изменение данных происходит не только вследствие функционирования механизма исполняемых утверждений, но также в связи с изменениями окружения задачи, которые моделируются специальной подсистемой или поступают извне. Механизмы вывода в различных средах могут отличаться способами реализации следующих процедур.

1. Структура процесса получения решения:

- построение дерева вывода на основе обучающей выборки (индуктивные методы приобретения знаний) и выбор маршрута на дереве вывода в режиме решения задачи;
- компиляция сети вывода из специфических правил в режиме приобретения знаний и поиск решения на сети вывода в режиме решения задачи;

генерация сети вывода и поиск решения в режиме решения задачи, при этом генерация сети вывода осуществляется в ходе выполнения операции сопоставления, определяющей пары «правило – совокупность данных», на которых условия этого правила удовлетворяются;

- в режиме решения задач ЭС осуществляет выработку правдоподобных предположений (при отсутствии достаточной информации для решения); выполнение рассуждений по обоснованию (опровержению) предположений; генерацию альтернативных сетей вывода; поиск решения в сетях вывода.

2. Поиск (выбор) решения:

- направление поиска – от данных к цели, от целей к данным, двунаправленный поиск;
- порядок перебора вершин в сети вывода – «поиск в ширину», при котором сначала обрабатываются все вершины, непосредственно связанные с текущей обрабатываемой вершиной G; «поиск в глубину», когда сначала раскрывается одна наиболее значимая вершина – G1 связанная с текущей G, затем вершина G1 делается текущей, и для нее раскрывается одна наиболее значимая вершина G2 и т. д.

3. Процесс генерации предположений и сети вывода:

- режим – генерация в режиме приобретения знаний, генерация в режиме решения задачи;
- полнота генерируемой сети вывода – операция сопоставления применяется ко всем правилам и ко всем типам указанных в правилах сущностей в каждом цикле работы механизма вывода (обеспечивается полнота генерируемой сети); используются различные средства для сокращения количества правил и (или) сущностей, участвующих в операции сопоставления; например, применяется алгоритм сопоставления или используются знания более общего характера (метазнания).

Механизм вывода для динамических проблемных сред дополнительно содержит: планировщик, управляющий деятельностью ЭС в соответствии с приоритетами; средства, гарантирующие получение лучшего решения в условиях ограниченности ресурсов; систему поддержания истинности значений переменных, изменяющихся во времени.

В динамических инструментальных средствах могут быть реализованы следующие варианты подсистемы моделирования:

- система моделирования отсутствует;
- существует система моделирования общего назначения, являющаяся частью инструментальной среды;

существует специализированная система моделирования, являющаяся внешней по отношению к программному обеспечению, на котором реализуется ЭС.

2.5 Средства приобретения знаний

В инструментальных системах они характеризуются следующими признаками:

1. Уровень языка приобретения знаний:

- формальный язык;
- ограниченный естественный язык;
- язык пиктограмм и изображений;
- ЕЯ и язык изображений.

2. Тип приобретаемых знаний:

- данные в виде таблиц, содержащих значения входных и выходных атрибутов, по которым индуктивными методами строится дерево вывода;

- специализированные правила;
- общие и специализированные правила.

Тип приобретаемых данных:

- атрибуты и значения;
- объекты;

классы структурированных объектов и их экземпляры, получающие значения атрибутов путем наследования.

2.6. Экспертные системы реального времени

Среди специализированных систем, основанных на знаниях, наиболее значимы экспертные системы реального времени, или динамические экспертные системы. На их долю приходится 70 процентов этого рынка.

Значимость инструментальных средств реального времени определяется не столько их бурным коммерческим успехом (хотя и это достойно тщательного анализа), но, в первую очередь, тем, что только с помощью подобных средств создаются стратегически значимые приложения в таких областях, как управление непрерывными производственными процессами в химии, фармакологии, производстве цемента, продуктов питания и т.п., аэро-космические исследования, транспортировка и переработка нефти и газа, управление атомными и тепловыми электростанциями, финансовые операции, связь и многие другие.

Классы задач, решаемых экспертными системами реального времени, таковы: мониторинг в реальном масштабе времени, системы управления верхнего уровня, системы обнаружения неисправностей, диагностика, составление расписаний, планирование, оптимизация, системы-советчики оператора, системы проектирования.

Статические экспертные системы не способны решать подобные задачи, так как они не выполняют требования, предъявляемые к системам, работающим в реальном времени:

1. Представлять изменяющиеся во времени данные, поступающие от внешних источников, обеспечивать хранение и анализ изменяющихся данных.
2. Выполнять временные рассуждения о нескольких различных асинхронных процессах одновременно (т.е. планировать в соответствии с приоритетами обработку поступивших в систему процессов).
3. Обеспечивать механизм рассуждения при ограниченных ресурсах (время, память). Реализация этого механизма предъявляет требования к высокой скорости работы системы, способности одновременно решать несколько задач (т.е. операционные системы UNIX, VMS, Windows).
4. Обеспечивать "предсказуемость" поведения системы, т.е. гарантию того, что каждая задача будет запущена и завершена в строгом соответствии с временными ограничениями. Например, данное требование не допускает использования в экспертной системе реального времени механизма "сборки мусора", свойственного языку Lisp.

5. Моделировать "окружающий мир", рассматриваемый в данном приложении, обеспечивать создание различных его состояний.
6. Протоколировать свои действия и действия персонала, обеспечивать восстановление после сбоя.
7. Обеспечивать наполнение базы знаний для приложений реальной степени сложности с минимальными затратами времени и труда (необходимо использование объектно-ориентированной технологии, общих правил, модульности и т.п.).
8. Обеспечивать настройку системы на решаемые задачи (проблемная/предметная ориентированность).
9. Обеспечивать создание и поддержку пользовательских интерфейсов для различных категорий пользователей.
10. Обеспечивать уровень защиты информации (по категориям пользователей) и предотвращать несанкционированный доступ.

Подчеркнем, что кроме этих десяти требований средства создания экспертных систем реального времени должны удовлетворять и перечисленным выше общим требованиям.

История развития ИС для создания ЭС реального времени началась в 1985 г., когда фирма Lisp Machine Inc. выпустила систему Picon для символьных ЭВМ Symbolics.

Успех этого ИС привел к тому, что группа ведущих разработчиков Picon в 1986 г. образовала частную фирму Gensym, которая, значительно развив идеи, заложенные в Picon, в 1988 г. вышла на рынок с ИС под названием G2, версия 1.0.

Основное предназначение программных продуктов фирмы Gensym (США) - помочь предприятиям сохранять и использовать знания и опыт их наиболее талантливых и квалифицированных сотрудников в интеллектуальных системах реального времени, повышающих качество продукции, надежность и безопасность производства и снижающих производственные издержки. О том, как фирме Gensym удастся справиться с этой задачей, говорит хотя бы то, что сегодня ей принадлежат 50% мирового рынка экспертных систем, используемых в системах управления.

С отставанием от Gensym на 2 - 3 года другие фирмы начали создавать свои ИС для ЭС РВ. С точки зрения независимых экспертов NASA, проводивших комплексное исследование характеристик и возможностей некоторых из перечисленных систем, в настоящее время наиболее продвинутой ИС, безусловно, остается G2 (Gensym, США); следующие места со значительным отставанием (реализовано менее 50% возможностей G2) занимают RTWorks – фирма Talarian (США), COMDALE/C (Comdale Techn. – Канада), COGSYS (SC – США), ILOG Rules (ILOG – Франция).

Классы задач, для которых предназначена G2 и подобные ей системы:

- мониторинг в реальном масштабе времени;
- системы управления верхнего уровня;
- системы обнаружения неисправностей;
- диагностика;

- составление расписаний;
- планирование;
- оптимизация;
- системы – советчики оператора;
- системы проектирования.

Инструментальные средства фирмы Gensym являются эволюционным шагом в развитии традиционных экспертных систем от статических предметных областей к динамическим. Немалую долю успеха фирме Gensym обеспечивают основные принципы, которых она придерживается в своих новых разработках:

- проблемно/предметная ориентация;
- следование стандартам;
- независимость от вычислительной платформы;
- совместимость снизу-вверх с предыдущими версиями;
- универсальные возможности, не зависящие от решаемой задачи;
- обеспечение технологической основы для прикладных систем;

комфортная среда разработки;

- поиск новых путей развития технологии;
- распределенная архитектура клиент-сервер;
- высокая производительность.

Основным достоинством оболочки экспертных систем G2 для российских пользователей является возможность применять ее как интегрирующий компонент, позволяющий за счет открытости интерфейсов и поддержки широкого спектра вычислительных платформ легко объединить уже существующие, разрозненные средства автоматизации в единую комплексную систему управления, охватывающую все аспекты производственной деятельности – от формирования портфеля заказов до управления технологическим процессом и отгрузки готовой продукции. Это особенно важно для отечественных предприятий, парк технических и программных средств которых формировался по большей части бессистемно, под влиянием резких колебаний в экономике. Кроме системы G2, как базового средства разработки, фирма Gensym предлагает комплекс проблемно/предметно-ориентированных расширений для быстрой реализации сложных динамических систем на основе специализированных графических языков, включающих параметризуемые операторные блоки для представления элементов технологического процесса и типовых задач обработки информации. Набор инструментальных средств фирмы Gensym, сгруппированный по проблемной ориентации, охватывает все стадии производственного процесса и выглядит следующим образом:

- интеллектуальное управление производством – G2, G2 Diagnostic Assistant (GDA), NeurOn-Line (NOL), Statistical Process Control (SPC), BatchDesign_Kit;
- оперативное планирование – G2, G2 Scheduling Toolkit (GST), Dynamic Scheduling Packadge (DSP);
- разработка и моделирование производственных процессов – G2, ReThink, BatchDesign_Kit;

- управление операциями и корпоративными сетями – G2, Fault Expert.

Несмотря на то, что первая версия системы G2 появилась не так давно – в 1988 г., ее даже в богатой Америке никто не назовет дешевой. G2 можно назвать бестселлером на рынке программных продуктов – на начало 1996 г. в мире было установлено более 5000 ее копий. Фирма Gensym обслуживает более 30 отраслей – от аэрокосмических исследований до производства пищевых продуктов. Список пользователей G2 выглядит как справочник Who-Is-Who в мировой промышленности. 25 самых крупных индустриальных мировых корпораций используют G2. На базе G2 написано более 500 действующих приложений.

Успех G2 объясняется, прежде всего, тем, что G2 – динамическая система в полном смысле этого слова. G2 – это объектно-ориентированная интегрированная среда для разработки и сопровождения приложений реального времени, использующих базы знаний. G2 функционирует на большинстве существующих платформ.

База знаний G2 сохраняется в обычном ASCII-файле, который однозначно интерпретируется на любой из поддерживаемых платформ. Перенос приложения не требует его перекомпиляции и заключается в простом переписывании файлов. Функциональные возможности и внешний вид приложения не претерпевают при этом никаких изменений.

G2 объединяет в себе как универсальные технологии построения современных информационных систем (стандарты открытых систем, архитектура клиент/сервер, объектно-ориентированное программирование, использование ОС, обеспечивающих параллельное выполнение в реальном времени многих независимых процессов), так и специализированные методы (рассуждения, основанные на правилах, рассуждения, основанные на динамических моделях, или имитационное моделирование, процедурные рассуждения, активная объектная графика, структурированный естественный язык для представления базы знаний), а также интегрирует технологии систем, основанных на знаниях с технологией традиционного программирования (с пакетами программ, с СУБД, с контроллерами и концентраторами данных и т.д.) (рисунок 4.5).

Поскольку многие организации увеличивают свои инвестиции в информационные технологии и системы автоматизации, то людям становится все труднее интерпретировать и реагировать на большое количество данных, которые динамически изменяются во времени выполнения, не говоря уже о предупреждении критических ситуаций. Интеллектуальные справочные системы предохраняют людей от информационной перегрузки, перерабатывая оперативные данные в полезную информацию так, чтобы нужные действия могли быть приняты в нужные времена.

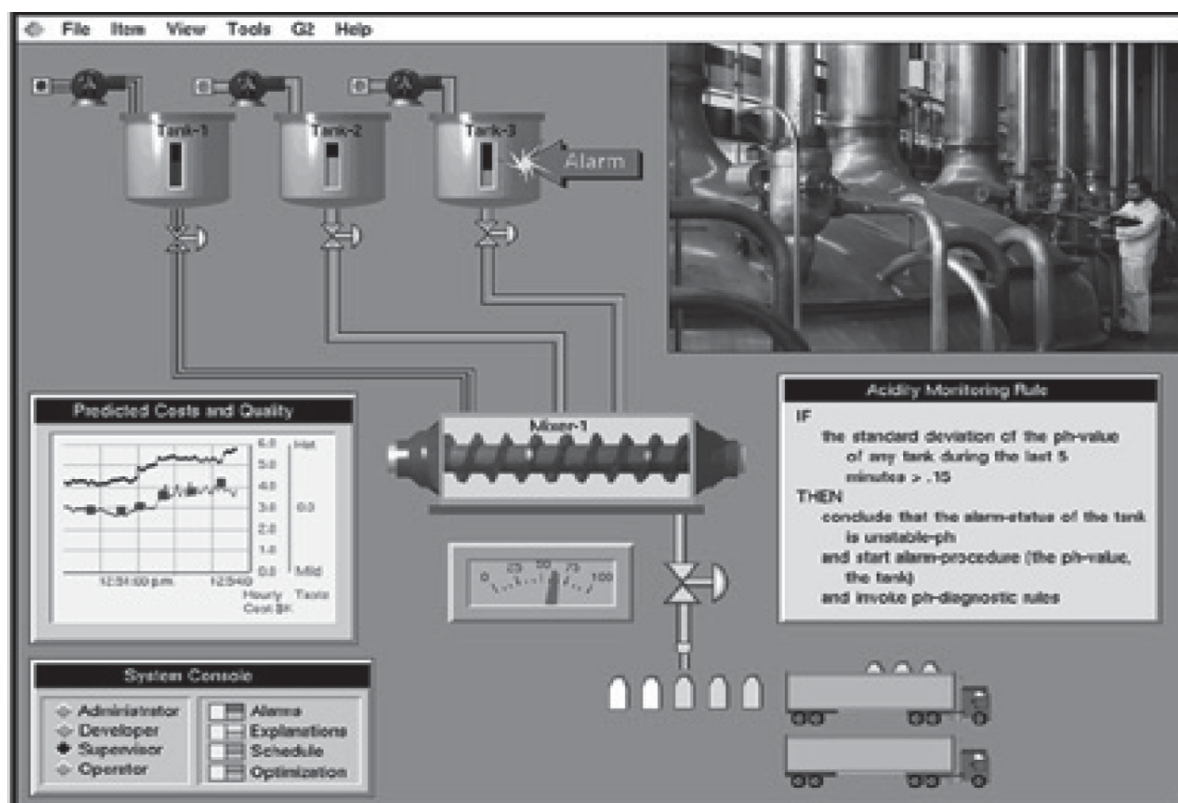


Рис. 5. Экспертная система, построенная в оболочке G2

G2 – мощная среда для разработки и развертывания интеллектуальных динамических систем управления. Прикладные программы, написанные в среде G2, могут существенно повысить эффективность выполнения операций, благодаря следующим факторам:

- непрерывному контролю над потенциальными проблемами прежде, чем они проявят неблагоприятное воздействие;
- принятие комплексных оперативных решений на основе информации, полученной посредством рассуждений и анализа данных, содержащихся в интеллектуальной модели процесса;
- диагностирование основных случаев возникновения проблем, критичных ко времени выполнения и выработки последовательности правильных действий;
- поддержание оптимальных рабочих условий;
- координирование действий и информации в выполнении сложных оперативных процессах.

G2 увеличивает Вашу производительность так, что Вы получаете готовые прикладные программы быстрее и они обладают более мощными функциональными возможностями, чем при использовании других программных средств. Результатом является более быстрое получение интеллектуальных прикладных программ. Это происходит из-за:

- более быстрого комплексного использования новой системы;
- сокращения времени разработки;
- уменьшения проектного риска;
- понижения расходов на сопровождение программы;

- более простого интегрирования с другим программным обеспечением и системами
- быстрого создания прототипа системы, чтобы продолжить проектирование совместно с конечным пользователем

Объекты, используемые в прикладных программах, написанных в среде G2, описывают свойства и действия динамических операций. Эти объекты легко определяются, а затем многократно используются. Синтаксические ошибки в описаниях объектов, правил и процедур исправляются мгновенно. Результат – быстрая разработка прикладных систем реального времени.

Объекты в G2 – это мощный и интуитивный способ представления материальных и абстрактных сущностей в прикладных программах. Мощностью объектно-ориентированного подхода к разработке программ, позволяет быстро и легко:

- встроить модули и объекты из других прикладных программ;
- графически определить объекты, их свойства и действия;
- создать новые образцы объектов, имитируя существующие объекты.

Объекты или класс объектов определенные один раз могут использоваться многократно. Любой объект или группа объектов могут иметь несколько экземпляров. Каждый экземпляр наследует все свойства и поведение первоначального объекта(ов). Объекты, правила, и процедуры можно группировать в библиотеки, которые будут общими для всех прикладных программ. Для моделирования широкого разнообразия действий типа производственных процессов, сетевых топологий, информационных маршрутизаций, или логических потоков объекты можно объединять графически на экране компьютера.

В G2 можно эффективно создавать и применять общие знания, создавая универсальные правила, процедуры, формулы и зависимости, которые являются применимыми для полных классов объектов. В результате сокращается время на разработку и увеличивается эффективность приложений.

В G2 для представления знаний используется структурный естественный язык, что позволяет облегчить чтение, редактирование и поддержку баз знаний. Это облегчает использование и редактирование приложений пользователем непрограммистом. Для создания и редактирования баз знаний используется Редактор Баз Знаний.

Для представления знаний эксперта о проблемной области используются правила. Правила могут быть как общими, то есть относящимся ко всему классу, так и специфическим, относящимся к конкретным экземплярам класса. Правила возбуждаются автоматически в следующих случаях:

- появились новые данные (вывод от фактов к цели);
- требуется найти данные (вывод от цели к фактам) для автоматического вызова других правил, процедур, или формул;
- требуется определить значения переменных;
- каждые n секунд для оценивания правила в указанном интервале времени.

Подобно правилам, процедуры в G2 выполняются в реальном времени. Процедуры, правила, и модели выполняются одновременно согласно их приоритетам. Процедуры могут использоваться для эффективного представления поведения объектов. Для процедур можно определить состояния ожидания и выполнять обработку информации параллельно. В результате это позволяет формировать мощные прикладные системы реального времени проще и быстрее, чем с помощью традиционных инструментальных средств программирования.

Работа в реальном времени, операционные решения и реакции зачастую должны быть выполнены мгновенно. Среда G2 предназначена для эффективной работы в реальном времени. Прикладные программы в G2 могут одновременно выполнять рассуждения относительно многократно выполняемых действий в реальном масштабе времени, перерабатывая тысячи правил, выполняя процедуры и модели, согласно их приоритетов. Для хранения хронологий данных и событий и для рассуждения относительно поведения через какое-то время используются переносимые типа время. G2 может сохранять мгновенные снимки состояния прикладной программы во время ее выполнения. Поэтому, в случае сбоя питания, G2 может выполнить "перезапуск из памяти" самого последнего сохраненного состояния.

G2 графика может моделировать знание, представляя объекты, связи и зависимости между объектами. G2 может рассуждать в терминах связи, следуя сети связанных объектов для определения причин и результатов. Графическая связность объектов G2 позволяет расширить прикладную программу используя графическое объединение аналогов. Графика включает встроенные диаграммы(графики), таблицы и рисунки и т.д.

G2 также работает с утилитами графического интерфейса Windows. Эти утилиты используют все преимущества объектно-ориентированных возможностей G2.

G2 позволяют динамически моделировать системы и процессы, используя объекты, правила, процедуры и формулы. Во время разработки модели используются вместо объектов реального мира, что позволяет непрерывно проверять прикладные программы в течение их разработки. Модели могут использоваться на этапе эксплуатации как часть прикладной программы G2 для сравнения фактических и модельных знаний.

Модели можно также использовать для проведения анализа и ответа на вопросы "что если", определения, например, лучших рабочих условий или лучших проектов. Модели можно также использовать для предсказания важных параметров действий в реальном времени, например, качество изделия или затрат.

Поскольку G2 имеет способность объединять опытное знание с аналитическими методами, большие индустриальные компании используют прикладные программы, основанные на G2 для минимизации потребления энергии, максимизации производительности, оптимизации ресурсов и управления процессами в окружающей среде. Способность G2 обрабатывать события, критичные ко времени выполнения и развитый интерфейс, помогают операторам избежать аварийных ситуаций, которые могут приводить к ухудшению качества изделия (программы), задержкам графика, по

- введению оборудования, и другим существенным затратам.

G2 разработан для эффективного управления сложными, динамическими проблемами.

Технологии G2 позволяют:

- эффективно выполнить прикладные программы критичные ко времени выполнения;
- использовать архитектуру клиент/ сервер;
- использовать эффективный интерфейс с базами данных в реальном масштабе времени;
- улучшить принятие решений, эффективно применяя знания экспертов совместно с рассуждениями, основанными на правилах моделей;
- облегчить редактирование прикладных программ, благодаря использованию структурированного естественного языка и возможностей удаленного доступа;
- быстро поставить сетевые прикладные программы;
- многократно использовать знания, хранящиеся в библиотеках.

Шлюз G2 позволяет прикладным программам, чтобы связаться с помощью интерфейса с широким разнообразием систем передачи и обработки данных в реальном масштабе времени и с базами данных. Шлюз может осуществлять буферизацию данных, обработку протокола, подтверждение связи, восстановление данных и многие другие функции. Межсетевой проект позволяет одновременно получать сообщения от многих источников данных. Шлюз действует как сервер данных для G2. Шлюз может выполняться на той же платформе, что и G2.

Все это позволяет с помощью данной оболочки создавать практически любые большие приложения значительно быстрее, чем с использованием традиционных методов программирования, и снизить трудозатраты на сопровождение готовых приложений и их перенос на другие платформы.

Вопросы для самоконтроля и текущего контроля

- 1) В чём заключается технология создания экспертных систем?
- 2) Какие этапы создания экспертных систем выделяются?
- 3) Что изменяется при переходе от прототипа экспертной системы к окончательной версии?
- 4) Кто из команды разработчиков отвечает за наполнение базы знаний?
- 5) Можно ли использовать оболочки экспертных систем для создания экспертных систем в разных предметных областях?
- 6) В чём заключается назначение оболочки экспертных систем?
- 7) Какие преимущества оболочек экспертных систем при разработке ЭС?

Задание на самостоятельную работу

- 1) Практическое задание

Выполнить проектирование структуры экспертной системы решающей задачу предметной области, определенной номером варианта практической работы №2.

- 2) Самостоятельная работа

Исследовать примеры современных экспертных систем.

Список рекомендуемой литературы¹:

[1] Сидоркина, Ирина Геннадьевна. Системы искусственного интеллекта [Текст] : [учеб. пособие для вузов по направлению 230100 "Информатика и вычисл. техника"] / И. Г. Сидоркина. М.: Кнорус, 2011. - 245 с.

Рассмотрены теоретические аспекты проектирования систем искусственного интеллекта и даны модели представления знаний, изложены материалы информационного, справочного, консультирующего характера по использованию инструментальных средств и технологическим особенностям разработки систем данного класса.

Подробнее: <https://www.labyrinth.ru/books/276217/>

[2] Богомолова, М. А. Экспертные системы (техника и технология проектирования) [Текст] : Методические указания к лабораторным работам / М. А. Богомолова. Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015. - 47 с.

Методические указания являются руководством при выполнении лабораторных занятий, проводимых по курсу «Экспертные системы (техника и технология проектирования)» с магистрами направления 09.04.03 - «Прикладная информатика» в терминальном классе. Лабораторные работы связаны с изучением принципов построения экспертных систем на основе различных моделей представления знаний, и могут быть полезны студентам других специальностей.

Подробнее: <https://www.iprbookshop.ru/71908.html>

[3] Советов, Борис Яковлевич. Представление знаний в информационных системах [Текст] : [учеб. для вузов по направлению подгот. "Информ. системы и технологии"] / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. М.: Академия, 2011.

Учебник создан в соответствии с Федеральным государственным образовательным стандартом по направлению бакалавриата «Информационные системы и технологии». Рассмотрены современное состояние теоретических и прикладных вопросов представления знаний в информационных системах, идеология построения интеллектуальных систем, математический аппарат представления знаний, возможности и пути использования искусственного интеллекта при проектировании информационных систем. Изложены новые аспекты представления знаний на основе искусственных нейронных сетей, расчетно-логических систем, генетических алгоритмов, мультиагентных систем. Приведены примеры практической реализации представления знаний на базе декларативного языка Пролог. Для студентов учреждений высшего профессионального образования. Может быть полезен разработчикам и пользователям информационных систем; преподавателям и научным сотрудникам, сферой интересов которых является интеллектуализация различных предметных областей; менеджерам и руководителям различного ранга, желающим самостоятельно ознакомиться с современным состоянием информационных технологий.

Подробнее: <https://lavkababuin.com/predstavlenie-znaniy-v-informacionnyh-sistemah-2-e-izd-ster-650564/>

[4] Остроух, А. В. Интеллектуальные информационные системы и технологии [Электронный ресурс] : монография / А. В. Остроух, А. Б. Николаев. Санкт-Петербург: Лань, 2019. - 308 с.

В монографии изложены концептуальные основы и методы представления знаний в интеллектуальных системах. Рассмотрены различные подходы, применяемые при проектировании и разработке интеллектуальных систем и технологий в транспортном комплексе, а также рассмотрены тенденции развития систем искусственного интеллекта.

Монография может быть использована для формирования профессиональной компетентности студентов высших учебных заведений, аспирантов и научных сотрудников обучающихся и ведущих научные исследования в области разработки и практического применения систем искусственного интеллекта по укрупнённой группе направлений подготовки "Информатика и вычислительная техника".

Подробнее: <https://www.labyrinth.ru/books/692861/>

Ссылки на открытые ресурсы (он-лайн-курсы, видеоуроки и т.д.)

[5]

ⁱСписок рекомендуемой литературы может быть более широким по сравнению со списком литературы, указанным в рабочей программе

Прикладные интеллектуальные системы

Конспект лекции 7. Применение оболочки для создания экспертных систем CLIPS 6.31

Аннотация: В лекции рассказывается о применении оболочки для создания экспертных систем CLIPS 6.31.

План лекции:

1. Краткая история и характеристика CLIPS
2. Основные элементы программирования
3. Факты
4. Правила
5. Дополнительные средства CLIPS
6. Задания для самостоятельной работы.

Ключевые слова: экспертная система, CLIPS 6.31, правила, факты, стратегии, функции.

СОДЕРЖАНИЕ ЛЕКЦИИ:

1. Краткая история и характеристика CLIPS

Название языка CLIPS - аббревиатура от C Language Integrated Production System. Язык был разработан в Центре космических исследований NASA (национальное Аэрокосмическое Агентство США)

CLIPS является одним из распространенных инструментальных средств разработки экспертных систем (ЭС). Представляя собой логически полную среду, содержащую встроенный редактор и средства отладки, CLIPS является оболочкой ЭС. Первая версия системы вышла в 1984 году и представляла собой интерпретатор порождающих правил. Процедурный язык и объектно-ориентированное расширение CLIPS были включены в этот программный продукт только в 1990-х годах. Существующая в настоящее время версия может эксплуатироваться на платформах UNIX, DOS, Windows и Macintosh. Она является хорошо документированным общедоступным продуктом (адрес в Интернет - <http://www.cosmic.uga.edu/>) и доступна по сети FTP.

Принципиальным отличием данной системы от аналогов является то, что она полностью реализована на языке C. Причем исходные тексты ее программ опубликованы в сети Интернет.

В CLIPS используется оригинальный LISP-подобный язык программирования, ориентированный на разработку ЭС. Кроме того, CLIPS поддерживает еще две парадигмы программирования: объектно-ориентированную и процедурную. Аспекты объектно-ориентированного программирования в CLIPS нами рассматриваться не будут.

Широкое распространение системы CLIPS объясняется тем, что CLIPS представляет собой комбинацию продукционного и объектно-ориентированного и фреймового подходов и обладает следующими ее преимуществами:

- относительная дешевизна;
- использование множества опробованных на практике конструкций из других инструментальных средств;
- наличие четко сформулированного синтаксиса, позаимствованного у LISP;
- высокая производительность;
- возможность использования внешних функций, написанных на других языках программирования;
- включение средств, позволяющих комбинировать правила и объекты.

CLIPS использует продукционную модель представления знаний и поэтому содержит три основных элемента:

1. базу фактов;
2. базу правил;
3. блок вывода.

База фактов представляет исходное состояние проблемы. Блок вывода CLIPS сопоставляет 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 называют «циклом распознавание – действие».

2. Основные элементы программирования

CLIPS предоставляет три основных элемента для написания программ:

1. Простые типы данных;
2. Функции для манипулирования данными;
3. Конструкторы для пополнения БЗ, для создания фактов, функций, правил, классов.

2.1 Простые типы данных

Для представления информации в CLIPS предусмотрено восемь простых типов данных.

Простые типы данных:

Integer	external address	-	внешние адреса
Float	fact – address	-	адреса факты
Symbol	instance –name	-	имена объектов
String	instance –address	-	адреса объектов

Для представления числовой информации используются типы float и integer, символьной - symbol, string . Остановимся на рассмотрении этих четырех типов данных.

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 (тип integer представляется внутри CLIPS как тип языка C long integer). Все остальные числа сохраняются как действительные (float - C double float).

Количество значащих цифр зависит от аппаратной реализации. В этой же связи могут возникать ошибки округления.

Как в любом языке программирования, особенную осторожность необходимо проявлять при сравнении чисел с плавающей точкой, а также при сравнении с ними целых чисел.

Примеры целых чисел: **237 15 +12 -32**

Примеры чисел с плавающей точкой: **237e3 15.09 +12.0 -32.3e-7**

Последовательность символов, которая не удовлетворяет числовым типам, обрабатывается как тип данных symbol.

Тип данных symbol в CLIPS - последовательность символов, состоящая из одного или нескольких любых печатных символов кода ASCII. Как только в последовательности символов встречается символ-разделитель, данное типа symbol заканчивается.

Следующие символы служат разделителями: любой непечатный ASCII символ (включая пробел, символ табуляции, CR, LF), двойные кавычки, (,), &, |, <, ~, ;

Символы-разделители не могут включаться в данные типа symbol за исключением символа "<", который может быть первым символом в данных типа symbol. Кроме того, данные типа symbol не могут начинаться с символа "?" или последовательности

символов "\$?", поскольку эти символы зарезервированы для переменных. Заметим, что *CLIPS* различает регистр символов. Ниже приведены примеры выражений символьного типа:

```
foo Hello B76-H1 bao_value 127A 742-42-42 @+==-%
```

Тип данных *string* - это последовательность символов, состоящая из нуля и более печатных символов и заключенная в двойные кавычки. Если внутри строки встречаются двойные кавычки, то перед ними необходимо поместить символ (\). То же справедливо и для самого (\).

Несколько примеров: "foo" "a and. b" "1 number" "a\"quote"

Отметим, что строка "abcd" не тоже самое, что abcd. Они содержат одинаковые наборы символов, но являются экземплярами различного типа.

Не допускается использование букв русского алфавита.

2.2 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:

1. Пользовательские.
2. Системные.
3. Внешние.

Пользовательские функции определены программистом с помощью конструктора **Deffunction**.

Системные функции – те функции, которые были определены изначально внутри среды *Clips*.

Внешние функции – функции, которые были определены вне *Clips* и написаны на других языках.

Хотя *CLIPS* и не ориентирована на численные вычисления, в ней предусмотрен ряд стандартных арифметических и математических функций. Среди них:

- + - сложение, - - вычитание;
- * - умножение, / - деление;
- ** - возведение в степень;
- **div** – целочисленное деление, **mod** - остаток от целочисленного деления;
- **abs** определение абсолютного значения;
- **Sqrt** вычисление квадратного корня;
- **min** - нахождение минимума, **max** - нахождение максимума;
- **cos, sin, tan** –тригонометрические функции синус, косинус, тангенс;
- **acos, asin, atan** - арксинус, арккосинус, арктангенс;
- **pi** – получение числа π ;
- **exp** – вычисление экспоненты;
- **log, log10** – вычисление натурального и десятичного логарифмов;
- **deg-rad, rad-deg** – преобразование из градусов в радианы и наоборот;
- **round** – округление числа.

Конструкция *deffunction* позволяет пользователю определять новые функции непосредственно в среде *CLIPS* с использованием синтаксиса *CLIPS*.

Вызовы функций в CLIPS имеют префиксную форму: аргументы функции могут стоять только после ее названия.

Синтаксис вызова функции:

(имя аргумент1 аргумент2 аргумент3).

Аргументами могут быть данные простых типов, переменные или вызовы других функций. Аргументы отделяются друг от друга одним или несколькими пробелами.

Ниже приведены примеры вызовов функций:

Запись в среде CLIPS

(+3 4 5)

(* 5 6.0 2)

(+3 (* 8 9) 4)

(* 8 (+3 (* 2 3 4) 9) (* 3 4))

(Sqrt (+ 10 6))

(log10 100)

Обычная запись

(3+4+5)

(5*6.0*2)

(3+8*9+4)

(8* (3+(2*3*4)+9)*(3*4))

Sqrt(10+6)

lg100

2.3 Конструкторы

В CLIPS существует несколько описывающих конструкторов: **defmodule**, **defrule**, **deffacts**, **deftemplate**, **defglobal**, **deffunction**, **defclass**, **definstances**, **defmessage-handler**, **defgeneric**.

При записи все они заключаются в скобки. Определение конструктора отличается от вызова функции главным образом по производимому эффекту. Обычно вызов функции оставляет состояние среды CLIPS без изменений (за исключением, когда речь идет о функциях сброса, очистки, открытия файла и т.п.). Определение конструктора, напротив, в точности направлено на изменение состояния среды путем внесения изменений в базу знаний CLIPS. В отличие от функций конструкторы никогда не возвращают значений.

Все конструкторы (за исключением **defglobal**) позволяют размещать комментарии сразу вслед за именем конструкции. Комментарии внутри конструктора заключены в двойные кавычки. 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.

3. Факты

3.1 Представление фактов

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* (рабочая память). 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).

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. Команда *reset* добавляет в список фактов *initial-fact*. 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.

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:

1. Позиционный.
2. Непозиционный (шаблонный – фреймовый).

Позиционные факты в общем виде представляются:

<позиционный_факт> ::= (данные_типа_symbol [поле]).*

Позиционные факты состоят из выражения типа *symbol*, за которым следует последовательность (возможно пустая) из полей, разделенных пробелом.

Вся запись заключается в скобки.

Первое поле (*Symbol*) определяет отношение или связь, которая применяется к оставшимся полям.

Примеры:

(altitude is 1000 feet)

(grocery – list bread milk eggs)

(schoolboys is Bob Mike)

Следующие символьные выражения *зарезервированы* и не могут быть использованы как первое поле позиционного факта: *test*, *and*, *or*, *not*, *declare*, *logical*, *object*, *exists*, *forall*.

Для обращения к информации в позиционном факте пользователь должен знать, не только какие данные в нем, но и в каком поле они хранятся.

Непозиционные факты (шаблонные) факты дают возможность пользователю абстрагироваться от структуры факта, задавая имена каждому из полей факта, называемых *слотами*. 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 C и Паскаль. Как и факт, слот с обеих сторон ограничивается круглыми скобками.

В общем виде конструктор *Deftemplate* имеет вид:

(Def template <name> [“комментарий”]

<определение слота>)*

< определение слота> ::= (slot <имя_слота> [(<тип_поля>) (<значение по умолчанию>)])

<тип поля> ::= (type <стандартный тип >)

<значение по умолчанию> ::= (default <значение>)

Все имена шаблонов и слотов имеют тип *symbol*.

Пример:

```
(deftemplate student "student record"
  (slot name (type string))
  (slot age (type integer) (default 18)))
```

Приведем примеры фактов, созданных по этому шаблону:

```
(student (name "Fred")
  (student (name "Din") (age 19))
```

В первом факте возраст будет задан по умолчанию, равным 18.

Как и все конструкторы CLIPS, конструктор *Deftemplate* не возвращает никакого значения. Для просмотра всех определенных в текущей базе знаний шаблонов можно воспользоваться командой *get-deftemplate-list* или специальным инструментом *Deftemplate Manager* (Менеджер шаблонов). Для запуска менеджера шаблонов воспользуйтесь меню *Browse* и выберите пункт *Deftemplate Manager*.

Менеджер шаблонов позволяет в отдельном окне просматривать список всех шаблонов, доступных в текущей базе знаний; выводить в главное окно среды определение выбранного шаблона с помощью команды *Pprint*; удалять выбранный шаблон с помощью команды *Remove*.

Система CLIPS различает шаблонные от позиционных фактов по первому полю факта. Первое поле любого факта является значением типа *symbol*. Если это значение соответствует имени некоторого шаблона, то факт – шаблонный (непозиционный). Порядок слотов в непозиционных слотах значения не имеет.

Примеры:

```
(client (name "Brown") (id x9334A))
(point –mass (x-velocity 100) (y-velocity-200))
(class (teacher "Martina Jones") (#-students 30) (room "37"))
```

Непозиционные факты удобны для описания стереотипных ситуаций.

3.2 Манипуляция над фактами

Факты могут задаваться с помощью конструктора *deffacts*, который позволяет определить список из любого количества любых фактов и присвоить ему имя. Список фактов будет автоматически добавляться всякий раз после выполнения команды *reset*, очищающий текущий список 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*.

В общем виде конструктор *deffacts* имеет вид:

```
(def facts < имя-набора фактов> ["комментарий"]
  (факт*))
```

Имя набора фактов должно быть значением типа *symbol*.

Можно определить с помощью конструктора *deffacts* множество исходных или априорных фактов. Факты могут быть *позиционные* и *непозиционные*.

Примеры конструктора *deffacts*:

1.

```
(deffacts walk "Some facts about walking"
  (status walking)
  (sun is shine)
  (walk-sing walk))
```
2.

```
(deffacts students
```

```
(student (name "Fred"))
(student (name "Din") (age 19))
(class is 14B ))
```

3. (deffacts startup "Refrigerator status"
 (refrigerator light on)
 (refrigerator door open)
 (refrigerator temp (+5 10 15)))

В первом примере walk - имя набора фактов, далее следует в кавычках необязательный комментарий, затем в скобках – отдельные факты.

Набор фактов, определенных конструкцией deffacts можно удалить командой undeffects: (undeffects <имя-набора фактов>)

Например, команда (undeffects walk) удалит весь набор фактов из примера 1.

Проверить работу конструктора deffacts можно воспользовавшись диалогом Watch Options, выбрав пункт Watch меню Execution. В диалогом окне Watch Options включите режим просмотра изменения списка фактов, поставив галочку в поле Facts. CLIPS предоставляет также визуальный инструмент для манипуляции с определенными в данный момент конструкторами deffacts – *Deffacts Manager*. Для запуска *Deffacts Manager* воспользуйтесь меню Browse и выберите пункт *Deffacts Manager*.

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. Добавить факты в список фактов текущей базы фактов можно также с помощью команды assert. Приведем общий вид команды:

```
(assert <факт>*)
```

Пример: (assert (cat is gray) (value(+7 8)))

В случае удачного добавления фактов в базу данных, функция возвращает индекс последнего добавленного факта. Если во время добавления произошла ошибка, то команда прекращает свою работу и возвращает значение false. Слотам непозиционных фактов, значения которых не заданы, будут присвоены значения по умолчанию.

Удаление факта выполняется командой retract:

```
(retract <определение-факта>* | *)
```

<определение-факта>::=<переменная, связанная с адресом с помощью правила>|<индекс факта без префикса f> .

*-удаляются все факты из списка фактов.

Примеры: (retract 2) – удаляется факт с индексом 2.

(retract *) – удаляются все факты из базы фактов.

(retract 0 (+ 2 2)) - удаляются факты с индексами 0 и 4.

Непозиционные факты можно изменять с помощью команды modify:

```
(modify <определение-факта><новое-значение-слота>*)
```

Аргументом <определение-факта> может быть либо переменная, связанная с адресом с помощью правила, либо индекс факта без префикса f, например, 3 для факта с индексом f-3. После определения факта следует список из одного или более новых значений слотов указанного шаблона.

Пример изменения существующего непозиционного факта:


```
(deftemplate temp (slot value))
(assert (temp (value low)))
(modify 0 (value high))
```

Функция *duplicate* создает новый непозиционный факт заданного шаблона и копирует в него определенную пользователем группу полей уже существующего факта того же шаблона. По действиям, которые выполняет функция *duplicate*, она аналогична *modify*, за исключением того, что *duplicate* не удаляет старый факт из списка фактов. Синтаксис команды *duplicate*:

```
(duplicate <определение-факта><новое-значение-слота>*)
```

Значение аргументов здесь те же самые, что и для команды *modify*.

Пример создания копии существующего непозиционного факта:

```
(deftemplate car
  (slot name)
  (slot producer)
  (slot type)
  (slot max-speed))
(assert (car
  (name scorio)
  (producer ford)
  (type sedan)
  (max-speed 180)))
(duplicate 0
  (type off-road)
  (max-speed 130))
```

В этом примере определяется шаблон, описывающий свойства автомобиля, и добавляется факт об автомобиле *Ford Scorio* с типом кузова – седан и максимальной скоростью -180 км/час. После этого с помощью функции *duplicate* добавляется факт с информацией об еще одном автомобиле с похожими характеристиками – это внедорожник *Ford Scorio* с максимальной скоростью -130 км/час.

Наполнение списка фактов в CLIPS довольно кропотливое и длительное занятие, поэтому система предоставляет команды сохранения и загрузки списка фактов в файл – *save-facts* и *load-facts* соответственно.

Синтаксис команды *save-facts*:

```
(save-facts <имя-файла>[ <границы-видимости> <список шаблонов>])
<границы-видимости> ::= visible / local
```

Команда *save-facts* сохраняет факты из текущего списка фактов в текстовый файл. На каждый файл отводится одна строка. Непозиционные факты сохраняются вместе с именами слотов. По умолчанию аргумент *<границы-видимости>* принимает значение *local*, т.е. сохраняются факты только из текущего модуля.

Пример использования функции *save-facts*:

```
(clear)
(deftemplate t
  (slot a)
  (slot b))
(assert (t (a 1) (b 2)))
```

(*save-facts f1 t*)

Синтаксис команды *load-facts*:

(*load-facts <имя-файла>*)

4. Правила

4.1 Общая структура 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 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*.

Левая часть правила представляет собой ряд *условий* (*условных элементов-УЕ*), 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 применимыми. Если все условия правила выполняются, то оно *активируется* и помещается в *список активированных правил*.

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*. Действия, описанные в выбранном правиле, выполняются (при этом список применимых правил может измениться), а затем блок вывода выбирает другое правило и т.д. Этот процесс продолжается до тех пор, пока не остается ни одного применимого правила, т.е. пока список активированных правил не окажется пуст.

Во многом правила похожи на операторы типа If-then процедурных языков программирования. Однако условие If-then оператора в процедурном языке проверяется только тогда, когда программа передает ему управление. С правилами ситуация иная. Блок вывода постоянно отслеживает все правила, условия которых выполняются, и, таким образом, правило может быть выполнено в любой момент, как только оно становится применимым. В этом смысле правила подобны обработчикам исключительных ситуаций в процедурных языках.

Для определения правил используется конструктор *defrule*:

(*defrule <имя-правила> [" комментарий"]*

[*<определение-свойства-правила>*]

[*<предпосылки>*] ; левая часть правила

=>

<следствие> ; правая часть правила

Имя правила должно быть значением типа `symbol`, в качестве имени нельзя использовать зарезервированные слова.

<определение-свойства-правила>::=(declare <свойство-правила>)

<свойство-правила>::=(*Salience*<целочисленное_выражение>)/(*auto-focus* *true* /*false*)

Целочисленное выражение свойства *Salience* задает приоритет, его значение находится в интервале -10000...10000, свойство влияет на стратегию разрешения конфликта. По умолчанию принимается значение 0. Свойство *auto-focus* позволяет автоматически выполняться команде *focus*, передающей управление модулю, из которого извлечено данное правило, при каждой активации правила. Если свойство *auto-focus* установлено в значение *true*, то команда *focus* в модуле, в котором определено данное правило, автоматически выполняется при каждой активации правила, в противном случае никаких действий не производится. По умолчанию принимается значение *false*.

Пример правила:

(*defrule vacation*

(*work done*) ; 1 посылка

(*money plenty*) ; 2 посылка

(*reservations made*) ; 3 посылка

=>

(*printout t "Let's go!" crlf*) ; заключение или действие – вывод сообщения.

CLIPS предоставляет также визуальный инструмент для манипуляции с определенными в данный момент конструкторами *defrule* – *Defrule Manager*. Для запуска *Defrule Manager* воспользуйтесь меню *Browse* и выберите пункт *Defrule Manager*.

Пример правила системы MYSIN, записанного на языке CLIPS:

(*defrule diagnosis*

(*patient (name Jones) (organism organism-1)*) ;если у пациента Джона обнаружен организм-1

(*organism (name organism-1) (morphology rod)*) ; и организм-1 имеет форму ;

; палочки и

(*aerobicity aerobic*) ;является аэробным (воздушная среда

; способствует его росту)

=>

(*assert (organism (name organism-1)* ; то добавить факт, что с уверенностью ;0.8
этот

(*identify enterobacteriaceae*) ; микроорганизм относится к классу

(*confidence 0.8*))) ; enterobacteriaceae

4.2 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:

?<имя-переменной>

Примеры переменных:

?x ?sensor ?noun ?color

Имя переменной имеет тип `symbol`. Перед использованием переменной ей необходимо присвоить значение. Все переменные могут быть *локальными* или *глобальными*. Все переменные, кроме глобальных, считаются локальными и могут использоваться только в рамках описания конструктора. К этим локальным переменным можно обращаться внутри описания, но они не определены вне него.

Глобальные переменные определены внутри модуля и задаются с помощью конструктора:

```
(defglobal [<имя-модуля>] <определение-переменной>* )
<определение-переменной>::= <имя-переменной>=<выражение>
<имя_переменной>::=? Значение-типа-symbol
```

Пример:

```
(defglobal ?y=5 ?color=red)
```

К глобальной переменной можно обратиться в любом месте, и ее значение остается независимым от других конструкций. Глобальные переменные CLIPS подобны глобальным переменным в процедурных языках программирования, но они значительно слабее типизированы (на них не налагается ограничения хранения данных только одного типа).

Чаще всего локальные переменные описываются и получают значения в левой части правила. Например:

```
(defrule make-quack
  (quack-sound ?sound)
=>
```

```
(assert (sound is ?sound) ))
```

Получив значение, переменная сохраняет его неизменным при использовании как в левой, так и в правой части правила, если только это значение не изменяется в правой части при помощи функции *bind*.

Пример использования функции *bind*:

```
(defrule addition
  (numbers ?x ?y)
=>
(assert (answer (+ ?x ?y) ) )
(bind ?answer (+ ?x ?y) )
(printout t "answer is " ?answer crlf) )
```

Кроме значения самого факта, переменной может быть присвоено значение адреса факта. Это может оказаться удобным при необходимости манипулировать фактами непосредственно из правила. Для такого присвоения используется комбинация "<-". Следующий пример иллюстрирует присвоение переменной значения адреса факта и ее последующее использование:

```
(defrule get-married
?duck <- (bachelor Dopey)
=> (retract ?duck) )
```

Пример правила системы MYSIN, записанного на языке CLIPS с использованием переменных:

```
(defrule diagnosis ;если у пациента ?pat обнаружен
организм
(patient (name ?pat) (organism ?org)) ; ?org и этот организм имеет форму
палочки и
(organism (name ?org) (morphology rod) ;является аэробным (воздушная среда
(aerobicity aerobic)) ; способствует его росту)
```

=>

```
(assert (organism (name ?org) ; то добавить факт, что с уверенностью 0.8 этот
(identify enterobacteriaceae) ; микроорганизм ?org относится к классу
(confidence 0.8))) ; enterobacteriaceae
```

4.3 Условные элементы в левой части правила

Левая часть правила задается набором УЕ, который обычно состоит из условий, примененных к некоторым образцам. Заданный набор образцов используется системой для сопоставления с имеющимися фактами и объектами. Все условия в левой части правила объединяются с помощью неявного логического оператора *and*. 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 инициализировать работу программы. Поэтому перед запуском таких программ необходимо произвести сброс состояния среды CLIPS командой (*reset*). Синтаксис условных элементов:

```
<условный-элемент>::=<образец-УЕ> / <not-УЕ> / <and-УЕ> / <or-УЕ> /
<logical-УЕ> / <test- УЕ> / <exists- УЕ> / <forall- УЕ>
```

Образец-УЕ как бы определяет маску, которой должны соответствовать данные, и состоит из списка ограничений полей, групповых символов (?, \$) и переменных. Условный элемент *test* имеет следующий синтаксис:

```
<test- УЕ>::=(test <вызов функции>).
```

Функция проверки условия *test* представляет собой мощное средство, при помощи которого можно сравнивать числа, переменные и строки в левой части правила. Она записывается точно так же, как и образцы. Правило может выполняться только тогда, когда наряду с совпадением всех образцов, записанных в левой части правила, справедливо и условие, описываемое в *test*.

Функция *test* имеет следующий синтаксис:

```
(test (аргумент-сравнения аргумент-1 аргумент-2)) ,
```

где аргумент-сравнения - это тот параметр, по которому сравниваются два следующих аргумента. В CLIPS существует ряд предопределенных аргументов сравнения:

Аргумент сравнения	Значение
eq	равно (сравнивает тип и значение)
neq	неравно
=	равно (сравнивает только значение)

Аргумент сравнения	Значение
<>	неравно
>	больше
>=	больше или равно
<	меньше
<=	меньше или равно

Все аргументы, кроме *eq* и *neq*, используются только для сравнения чисел. При интерпретации выражения сравнения считается, что аргумент-1 стоит слева от аргумента-сравнения, а аргумент-2 - справа.

Пример применения функции *test*:

```
(defrule example
  (data ?x)
  (data ?y)
  (test (>= (abs (- ?y ?x)) 3))
=> )
```

Условный элемент *not* имеет следующий синтаксис:

<not-УЕ>::=(not <УЕ>)

Этот элемент проверяет отсутствие факта в списке фактов.

Условный элемент *or* имеет следующий синтаксис:

<or- УЕ>::=(or <УЕ>*)

Пример применения УЕ *or*:

```
(defrule system-fault
  (error-status unknown)
  (or (temp high )
      (valve broken)
      (pump off))
=>
```

```
(printout t "System has a fault" crlf))
```

УЕ *or* позволяет активировать правило любым из условных элементов, объединенных с помощью *or*.

Синтаксис всех остальных элементов подобный синтаксису УЕ *or*.

5. Дополнительные средства CLIPS

CLIPS предоставляет ряд дополнительных средств, необходимых при написании программ. Основными из них являются:

- ограничения на значения полей;
- использование функций в правилах;
- использование процедурных знаний.

Рассмотрим каждое из этих средств по очереди.

5.1 Ограничения на значения полей

Использование ограничений на значения полей позволяет ограничить значения, принимаемые образцами в левой части правила.

Рассмотрим три вида ограничений: $\sim$, $|$ и $\&$.

Ограничение первого типа действует на следующее прямо за ним значение и говорит о том, что поле не может принимать это значение. Например:

```
(defrule walk
  (light  $\sim$  green)
```

=>

```
(printout t "Don't walk" crlf) )
```

Ограничение второго типа указывает на то, что поле может принимать одно из следующих значений. Например:

```
(defrule cautious
  (light yellow | blinking-yellow)
```

=>

```
(printout t "Be cautious" crlf) )
```

Ограничение третьего типа используется только вместе с ограничениями первых двух типов и указывает на то, что должны удовлетворяться оба соединяемых при его помощи ограничения. Например:

```
(defrule cautious
  (light ?color & yellow | blinking-yellow)
```

=>

```
(printout t "Be cautious because light is" ?color crlf) )
```

5.2 Использование функций

Функции могут использоваться и в левой, и в правой части правила. Например:

```
(defrule addition
  (numbers ?x ?y)
```

=>

```
(assert (answer (+ ?x ?y) ) ) )
```

При использовании функции в левой части правила перед ней должен стоять знак "=" , показывающий CLIPS, что следующее выражение необходимо вычислить, а не использовать буквально. Например:

```
(defrule addition
  (numbers ?x ?y)
  (stock ?ID = (sqrt (+ (** ?x 2) (** ?Y 2))))
```

=>

```
(printout t "stock ID = " ?ID crlf) )
```

Конструктор *deffunction* позволяет пользователю создавать свои функции непосредственно в среде CLIPS. Способ вызова функции, определенной пользователем эквивалентен способу вызова внутренних функций CLIPS. Вызов функции осуществляется по имени, за которым следует список необходимых аргументов, отделенных одним или большим числом пробелов. Синтаксис конструктора *deffunction*:

```
(deffunction <имя-функции> ["комментарий"]
  <обязательные-параметры>
  [<групповой-параметр>] <действия >)
```

<обязательные-параметры>::=(<выражение-простое-поле>*)

<групповой-параметр>::=<выражение-составное-поле>

Групповой параметр указывает, что функция может иметь переменное число аргументов. Действия – это последовательность действий или выражений, которые будут выполнены (вычислены) по порядку в момент вызова функции. Имя функции должно быть уникальным. Конструктор *deffunction* должен быть объявлен до первого использования создаваемой им функции, исключение составляют рекурсивные функции.

Пример функции:

```
(deffunction hypotenuse (?a ?b)
  (sqrt (+ (* ?a ?a) (* ?b ?b))) )
```

Функция возвращает результат последнего выражения в списке. Иногда функция имеет побочные эффекты, как в приведенном ниже примере:

```
(deffunction init (?day)
  (reset) (assert (today is ?day) )
```

В результате после запуска функции на выполнение командой

```
CLIPS> (init Sunday)
```

будет выполнена команда *reset*, и следовательно очищена база фактов, а затем в нее включен новый факт (*today is Sunday*).

CLIPS поддерживает также процедурную парадигму представления знаний, подобную принятой в обычных языках программирования (С, Паскаль). Конструктор *deffunction* позволяет пользователю определять новые функции. Эти новые функции могут вызываться точно так же, как и встроенные функции CLIPS. Конструктор *defmodule* позволяет разбивать базу знаний на части.

Также CLIPS предоставляет 9 функций, которые реализуют возможности процедурного программирования:

Функция	Назначение
bind	Создание и связывание переменной
if	Оператор ветвления
while	Цикл с предусловием
loop-for-count	Итеративный цикл
progn	Объединение нескольких действий в рамках одной логической команды.
progn\$	Выполнение заданного набора действий над каждым элементом составного поля.
return	Прерывание функции, цикла, правила, обработчика сообщений.
break	Прерывание текущей работы циклов, функции <i>progn</i> и некоторых функций, выполняющих действия над набором объектов.
switch	Оператор множественного ветвления.

5.3 Команды управления интерактивной средой

Приведем в таблице наиболее часто применяемые команды управления интерактивной средой.

Команда	Описание
load	Загрузка конструкторов из текстового файла
load*	Загрузка конструкторов из текстового файла без отображения процесса загрузки в диалоговом окне CLIPS
save	Сохранение созданных конструкторов в текстовом файле.
clear	Очистка рабочей памяти системы. Удаляет все конструкторы и связанные с ним данные.
reset	Сброс рабочей памяти системы, добавляется initial-fact, который имеют индекс f-0
batch	Запуск командного файла, который содержит любые команды CLIPS
batch*	Запуск командного файла без отображения результатов
exit	Окончание работы, выход из CLIPS
system	Выполнение команд, операционной системы.
options	Вывод в диалоговое окно информации обо всех текущих установках системы

Все команды записываются в круглых скобках. Для команд *batch**, *batch*, *save*, *load**, *load* задается параметр – имя файла.

5.4 Команды работы с конструктором deffacts

ppdeffacts – вывод определения конструктора в диалоговое окно. Синтаксис команды: (*ppdeffacts* <имя конструктора>).

list-deffacts – вывод в диалоговое окно системы списка всех определенных конструкторов *deffacts*.

Undeffacts – удаление определенного конструктора, этих фактов в списке фактов не будет. Синтаксис команды: (*undeffacts* <имя конструктора>/ *). В случае использования символа * в качестве параметра команды будут удалены все определенные пользователем конструкторы *deffacts*.

5.5 Команды работы с конструктором deftemplate

ppdeftemplate вывод определения конструктора в диалоговое окно. Синтаксис команды: (*ppdeftemplate* <имя конструктора>).

list-deftemplate– вывод в диалоговое окно системы списка всех определенных конструкторов *deftemplate*

undeftemplate – удаление определенного конструктора, этих фактов в списке фактов не будет. Синтаксис команды: (*undeftemplate* <имя конструктора>/ *).

В случае использования символа * в качестве параметра команда попытается удалить все определенные пользователем конструкторы *deftemplate*. Если выбранный конструктор используется фактом или правилом, его удаление закончится неудачей.

5.6 Команды работы с правилами

ppdefrule- вывод определение конструктора *defrule* в диалоговое окно CLIPS. Синтаксис команды: (*ppdefrule* <имя конструктора>).

list-defrules – вывод в диалоговое окно системы списка всех определенных конструкторов *defrule*.

undefrule – удаление определенного конструктора *defrule*. Синтаксис команды: (*undefrule* <имя конструктора>/ *). В случае использования символа * команда попытается удалить все определенные пользователем правила.

matches- просмотр списка набора данных (фактов), способных активировать заданное правило Синтаксис команды: (*matches* <имя правила>)

refresh –помещение всех текущих активаций заданного правила в план решения задачи. Синтаксис команды: (*refresh* <имя правила>).

agenda – вывод текущего плана решения задачи. Команда отображает в диалоговом окне все активации, содержащиеся в плане решения задачи в соответствующем порядке вместе с данными, активировавшими правила.

run – запуск цикла выполнения правил. Синтаксис команды: (*run* [<целочисленное выражение >]). Если необязательный параметр <целочисленное выражение > является положительным, то выполнение правил прекращается после заданного числа запусков правил или в случае, если текущий план не содержит ни одной активации. Если данный параметр не указан или отрицателен, то текущий план выполняется полностью.

5.7 Функции ввода-вывода

Система ввода – вывода, используемая CLIPS, называется маршрутизацией ввода-вывода (I/O routers). Одна из особенностей системы - это использование логических имен, которые позволяют обращаться к устройствам ввода – вывода унифицированным способом без необходимости учета особенностей конкретного устройства.

Некоторые предопределенные логические имена CLIPS:

stdin (t) - логическое имя, определяющее устройство для ввода информации от пользователя по умолчанию. Его используют функции *read*, *readline*. Его синоним - *t*.

stdout (t) - логическое имя, которое определяет устройство для вывода информации. Его синоним - *t*. Используется командами *printout*, *format*

wclips – имя устройства, которая использует справочная система CLIPS.

werror – логическое имя устройства, которое будет выводить сообщение об ошибках.

wwarning – логическое имя устройства, на котором будут выдаваться предупреждения.

wtrace – логическое имя устройства, на котором отображается отладочная информация.

Функции ввода-вывода:

open - открытие файла. Синтаксис функции:

(*open* <имя файла><логическое имя>[<параметр открытия>])

Возможные варианты значений параметра, определяющих способ открытия файла:

“r”- доступ только для чтения;

“w”- доступ только для записи;

“r +” – доступ для чтения и записи;

“a” – добавление только в конец файла.

Пример функции: (*open* “myfile.clp” *writeFile* “w”)

close -закрытие файла. Параметром может быть логическое имя, но может быть без параметров, тогда CLIPS закрывает все открытые файлы.

printout - вывод информации на заданное устройство. Синтаксис функции:

(*printout* <логическое имя><выражение>*)

Функция *printout* принимает и выводит на устройство, связанное с заданным логическим именем, любое число параметров.

Каждое выражение вычисляется и выводится на соответствующее устройство без пропусков. Используемые символы: *crlf*- перевод строки.; *tab* – горизонтальная табуляция (8символов); *vtab* – вертикальная табуляция; *ff* – переход на новую страницу.

read- ввод данных с заданного устройства. Параметром может быть логическое имя, а может быть без него.

format – форматированный вывод информации на заданное устройство.

readline – ввод строки с заданного устройства.

rename – переименование файла. Синтаксис функции:

(*rename* <старое- имя><новое-имя>)

remove-удаление файла. Синтаксис функции:

(*remove* < имя-файла>).

Вопросы для самоконтроля и текущего контроля

- 1) Каков механизм организации логического вывода в ИО CLIPS?
- 2) Какие инструменты обеспечивают логический вывод в ИО CLIPS?
- 3) Как представлен интерфейс работы системы, разработанной в ИО CLIPS?
- 4) На каком языке программирования написана ИО CLIPS?
- 5) Определите основные структурные элементы программирования в ИО CLIPS.
- 6) Опишите особенности интерфейса ИО CLIPS.
- 7) Каковы особенности реализации логического вывода в ИО CLIPS?
- 8) К какому типу ИО относится CLIPS?
- 9) Опишите общую схему функционирования ЭС, разработанных с помощью ИО CLIPS.
- 10) Какие составляющие входят в структуру базы знаний CLIPS?
- 11) Какие стратегии принятия решений реализованы в ИО CLIPS?
- 12) Как проявляется свойство активности знаний?

Задание на самостоятельную работу

- 1) Практическое задание

Выполнить создание экспертной системы в оболочке CLIPS 6.31 по предметной области, определенной номером варианта практической работы №3.

- 2) Самостоятельная работа

Исследовать стратегии логического вывода, применяемые в современных интеллектуальных экспертных системах.

Список рекомендуемой литературы¹:

[1] Сидоркина, Ирина Геннадьевна. Системы искусственного интеллекта [Текст] : [учеб. пособие для вузов по направлению 230100 "Информатика и вычисл. техника"] / И. Г. Сидоркина. М.: Кнорус, 2011. - 245 с.

Рассмотрены теоретические аспекты проектирования систем искусственного интеллекта и даны модели представления знаний, изложены материалы информационного, справочного, консультирующего характера по использованию инструментальных средств и технологическим особенностям разработки систем данного класса.

Подробнее: <https://www.labyrinth.ru/books/276217/>

[2] Богомолова, М. А. Экспертные системы (техника и технология проектирования) [Текст] : Методические указания к лабораторным работам / М. А. Богомолова. Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015. - 47 с.

Методические указания являются руководством при выполнении лабораторных занятий, проводимых по курсу «Экспертные системы (техника и технология проектирования)» с магистрами направления 09.04.03 - «Прикладная информатика» в терминальном классе. Лабораторные работы связаны с изучением принципов построения экспертных систем на основе различных моделей представления знаний, и могут быть полезны студентам других специальностей.

Подробнее: <https://www.iprbookshop.ru/71908.html>

[3] Советов, Борис Яковлевич. *Представление знаний в информационных системах* [Текст] : [учеб. для вузов по направлению подгот. "Информ. системы и технологии"] / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. М.: Академия, 2011.

Учебник создан в соответствии с Федеральным государственным образовательным стандартом по направлению бакалавриата «Информационные системы и технологии». Рассмотрены современное состояние теоретических и прикладных вопросов представления знаний в информационных системах, идеология построения интеллектуальных систем, математический аппарат представления знаний, возможности и пути использования искусственного интеллекта при проектировании информационных систем. Изложены новые аспекты представления знаний на основе искусственных нейронных сетей, расчетно-логических систем, генетических алгоритмов, мультиагентных систем. Приведены примеры практической реализации представления знаний на базе декларативного языка Пролог. Для студентов учреждений высшего профессионального образования. Может быть полезен разработчикам и пользователям информационных систем; преподавателям и научным сотрудникам, сферой интересов которых является интеллектуализация различных предметных областей; менеджерам и руководителям различного ранга, желающим самостоятельно ознакомиться с современным состоянием информационных технологий.

Подробнее: <https://lavkababuin.com/predstavlenie-znaniy-v-informacionnyh-sistemah-2-e-izd-ster-650564/>

[4] Остроух, А. В. *Интеллектуальные информационные системы и технологии* [Электронный ресурс] : монография / А. В. Остроух, А. Б. Николаев. Санкт-Петербург: Лань, 2019. - 308 с.

В монографии изложены концептуальные основы и методы представления знаний в интеллектуальных системах. Рассмотрены различные подходы, применяемые при проектировании и разработке интеллектуальных систем и технологий в транспортном комплексе, а также рассмотрены тенденции развития систем искусственного интеллекта.

Монография может быть использована для формирования профессиональной компетентности студентов высших учебных заведений, аспирантов и научных сотрудников обучающихся и ведущих научные исследования в области разработки и практического применения систем искусственного интеллекта по укрупнённой группе направлений подготовки "Информатика и вычислительная техника".

Подробнее: <https://www.labirint.ru/books/692861/>

Ссылки на открытые ресурсы (он-лайн-курсы, видеоуроки и т.д.)

[5]

ⁱСписок рекомендуемой литературы может быть более широким по сравнению со списком литературы, указанным в рабочей программе

Прикладные интеллектуальные системы

Конспект лекции 8. Знания. Методы представлений знаний. Формализация базы знаний. Обработка знаний. Извлечение знаний из данных. Применение баз знаний в профессиональной деятельности.

Аннотация: В лекции рассказывается о хранении и применении знаний о проблемной области в интеллектуальных системах. Описаны методы представления знаний. Описаны способы приобретения знаний и наполнения базы знаний. Описано применение баз знаний в профессиональной деятельности.

План лекции:

1. Понятие знаний
2. Классификация знаний
3. Базы знаний
4. Базы знаний в архитектуре интеллектуальных систем
5. Моделей представления знаний
6. Технологии извлечения знаний и интеллектуального анализа данных
7. Задания для самостоятельной работы.

Ключевые слова: знания, извлечение знаний, база знаний, логическая модель, продукционная модель, фреймовая модель, семантическая модель, Data Mining, Big Data, анализ данных

СОДЕРЖАНИЕ ЛЕКЦИИ:

1. Понятие знаний

Данные – это информация, полученная в результате наблюдений или измерений отдельных свойств (атрибутов), характеризующих объекты, процессы и явления предметной области.

Знания (с точки зрения представления знаний в интеллектуальных системах) – это связи и закономерности предметной области (принципы, модели, законы), полученные в результате практической деятельности и профессионального опыта, позволяющего специалистам ставить и решать задачи в данной области.

Знания от данных отличаются рядом свойств:

- внутренняя интерпретируемость;
- структурированность;
- связность;
- семантическая метрика;
- активность.

Внутренняя интерпретируемость. Данные, хранящиеся в памяти или на внешних носителях, лишены имен, таким образом, отсутствует возможность их однозначной идентификации системой. Данные может идентифицировать лишь программа, извлекающая их по определенному алгоритму. При переходе к знаниям в память вводится дополнительная информация (атрибуты: фамилия, год рождения, специальность, стаж). Атрибуты могут играть роль имен. По ним можно осуществлять поиск нужной информации.

Структурированность. Информационные единицы должны обладать гибкой структурой. Иначе говоря, должна существовать возможность произвольного установления между отдельными информационными единицами отношений типа «часть – целое», «род – вид» или «элемент – класс».

Связность. Между информационными единицами должна быть предусмотрена возможность установления связей различного типа. Семантика отношений может носить декларативный или процедурный характер. Например, две и более информационные единицы могут быть связаны отношением «одновременно», две информационные единицы – отношением «причина – следствие» или «быть рядом».

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

Активность. Все вычислительные процессы инициируются командами, а данные используются этими командами лишь в случае необходимости. Иначе говоря, данные

пассивны, а команды активны.

Знания позволяют адаптироваться и действовать в реальной действительности.

Существует огромное множество различных знаний, начиная от рецепта приготовления омлета до квантовой физики.

2. Классификация знаний

Знания можно классифицировать по нескольким критериям (**рисунок 1**).

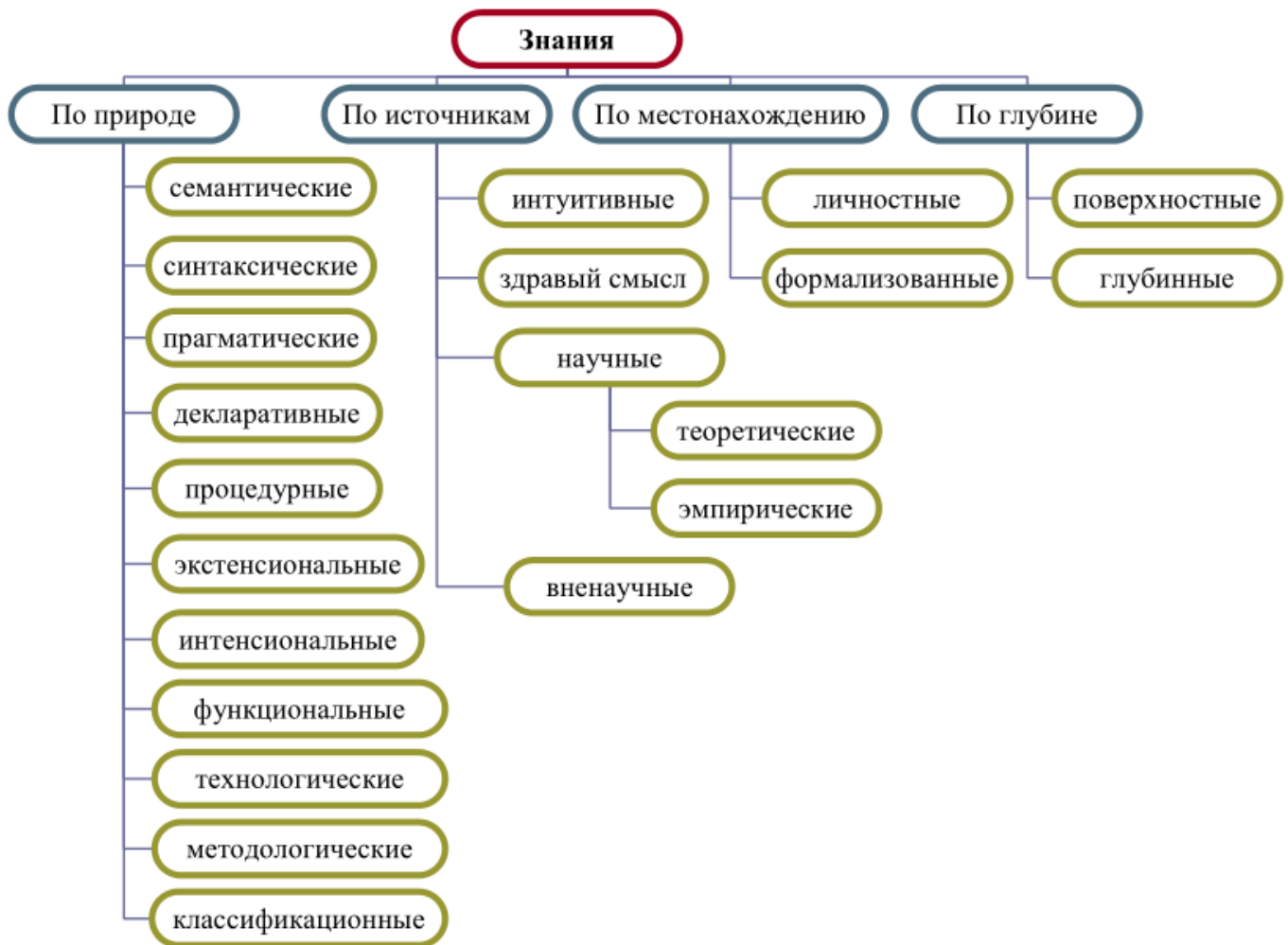


Рисунок 1 – Классификация знаний

Знание *синтаксического* типа характеризует синтаксическую структуру потока информации, которая не зависит от смысла и содержания используемых при этом понятий, то есть интеллектуальную систему не образует.

Семантическое знание рассматривается как структура, образующая текущий контекст. Оно содержит информацию, непосредственно связанную с текущими значениями и смыслом описываемых понятий, и предопределяет состояние связей данных в информационной базе.

Прагматическое знание предопределяет наиболее вероятные связи, описывающие данные с точки зрения решаемой задачи (обобщенный или «объективный» контекст), например, с учетом действующих в данной задаче специфических критериев и соглашений.

Декларативные знания содержат в себе представление о структуре понятий. Эти знания приближены к данным, фактам. Например, высшее учебное заведение есть совокупность факультетов, а каждый факультет в свою очередь есть совокупность кафедр.

Процедурные знания имеют активную природу. Они определяют представления о средствах и путях получения новых знаний, проверке знаний. Это алгоритмы разного рода. С развитием информатики все большая часть знаний сосредотачивалась в структурах данных (таблицы, списки, абстрактные типы данных), то есть увеличивалась роль декларативных.

Существенными для понимания природы знаний являются способы определения понятий. Один из широко применяемых способов основан на идее интенционала и экстенционала.

Интенционал понятия – это определение его через соотнесение с понятием более высокого уровня абстракции с указанием специфических свойств.

Экстенционал понятия – это определение понятия через перечисление его конкретных примеров, то есть понятий более низкого уровня абстракции.

Интенционалы формируют знания об объектах, в то время как экстенционалы объединяют данные.

Отсюда *интенциональные знания* – это знания о предметной области, которые отражают факты, закономерности, свойства и характеристики, справедливые для любых ситуаций, которые могут возникнуть в этой предметной области.

Экстенциональные знания – это знания о предметной области, отражающие факты, закономерности, свойства и характеристики, типичные для конкретных ситуаций или классов однотипных ситуаций, которые могут возникнуть в этой области.

Функциональные знания – это знания о выполняемых функциях отдельных предметов и о применении их в реальной действительности.

Технологические знания – специализированные знания, обеспечивающие поддержание технологических параметров производства; производственный опыт и навыки, используемые при решении повседневных производственных вопросов. Это может быть знание последовательности операций или знание технологической цепочки, позволяющие достигать поставленные цели в соответствии с принятой технологией.

Методологические знания – знания о методах преобразования действительности, научные знания о построении эффективной деятельности. К методологическим знаниям относят знание целей, форм и направлений развития теории, методов и способов эффективного преобразования практики.

Классификационные знания применяются главным образом в науке, являются обобщенными, системными знаниями. Пример – система элементов Д. И. Менделеева.

Интуиция – это вид знания, специфика которого обусловлена способом его приобретения. Это знание, не нуждающееся в доказательстве и воспринимаемое как достоверное. По способу получения интуиция – это прямое усмотрение объективной

связи вещей, не опирающееся на доказательство (интуиция есть усмотрение внутренним зрением; от лат. *intueri* – созерцать).

Под *здравым смыслом* понимают знания, позволяющие принимать правильные решения и делать правильные предположения, основываясь на логическом мышлении и накопленном опыте. В этом значении термин зачастую акцентирует внимание на способности человеческого разума противостоять предрассудкам, заблуждениям, мистификациям.

Научные знания в любом случае должны быть основанными на эмпирической или теоретической доказательной основе.

Теоретические знания – абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов. Теоретический уровень научного знания предполагает установление законов, дающих возможность идеализированного восприятия, описания и объяснения эмпирических ситуаций, то есть познания сущности явлений. Теоретические законы имеют более строгий, формальный характер по сравнению с эмпирическими. Термины описания теоретического знания относятся к идеализированным, абстрактным объектам. Подобные объекты невозможно подвергнуть непосредственной экспериментальной проверке.

Эмпирические знания получают в результате применения эмпирических методов познания: наблюдения, измерения, эксперимента. Это знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области. Эмпирические знания, как правило, констатируют качественные и количественные характеристики объектов и явлений. Эмпирические законы часто носят вероятностный характер и не являются строгими.

Вненаучные знания могут быть различными. *Паранормальные знания* – знания, несовместимые с имеющимся гносеологическим стандартом. Широкий класс *паранаучного* (пара от греч. около, при) знания включает в себя учения или размышления о феноменах, объяснение которых не является убедительным с точки зрения критериев научности. *Лженаучные знания* – сознательно эксплуатирующие домыслы и предрассудки. В качестве симптомов лженауки выделяют малограмотный пафос, принципиальную нетерпимость к опровергающим доводам, а также претенциозность. Лженаучные знания сосуществуют с научными знаниями.

Личностные (неявные, скрытые) знания – это знания людей, полученные из практики и опыта.

Формализованные (явные) знания – знания, содержащиеся в документах, на компакт-дисках, в персональных компьютерах, в Интернете, в базах знаний, в экспертных системах. Формализованные знания объективизируются знаковыми средствами языка, охватывают те знания, о которых мы знаем, их можно записать, сообщить другим.

3. Базы знаний

Перечисленные ниже пять особенностей информационных единиц определяют ту грань, за которой данные превращаются в знания, а базы данных перерастают в **базы знаний**.

База знаний (БЗ) – основа любой интеллектуальной системы, где знания описаны на некотором языке представления знаний, приближенном к естественному. Сегодня знания приобрели чисто декларативную форму, то есть знаниями считаются предложения, записанные на языках представления знаний, приближенных к естественному языку и понятных неспециалистам.

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

Таблица 1 – Данные о сотрудниках предприятия

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

Если, например, в память компьютера нужно было записать сведения о сотрудниках учреждения, представленные в таблице 1.1, то без внутренней интерпретации в память компьютера была бы занесена совокупность из четырех машинных слов, соответствующих строкам этой таблицы. При этом информация о том, какими группами двоичных разрядов в этих машинных словах закодированы сведения о специалистах, у системы отсутствуют. Они известны лишь программисту, который использует данные таблице 1.1 для решения возникающих у него задач.

При переходе к знаниям в память компьютера вводится информация о некоторой протоструктуре информационных единиц. В рассматриваемом примере она представляет собой специальное машинное слово, в котором указано, в каких разрядах хранятся сведения о фамилиях, годах рождения, специальностях и стажах. При этом должны быть заданы специальные словари, в которых перечислены имеющиеся в памяти системы фамилии, года рождения, специальности и продолжительности стажа. Все эти атрибуты могут играть роль имен для тех машинных слов, которые соответствуют строкам таблицы. По ним можно осуществлять поиск нужной информации. Каждая строка таблицы будет экземпляром

протоструктуры. В настоящее время СУБД обеспечивают реализацию внутренней интерпретируемости всех информационных единиц, хранящихся в базе данных.

Структурированность. Информационные единицы должны обладать гибкой структурой. Для них должен выполняться "принцип матрешки", т.е. рекурсивная вложенность одних информационных единиц в другие. Каждая информационная единица может быть включена в состав любой другой, и из каждой информационной единицы можно выделить некоторые составляющие ее информационные единицы. Другими словами, должна существовать возможность произвольного установления между отдельными информационными единицами отношений типа "часть - целое", "род - вид" или "элемент - класс".

Связность. В информационной базе между информационными единицами должна быть предусмотрена возможность установления связей различного типа. Прежде всего эти связи могут характеризовать отношения между информационными единицами. Семантика отношений может носить декларативный или процедурный характер. Например, две или более информационные единицы могут быть связаны отношением "одновременно", две информационные единицы - отношением "причина - следствие" или отношением "быть рядом". Приведенные отношения характеризуют декларативные знания. Если между двумя информационными единицами установлено отношение "аргумент - функция", то оно характеризует процедурное знание, связанное с вычислением определенных функций. Далее будем различать отношения структуризации, функциональные отношения, каузальные отношения и семантические отношения. С помощью первых задаются иерархии информационных единиц, вторые несут процедурную информацию, позволяющую находить (вычислять) одни информационные единицы через другие, третьи задают причинно-следственные связи, четвертые соответствуют всем остальным отношениям.

Между информационными единицами могут устанавливаться и иные связи, например, определяющие порядок выбора информационных единиц из памяти или указывающие на то, что две информационные единицы несовместимы друг с другом в одном описании. Перечисленные три особенности знаний позволяют ввести общую модель представления знаний, которую можно назвать семантической сетью, представляющей собой иерархическую сеть, в вершинах которой находятся информационные единицы. Эти единицы снабжены индивидуальными именами. Дуги семантической сети соответствуют различным связям между информационными единицами.

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

Активность. Все процессы, протекающие в ЭВМ, инициируются командами, а данные используются этими командами лишь в случае необходимости. Для ИС эта ситуация не приемлема. Как и у человека, в ИС актуализации тех или иных действий способствуют знания, имеющиеся в системе. Таким образом, выполнение программ в ИС должно инициироваться текущим состоянием информационной базы. Появление в базе фактов или описаний событий, установление связей может стать источником активности системы.

Совокупность средств, обеспечивающих работу с знаниями, образует **Систему Управления Базой Знаний (СУБЗ)**. В настоящее время не существует баз знаний, в которых в полной мере были бы реализованы внутренняя интерпретируемость, структуризация, связность, введена семантическая мера и обеспечена активность знаний.

4. Базы знаний в архитектуре интеллектуальных систем

Архитектура интеллектуальных систем включает три комплекса вычислительных средств (рисунок 2).

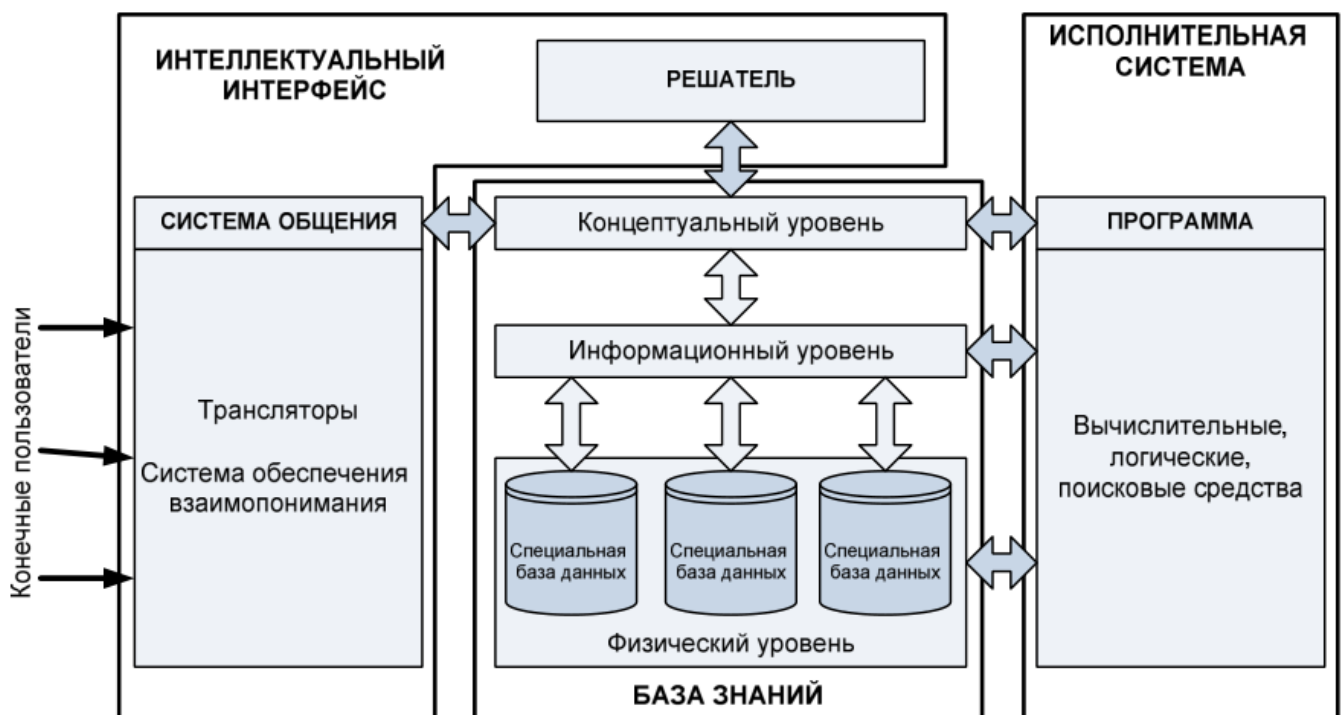


Рисунок 2 – Архитектура интеллектуальных систем

Первый комплекс представляет собой совокупность средств, выполняющих программы (исполнительную систему), спроектированных с позиций эффективного решения задач, имеет в ряде случаев проблемную ориентацию.

Второй комплекс представляет собой совокупность средств интеллектуального интерфейса, имеющих гибкую структуру, которая обеспечивает возможность адаптации в широком спектре интересов конечных пользователей.

Третьим комплексом средств, с помощью которых организуется взаимодействие первых двух, является база знаний, обеспечивающая использование вычислительными средствами первых двух комплексов целостной и независимой от

обрабатывающих программ системы знаний о проблемной среде. Исполнительная система объединяет всю совокупность средств, обеспечивающих выполнение сформированной программы.

Интеллектуальный интерфейс – система программных и аппаратных средств, обеспечивающих для конечного пользователя использование компьютера для решения задач, которые возникают в среде его профессиональной деятельности либо без посредников, либо с незначительной их помощью.

БЗ занимает центральное положение по отношению к остальным компонентам вычислительной системы. В целом, через БЗ осуществляется интеграция средств вычислительной системы, участвующих в решении задач.

5. Моделей представления знаний

5.1. Логическая модель представления знаний

Логическая модель представляет собой формальную систему — некоторое логическое исчисление как правило, исчисление предикатов первого порядка, когда предметная область или задача описывается в виде набора аксиом.

Все знания о предметной области описываются в виде формул этого исчисления или правил вывода. Описание в виде формул дает возможность представить декларативные знания, а правила вывода — процедурные знания.

Предикат висчисления предикатов специальный знак, отражающий определенное отношение между конечным множеством сущностей - аргументов. Предикат первого порядка имеет два состояния - истина и ложь.

Рассмотрим в качестве примера знание: "Когда температура в печи достигает 120° и прошло менее 30 минут с момента включения печи, давление не может превосходить критическое. Если с момента включения печи прошло более 30 мин, то необходимо открыть вентиль №2".

Логическая модель представления этого знания имеет вид:

$P(p = 120) \wedge T(t < 30) \rightarrow (D < D_{кр}); P(p = 120) \wedge T(t > 30) \Rightarrow F(N^{\circ}2).$

В этой записи использованы следующие обозначения:

$P(p = 120)$ — предикат, становящийся истинным, когда температура достигает 120°;

$T(t < 30)$ — предикат, остающийся истинным в течение 30 мин с начала процесса;

$T(t > 30)$ — предикат, становящийся истинным по истечении 30 мин с начала процесса;

$D < D_{кр}$ — утверждение о том, что давление ниже критического;

$F(N^{\circ}2)$ — команда открыть вентиль №2.

Кроме того, в этих записях использованы типовые логические связки И ($\wedge$), импликации ($\rightarrow$) и логического следования ($\Rightarrow$).

Первая строчка в записи представляет декларативные знания, а вторая — процедурные.

Языки представления знаний логического типа широко использовались на ранних стадиях развития интеллектуальных систем, но вскоре были вытеснены (или, во

всяком случае, сильно потеснены) языками других типов. Объясняется это громоздкостью записей, опирающихся на классические логические исчисления. При формировании таких записей легко допустить ошибки, а поиск их очень сложен. Отсутствие наглядности, удобочитаемости (особенно для тех, чья деятельность не связана с точными науками) затрудняло распространение языков такого типа. Исчисление предикатов 1-го порядка в промышленных экспертных системах практически не используется. Эта логи- ческая модель применима в основном в исследовательских «игрушечных» системах, так как предъявляет очень высокие требования и ограничения к предметной области.

5.2 Продукционная модель представления знаний.

Продукционная модель (модель правил) - это модель, основанная на правилах, в которой знания представлены в виде предложений типа «**Если** (условие), **то** (действие)».

Под «условием» (*антецедентом*) понимается некоторое предложение-образец, по которому осуществляется поиск в базе знаний, а под «действием» (*консеквентом*) — действия, выполняемые при успешном исходе поиска (они могут быть промежуточными, выступающими далее как условия и терминальными или целевыми, завершающими работу системы).

Основу модели составляют системы продукций. Каждая **продукция** в наиболее общем виде записывается как стандартное выражение следующего вида:

"Имя продукции":

Имя сферы; Предусловие; Условие для ядра; Ядро: «Если А, то В»; Постусловие.

В наиболее простом виде продукция может состоять лишь из имени (например, ее порядкового номера в системе продукций) и ядра.

Имя продукции может выражаться в виде номера или слова (словосочетания). Служит для определения местоположения в системе продукций.

Имя сферы указывает ту предметную область, к которой относятся знания, зафиксированные в данной продукции. В интеллектуальной системе может храниться совокупность знаний (ее называют *базой знаний*), относящихся к разным областям (например, знания о различных заболеваниях че- ловека или знания из различных разделов математики). Ясно, что если в данный момент решается задача из области физики твердого тела или из геометрии треугольника, то надо использовать знания, относящиеся именно к этой области. Сферы и выделяют такие подобласти знаний.

Предусловие определяет необходимые предпосылки применения условия для ядра продукции. Предусловия может и не быть вовсе.

Условие для ядра определяет те ситуации, при выполнении которых можно (надо) проверять наличие или истинность А в ядре продукции. Следующий пример иллюстрирует употребление условия для ядра: a, b, c —

стороны треугольника; если $c = \sqrt{a^2 + b^2}$, то треугольник

является прямоугольным. Ясно, что при другой интерпретации a , b и c не имеет никакого смысла использовать данное ядро продукции.

Ядро - основная часть продукции. Ядро имеет вид: "Если A , то B ", где A и B могут иметь разные значения. Остальные элементы, образующие продукцию, носят вспомогательный характер. Несколько примеров ядра:

"Если сверкает молния, то гремит гром".

"Если в доме вспыхнул пожар, то вызывайте по телефону 01 пожарную команду".

"Если в путеводителе указано, что в городе есть театр, то надо пойти туда".

Первый пример иллюстрирует тот случай, когда ядро продукции описывает причинно-следственную связь явлений A и B . Во втором примере A и B представляют собой некоторые действия. В третьем примере A — это некоторые знания, а B — действие. Возможны и другие варианты ядра продукции. Таким образом, при помощи ядер можно представлять весьма разнообразные знания, как декларативные знания, так и процедурные, хотя сама форма продукции весьма удобна для задания именно процедурных знаний.

Постусловие определяет специфику ядра. Используется редко. Обычно содержит пояснения.

Пример продукции декларативного знания: «Пусть a , b , c — стороны треугольника и

при выполнении равенства $c = \sqrt{a^2 + b^2}$ треугольник является прямоугольным».

Имя продукции: **№5**

Имя сферы: *Геометрия;*

Предусловие: *Фигуры;*

Условие для ядра: a , b , c — стороны треугольника;

Ядро: **Если $c = \sqrt{a^2 + b^2}$, то треугольник**
прямоугольный;

Постусловие: Теорема Пифагора.

Основные достоинства систем продукционных:

- Простота создания и понимания, отдельных правил.
- Простота понимания и модифицирования знаний.
- Простота программной реализации механизма логического вывода.

Слабые стороны систем:

1. Неясность взаимоотношений правил.
2. Сложность оценки целостного образа знаний.
3. Весьма низкая эффективность обработки.
4. Существенные отличия от человеческих систем знаний.
5. Отсутствие гибкости в логическом выводе.

Данная модель широко применяется в экспертных системах, в том числе промышленного (коммерческого) уровня.. Имеется большое число программных средств, реализующих продукционный подход (язык OPS 5; «оболочки» или «пустые» ЭС — EXSYSProfessional, Карра, ЭКСПЕРТ; ЭКО и др.).

5.3. Семантическая модель представления знаний

Термин «семантическая» означает «смысловая», а сама семантика — это наука, устанавливающая отношения между символами и объектами, которые они обозначают, то есть наука, определяющая смысл знаков.

Семантическая сеть – это модель, в которой структура знаний предметной области формализуется в виде ориентированного графа вершины которого — понятия, а дуги — отношения между ними

В качестве понятий обычно выступают абстрактные или конкретные объекты, события, свойства, операции. Отношения — это связи различного типа. Существуют отношения разных типов:

- логические (дизъюнкция, конъюнкция, отрицание, импликация);
- теоретико-множественные (часть – целое, множество – подмножество, класс – элемент класса, пример элемента класса);
- функциональные (количественные, временные, пространственные и другие характеристики: объект- свойство, свойство-значение);
- квантификационные (логические кванторы общности и существования, нелогические кванторы, например, много, несколько);
- Наиболее часто в семантических сетях используются следующие отношения:
- связи типа «часть— целое» («класс— подкласс»,
- «элемент—множество», и т. п.). Отношения — является|| и
- — имеет часть|| определяют иерархическую структуру, в которой свойства "высших" понятий автоматически переносятся на "низшие" понятия. Это позволяет избежать дублирования информации в сети.;
- функциональные связи (определяемые обычно глаголами «производит», «влияет»...);
- количественные (больше, меньше, равно...);
- пространственные (далеко от, близко от, за, под, над...);
- временные (раньше, позже, в течение...);
- атрибутивные связи (иметь свойство, иметь значение);

Классификация семантических сетей:

Можно предложить несколько классификаций семантических сетей, связанных с типами отношений между понятиями.

По количеству типов отношений:

Однородные (с единственным типом отношений). Неоднородные (с различными типами отношений).

По типам отношений:

Бинарные (в которых отношения связывают два объекта).

N-арные (в которых есть специальные отношения, связывающие более двух понятий).

Проблема поиска решения в базе знаний типа семантической сети сводится к задаче поиска фрагмента сети, соответствующего некоторой подсети, отражающей поставленный запрос к базе.

В качестве примера, рассмотрим текст, который содержит некоторые знания (декларативные): «Иванов имеет личный автомобиль «Волга» красного цвета с мощностью двигателя 75 л.с.»

На рис.1.3 изображена семантическая сеть. В качестве вершин тут выступают понятия: «человек», «Иванов», «Волга», «автомобиль», «вид транспорта» и «двигатель».



Рис. 3. Семантическая сеть.

Основным преимуществом семантической модели является то, что она более других соответствует современным представлениям об организации долговременной памяти человека.

Недостатком этой модели является сложность организации процедуры поиска вывода на семантической сети.

Для реализации семантических сетей существуют специальные сетевые языки, например NET, язык реализации систем SIMER+MIR [и др. Широко известны экспертные системы, использующие семантические сети в качестве языка представления знаний — PROSPECTOR, CASNET, TORUS.

5.4 Фреймовая модель представления знаний Понятие и структура фрейма.

Теория фреймов - это парадигма для представления знаний с целью использования этих знаний компьютером. Теория фреймов (теория представления знаний фреймами) была разработана Марвином Минским (США) в 70-е г.г. XX в. В ее основе лежит восприятие фактов посредством сопоставления полученной извне информации с конкретными элементами и значениями, а также с рамками, определенными для каждого объекта в памяти человека. Эта модель, как и семантическая сеть, имеет глубокое психологическое обоснование.

Минский разработал такую схему, в которой информация содержится в специальных ячейках, называемых фреймами, объединенными в сеть, называемую системой фреймов. Новый фрейм активизируется с наступлением новой ситуации. Отличительной его чертой является то, что он одновременно содержит большой объем знаний и в то же время является достаточно гибким для того, чтобы быть использованным как отдельный элемент БД. Термин

«фрейм» был наиболее популярен в середине семидесятых годов, когда существовало много его толкований, отличных от интерпретации Минского.

Чтобы лучше понять эту теорию, рассмотрим один из примеров Минского, основанный на связи между ожиданием, ощущением и чувством человека, когда он открывает дверь и входит в комнату. Предположим, что вы собираетесь открыть дверь и зайти в комнату незнакомого вам дома. Находясь в доме, перед тем как открыть дверь, у вас имеются определенные представления о том, что вы увидите, войдя в комнату. Например, если вы увидите какой либо пейзаж или морской берег, поначалу вы с трудом узнаете их. Затем вы будете удивлены, и в конце концов дезориентированы, так как вы не сможете объяснить поступившую информацию и связать ее с теми представлениями, которые у вас имелись до того. Также у вас возникнут затруднения с тем, чтобы предсказать дальнейший ход событий. С аналитической точки зрения это можно объяснить как активизацию фрейма комнаты в момент открывания двери и его ведущую роль в интерпретации поступающей информации. Если бы вы увидели за дверью кровать, то фрейм комнаты приобрел бы более узкую форму и превратился бы во фрейм кровати. Другими словами, вы бы имели доступ к наиболее специфичному фрейму из всех доступных.

Термин *фрейм* (от английского слова *frame*, что означает «каркас» или «рамка») был предложен М.Минским для обозначения структуры знаний для восприятия пространственных сцен.

Фрейм – это структура для представления знаний, которая при заполнении ее соответствующими значениями превращается в описание конкретного факта, события или ситуации.

Фрейм является минимально возможным описание сущности какого-либо явления, события, ситуации, процесса или объекта. (Минимально возможное означает, что при дальнейшем упрощении описания теряется его полнота, оно перестает определять ту единицу знаний, для которой оно предназначено.)

В психологии и философии известно понятие абстрактного образа. Например, произнесение вслух слова

«комната» порождает у слушающих образ комнаты: «жилое помещение с четырьмя стенами, полом, потолком, окнами и дверью, площадью 6-20 м²». Из этого описания ничего нельзя убрать (например, убрав окна, мы получим уже чулан, а не комнату), но в нем есть «дырки» или «слоты» — это незаполненные значения некоторых атрибутов — например, количество окон, цвет стен, высота потолка, покрытие пола и др. В теории фреймов такой образ комнаты называется *фреймом комнаты*. Фреймом также называется и формализованная модель для отображения образа.

Структура фрейма. Фрейм имеет почти однородную структуру и состоит из стандартных единиц, называемых слотами. Каждая такая единица — слот — содержит название и свое значение.

Фрейм можно представить: а) в виде цепочки:

Фрейм: <слот 1>, <слот 2>, ... , <слот N>.

или более подробно, как список свойств:

Имя ФРЕЙМА: <имя 1-го слота: значение 1-го слота>, <имя 2-го слота: значение 2-го слота>, <...>, <имя N-го слота: значение N-го слота>.

б) в виде таблицы:

Имя фрейма	
Имя слота 1	Значение слота 1
Имя слота 2	Значение слота 2
Имя слота N	Значение слота N

В качестве примера, рассмотрим фрейм для понятия "Лекция". Ситуация "лекция" может быть определена как "чтение лектором учебного материала слушателям". Фрейм "лекция" может содержать слоты "предмет" (предмет, по которому проводится лекция), "лектор" (ФИО лектора), "аудитория" (место проведения лекции), "слушатели" (количество слушателей).

Лекция	
Предмет	Физика
Лектор	Иванов И.И.

Аудитория	109
Слушатели	Пи41

В данном случае "ЛЕКЦИЯ" - название фрейма; "ПРЕДМЕТ", "ЛЕКТОР", "АУДИТОРИЯ", "СЛУШАТЕЛИ" - слоты; "Физика", "Иванов И.И.", "109", "ПИ41" - значения слотов.

Типы фреймов. Различают фреймы-прототипы, хранящиеся в базе знаний, и фреймы-экземпляры, которые создаются для отображения реальных фактических ситуаций на основе поступающих данных.

Фрейм-прототип (протофрейм) – это фрейм, содержащий знания о самом понятии. Например, фрейм понятия «Битва» можно изобразить следующим образом:

Битва	
Субъект (кто?)	X1
Объект (с кем?)	X2
Место (где?)	X3
Время (когда?)	X4
Результат	X5

В этом фрейме указаны имена слотов (субъект, объект и т. д.), но вместо их значений стоят переменные (X1, X2 и т. д.).

Фрейм-экземпляр (экзофрейм) - это фрейм, содержащий конкретное описание понятия или знания.

В примере, наверное, основными для фрейма "Битва" можно считать слоты с именами "субъект" и "объект". Битву осуществляет субъект X1 с объектом X2 в месте X3 во время X4, при этом получается результат X5. Подставляя вместо всех переменных конкретные значения, получим конкретный факт-описание:

Куликовская битва	
Субъект (кто?)	Князь Дмитрий
Объект (с кем?)	Хан Мамай
Место (где?)	Куликово поле

Время (когда?)	Утром в сентябре 1380 года
Результат	Победа князя Дмитрия

Исключение из фрейма любого слота делает его принципиально не полным, а иногда вообще бессмысленным.

Модель фрейма является достаточно универсальной, поскольку в состав фрейма могут входить слоты с именами действий, то фреймы годятся для представления как декларативных, так и процедурных знаний.

Фрейм является *простым*, если он не содержит в себе других фреймов. *Сложный (составной)* фрейм содержит в себе два и более фрейма, и по существу представляет сеть фреймов.

Фрейм позволяет отобразить все многообразие знаний о мире через:

- фреймы – *структуры*, использующиеся для обозначения объектов и понятий (заем, залог, вексель);
- фреймы – *роли* (менеджер, кассир, клиент);
- фреймы – *сценарии* (банкротство, собрание акционеров, празднование именин);
- фреймы – *ситуации* (тревога, авария, рабочий режим устройства) и др.

Фреймовая модель представления знаний – это модель, в которой структура знаний предметной области формализуется в виде совокупности взаимосвязанных фреймов, описывающих объекты, а свойства этих объектов и факты, относящиеся к ним, описываются в структурных элементах фрейма.

Чтобы представить семантическую сеть в виде совокупности фреймов, надо уметь представлять отношения между вершинами сети. Для этого также используются слоты фреймов. Эти слоты могут иметь имена вида "Связь Y", где Y есть имя того отношения (его тип), которое устанавливает данный фрейм-вершина с другим фреймом-вершиной. Заметим также, что в качестве значения слота может выступать новый фрейм, что позволяет на множестве фреймов осуществлять иерархическую классификацию. Это очень удобное свойство фреймов, так как человеческие знания, как правило, упорядочены по общности.

Важнейшим свойством теории фреймов является заимствование из теории семантических сетей — так называемое *наследование свойств*. Во фреймах наследование происходит по **АКО-связям** (A-Kind-Of= это), которые связывают фреймы с фреймами, находящимися на уровень выше в иерархии, откуда неявно наследуются (переносятся) значения слотов. Слот АКО указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, то есть переносятся, значения аналогичных слотов.

ПРИМЕР: Понятие «Лектор».



В данном случае представлено одно звено иерархии (ЧЕЛОВЕК-ЛЕКТОР). Здесь фрейм "ЧЕЛОВЕК" является обобщающим для фрейма "ЛЕКТОР". Таким образом, фрейм "ЛЕКТОР" наследует от фрейма "ЧЕЛОВЕК" значение слота "УМЕЕТ" (а также других слотов, не показанных в примере). Цепочка наследования может быть продолжена вплоть до, например, фрейма "ЖИВОЕ СУЩЕСТВО".

Кроме того связь фреймов осуществляется по **значению слота**. Фрагмент сети фреймов представлен на рис.4. Такая структура позволяет систематизировать большой объем информации, оставляя ее при этом максимально удобной для использования. Кроме того, система (сеть) фреймов способна отражать концептуальную основу организации памяти человека.

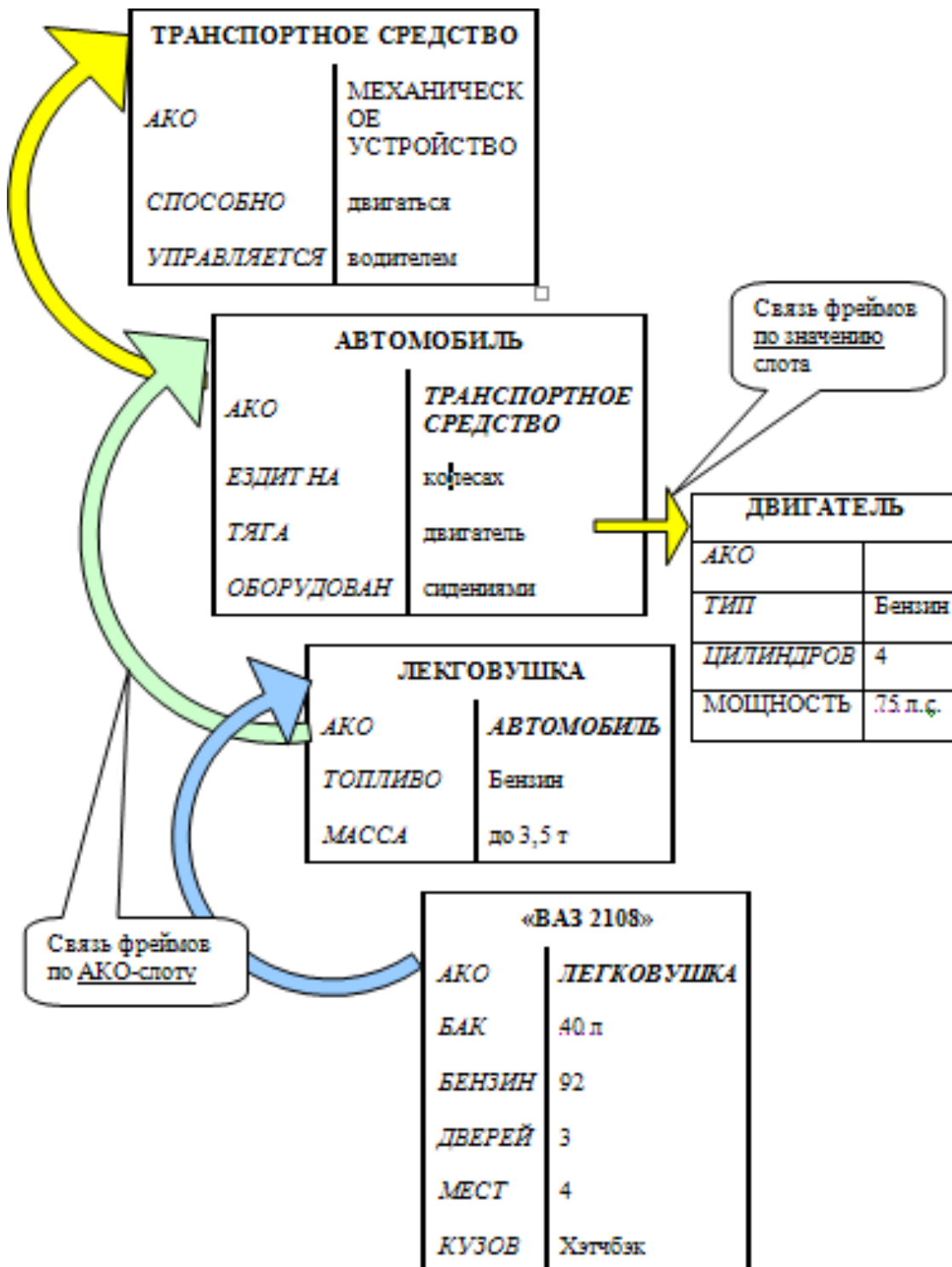


Рис.4 Фрагмент сети фреймов.

6. Технологии извлечения знаний и интеллектуального анализа данных

В настоящее время интенсивно разрабатывались методы автоматического извлечения знаний из накопленных фактов, хранящихся в различных базах данных. Для анализа информации, накопленной в современных базах данных, методы анализа должны быть эффективными, т.е. простыми в использовании, обладать значительным уровнем масштабируемости и определенным автоматизмом. Это концепция, зародившаяся в 1989 г., лежит в основе двух современных технологий анализа данных **Data Mining** и **KDD** –

Knowledge Discovery in Databases, которые на русский язык переводятся как «добыча (раскопка) данных» и

«извлечение знаний из баз данных».

Knowledge Discovery in Database(KDD) — процесс получения из данных знаний в виде зависимостей, правил и моделей позволяющих моделирование и прогнозирование различных процессов.

В отечественной литературе применяется термин

«**Интеллектуальный анализ данных**» (ИАД). В этой области необходимо отметить пионерские работы отечественных исследователей, в частности, М.М. Бонгарда (программа «Кора»), В.К. Финна (JSM-метод), А.Г. Ивахненко (МГУА), выполнивших свои работы задолго до того, как в этой области на Западе возник настоящий бум.

ИАД— это процесс поддержки принятия решений, основанный на поиске в данных скрытых закономерностей, то есть извлечения информации, которая может быть охарактеризована как знания.

Интеллектуальный анализ данных является кратким обозначением довольно широкого спектра процедур автоматического анализа данных высокоинтеллектуальными технологиями.

В общем случае процесс ИАД состоит из трех стадий:

- выявление закономерностей (свободный поиск);
- использование выявленных закономерностей для предсказания неизвестных значений (прогностическое моделирование);
- анализ исключений, предназначенный для выявления и толкования аномалий в найденных закономерностях.

Причины распространения KDD и Data Mining.

В KDD и Data Mining нет ничего принципиально нового. Специалисты в различных областях человеческого знания решали подобные задачи на протяжении нескольких десятилетий. Однако в последние годы интеллектуальная составляющая бизнеса стала возрастать, и для распространения технологий KDD и Data Mining были созданы все необходимые и достаточные условия.

Развитие технологий автоматизированной обработки информации создало основу для учета сколь угодно большого количества факторов и объема данных.

Возникла острая нехватка высококвалифицированных специалистов в области статистики и анализа данных. Поэтому потребовались технологии обработки и анализа, доступные для специалистов любого профиля за счет применения методов визуализации и самообучающихся алгоритмов.

Возникла объективная потребность в тиражировании знаний. Полученные в процессе KDD и Data Mining результаты являются формализованным описанием некоего процесса, а следовательно, поддаются автоматической обработке и повторному использованию на новых данных.

На рынке появились программные продукты, поддерживающие технологии KDD и Data Mining, – [аналитические платформы](#). С их помощью можно создавать полноценные аналитические решения и быстро получать первые результаты.

Технология Knowledge Discovery in Databases описывает не конкретный алгоритм или математический аппарат, а последовательность действий, которую необходимо выполнить для построения модели с целью извлечения знания. Она не зависит от предметной области; это набор атомарных операций, комбинируя которые, можно получить нужное решение.

KDD включает в себя этапы подготовки данных, предобработки (очистки) данных, [трансформации данных](#), построения моделей (применения методов Data Mining) и интерпретации полученных результатов. Ядром этого процесса являются методы Data Mining, позволяющие обнаруживать закономерности и знания (рис.5).



Рис. 5. Этапы KDD.

Кратко рассмотрим каждый этап.

- 1 этап. Выборка данных.** Этот этап заключается в подготовке набора данных, в том числе из различных источников, выбора значимых параметров и т.д. Для этого должны быть различные инструменты доступа к различным источникам данных – конверторы, запросы, фильтрация данных и т.п. В качестве источника рекомендуется использовать специализированное хранилище данных, агрегирующее всю необходимую для анализа информацию.
- 2 этап. Очистка данных.** Реальные данные для анализа редко бывают хорошего [качества](#). Поэтому для эффективного применения методов Data Mining следует обратить серьезное внимание на вопросы предобработки данных. Данные могут содержать пропуски, шумы, аномальные значения и т.д. Кроме того, данные могут быть противоречивы, избыточны, недостаточны, содержать ошибки и т.д.

Для решения каждой из этих проблем есть отработанные методы. Конечно, ошибки можно править и вручную, но при больших объемах данных это становится довольно проблематично. Поэтому рассмотрим варианты решения этих задач в автоматическом режиме при минимальном участии человека.

Противоречивость информации. Для начала нужно решить, что именно считать противоречием. Как ни странно, это задача нетривиальная. Например, пенсионную карточку в России нужно менять в случае изменения фамилии, имени, отчества и пола. Оказывается, в том, что человек родился женщиной, а вышел на пенсию мужчиной, противоречия нет!

После того, как мы определимся с тем, что считать противоречием и найдем их, есть несколько вариантов действий.

При обнаружении нескольких противоречивых записей, удалять их. Метод простой, а потому легко реализуемый. Иногда этого бывает вполне достаточно. Тут важно не переусердствовать, иначе мы можем вместе с водой выплеснуть младенца.

Исправить противоречивые данные. Можно вычислить вероятность появления каждого из противоречивых событий и выбрать наиболее вероятное. Это самый грамотный и корректный метод работы с противоречиями.

Пропуски в данных. Очень серьезная проблема. Это вообще бич для большинства хранилищ данных. Большинство методов [прогнозирования](#) исходят из предположения, что данные поступают равномерным постоянным потоком. На практике такое встречается крайне редко. Поэтому одна из самых востребованных областей применения хранилищ данных – прогнозирование – оказывается реализованной некачественно или со значительными ограничениями. Для борьбы с этим явлением можно воспользоваться следующими методами:

[Аппроксимация](#). Т.е. если нет данных в какой-либо точке, мы берем ее окрестность и вычисляем по известным формулам значение в этой точке, добавляя соответствующую запись в хранилище. Хорошо это работает для [упорядоченных данных](#). Например, сведения об ежедневных продажах продуктов.

Определение наиболее правдоподобного значения. Для этого берется не окрестность точки, а все данные. Этот метод применяется для неупорядоченной информации, т.е. случаев, когда мы не в состоянии определить, что же является окрестностью исследуемой точки.

Аномальные значения. Довольно часто происходят события, которые сильно выбиваются из общей картины. И лучше всего такие значения откорректировать. Это связано с тем, что средства прогнозирования ничего не знают о природе процессов. Поэтому любая аномалия будет восприниматься как совершенно нормальное значение. Из-за этого будет сильно искажаться картина будущего. Какой-то случайный провал или успех будет считаться закономерностью.

Есть метод борьбы и с этой напастью – это робастные оценки. Это методы устойчивые к сильным возмущениям. Мы оцениваем имеющиеся данные ко всему, что выходит за допустимые границы, и применяем одно из следующих действий:

- Значение удаляется;
- Заменяется на ближайшее граничное значение.

Шум. Почти всегда при анализе мы сталкиваемся с шумами. Шум не несет никакой полезной информации, а лишь мешает четко разглядеть картину. Методов борьбы с этим явлением несколько.

Спектральный анализ. При помощи него мы можем отсечь высокочастотные составляющие данных. Проще говоря, это частые и незначительные колебания около основного сигнала. Причем, изменяя ширину спектра, можно выбирать какого рода шум мы хотим убрать.

Авторегрессионные методы. Этот довольно распространенный метод **активно** применяется при анализе временных рядов и сводится к нахождению функции, которая описывает процесс плюс шум. Собственно шум после этого можно удалить и оставить основной сигнал.

Ошибки ввода данных. Вообще это тема для отдельного разговора, т.к. количество типов такого рода ошибок слишком велико, например, опечатки, сознательное искажение данных, несоответствие форматов, и это еще не считая типовых ошибок, связанных с особенностями работы приложения по вводу данных. Для борьбы с большинством из них есть отработанные методы. Некоторые вещи очевидны, например, перед внесением данных в хранилище можно провести проверку форматов. Некоторые более изощренные. Например, можно исправлять опечатки на основе различного рода тезаурусов. Но, в любом случае, очищать нужно и от такого рода ошибок.

К задачам очистки данных относятся: заполнение пропусков, подавление аномальных значений, сглаживание, исключение дубликатов и противоречий и пр.

Ошибочно предполагать, что если подать данные на вход системы в существующем виде, то на выходе будут получены полезные знания. Входные данные должны быть качественными и корректными.

3 этап. Трансформация данных. Этот шаг необходим для тех методов, которые требуют, чтобы исходные данные были в каком-то определенном виде. Дело в том, что различные алгоритмы анализа требуют специальным образом подготовленные данные. Например, для прогнозирования необходимо преобразовать временной ряд при помощи скользящего окна или вычисление агрегируемых показателей. К задачам трансформации данных относятся: скользящее окно, приведение типов, выделение временных интервалов, преобразование непрерывных значений в дискретные и наоборот, сортировка, группировка и прочее.

4 этап. Data Mining. На этом этапе строятся модели, в которых применяются различные алгоритмы для нахождения знаний. Это нейронные сети, деревья решений, алгоритмы кластеризации и установления ассоциаций и т.д.

5 этап. Интерпретация. На данном этапе осуществляется применение пользователем полученных моделей (знаний) в бизнес приложениях. Для оценки качества полученной модели нужно использовать как формальные методы, так и знания **аналитика**. Именно аналитик может сказать, насколько применима полученная модель к реальным данным.

Пример. Имеется сеть магазинов розничной торговли. Требуется получить прогноз объемов продаж на следующий месяц. Первым шагом будет сбор истории продаж в каждом магазине и объединение ее в общую выборку данных. Следующим шагом будет предобработка собранных данных: их группировка по месяцам, сглаживание кривой продаж, устранение факторов, слабо влияющих на объемы продаж. Далее следует

построить модель зависимости объемов продаж от выбранных факторов. Это можно сделать с помощью линейной регрессии или нейронных сетей. Имея такую модель, можно получить прогноз, подав на вход модели историю продаж. Зная прогнозное значение, его можно использовать, например, в приложениях оптимизации для лучшего размещения товара на складе.

Полученные модели являются, по сути, формализованными знаниями эксперта, а следовательно, их можно **тиражировать**. В этом заключается самое главное преимущество KDD. Т.е. построенную одним человеком модель могут применять другие, без необходимости понимания методик, при помощи которой эти модели построены. Найденные знания должны быть использованы на новых данных с некоторой степенью достоверности.

Задачи и методы Data Mining.

Термин Data Mining дословно переводится как «добыча данных» или «раскопка данных» и имеет в англоязычной среде несколько определений.

Data Mining — обнаружение в «сырых» данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности.

Информация, найденная в процессе применения методов Data Mining, должна быть нетривиальной и ранее неизвестной, например, сведения о средних продажах таковыми не являются. Знания должны описывать новые связи между свойствами, предсказывать значения одних признаков на основе других.

Нередко KDD отождествляют с Data Mining. Однако правильнее считать Data Mining шагом процесса KDD.

Приведенные ниже примеры из разных областей экономики демонстрируют основное преимущество методов Data Mining – способность обнаружения новых знаний, которые невозможно получить методами статистического, регрессионного анализа или эконометрики.

1. Множество клиентов компании с помощью одного из инструментов Data Mining были объединены в группы, или сегменты со схожими признаками. Это позволило проводить компании различную маркетинговую политику и строить отдельные модели поведения для каждого клиентского сегмента. Наиболее значимыми факторами для разделения на группы оказались следующие: удаленность региона клиента, сфера деятельности, среднегодовые суммы сделок, количество сделок в неделю.
2. Автоматический анализ банковской базы данных кредитных сделок физических лиц выявил правила, по которым потенциальным заемщикам отказывалось в выдаче кредита. В частности, решающими факторами при выдаче кредитов на небольшие суммы, оказались: срок кредита, среднемесячный доход и расход заемщика. В дальнейшем это учитывалось при экспресс-кредитовании наиболее дешевых товаров.

3. При анализе базы данных клиентов страховой компании был установлен социальный портрет человека, страхующего жизнь - это оказался мужчина 35-50 лет, имеющий 2 и более детей и среднемесячный доход выше \$2000.

Задачи, решаемые методами Data Mining:

Классификация – отнесение объектов (наблюдений, событий) к одному из заранее известных классов. Это делается посредством анализа уже классифицированных объектов и формулирования некоторого набора правил.

Классификация используется в случае, если заранее известны классы отнесения объектов. Например, отнесение нового товара к той или иной товарной группе, отнесение клиента к какой-либо категории. При кредитовании это может быть, например, отнесение клиента по каким-то признакам к одной из групп риска.

Кластеризация – это группировка объектов (наблюдений, событий) на основе данных (свойств), описывающих сущность объектов. Причем в отличие от классификации, группы заранее не заданы. Объекты внутри кластера должны быть «похожими» друг на друга и отличаться от объектов, вошедших в другие кластеры. Чем больше похожи объекты внутри кластера и чем больше отличий между кластерами, тем точнее кластеризация. Часто применительно к экономическим задачам вместо кластеризации употребляют термин *сегментация*.

Кластеризация может использоваться для сегментации и построения профилей клиентов (покупателей). При достаточно большом количестве клиентов становится трудно подходить к каждому индивидуально. Поэтому клиентов удобно объединить в группы – сегменты с однородными признаками. Выделять сегменты клиентов можно по нескольким группам признаков. Это могут быть сегменты по сфере деятельности, по географическому расположению. После сегментации можно узнать, какие именно сегменты являются наиболее активными, какие приносят наибольшую прибыль, выделить характерные для них признаки. Эффективность работы с клиентами повышается за счет учета их персональных предпочтений.

Прогнозирование (регрессия). Это установление зависимости непрерывных выходных переменных от входных. К этому же типу задач относится прогнозирование временного ряда на основе исторических данных.

Регрессия используется для установления зависимостей в факторах. Например, в задаче прогнозирования зависимой величиной является объемы продаж, а факторами, влияющими на эту величину, могут быть предыдущие объемы продаж, изменение курса валют, активность конкурентов и т.д. Или, например, при кредитовании физических лиц вероятность возврата кредита зависит от личных характеристик человека, сферы его деятельности, наличия имущества.

Ассоциация – выявление закономерностей между связанными событиями. Примером такой закономерности служит правило, указывающее, что из события X следует событие Y. Такие правила называются ассоциативными.

Впервые это задача была предложена для нахождения типичных шаблонов покупок, совершаемых в супермаркетах, поэтому иногда ее еще называют анализом рыночной корзины (market basket analysis). Ассоциации помогают выявлять совместно приобретаемые товары. Это может быть полезно для более удобного размещения товара на прилавках, стимулирования продаж. Тогда человек, купивший пачку спагетти, не забудет купить к ним бутылочку соуса.

Последовательность (последовательные шаблоны) – установление закономерностей между связанными во времени событиями.

Последовательные шаблоны могут быть использованы при планировании продаж или предоставлении услуг. Пример последовательного шаблона: если человек приобрел фотопленку, то через неделю он отдаст ее на проявку и закажет печать фотографий.

Методы в Data Mining.

Для решения вышеперечисленных задач используются различные методы и алгоритмы Data Mining. Ввиду того, что Data Mining развивался и развивается на стыке таких дисциплин, как математика, статистика, теория информации, машинное обучение, теория баз данных, вполне закономерно, что большинство алгоритмов и методов Data Mining были разработаны на основе различных методов из этих дисциплин.

На сегодня наибольшее распространение получили самообучающиеся методы и машинное обучение. Рассмотрим кратко наиболее известные алгоритмы и методы, применяющиеся для решения каждой задачи Data Mining.

Деревья решений. Деревья решений (decision trees) предназначены для решения задач классификации. Они создают иерархическую структуру классифицирующих правил типа «ЕСЛИ...ТО...» (if-then), имеющую вид дерева. Дерево решений состоит из узлов (рис.4.2), где производится проверка условия, и листьев – конечных узлов дерева, указывающих на класс (узлов решения).

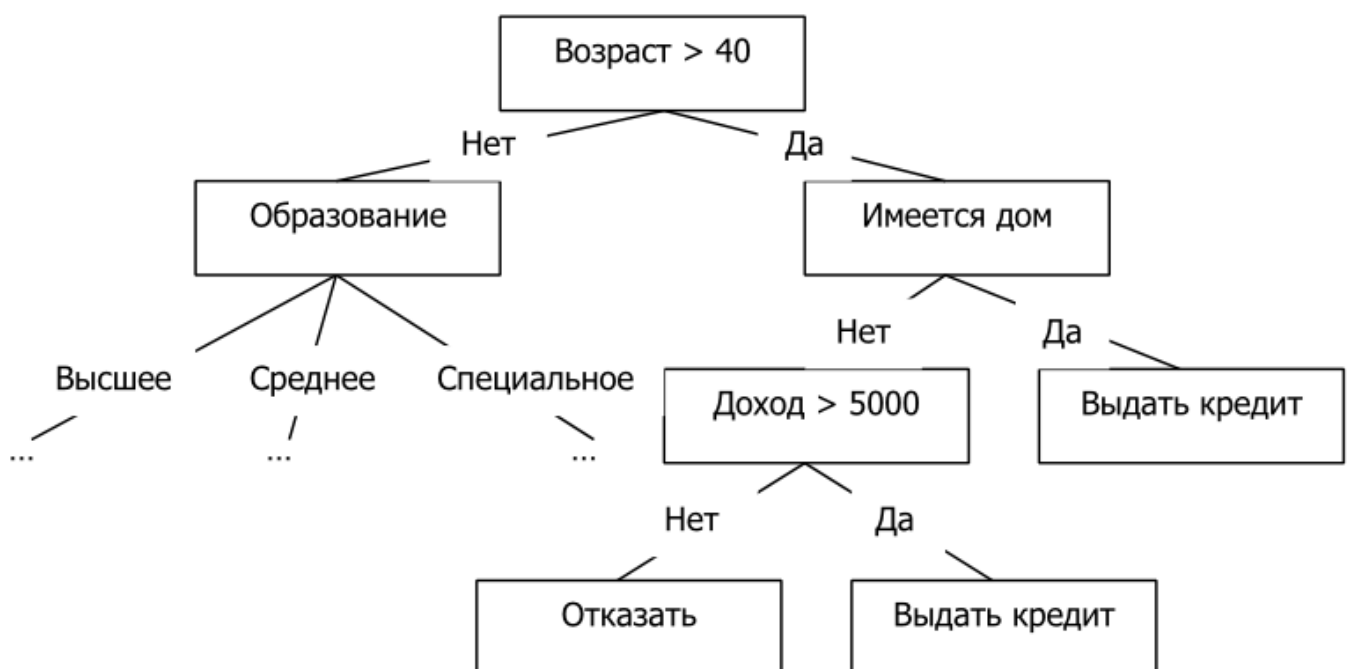


Рис. 5. Пример дерева решений.

Дерево решений строится по определенному алгоритму. Наибольшее распространение получили алгоритмы CART и C4.5(C5.0).

Искусственные нейронные сети (ИНС). Искусственные нейронные сети, в частности, многослойный персептрон, решают задачи регрессии и классификации. Однако, в отличие от дерева решений, нейронные сети не способны объяснять выдаваемое решение, поэтому их работа напоминает «черный ящик» со входами и выходами. Нейронные сети моделируют простые биологические процессы, аналогичные процессам, происходящим в человеческом мозге. ИНС способны к адаптивному обучению путем реакции на положительные и отрицательные воздействия. В основе их построения лежит элементарный преобразователь, называемый *искусственным нейроном* или просто *нейроном* по аналогии с его биологическим прототипом.

Структуру нейросети – многослойного персептрона (рис.4.3) - можно описать следующим образом. Нейросеть состоит из нескольких слоев: входной, внутренний (скрытый) и выходной слои. Входной слой реализует связь с входными данными, выходной – с выходными. Внутренних слоев может быть от одного и больше. В каждом слое содержится несколько нейронов. Все нейроны соединяются между собой связями, называемые *весами*.

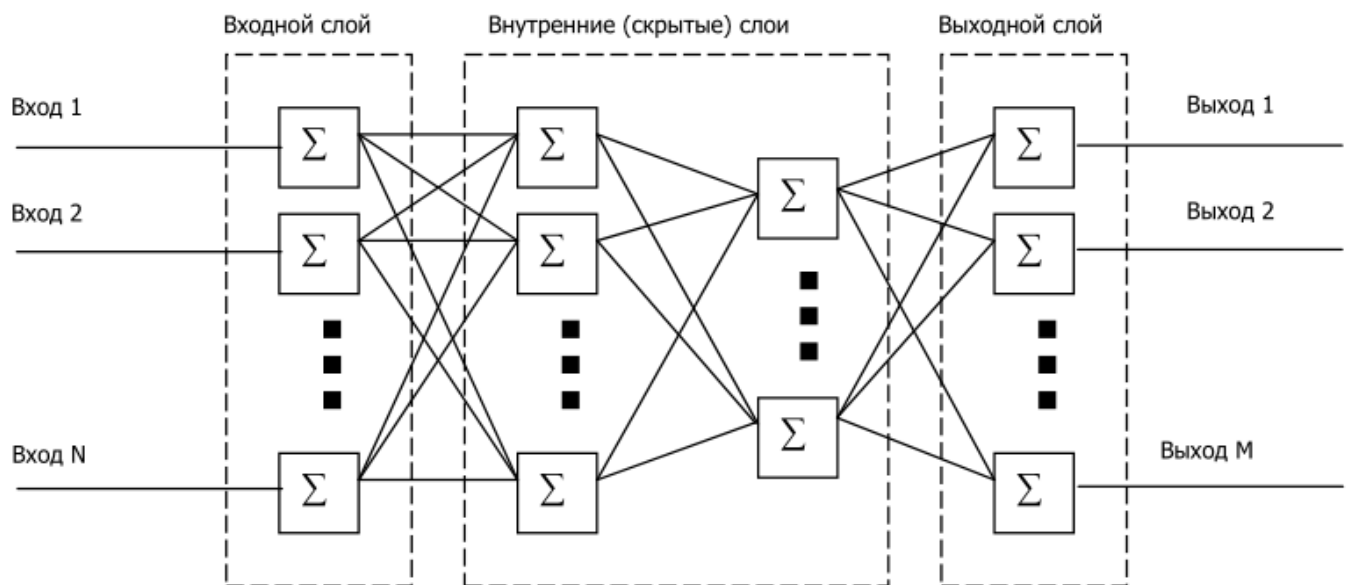


Рис.6. Многослойный персептрон.

Перед использованием нейронной сети производится ее обучение, что представляет собой итерационный процесс настройки весовых коэффициентов. Для обучения применяются специальные алгоритмы. Наибольшее распространение получили градиентные методы обучения – алгоритм обратного распространения ошибки (Back Propagation), сопряженных градиентов, RProp и другие. Для проверки адекватности построенной нейронной сети используется специальный прием - тестовое подтверждение.

Основное достоинство нейронных сетей состоит в том, что они моделируют сложные нелинейные зависимости между входными и выходными переменными.

Линейная регрессия. Линейная регрессия, как это следует из названия, решает задачи регрессии. Но она предназначена для поиска линейных зависимостей в данных. Если же зависимости нелинейные, то модель с использованием линейной регрессии может быть не построена вообще. Для этого лучше воспользоваться более универсальным методом нахождения зависимостей, например, искусственной нейронной сетью.

Кластерный анализ. Главное назначение кластерного анализа – разбиение множества исследуемых объектов и признаков на однородные кластеры (группы). Большое достоинство кластерного анализа в том, что он позволяет производить разбиение объектов не по одному параметру, а по целому набору признаков. Кроме того, кластерный анализ в отличие от большинства математико-статистических методов не накладывает никаких ограничений на вид рассматриваемых объектов, и позволяет рассматривать множество исходных данных практически произвольной природы.

Самоорганизующиеся карты. Самоорганизующиеся карты (Self Organizing Maps – SOM), или карты Кохонена, так же как и методы кластерного анализа, используются для решения задач кластеризации и сегментирования. Самоорганизующаяся карта является разновидностью нейронной сети.

Ассоциативные правила. Ассоциативные правила (association rules) позволяют находить закономерности между связанными событиями, т.е. выявление ассоциаций. Примером ассоциативного правила, служит утверждение, что покупатель, приобретающий хлеб, приобретет и молоко с вероятностью 75%. Впервые эта задача была предложена для поиска ассоциативных правил для нахождения типичных шаблонов покупок, совершаемых в супермаркетах, поэтому иногда ее еще называют анализом рыночной корзины (market basket analysis). Ассоциативные правила эффективно используются в сегментации покупателей по поведению при покупках, анализе предпочтений клиентов, планировании расположения товаров в супермаркетах, адресной рассылке. Классическим алгоритмом нахождения ассоциативных правил считается алгоритм APriori.

Последовательные шаблоны. Последовательные шаблоны (sequential patterns) представляют собой закономерности между связанными во времени событиями. Алгоритмы последовательных шаблонов похожи на алгоритмы ассоциативных правил. Распространение получили AprioriAll и AprioriSome.

Вопросы для самоконтроля и текущего контроля

- 1) Дайте определение понятию "знание"
- 2) Перечислите виды знаний
- 3) Опишите назначение баз знаний в интеллектуальных системах
- 4) Перечислите виды баз знаний
- 5) Перечислите основные модели представления знаний
- 6) Опишите логическую модель представления знаний
- 7) Опишите продукционную модель представления знаний
- 8) Опишите семантическую модель представления знаний
- 9) Опишите фреймовую модель представления знаний
- 10) Дайте определение задаче анализа данных
- 11) Опишите способы извлечения знаний из данных

Задание на самостоятельную работу

- 1) Практическое задание

Выполнить создание спецификации знаний о предметной области в виде онтологии в редакторе Protege 5.5.0 согласно заданию практической работы №4.

- 2) Самостоятельная работа

Исследовать синтаксис и лексику современных языков создания онтологий.

Список рекомендуемой литературы¹:

[1] Сидоркина, Ирина Геннадьевна. Системы искусственного интеллекта [Текст] : [учеб. пособие для вузов по направлению 230100 "Информатика и вычисл. техника"] / И. Г. Сидоркина. М.: Кнорус, 2011. - 245 с.

Рассмотрены теоретические аспекты проектирования систем искусственного интеллекта и даны модели представления знаний, изложены материалы информационного, справочного, консультирующего характера по использованию инструментальных средств и технологическим особенностям разработки систем данного класса.

Подробнее: <https://www.labyrinth.ru/books/276217/>

[2] Богомолова, М. А. Экспертные системы (техника и технология проектирования) [Текст] : Методические указания к лабораторным работам / М. А. Богомолова. Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015. - 47 с.

Методические указания являются руководством при выполнении лабораторных занятий, проводимых по курсу «Экспертные системы (техника и технология проектирования)» с магистрами направления 09.04.03 - «Прикладная информатика» в терминальном классе. Лабораторные работы связаны с изучением принципов построения экспертных систем на основе различных моделей представления знаний, и могут быть полезны студентам других специальностей.

Подробнее: <https://www.iprbookshop.ru/71908.html>

[3] Советов, Борис Яковлевич. Представление знаний в информационных системах [Текст] : [учеб. для вузов по направлению подгот. "Информ. системы и технологии"] / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. М.: Академия, 2011.

Учебник создан в соответствии с Федеральным государственным образовательным стандартом по направлению бакалавриата «Информационные системы и технологии». Рассмотрены современное состояние теоретических и прикладных вопросов представления знаний в информационных системах, идеология построения интеллектуальных систем, математический аппарат представления знаний, возможности и пути использования искусственного интеллекта при проектировании информационных систем. Изложены новые аспекты представления знаний на основе искусственных нейронных сетей, расчетно-логических систем, генетических алгоритмов, мультиагентных систем. Приведены примеры практической реализации представления знаний на базе декларативного языка Пролог. Для студентов учреждений высшего профессионального образования. Может быть полезен разработчикам и пользователям информационных систем; преподавателям и научным сотрудникам, сферой интересов которых является интеллектуализация различных предметных областей; менеджерам и руководителям различного ранга, желающим самостоятельно ознакомиться с современным состоянием информационных технологий.

Подробнее: <https://lavkababuin.com/predstavlenie-znaniy-v-informacionnyh-sistemah-2-e-izd-ster-650564/>

[4] Остроух, А. В. Интеллектуальные информационные системы и технологии [Электронный ресурс] : монография / А. В. Остроух, А. Б. Николаев. Санкт-Петербург: Лань, 2019. - 308 с.

В монографии изложены концептуальные основы и методы представления знаний в интеллектуальных системах. Рассмотрены различные подходы, применяемые при проектировании и разработке интеллектуальных систем и технологий в транспортном комплексе, а также рассмотрены тенденции развития систем искусственного интеллекта.

Монография может быть использована для формирования профессиональной компетентности студентов высших учебных заведений, аспирантов и научных сотрудников обучающихся и ведущих научные исследования в области разработки и практического применения систем искусственного интеллекта по укрупнённой группе направлений подготовки "Информатика и вычислительная техника".

Подробнее: <https://www.labirint.ru/books/692861/>

Ссылки на открытые ресурсы (он-лайн-курсы, видеоуроки и т.д.)

[5]

Прикладные интеллектуальные системы

Конспект лекции 9. Онтологические модели представления знаний о предметной области. Применение онтологий в решении профессиональных задач

Аннотация: В лекции рассказывается о понятии онтологической модели представления знаний.

План лекции:

1. Определение понятия «онтология»
2. Структура онтологии
3. Классификация онтологий
4. Методы построения онтологий
5. Применение онтологий
6. Языки представления (Semantic web)
7. Существующие онтологические ресурсы
8. Задания для самостоятельной работы.

Ключевые слова: онтология, онтологические модели, классы, фреймы, понятия, атрибуты, отношения, экземпляры, структура онтологии, классификация онтологий, языки представления онтологий

СОДЕРЖАНИЕ ЛЕКЦИИ:

1. Определение понятия «онтология»

Развитие наукоемких областей человеческой деятельности в современном обществе сопровождается возрастанием роли компьютерных технологий. Сейчас значительно увеличивается поток информации, появилась необходимость поиска новых способов ее хранения, представления, формализации и систематизации, а также автоматической обработки. Таким образом, растет интерес к всеобъемлющим базам знаний, которые возможно использовать для различных практических целей. Огромный интерес вызывают системы, способные без участия человека извлечь какие-либо сведения из текста. Как результат, на фоне вновь возникающих потребностей развиваются новые технологии, призванные решить заявленные проблемы. Наряду с World Wide Web появляется его расширение, Semantic Web, в котором гипертекстовые страницы снабжаются дополнительной разметкой, несущей сведения о семантике включаемых в страницы элементов. Неотъемлемым компонентом Semantic Web является понятие онтологии, описывающее смысл семантической разметки. В общих чертах под онтологией понимается система понятий некоторой предметной области, которая представляется как набор сущностей, соединенных различными отношениями (подробнее см. раздел 1.1). Онтологии используются для формальной спецификации понятий и отношений, которые характеризуют определенную область знаний. Преимуществом онтологий в качестве способа представления знаний является их формальная структура, которая упрощает их компьютерную обработку. Можно говорить о неявном применении онтологий в качестве систем понятий в естественных науках (биология, медицина, геология и другие), где они служат своего рода фундаментом для построения теорий. Поскольку классификационная структура (таксономия) является неотъемлемой частью любой онтологии, можно говорить о присутствии элементов онтологий в специальных классификациях и системах индексации (например, в библиотечных классификационных кодах). В явном виде онтологии используются как источники данных для многих компьютерных приложений (для информационного поиска, анализа текстов, извлечения знаний и в других информационных технологиях), позволяя более эффективно обрабатывать сложную и разнообразную информацию. Этот способ представления знаний позволяет приложениям распознавать те семантические отличия, которые являются само собой разумеющимися для людей, но не известны компьютеру. Само понятие онтологии известно давно, но, будучи переосмысленным, оно стало применяться в компьютерных технологиях лишь недавно. Полноценная разработка онтологий в новом смысле этого термина началось лишь в конце 90-х. Это достаточно новая и мало разработанная отрасль прикладной лингвистики. Большинство работ по созданию и использованию онтологий проводится за рубежом, однако и в России существует ряд исследователей, работающих в этой области, см. также работы по онтологиям, опубликованные в России ([1], [3], [4], [6] и др.). Уже на данном этапе

создан ряд обширных онтологий, включающих несколько тысяч понятий: OMEGA, SUMO, DOLCE и другие.

Онтологии широко используются во всех областях, занимающихся обработкой данных на естественном языке. В связи с использованием онтологий в различных приложениях возникла необходимость создания стандартизированных способов их представления. Началось развитие разнообразных языков, которые могли бы применяться повсеместно во всех системах, самыми известными являются RDF и OWL. Возникло также большое количество редакторов для создания, пополнения и изменения онтологий. Каждое из этих средств обычно направлено на работу с определенным форматом данных и обладает своими особенностями.

и других сущностей и отношений, связывающих их друг с другом. Можно сказать, что концептуализация – это абстрактное, упрощенное видение мира, который мы хотим представить для каких-то целей. Таким образом, концептуализация расчленяет какую-либо область знаний, существующую в целостном виде, выделяет из этой области отдельные объекты, а затем формулирует отношения, свойственные для данной области. Основная часть формально представленного знания базируется на концептуализации, каждая база знаний или система, основанная на знаниях, явно или неявно связывается с какой-то концептуализацией.

Однако существует две различные трактовки природы концептуализации, их можно охарактеризовать, как интенциональную и экстенциональную. Экстенциональная трактовка, которой придерживаются авторы работы, подразумевает, что каждое понятие и отношение может исчерпывающе описываться перечислением индивидуальных сущностей, к которым оно применимо. Н.Гуарино же считает эту точку зрения узкой и обобщает ее, развивая интенциональный подход. Он предлагает идентифицировать понятия не через их перечисление, то есть экстенционал, а через их внутренние свойства и характеристики, так называемое «предполагаемое содержание». Этот универсальный подход дает специалисту возможность подводить под одну и ту же концептуализацию разные положения вещей. Таким образом, концептуализация становится относительно независимой от индивидуальных сущностей, необходимым является лишь сохранение заданных типов отношений. Важно внутреннее содержание понятий, а не перечисление соответствующих конкретных индивидов.

Однако приведенное ранее определение, связанное с понятием «концептуализации», является далеко не единственным. В литературе можно также найти определение онтологии как «теории того, какие сущности могут существовать в уме хорошо осведомленного (knowledgeable) агента». Данное определение выявляет другой подход к этому понятию. Эта формулировка позволяет включать в онтологию набор понятий, но не дает возможность задавать их структуру. Этот пробел в определении особенно значим, так как в рамках искусственного интеллекта в качестве синонима онтологии часто используется понятие «терминология», а в ней структура безусловно содержится.

Объединением выше упомянутых дефиниций становится определение из работы , вводящее «онтологию», как «спецификация концептуализации на уровне эксплицитных знаний, зависящее от предметной области или задачи, для которой она предназначена». Таким образом, онтология зависит от определенной точки зрения, однако, как упоминает Н.Гуарино, как раз степень этой зависимости является определяющим фактором для возможности ее многократного использования. А ведь именно в возможности многократного использования онтологий видится их значимость и ценность.

В других определениях, приводимых Н.Гуарино, делается упор на иной аспект онтологий, и они определяются как соглашения о совместно используемых концептуализациях. При этом поясняется, что эти совместные концептуализации включают в себя понятийные структуры для моделирования знания какой-то предметной области. Это своего рода соглашение, какие схемы и теории использовать при описании предметной области. Таким образом, здесь проводится разграничение понятий «онтология» и «концептуализация». Об онтологии говорится уже не как о спецификации концептуализации, а лишь как о соглашении о концептуализации. Однако степень детализации этого соглашения будет напрямую зависеть от предназначения конкретной онтологии и целей, поставленных перед исследователем.

Н. Гуарино упоминает еще один подход к определению «онтологии»: «Онтология – это конкретный артефакт, созданный для выражения значений, подразумеваемых у совместно используемой лексики». Здесь упоминается, что онтология предоставляет средства для передачи подразумеваемого значения. Н.Гуарино предлагает также определять онтологию, как «логическую теорию, которая ограничивает подразумеваемые модели логического языка». Таким образом, под онтологией понимается нечто большее, чем просто детализированный набор понятий и отношений. В онтологию включаются и ограничения, накладываемые на отношения в рамках данной области. Это некоторый набор аксиом, который строится на базе понятий и отношений между ними. Таким образом, например, в рамках искусственного интеллекта можно описать онтологию программы, определив множество объектов и связав их с описаниями, а также введя формальные аксиомы, которые ограничивают интерпретацию и совместное употребление этих терминов. Формально онтологию можно назвать формулировкой логической теории, некоего исчисления со своими правилами. Эта теория позволяет систематизировать категории действительности и/или выражаемые в языке значения. Следовательно, такое определение онтологии можно считать более широким взглядом на данное понятие, нежели предыдущие.

В качестве рабочего определения, наиболее приспособленного для целей компьютерной лингвистики, можно взять дефиницию, предложенную Эдвардом Хови : «Онтология – это структура данных с заданными в ней символами, позволяющими представлять концептуализации для обработки компьютерными программами».

Рассмотрев оба понимая значения термина «онтология», вводимые в философии и в инженерной области знания, можно обнаружить такое соотношение двух значений, как «процесс-результат». В философии «онтология» - это наука, изучающая бытие, а «онтология» в инженерии – это отображение бытия в формализованном виде. Однако С. Ниренбург и В. Раскин указывают на то, что предмет описания формальной онтологии соотносится внутри философии скорее не с наукой о бытии (онтологией, или метафизикой), а с наукой о познании («гносеологией»). Для формальной онтологии важны не столько сами понятия, сколько использование их людьми, знания людей о данных понятиях. В связи с этим не встает, например, вопрос, правомерно ли включать в формальные онтологии несуществующие сущности (например, вымышленные существа). Метафизика будет утверждать, что в бытии такого не существует, однако, обратившись к гносеологии, мы обнаружим, что это существует в умах людей и на данном основании может включаться в онтологию.

Таким образом, следует заключить, что четкой взаимообусловленности между двумя значениями термина «онтология» – в философии и в инженерии знаний – не прослеживается. Связь между ними носит скорее произвольный ассоциативный характер и не будет обсуждаться далее в нашем исследовании.

2. Структура онтологии

Рассмотрев возможные содержательные интерпретации понятия «онтология», остановимся подробнее на структуре онтологии, ее составляющих. В общем виде структура онтологии представляет собой набор элементов четырех категорий:

- понятия;
- отношения;
- аксиомы;
- отдельные экземпляры;

Понятия рассматриваются как концептуализации класса всех представителей некой сущности или явления (например, Животное, Чувство). Классы (или понятия) являются общими категориями, которые могут быть упорядочены иерархически. Каждый класс описывает группу индивидуальных сущностей, которые объединены на основании наличия общих свойств.

Понятия могут быть связаны различного рода отношениями (например, Длина, Местоположение), которые связывают воедино классы и описывают их. Самым распространенным типом отношений, использующимся во всех онтологиях, является отношение категоризации, то есть отнесение к определенной категории. Этот тип отношений имеет ряд других названий [3], встречающийся в различных исследованиях:

- таксономическое отношение;
- отношение IS-A;
- класс – подкласс;
- лингвистика: гипоним – гипероним;
- родовидовое отношение;

- отношение a-kind-of.

Аксиомы задают условия соотнесения категорий и отношений, они выражают очевидные утверждения, связывающие понятия и отношения. Под аксиомой можно понимать утверждение, вводимое в онтологию в готовом виде, из которого могут быть выведены другие утверждения. Они позволяют выразить ту информацию, которая не может быть отражена в онтологии посредством построения иерархии понятий и установки различных отношений между понятиями. В качестве примера аксиомы можно привести следующее высказывание: «Если X смертен, то X когда-нибудь умрет». Аксиомы позволяют в дальнейшем осуществлять умозаключения в рамках онтологии. Они могут снабжать исследователей информацией о правилах, позволяющих автоматически добавлять информацию. Аксиомы могут также представлять собой ограничения, накладываемые на какие-либо отношения, делающие возможным выведение умозаключений. Приведем несколько примеров таких ограничений. Понятийные ограничения указывают на то, какой тип понятий может выражать данное отношение (например, свойство Цвет может выражаться только понятиями категории Цвет). Примером числовых ограничений является утверждение того, что для Человека количество биологических родителей равно 2. Количество и степень детализации аксиом обычно зависят от типа онтологии, о чем будет подробнее сказано далее. Наряду с указанными элементами онтологии в нее также входят так называемые «экземпляры». В литературе они могут выступать также под названиями:

- конкретные экземпляры;
- инстанции;
- индивидуальные экземпляры.

Экземпляры – это отдельные представители класса сущностей или явлений, то есть конкретные элементы какой-либо категории (например, экземпляром класса Человек будет королева Виктория).

Составляющие онтологии подчиняются своеобразной иерархии. На нижнем уровне этой иерархической лестницы находятся экземпляры, конкретные индивиды, выше идут понятия, то есть категории. На уровень выше располагаются отношения между этими понятиями, а обобщающей и связующей является ступень правил или аксиом. Как упомянуто в работах [3] и [5], «термину «онтология» удовлетворяет широкий спектр структур, представляющих знания о той или иной предметной области». Так к онтологиям можно отнести ряд структур, отличающихся разной степенью формализованности:

- глоссарий;

простая таксономия;

- тезаурус (таксономия с терминами);
- понятийная структура с произвольным набором отношений;
- полностью аксиоматизированная теория.

Однако в этих структурах не всегда представлены все составляющие онтологии, которые описывались в данном разделе.

3. Классификация онтологий

Онтологии сильно различаются по ряду параметров, и исследователи выделяют различные основания для их классификации. Так Э. Хови говорит, что онтологии различаются в зависимости от набора элементов, содержащихся в них, а также типов вводимых отношений. Он выделяет так называемые «терминологические онтологии» и «настоящие онтологии». Под первыми Э.Хови понимает онтологии, включающие сущности, явления, свойства, связи предметной области и объединяющие их структурные отношения. «Настоящие» же онтологии включают в себя также дефиниционные отношения и отношения дополнительной информации. Наряду с этим в них входят аксиомы, определяющие взаимозависимости между отношениями и понятиями.

Э.Хови выстраивает подробную классификацию различных характеристик онтологий. Он упоминает, что основными параметрами могут быть: форма (то, как формируется онтология), содержание, а также средства использования онтологий.

Можно упомянуть о том, что существует разбиение онтологий по количеству и качеству понятий, включаемых в них. Онтологии верхней зоны обычно насчитывают примерно 100-500 концептов. В них включены наиболее абстрактные категории, обладающие свойством универсальности. Они являются базовым разбиением наблюдаемой действительности на категории. Обычно они строятся теоретиками, философами. Зачастую концепты даже не лексикализуются. Составление аксиом в данном типе онтологий с высоким уровнем обобщения достаточно сложно и требует некоторого воображения. Преимуществом таких онтологий является возможность их использования во многих областях и даже во многих языках. Для данного рода онтологий характерен ограниченный набор обобщенных отношений, которые можно отнести к базовым (таких как родовидовые отношения, отношения часть-целое и ассоциативные отношения). В таких онтологиях типичными на верхнем уровне разбиения являются такие понятия, как:

- сущность;
- явление;

объект;

- процесс;
- роль [3].

Однако этот типичный набор может быть представлен в усеченном виде, например, в онтологии MicroKosmos, разработанной С. Ниренбургом и В. Раскиным

[26] на верхнем уровне появляются лишь три категории понятий: «объект», «процесс» и «роль». При этом авторы претендуют на универсальность этого разбиения для онтологической семантики в целом и, таким образом, для всех онтологий верхнего уровня.

Другим типом являются онтологии средней зоны, здесь элементов обычно уже больше (500 – 100000 концептов). Они представляют мир в целом и в общем случае это неаксиоматизированная область. Сложность заключается в том, что для данного вида онтологий требуется выводить слишком большое количество аксиом. Обычно

выходом является использование методов автоматизированного вывода аксиом из уже существующих онтологий. Построением онтологий этого уровня чаще всего занимаются когнитологи и лингвисты.

Онтологии нижней зоны или так называемые онтологии предметной области наиболее обширные, обычно они насчитывают около 200 – 2 000 концептов. Они описывают конкретные предметные области с их спецификой. При этом круг решаемых задач и вопросов, на которые онтология отвечает, ограничен выбранной областью. Для данного типа онтологий характерно наличие отношений, специфичных для конкретной области [3]. Это высокоаксиоматизированная зона, то есть для нее возможно построение большого количества аксиом и правил. В большинстве случаев этот тип онтологий строится экспертами области знания или при их содействии. В связи с большой спецификой каждой отдельной предметной онтологии ее повторное использование зачастую возможно только в рамках предметной области.

Наряду с описанным делением все онтологии могут быть разделены на глубинные и поверхностные. Поверхностные онтологии строятся на поверхностной семантике, они определяют понятия через значения слов. Однако здесь возникает проблема, какое количество смыслов выделять для каждого слова. Глубинные же онтологии используют глубинную семантику.

Б. Й. Вилинга и А. Т. Шрайбер выделяют два измерения для оценки онтологий: «объем и тип структуры концептуализации и предмет концептуализации». Однако, как указывает Н. Гуарино, критерий для выделения в рамках первого измерения информационных, терминологических онтологий и онтологий, моделирующих знания, не отличается четкостью. Проще разделить онтологии по степени детальности, используемой для характеристики концептуализации. Очень детализированная онтология подробно специализирует подразумеваемую концептуализацию, но платой за это оказывается более громоздкий язык, который может быть сложно применять на практике. Простая же онтология может развиваться с допущением каких-то скрытых условий, которые подразумеваются создателями, и ее могут использовать те, кто уже договорился о лежащей в основе концептуализации (и осознает эти допущения). Можно различать «отсылочные (также называемые офф-лайн) онтологии» (reference ontologies) и «осуществляемые (совместно используемые, он-лайн) онтологии» (implemented (shareable) ontologies). Несложная структура, описывающая, например, лексикон, может помещаться он-лайн, в то время как замысловатые теории, определяющие значение терминов из лексикона, могут находиться офф-лайн. С точки зрения предмета концептуализации исследователи выделяют прикладные онтологии, онтологии области знания, общие (родовые) онтологии и репрезентационные онтологии (речь идет об онтологиях метауровня, включающих в себя репрезентационные первоэлементы).

Онтологии могут быть также разделены на одноязычные и многоязычные. Уже существует ряд онтологий, ориентированных на представление знаний на нескольких языках, например, EuroWordNet, MikroKosmos и некоторые другие. Сложность создания таких онтологий обычно заключается в том, что возможно

наличие различий в понятийных системах разных языков.

В рамках работы [3] также выделяется особый тип онтологий – лексические (или лингвистические). Отличительным свойством таких онтологий является «фиксация в одном ресурсе (лексикализованных) понятий (слов) вместе с их языковыми свойствами» [3]. Такие онтологии тесно взаимосвязаны с семантикой грамматических элементов (слов, именных групп и др.). Основным источником понятий в онтологиях данного типа являются значения языковых единиц. Их также отличает своеобразный

набор отношений, обычно свойственный для языковых элементов: синонимия, гипонимия, меронимия, а также ряд других. К лингвистическим онтологиям авторы [3] относят WordNet, MikroKosmos, Sensus, PyТез и другие. Круг задач, решаемых такими онтологиями, тесно взаимосвязан с обработкой естественного языка.

С.А.Коваль [8] предлагает различать безэкземплярные и экземплярные онтологии. Как понятно из названия данного типа онтологий, безэкземплярные онтологии отличаются отсутствием конкретных экземпляров. На нижних уровнях иерархии таких онтологий находятся не конкретные экземпляры, а понятия. Эта особенность онтологии накладывает некоторый отпечаток и на вводимые в данной онтологии отношения.

Таким образом, существует множество подразделений онтологий, но эти классификации не всегда бывают достаточно четкими и последовательными. Создаваемая нами онтология относилась к онтологиям нижней зоны и описывала конкретную предметную подобласть. В связи с этим для нашей онтологии были характерны некоторые отношения, которые являются специфичными для данной области, и мала вероятность, что могут использоваться в других областях. Она также являлась безэкземлярной, что наложило свой отпечаток на установление отношений между элементами, однако об этом будет упомянуто подробнее в разделе 2.5. На данном этапе создания можно говорить об одноязычности нашей онтологии, хотя для некоторых элементов уже была сделана попытка связать русские языковые выражения с английскими.

4. Методы построения онтологий

4.1 Принятие исходных решений

Определившись с понятием «онтологии» и ее структурой, перейдем к обсуждению методов их построения. Говоря в общих чертах, для создания онтологии надо сначала перечислить категории, обозначающие сущности или явления в моделируемой области. Затем следует связать эти категории определенными отношениями. И на последнем шаге надо соотнести категориям набор конкретных экземпляров. Но это лишь общий, упрощенный алгоритм, а в реальности этот процесс противоречив и рождает много дискуссий.

Так, ряд решений должен быть принят уже на начальных этапах создания онтологий. Надо определить, создавать новый элемент и должен ли этот элемент быть включен в структуру. Следует понять, где по отношению к другим объектам должен

располагаться вновь создаваемый, должен ли он быть видом какого-либо класса или же сам представляет собой родовой термин. Помощь в этом может оказать формулировка особых, уникальных свойств термина, то есть его отличительных характеристик. При этом не следует смешивать свойства понятия и его отличительные признаки.

При формировании онтологий могут привлекаться специалисты различных областей, и для каждой области есть свои базовые методы работы. Так, философы используют абстракцию и комбинирование свойств, например, по Аристотелю следует выделять понятия как атомарные понятия, строящиеся из набора дифференциальных признаков. Когнитивисты склонны полагаться на интуитивные отличия. Например, Э.Рош полагает, что функциональным предназначением классов является предоставление максимума информации при минимуме когнитивных усилий. Считая, что люди склонны формировать классы на основе прототипов, она вводит такие классы предлагает в дополнение к родовидовым иерархическим системам или вместо них. Специалисты в компьютерной области оперируют логическими теориями и используют структуры, построенные на умозаклучениях. Лингвисты уделяют большое внимание описаниям межъязыковых соответствий. Общим же методом для всех является использование классификаций.

Однако при реализации упомянутых ранее в общих чертах методах построения онтологии может возникнуть ряд проблем. Так при составлении классификаций существует возможность появления неоднозначностей в зависимости от того, какому дифференциальному признаку отдавать предпочтение. Можно привести пример разбиения понятия «человек», предлагаемый Э.Хови . С одной стороны, можно сначала использовать отличительный признак «пол», выделив понятия «лица мужского пола» и «лица женского пола», а затем «возраст», выделив «мужчина» и «мальчик», а также «женщина» и «девочка». Но, с другой стороны, в равной степени правомерным будет разделить понятие «человек» на «взрослые» и «дети», а потом уже выделить пару «мужчина» и «женщина» и пару «мальчик» и «девочка». В данном случае проблема заключается в неопределенности иерархии отличительных признаков.

Выбор того или иного решения упомянутых ранее проблем порой достаточно сложно обосновать. В данном случае нет авторитетов, это может быть данью традиции, общественным соглашением или просто подстройка под определенные задачи. Самым важным здесь является выбрать какой-либо подход и придерживаться его на протяжении всей работы, поэтому стоит задуматься над рядом проблем и возможными универсальными методами их устранения еще до начала принятия решений.

При создании концептов исследователи сталкиваются еще с одной принципиальной проблемой, так называемой «проблемой примитивности». Э.Хови указывает на то, что почти вся семантика и представление знаний основываются на композиционной гипотезе: можно определить ограниченный набор единичных сущностей («атомов»), а все остальные («молекулы») представлять как комбинацию (композицию) атомов. При этом возникает вопрос: как много таких «атомов» необходимо? В современных исследованиях можно найти два различных подхода: экономный и неэкономный.

Экономный подход призывает создавать малое количество элементарных концептов, семантически простых, с помощью которых можно объяснить значение более сложных понятий. При таком положении вещей легко обнаружить связанность понятий, просто определять и осуществлять умозаключения, однако достаточно сложно составлять сложные значения. Примером такой точки зрения является исследование А.Вежбицкой, автора теории семантических примитивов и основанного на ней естественного метаязыка описания семантики. Семантические примитивы – это лексические единицы, выражающие элементарные, базовые значения. Количество таких единиц не превышает 100, и их значения универсальны для всех языков. Неэкономный подход позволяет создавать любое количество индивидуальных сущностей – столько, сколько захочется создателю онтологии. Это количество может варьироваться от 10 до 100 000 и более. Такая точка зрения свойственна, например, создателям WordNet. Здесь затруднительно определять связанность понятий, сложно работать с умозаключениями. Данный подход сопровождается, по существу, отказом от композиционной гипотезы, но это влечет за собой и преимущество: отсутствие необходимости составлять сложные значения. В данной проблеме нет единственно верного решения: все зависит от того, как много высказываний необходимо или как сложна исследуемая предметная область. Подытоживая, следует сказать, что современная практика показывает, что экономного подхода придерживаются в основном формалисты, а пользователи склоняются к неэкономному. Стоит упомянуть, что не всегда построение онтологии проходит очевидным образом, не всегда легко собрать понятия, выделить дифференциальные признаки. Здесь также существует несколько вариантов действий, зависящих от конкретных задач и исходного материала. Существует возможность сбора элементов для онтологии напрямую. При таком подходе сначала собираются и классифицируются понятия, подбираются слова, затем проводится соответствие между понятиями и лексиконом. Проблемой является близкое, но не идентичное пересечение значений в словах разных языков. Вторым вариантом является использование микротеорий. При таком подходе сначала надо понять явление, затем формируются примитивы теории, элементы лексикона определяются с точки зрения примитивов, после этого лексикон усложняется. Проблема данного метода заключается в выборе микротеории, а также в необходимости отдельной теории для каждого комплекса значений. Противопоставление этих двух точек зрения можно проиллюстрировать на предложенном Э.Хови примере, где рассматривается несколько подходов к трактовке понятия «цвет» и различных обозначений цветов. Можно напрямую собрать слова, обозначающие цвет и попробовать их как-нибудь классифицировать. А можно применить микротеоретизирование, понять, что представляет собой явление «цвет». С физической точки зрения цвета задаются длиной волны и интенсивностью, теперь с помощью этих параметров мы можем определить все конкретные значения цвета. Может быть применен и нейрофизиологический подход, представляющий все цвета с точки зрения восприятия их с помощью трех рецепторов.

Достаточно ясно, что в основании онтологии должны лежать понятия, но возникает вопрос: как при построении определить понятия, основываясь на словах? Обычно связующим звеном становится категория значения (лексико-семантического варианта). Слова существуют в рамках одного языка, значения же независимы от конкретного языка. Э.Хови предлагает формальную процедуру выявления понятий на базе совокупности слов. Он предлагает алгоритм перехода от слов к значениям, а затем к понятиям:

1. **Инициализация:** Для данного слова соберите несколько десятков предложений, содержащих его. Подберите определения из различных словарей.
2. Расположите значения слова в предварительные, грубо схожие группы
3. **Процесс дифференциации:** Начните строить дерево, расположив все группы в корне
4. Рассматривая все группы, определите группу, наиболее отличную от других:
 1. Если вы можете найти одну ясно выделяющуюся группу, выпишите ее наиболее яркое отличие в явной форме – оно послужит отличительным признаком и будет формализовано в виде аксиомы.
 2. Если отличия, по которым можно далее подразделить группу, не обнаруживаются, остановите работу с этой ветвью и перейдите на другую ветвь
 3. Если обнаруживается несколько отличий, позволяющих подразделить группу несколькими равнозначными способами, также прекратите работу с этой ветвью и перейдите на другую ветвь.

Создайте в древесной структуре две новые ветви, расположите новую группу под одной ветвью, а остальные под другой.

5. Повторите действия с шага 4, исследуя по отдельности группу/группы под каждой ветвью
6. **Формирование понятий:** Когда ветвление прекращается, конечным результатом является дерево все более дробных отличительных признаков, которые в явном виде перечислены на каждом уровне дерева. Каждый лист становится отдельным понятием, далее не делимым в настоящей задаче (приложении, предметной области). Каждое отличие должно быть формализовано в виде аксиомы, которая срабатывает для ветви, с которой ассоциируется.
7. **Добавление понятия в онтологию:** Начиная с вершины, пройдите каждый узел с ветвлением. Имеют ли уже созданная и добавляемая ветвь примерно одно и то же значение?
 1. Если так, объедините их в онтологии в подходящем узле и остановите прохождение этой ветви
 2. Если нет, разделите дерево и повторите шаг 8 для каждой ветви. Повторяйте, пока не дойдете до конца.

Исследования показывают, что обычно понятий оказывается меньше, чем значений-смыслов.

Однако существуют и другие источники нахождения новых понятий. Для этих целей можно использовать уже существующие онтологии и различные списки, словари и тезаурусы. На данном этапе также может помочь автоматическое выявление понятий путем кластеризации слов (о ней подробнее в раздел 1.4.2).

Достаточно много внимания методике создания онтологий уделяется в статье Н. Ной и Д. МакГиннес. Основополагающие правила разработки онтологии авторы формулируют следующим образом:

- 1) Не существует единственного правильного способа моделирования предметной области – всегда существуют жизнеспособные альтернативы. Лучшее решение почти всегда зависит от предполагаемого приложения и ожидаемых расширений.

Разработка онтологии – это обязательно итеративный процесс.

Под итеративным процессом понимается неоднократный проход по онтологии с целью ее уточнения, то есть на начальном этапе строится черновой вариант. Затем мы проверяем и уточняем составленную онтологию, добавляя детали, возможно, частично или даже полностью пересматривая начальную онтологию.

- 2) Элементы онтологии должны быть близки к объектам (физическим или логическим) и отношениям в интересующей вас предметной области. Скорее всего, они соответствуют существительным (объекты) или глаголам (отношения) в предложениях, которые описывают вашу предметную область.

Все же следует упомянуть, что каким бы образом не решались ранее упомянутые проблемы, эти решения должны быть последовательными. Таким образом, при построении онтологии нам нужны определенные ориентиры для принятия решений.

Целый ряд решений может приниматься на основе практической цели построения онтологии. Однако можно сформулировать и универсальные требования к онтологиям, не зависящие от конкретной задачи. Так, общая структура онтологии должна быть понятной и должна существовать возможность ее многократного использования. Как подчеркивает Н. Гуарино, онтология должна быть когнитивно прозрачной. Ряд требований к онтологии можно найти в работе С. Ниренбурга и В. Раскина:

- Ясность: онтология должна быть ясной и легко передавать подразумеваемый смысл. Она должна быть объективной;
- Последовательность: в ней должны содержаться утверждения, которые не противоречат друг другу, иерархии понятий, связывающим их отношениям, экземплярам.
- Возможность расширения: наличие возможности введения новых элементов без пересмотра остальных элементов;
- Минимальная степень специализации онтологии: нежелательность полного подчинения онтологии конкретной задаче, что может осложнить ее последующее использование в других задачах.

Нельзя утверждать, что этот список требований к онтологиям является исчерпывающим, но он может помочь при принятии тех или иных решений, касающихся структуры онтологии.

Существуют и более формализованные и подробные описания стандартов для онтологий. Так, можно привести пример инициативы EAGLES - Expert Advisory Group on Language Engineering Standards (<http://www.ilc.cnr.it/EAGLES96/home.html>), действовавшую с 1993 года по 1996, и далее сменившую ее ISLE - International Standard for Language Engineering (<http://www.mpi.nl/ISLE/>). Целью данных проектов является разработка универсальных стандартов и рекомендаций для создания языковых ресурсов и программ, обрабатывающих естественный язык. Созданные в рамках EAGLES стандарты уже давно стали общепринятыми и широко распространенными. Существует множество стандартов по языкам представления онтологий, а также по общим правилам создания лингвистических ресурсов для последующей компьютерной обработки. Список таких стандартов (в том числе и для онтологий) можно, например, найти на сайте The Language Technology Resource Center (<http://flrc.mitre.org/References/Standards/>). Использование стандартов должно стать залогом того, что созданный ресурс будет легко внедряться в уже существующие и учитывать все особенности технологии.

4.2 Автоматические методы построения онтологий

Как уже было упомянуто ранее, сейчас развитие онтологий начинает приобретать более массовый характер, и в настоящее время в этой области есть ряд масштабных разработок. Существует большое количество различных списков и баз данных, но возникает вопрос, как гарантировать их соответствие текущему положению вещей, как быть уверенным, что они точны и полны, а также как обеспечить достаточную детальность представляемых данных. В связи с тем, что мир очень быстро изменяется, идет развитие новых отраслей, существующие онтологии требуют постоянного пополнения и усовершенствования. На данном этапе появляются идеи использования автоматических и полуавтоматических методов для не только обновления онтологий, но даже для их создания.

Существует ряд методов расширения онтологий, которые специфичны для онтологий разных зон. Для расширения верхней, наиболее общей зоны необходимо подробное теоретизирование, после чего можно приступать к построению понятий и аксиом. Для онтологий средней зоны, которые отличаются большим количеством понятий, сбор понятий может выполняться автоматически с помощью кластеризации. В процессе обработки большого количества информации происходит сбор понятий и разбиение их по классам на основании каких-то общих характеристик. Существует целый ряд методов по увеличению точности извлечения семантически связанных семей понятий. При таком анализе в дальнейшем возможно также устанавливать перекрестные ссылки внутри онтологии. Однако для онтологии важно знать не только то, что понятия взаимосвязаны, но, и то, как именно они взаимосвязаны. Для выявления таких отношений между понятиями также могут быть использованы автоматические методы просмотра и анализа различных текстов, например, как предлагают авторы работы, можно извлекать данную информацию из словарных определений. Это обусловлено тем, что существует ограниченный набор фразовых

моделей, вводящих определяющее, по характеру которых можно сформулировать тип связей между понятиями и ввести эти данные в онтологию. Таким образом, можно выбрать словарные статьи, провести их разбор, выявить семантические толкования для слов и начальных конструкций определений. Данная идея пополнения онтологии встречается и в работе , где такой подход позволяет выявить отношения между элементами онтологии с помощью анализа корпуса и поиска моделей, соответствующих какому-либо роду отношений.

Выявление же аксиоматических знаний, правил может быть произведено на основании Интернета. Э.Хови указывает на то, что сбор отдельных примеров происходит за счет изучения большой базы ресурсов, однако вначале формулируется ряд параметров, указывающих, какие именно экземпляры нам нужны. Это можно пояснить на конкретном примере: если нам нужно собрать названия столиц, то можно, например, задать условия поиска с моделью "X – столица ...". При этом можно использовать не одну такую модель, а несколько. Таким образом, нахождение отдельных экземпляров будет сводиться к анализу текстов для выявления примеров с данными структурами. Такие шаблоны могут формироваться как вручную, так и автоматически с помощью самообучающихся программ.

Наряду с поиском отдельных экземпляров важным является установление различных связей между элементами онтологии, классами. В целом авторы работы предлагают разделить все методы извлечения отношений между элементами онтологии на два класса: подходы, основанные на использовании шаблонов, и методы, использующие кластеризацию. При использовании методов на основе шаблонов исследователи ищут языковые модели, которые указывают на какой-либо тип отношений между классами. В большинстве случаев осуществляется поиск родовидовых отношений и отношений «часть-целое».

При наличии базового перечня категорий можно наращивать их число, обращаясь к корпусу текстов. В корпусе выделяются кластеры близких по значению элементов, затем оценивается теснота связи между элементами, далее каждому кластеру приписывается имя, которое ассоциируется с категорией, включаемой в онтологию (подробнее см., например, ,). Таким образом, пополнение онтологии новыми элементами происходит автоматически и также автоматически определяется место нового элемента в иерархии категорий.

Как упоминалось ранее, в онтологии присутствуют не только классы и отдельные экземпляры, но и аксиомы. Они вносят большой вклад в усовершенствование анализа информации, дают возможность компьютеру дополнить текст дополнительными знаниями, которые для нас кажутся тривиальными. Аксиомы сообщают компьютеру, например, о том, что два высказывания имеют один и тот же смысл или же один факт влечет за собой наличие другого. В работе рассматривается метод автоматического извлечения аксиом из текстов и их дальнейшая классификация и проверка на релевантность. В качестве основной проблемы автоматического выявления аксиом авторы упоминают построение ошибочных предположений или слишком общих, а также проблему определения симметричности аксиом. Они берут за

основу гипотезу о схожести значений при схожести контекстов встречаемости, дополняя их своими ограничениями. Здесь ключевым становится определение методов вычисления близости контекстов и близости понятий. По утверждению авторов их методика выявления аксиом позволяет извлекать релевантные аксиомы с маленьким процентом ошибок.

В большинстве случаев проблемой автоматического извлечения становится большое количество «шума», который надо эффективно отсеивать. В связи с этим иногда наряду с автоматическими методами используют последующую ручную обработку полученного материала для получения данных большей точности.

В работе упоминаются общие требования, предъявляемые к системам автоматического извлечения данных для онтологий:

- Минимальный контроль – сведение к минимуму или исключение вообще участие человека.
- Универсальность – применимость к различным источникам, все зависимости от их размера, области знания и т.д.
- Точность – извлеченная информация должна содержать как можно меньше ошибок.
- др.

Выполнение данных требований, возможно, позволит построить эффективную систему автоматического построения онтологий, пока же все существующие системы нуждаются в доработках и улучшениях или же успешно работают лишь применительно к замкнутым областям знания.

Однако никакая онтология не полезна в изоляции, ее преимущество как раз заключается в возможности использоваться вновь в новом окружении. Сейчас существует много различных алгоритмов состыковки нескольких уже созданных онтологий (их общее описание приводится в ,). Эти методы используются для слияния онтологий и для нахождения соответствий для слов в других языках (в данном случае онтология служит межъязыковой точкой отсчета).

Соединение двух онтологий осуществляется в несколько этапов. На начальной стадии происходит нахождение связующего звена, на основании которого можно произвести слияние. Затем срабатывает выравнивающий алгоритм, который по описанию элементов находит их место в новой структуре. Потом уже идет выверка результатов состыковки в контексте.

Как упоминалось ранее, существует целый набор алгоритмов по нахождению связующего звена для двух онтологий, стоит рассказать о них немного подробнее. Выделяют несколько различных методов обнаружения связующих звеньев ():

- текстовые совпадения;
- совпадения иерархических отношений;
- совпадение форматов и данных.

Э.Хови поясняет, что под текстовыми совпадениями подразумевается идентичность имен понятий (здесь также учитываются родственные слова), текстовых определений (сравнение строк, трансформация, исключение стоп-слов и др.). Иерархическое

совпадение предусматривает поиск общих вышестоящих понятий, фильтрацию неоднозначностей, нахождение семантического расстояния, рассеивание семантических групп (semantic group dispersal). Совпадение форматов и данных опирается на внутрисемантические отношения и ограничения на заполнение слотов. После отработки алгоритмов используется функция, которая учитывает результаты всех процедур и выдает общий коэффициент совпадения. Нахождение связующего звена включает в себя также процедуры валидации, в ходе которых происходит проверка с учетом иерархических связей соотносимых понятий. Эта процедура пытается найти несоответствия, порочные круги, проверить наследование свойств. Для выявления идентичности понятий используются специально созданные критерии. Так, комплексный критерий, предложенный Н. Гуарино, проверяет сходство по нескольким параметрам:

- материал: идентичность материала, из которого сделаны экземпляры сравниваемых понятий;
- топологический: идентичность формы экземпляров сравниваемых понятий;
- морфологический: те части, из которых состоят экземпляры сравниваемых понятий;
- функциональный: использование;
- меронимический: экземпляры понятий;
- социальный: социальная роль.

Учитываются также возможные стандартные метонимические переносы, которые делают онтологию более гибкой и расширяют возможность нахождения близких по содержанию понятий.

Упомянутые ранее методы являются полуавтоматическими, то есть сначала автоматически генерируются варианты соответствий, а потом вручную в несколько этапов происходит соединение онтологий. Как упоминает Э.Хови, статистика показала, что эти процедуры обладают достаточно высокой степенью точности и дают хорошие результаты. Так, использование подобных автоматических алгоритмов выравнивания при построении онтологии SENSUS дало более 90% точности.

Побочным положительным результатом использования процедур слияния является возможность выявить ошибки и опущения в онтологиях. Слияние онтологий может использоваться также для устранения проблемы дублирования. На данном этапе развития происходит так, что соревнующиеся и взаимосвязанные коллективы формируют схожие модели областей знания, где наблюдается частичное пересечение. При такого рода дублировании неизбежно страдает и логичность. Следует признать, что некоторые пересечения конечно необходимы.

Однако Э.Хови отмечает, что при слиянии онтологий может возникнуть ряд проблем, которые может быть достаточно сложно решать автоматическими методами. Так эксперты в разных областях могут отсылать к одному и тому же понятию и понимать его различным образом. Проблемы возникают и тогда, когда одно и то же слово используется для обозначения различных понятий в каждом отдельном поле. Решением такой проблемы может быть более тесная коммуникация составителей

онтологий, а также использование более широких онтологий, применимых к различным областям знания.

5. Применение онтологий

В предыдущих разделах рассматривались определения и структура онтологий, но остался нераскрытым вопрос, зачем строят онтологии и где они применяются. Н. Ной упоминает ряд способов использования онтологий:

- для совместного использования людьми или программными агентами общего понимания структуры информации;
- для возможности повторного использования знаний в предметной области;
- для того чтобы сделать допущения в предметной области явными;
- для отделения знаний в предметной области от оперативных знаний;
- для анализа знаний в предметной области.

Построение онтологии часто не является само по себе конечной целью, обычно онтологии далее используются другими программами для решения практических целей. На данном этапе развития науки существует ряд задач, где применение онтологий может дать хорошие результаты. Однако сейчас лишь малое количество приложений на естественном языке включают в себя онтологические базы, откуда черпаются знания об окружающей действительности. С. Ниренбург и В. Раскин говорят о возможности использования онтологий в:

- машинном переводе;
- вопросно-ответных системах;
- информационном поиске;
- системах извлечения знаний;
- общих системах ведения диалога между компьютером и человеком;

системах понимания языка (автоматическое реферирование текста, рубрикация и др.)

Можно упомянуть также системы расширенного консультирования, которые включают в себя несколько уровней работы с информацией и строятся на базе других приложений.

В искусственном интеллекте онтологии используются для формальной спецификации понятий и отношений, которые характеризуют определенную область знаний.

Поскольку компьютер не может понимать, как человек, положение вещей в мире, ему необходимо представление всей информации в формальном виде. Таким образом, онтологии служат своеобразной моделью окружающего мира, а их структура такова, что легко поддаются машинной обработке и анализу. Онтологии снабжают систему сведениями о хорошо описанной семантике заданных слов и указывают иерархическое строение области, взаимосвязь элементов. Все это позволяет компьютерным программам при помощи онтологий делать умозаключения из представленной информации и манипулировать ими.

Онтологии используются также при построении корпуса определений, служащего справочным материалом. В дальнейшем результаты этой работы могут использоваться для сложных процедур обработки естественного языка, например, в снятии омонимии на основе контекста. Онтологии могут использоваться для вывода умозаключений, необходимых для понимания текстов на глубинно-семантическом уровне, что требуется для высококачественного машинного перевода и может служить базой для расширения и уточнения информационного поиска. Глубокий анализ текста необходим и для систем автоматического реферирования. Стоит упомянуть, что также онтологии могут способствовать систематизации понятий. На базе онтологий может осуществляться автоматическое аннотирование и разбор текстов, которое в дальнейшем может использоваться в первую очередь в информационном поиске, а также при различных видах анализа информации. Приведем некоторые примеры существующих систем, содержащих онтологические приложения. В сфере информационного поиска заслуживает упоминания европейский исследовательский проект под названием CROSSMARC (описание данной системы приводится в работе). Участники этого проекта делают упор на необходимости широкого использования онтологий для разделения отраслевых и общепонятийных знаний, считая, что это облегчит извлечение информации из различных источников, сузит поисковые запросы и улучшит качество выдаваемых результатов. Эта задача оказывается смежной с задачей автоматической рубрикации текстов, в ходе которой производится распределение текстов по рубрикам на основе автоматических методов и использования онтологий. В области машинного перевода известна система OntoLearn (описание данной системы приводится в работе), используемая при переводе многословных терминов с английского языка на итальянский. Система автоматически вычленяет и строит предметные онтологии. Промежуточные онтологические построения используются для прямого машинного перевода. Можно привести также пример системы машинного перевода, разработанной в Университете Sains в Малайзии . Она осуществляет снятие неоднозначности со слов, используя тексты определений и структурную информацию из онтологий. IAMTC (Interlingual Annotation of Multilingual Text Corpora) можно отнести к системам понимания языка. Этот многосторонний проект занимается аннотацией шести больших параллельных корпусов с целью извлечения межъязыковых соответствий. Система использует 110 000 записей онтологии OMEGA для частеречной разметки и дальнейшего анализа естественного языка. Для понимания естественного языка могут использоваться также аксиомы и умозаключения, содержащиеся в онтологиях, помогает и большой набор отдельных примеров, экземпляров. Как уже упоминалось ранее, онтологии могут также лежать в основе различных вопросно-ответных систем и способствовать улучшению анализа запросов и точности ответов. Можно привести пример демонстрационной вопросно-ответной системы YAWA [7]. YAWA по запросу выдает информацию о главах государств и правительствах стран мира. Она обладает сведениями о том, кто занимает указанную

должность в данной стране. Кроме этих связей, в нее заложены знания о соотношении названия и основной функции (глава государства и/или правительства) высших государственных должностей в отдельно взятой стране в зависимости от типа системы государственного управления. Таким образом, при выдаче ответов по запросу система учитывает заложенные в нее сведения об окружающей действительности: набор понятий, отношений между ними, ограничений на отношения и список конкретных экземпляров.

В работе рассматривается целое направление работ в сфере электронной коммерции, где онтологии предоставляют классификацию товаров и услуг и обеспечивают наличие стандартизованного представления информации. Таким образом, происходит систематизация понятий области бизнеса, упорядочение их описаний. Онтологии предоставляют эффективный доступ к информации, дают возможность лучше понять данную информацию и, следовательно, произвести её более широкий и сложный анализ. Это использование онтологий является ярким примером использования онтологий «для совместного использования общего понимания структуры информации людьми или программными агентами», упоминаемого Н. Ной .

Еще один пример использования онтологий представляет собой система понимания языка, разработанная в НПЦ «Интелтек Плюс» [1]. Сейчас уже «создана и внедряется в Совете Федерации Федерального Собрания Российской Федерации первая очередь информационной системы "Семантический контроль текстов редактируемых документов"... Она используется специалистами Управления информационного и документационного обеспечения аппарата Совета Федерации для проверки правильности расшифровки стенограмм и проверки редакций различных типов документов на предмет их соответствия эталонным словарям и базам данных»[1]. Эта система занимается поиском несоответствий в текстах редактируемых документов на основе эталонного описания предметной области, содержащегося в онтологии. К таким несоответствиям авторы относят «ошибочные должности сотрудников организаций, ссылки на устаревшие структурные подразделения организаций, неправильные телефонные номера должностных лиц» [1]. Таким образом, они стремятся выявить

«неэквивалентность факта, выявленного при анализе текста, имеющимся в базе знаний фактам». В данном случае под базой знаний понимается онтология, снабженная конкретными экземплярами. После извлечения знаний из текстов, модуль логического вывода сверяет их с данными в онтологии, проверяя наличие связей между элементами, отслеживая правильность этих связей, и таким образом подтверждает или опровергает достоверность фактов и отмечает нарушение семантических связей.

Онтологии занимают ключевую позицию во многих лингвистических комплексах, так, например, InfoMap (<http://infomap.stanford.edu>) широко использует иерархические структуры в работе с текстами на естественном языке. Целью данного проекта являлось извлечение значения слов на базе их употребления в

тексте. Этот материал в дальнейшем может использоваться для понимания языка и машинного перевода, а также для интеллектуального поиска, принимающего во внимание значения слов, и индексирования ([9]). В рамках данного проекта при группировке понятий и значений используется верхняя зона WordNet, а также анализ корпусов текстов путем извлечения кластеров слов автоматически. Затем эти данные используются для подбора вероятных родовых терминов в группах слов, то есть для формирования таксономий. Также возможно применение результатов данных процедур для сравнения структуры групп в параллельных текстах на разных языках, что позволяет улучшать качество машинного перевода и уточнять переводы слов в рамках многоязычной лексикографии. Существует большое количество проектов в области медицины, использующих онтологии в своих приложениях. Так, можно привести пример проекта MuchMore (<http://muchmore.dfki.de>), являющегося частью InfoMap, описанного ранее, занимающегося разработкой методов организации информации на различных языках и в частности медицинской области знания. Их исследование основывается на использовании иерархии понятий для предметных областей, и следовательно технологиях извлечения многоязычных терминов и отношений. Их продукт помогает осуществлять поиск документов на различных языках по медицинской области знания. Медицинская область знания очень перспективна в этой сфере, так как для нее уже создано большое количество онтологий и структурированных источников знания, а также присутствует множество текстов на данной области, требующих обработки. Это тексты, описывающие карты больных, случаи заболеваний, общие описания разных болезней и многие другие. Проект MuchMore помогает выстроить взаимосвязи между всеми

типами текстов в данной области. В задачи этого исследования входит:

Сокращение «пропасти» между медицинской документацией и многоязычными данными путем автоматического извлечения дескрипторов и составления метаописаний истории болезней для последующего использования в других источниках.

Организация информации в онтологии помогает в дальнейшем быстро строить экспертные системы и приложения для работы с данными.

- Устранение языкового барьера при поиске информации. Использование онтологий позволяет эффективно искать информацию на нескольких языках, тем самым значительно облегчая работу специалистов. Также это дает возможность сравнивать описания аналогичных случаев на разных языках и проводить более содержательные исследования.

В рамках работы описывается еще одно перспективное применение онтологий, их использование при семантической разметке текста. Учет семантических категорий, описанных в онтологии, позволяет сделать разметку корпусов более точной, уменьшить неоднозначность, так как в шаблоны, по которым производится разметка, связаны в категориями в онтологии. Такая семантическая разметка в дальнейшем позволяет проводить семантический анализ текста, различные статистические исследования, извлекать межъязыковые соответствия.

Еще одним современным проектом, широко использующим онтологии является компания Онтос (http://www.ontos.com/ru/company/about_us.php), разрабатывающая различные семантические технологии. При помощи их систем, основанных на обработке текстов на естественном языке (NLP), пользователь может генерировать и хранить релевантные знания, необходимые для различных задач. Данные системы ориентированы на пользователя, которому надо обрабатывать большие массивы информации, извлекать структурированную информацию. Для решения данных задач возможно использование продуктов Онтоса, обеспечивающих автоматическую обработку необходимых неструктурированных данных и получения прямого доступа к аналитическим (обработанным) данным. Как упоминается на сайте компании, их системы успешно решают следующие задачи:

- Поддержка принятия решений при проведении исследований
- Визуализация информации с помощью семантических сетей
- Автоматическая генерация семантических аннотаций из неструктурированного текста
- Дайджестирование больших документов на базе их семантического содержания
- Резюмирование больших объемов аннотированного текста

Поддержка мета-данных в соответствии со стандартами RDF/OWL

- Семантический поиск с применением технологии триплетов (Объект — Отношение — Объект)

Так, один из продуктов компании, OntosMiner

(http://www.ontos.com/ru/products/ontos_miner.php), анализирует текст на естественном языке, используя онтологии и семантические правила. Результатом работы данной системы становится распознавание объектов и связей между ними и добавление их как аннотации к соответствующим фрагментам текста.

Еще одним аналогичным проектом является RCO (Russian Context Optimizer) (<http://www.rco.ru/>). С помощью современных технологий исследователи строят онтологии, семантические представления. Продукты и технологии RCO позволяют решать такие прикладные задачи как составление содержательного портрета текста, извлечение именованных объектов, связей и фактов из массивов неструктурированных данных, анализ тональности текста, выявление заимствований, обнаружение дубликатов. Использование онтологий помогает при поиске заранее неизвестной информации, относящейся к некоторой теме, позволяя выдать пользователю возможные — подсказки|| для уточнения запроса. Также онтологии служат основой для решения различных аналитических задач, позволяя исследовать окружение выбранного объекта, находить цепочки и группы связности во множестве объектов.

Также в этой связи стоит упомянуть проект «Галактика ZOOM» (<http://www.galaktika-zoom.ru/>). Эта система предоставляет различные возможности для специалистов разных уровней: руководителей, аналитиков, маркетологов, специалистов по PR, сотрудников служб безопасности. Как упоминается на сайте, их разработки могут

использоваться для поиска информации, выявления сути текста, сравнения документов и исследования документов с учетом динамики во времени. Стоит упомянуть, что онтологии получили широкое распространения и для моделирования организационной структуры предприятий. Как упоминается в обзоре [2], онтологическое представление знаний о субъектах экономической деятельности, которые входят в состав какой-либо системы, можно использовать для объединения их информационных ресурсов в единое информационное пространство. Онтология предприятия включает в себя организационную онтологию, описывающую организационно-функциональную структуру предприятия: состав штатного расписания (работники, администрация, обслуживающий персонал), партнеры, ресурсы и т. п. и отношения между ними, а также онтологию по технологиям, описывающую терминологию технологий. Разработанные онтологии позволяют сотрудникам одной отрасли или корпорации использовать общую терминологию и избежать взаимных недоразумений, которые могут усложнить сотрудничество и привести к серьезным убыткам.

В качестве примера практического использования онтологических моделей технологий можно привести систему ONTOLOGIC (<http://www.intertech.ru/Production/sol+tech.asp>), предназначенную для создания и поддержки распределенных систем нормативно-справочной информации (НСИ), ведения словарей, справочников и классификаторов и поддержки системы кодирования объектов учета. Онтология обеспечивает непротиворечивое накопление любого количества информации в стандартной структуре классификации. Такой подход гарантирует однозначную идентификацию ресурсов независимо от различных трактовок их наименований разными производителями. При использовании данной системы осуществляется эффективный контроль и верификация данных, проверки корректности, полноты и непротиворечивости данных как на этапе анализа и нормализации существующих данных, так и при занесении новых элементов данных.

6. Языки представления (Semantic web)

Как упоминалось ранее, на данном этапе развития технологий появилась идея расширения существующей ныне World Wide за счет добавления к уже существующим документам семантической информации. Это расширение Интернет ресурсов получило название Semantic Web. Создатель WWW Тим Бернерс-Ли определяет Semantic Web как «расширение существующей WWW, в котором информации придается четко определенное значение, позволяющее людям и компьютерам сотрудничать» (цит. по [3]). Майкл Ушолд [3] поясняет, что «точного определения SW нет, но ее основная черта – наличие семантики, доступной для обработки машинами (программными агентами Сети) при взаимодействии друг с другом». Semantic Web дает возможность компьютерам обрабатывать гипертекстовые страницы, предоставляющие информацию людям, и автоматически получать данные.

Можно привести несложный пример такой обработки, упомянутый А.Сварцем : система оценки рейтинга книги. Можно добавлять баллы к конкретной книге X, если в тексте отзывов встречается фраза "Мне понравилась X", но есть и другие способы выражения того, что это хорошая книга и читатель оценил ее по достоинству. Однако компьютеру не понять, что выражает одобрительную оценку книги, для этого ему нужны данные, которые бы явно указали на это.

На решение этих и многих других задач и направлено создание такой системы, как Semantic Web. Использование стандартизованных языков позволяет точно описывать содержание, лежащее в основе HTML страниц, а также устанавливать скрытую информацию. Так, описанию могут подвергаться изображения и видеоматериалы, содержащиеся на сайтах или же какие-либо недоступные при простом чтении страницы данные.

Любой элемент в Semantic Web может получать идентификатор, называемый "Uniform Resource Identifier" или URI. Используется единообразная система идентификаторов, и каждый идентифицированный элемент рассматривается как ресурс, получивший определенное имя. Если что-то имеет URI, можно сказать, что оно находится в сети. Именно URI является основой и связующим звеном Semantic Web, и они не подлежат перемещению. URI децентрализованы, они напрямую не контролируются никакой организацией.

В курсе лекций В. Д. Соловьева и соавторов [3] приводится структура (рис. 1), отражающая строение Semantic Web, которая у Т. Бернерса-Ли получила название «слоеный пирог». Данную структуру отличает большое разнообразие элементов:

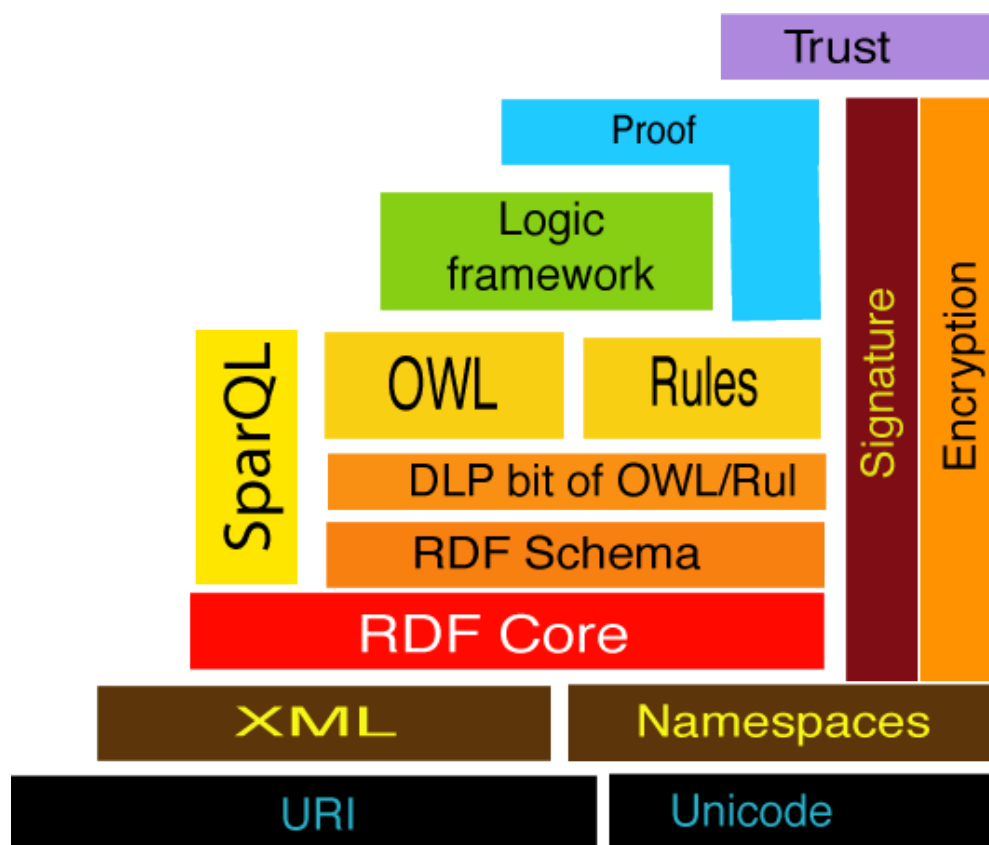


Рис. 1

Простой идентификации ресурса недостаточно, чтобы понять, как он может использоваться. Таким образом, URI бесполезны, если мы не опишем, что они значат. Для этого и нужны онтологии и так называемые схемы, они являются неотъемлемой частью Semantic Web и становятся основой для семантической разметки. В рамках Semantic Web онтологии занимают центральное положение, что можно увидеть на рис. 2, они задают отношения между понятиями и определяют логические правила для рассуждений о них. Таким образом, компьютер может понимать смысл данных, обращаясь к онтологиям за требующейся информацией.

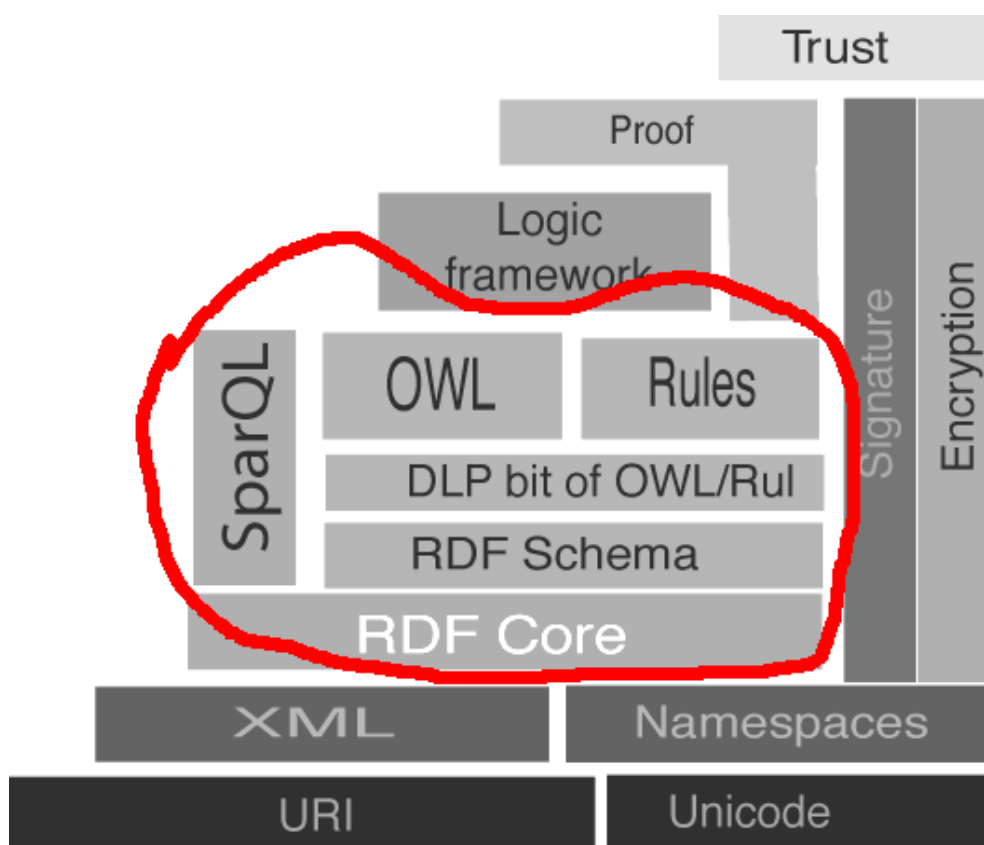


Рис. 2

Основные этапы развития языков представления онтологий в Semantic Web освещены в цикле лекций Б.В.Доброва и соавторов [3]. Изначально для описания Semantic Web использовался Extensible Markup Language (XML). Этот язык является простым инструментом создания документов с универсальными средствами описания. Он позволяет каждому описывать свою структуру, формат документа, в том числе различные виды разметки. Введенная с помощью XML «интеллектуальная» разметка поддается компьютерной обработке и может интерпретироваться особым образом. Однако недостаточно просто описать элементы Semantic Web, надо, чтобы компьютер мог понимать и анализировать составленные нами высказывания. Больше возможностей для обработки высказывания предоставляет формализм Resource

Description Framework (RDF). Выражение RDF имеет структуру простого предложения, с единственным отличием -- все слова являются URI. Каждое высказывание RDF представляет собой тройку (S,P,O), где S – субъект, P – предикат и O – объект (рис. 3). Выражения RDF позволяют утверждать какую-либо информацию об элементах Semantic Web и делать выводы. Этот язык удобен также для работы с базами данных. Однако стоит помнить, что базы данных постоянно меняются, а слишком сложное описание может затруднить их быстрое обновление.

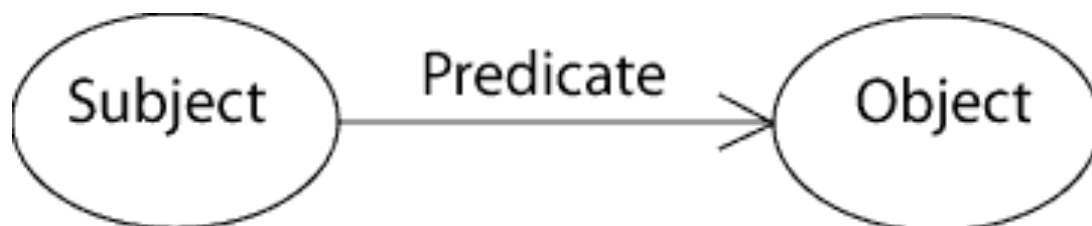


Рис. 3

Затем стал развиваться язык RDF Schema (RDFS) - язык описания словарей RDF-терминов. RDF Schema определяет классы, свойства и другие ресурсы. Таким образом, RDFS стал семантическим расширением RDF. RDF и RDFS позволяют работать с метаданными, обеспечивать компьютер семантической информацией и обрабатывать эту информацию автоматически. Однако, используя RDF, «кто угодно (т.е. любой пользователь RDF) может сказать что угодно (т.е. фиксировать произвольное утверждение) о чем угодно (т.е. о любом ресурсе Сети)». Кроме того, «RDF не запрещает делать бессмысленных утверждений или утверждений не согласующихся с другими. Следовательно, нет никакой гарантии целостности и непротиворечивости RDF-описаний» [3], поэтому при использовании информации пользователи должны сами проверять эту целостность.

Как уже упоминалось ранее, ядром Semantic Web должны были стать онтологии, однако, как отмечается в [1], отсутствие стандартизированных, единых языков онтологий затрудняло их использование внутри взаимосвязанных систем в одних и тех же областях знания. Было сложно устанавливать связи между онтологиями и соединять их. Онтологии, описывающие сходные специализированные знания, существенно отличались в синтаксисе и семантике в зависимости от используемого онтологического языка. Это препятствовало повсеместному использованию онтологий.

Создатели Semantic Web попытались найти решение проблемы совместимости описаний, создав в 2004 году World Wide Web Consortium (W3C) для обсуждения возникших проблем. Ими был предложен универсальный стандарт для сетевого обмена онтологической информацией – Web Ontology Language (OWL). Данный язык помогает

преодолеть проблему взаимодействия систем и становится основой для многих сетевых приложений. С его помощью эксперты предметной области и разработчики приложений могут создавать, модифицировать и соединять различные онтологии. Стоит упомянуть, что сейчас существуют различные онтологические языки, однако OWL является их своеобразным универсальным объединением. Так одним из первых таких языков являлся Simple HTML Ontology Extension (SHOE), разработанный в Университете Мериленда. Этот язык был дополнен специальными тэгами для включения семантической информации в обычный код HTML. Вскоре за ним в 2000 году в Амстердамском университете был создан язык Ontology Interchange Level (OIL). Здесь уже была добавлена формальная семантика, основанная на логическом описании. Этот язык базировался на таких упомянутых ранее стандартах W3C, как XML и RDF. Одновременно с ним в США появилась программа DARPA Agent Markup Language (DAML). Она выпустила DAML-ONT, язык спецификации онтологий для реализации более сложных определений RDF классов, чем реализуемые с помощью RDFS. Позднее проект DAML соединил усилия с OIL и выпустил DAML+OIL. В DAML+OIL была усовершенствована языковая семантика, она стала яснее и успешнее взаимодействует с инструментами построения.

Вдохновленные успешным развертыванием и использованием DAML+OIL и понимая необходимость стандартизированного языка онтологий, участники W3C создали Web Ontology Language (OWL). Данный язык строится на основании RDF, который сам по себе строится на синтаксисе XML. RDF и OWL дают возможность создавать классы, свойства и отдельные экземпляры. Таким образом данный язык реализует структуру онтологий. Можно привести пример использования этих трех компонентов, предложенный в работе : определим класс сущностей "People" и некоторые свойства "People", такие как "name", "birthday" и "friend".

Средствами синтаксиса RDF класс и его свойств будут описаны так:

```
<Class ID="Person"/>
```

```
<Property ID="name"/>
```

```
<Property ID="birthday"/>
```

```
<Property ID="friend"/>
```

Когда определены классы, их можно использовать при описании отдельных экземпляров. Информация — Joe Blog, born January 1, 1950, is friends with John Doe|| будет представляться следующим образом:

```
<Person ID="Joe">
```

```
<name>Joe Blog</name>
```

```
<birthday>January 1, 1950</birthday><friend resource="#John"/>
</Person>
```

```
<Person ID="John">
<name>John Doe</name>
</Person>
```

Такое описание может быть подвергнуто большей степени обобщения и использоваться как общее описание класса с указанием ограничений на свойства и отношения.

Упомянутый ранее язык OWL расширяет функциональность RDF, сохраняя при этом совместимость с другими программами и являясь открытой и свободной для распространения системой. Существует несколько вариантов данного языка: OWL Lite, OWL DL, и OWL Full. Каждая разновидность характеризуется своим соотношением детальности описания и объема, сложности прикладной области:

- OWL Lite предназначен для пользователей, которым необходима лишь классификационная иерархия и некоторые простые условия согласованности сущностей.
- OWL DL (Description Logic) рассчитан на пользователей, которым необходима максимальная степень выразительных возможностей языка без потери вычислительной полноты (с гарантией получения всех возможных умозаключений, получаемых формально-логическим путем) и разрешимости (все вычисления выполняются за конечное время). Уровень OWL DL ориентирован на существующие сегодня системы описания знаний и системы логического программирования.
- OWL Full рассчитан на тех пользователей, которым необходимы максимальные выразительные возможности языка и вся свобода выбора синтаксических средств, предоставляемая в RDF, но не обязательны вычислительная полнота и разрешимость. Онтология, записанная на OWL Full, позволяет расширять значения терминов, взятых из заданных словарей. , .

OWL обладает рядом преимуществ перед RDF. К дополнительным возможностям, появившимся в этом языке, можно отнести возможность создавать локальные ограничения области распространения. Ранее у каждого свойства, отношения мог задаваться домен, но он оставался неизменным во всех областях применения. Однако порой то, как отношение применяется, зависит от конкретного класса, с которым мы работаем. Можно привести пример, упоминаемый в названной работе : рассмотрим свойство «ест» для класса «Человек». Возможно, для уточнения мы захотим добавить ограничение, указав, что значение для «ест» берется из класса «Еда». Это утверждение верно для общего класса «Человек», однако оно может не распространяться на подклассы, например, подкласс «Вегетарианец». Значение

«ест» для этого более узкого класса не «Еда», а ее подмножество (все, кроме мяса). Однако такая детализация была невозможна на уровне RDF, в OWL же мы можем указать локальное ограничение свойства для конкретного класса. Это уточняет описание и расширяет возможности при создании онтологии.

Кроме того, в OWL вводятся основные функции над множествами, такие как объединение, пересечение, дополнение и непересекаемость. Ценность этой возможности можно проиллюстрировать на примере: пусть в нашем распоряжении есть два подкласса "Человек", называемых "Живой Человек" и "Мертвый Человек". При использовании описания RDF не существует никаких ограничений на определение объекта, который одновременно является экземпляром класса "Живой Человек" и класса "Мертвый Человек", так как ничто не определяет эти классы взаимоисключающими. Однако эта проблема успешно решается на уровне OWL.

Еще одним преимуществом OWL является введение понятия мощности. OWL позволяет накладывать ограничения на свойство, требуя чтобы оно использовалось для любого экземпляра минимальное количество раз (минимальная мощность), максимальное количество раз (максимальная мощность) или определенное количество раз (мощность). Это означает, что можно, например, потребовать, чтобы человек- билингв говорил на двух языках.

На данном этапе OWL применяется в ряде областей, так можно привести пример системы knOWLeR , утилиты по управлению потоками информации. Она демонстрирует, что использование онтологический умозаключений сопоставимо по результатам со стандартными системами информационного поиска. Данная система включает в себя примерно 100 миллионов утверждений. Система поддерживает сложное формирование логического вывода и работу с запросами, используя подмножество OWL. Следует упомянуть, что OWL используется также в системах, не относящихся к анализу естественного языка.

Хотелось бы отметить, что используемые ныне языки представления онтологий также рассчитаны и на моделирование нечетких знаний . Например, такие расширения RDF и OWL, как Fuzzy RDF и Fuzzy OWL, позволяют совмещать стандартные модели предметных областей и нечеткую логику, применяемую на уровне задания аксиом. Известно также расширение OWL, где используются вероятностные модели знаний, например, Bayes OWL

Подводя итог, можно сказать, что OWL обеспечивает достаточно богатую семантику для описания он-лайн онтологий, которые важны для современных проектов анализа естественного языка и совместимы с web-стандартами и архитектурой. Сейчас существуют планы дальнейшего развития OWL для усовершенствования и возможно расширения описаний.

7. Существующие онтологические ресурсы

Как упоминалось ранее, на данном этапе существует ряд разнообразных онтологических ресурсов. Эти онтологии, противопоставляемые по направленности, отличаются друг от друга по форме и содержанию. Опишем некоторые из онтологий,

соотнося их с ранее описанными классами.

Сейчас существует ряд известных онтологий верхней зоны. Так, в качестве примера можно привести онтологическую структуры, созданной под руководством Э.Хови, которая включает 400 записей и описывает наиболее общее разбиение понятий по категориям. Известной является также Penman Upper Model, состоящая всего лишь из 300 элементов. В ней нет лексических единиц и аксиом. Ее записи представляют понятийно-грамматические классы, выступающие как связующее звено между языком и реальным миром. Между классами верхней модели и синтаксическими типами установлены связи. Эта модель, протестированная для грамматик разных языков и машинного перевода, может быть использована при построении и координации онтологий областей знаний.

Еще одним примером такого рода онтологий является DOLCE. Она была создана под руководством Н. Гуарино в Италии, в институте LADSEB (по материалам). Это поверхностная модель, обладающая очень высокой степенью абстрактности. В ней содержатся примерно 500 понятий и нет лексических элементов. Описание производится с помощью языков XML или RDF, оно очень четкое и последовательное. Однако в данной онтологии малая степень иерархичности и все выглядит как верхние уровни. При конвертации этой онтологии в другие форматы могут возникать проблемы, так как пропадает часть функциональности, теряются некоторые аксиомы и ограничения.

Онтология, разработанная Дж. Совой (Sowa's KR Ontology), также относится к категории структур верхней зоны. Эта онтология базируется на структурах, разработанных философами. Главной целью, преследуемой ее автором, являлось создание основы, в которую могут быть включены все остальные онтологии или модели их верхних зон. Структурная решетка этой онтологии представляет собой 12 понятий, являющихся всеми комбинациями трех отличительных признаков:

- первичность/вторичность/третий порядок
- материальный/абстрактный
- длительный/мгновенный (continuant/occurrent)(объект/процесс)

Составление таких решеток помогает избежать проблемы очередности использования дифференциальных признаков при делении понятий.

В онтологии Дж. Совы используются также некоторые стандартные ситуативные роли:

- детерминирующий участник (участник, определяющий направление процесса, то есть инициатор или цель);
- неотъемлемый участник (присутствует на протяжении всего процесса, но активно не контролирует происходящее);
- источник (должен присутствовать в начале процесса, но не обязан принимать участие во всем процессе);
- продукт (может появляться в конце процесса, но не обязан принимать участие во всем процессе);

К онтологиям средней зоны можно отнести Mikrokosmos, созданную в Университете Нью-Мексико. Она содержит около 6 000 записей. Онтология Принстонского университета WordNet на несколько порядков больше, она состоит из 110 000 записей, в которых в большинстве случаев отражена конкретная, нетерминологическая лексика. В ее рамках заданы различные элементы с их значениями, объединенные в группы (синсеты) по сходству значений.

СYC (<http://www.opencyc.org/>) является одной из самых крупных и разработанных онтологий. Она направлена на использование в рамках искусственного интеллекта, в ней содержится большое количество (более миллиона) аксиом. Версия ResearchCYC, преобразованная в формат RDF, не включает вторичных понятийных выражений, и таким образом, в ней пропала часть родовых понятий высшего уровня. По мнению Э.Хови, проблемой использования этой онтологии является запутанность ее структуры, она очень сложна для понимания.

Кроме того, на данном этапе существует множество онтологий отдельных предметных областей. Их можно разделить на несколько категорий. Так, к компьютерной области относится ряд онтологий, составленных в рамках библиотеки онтологий Protégé, также более 700 различных онтологий были созданы в OntoSelect. Область медицины достаточно подробно описывается в UMLS, состоящей из метатезауруса (более 1 миллиона биомедицинских понятий и 5 миллионов понятийных имен), семантической сети и специального лексикона. Еще одним примером онтологии предметной области является NAICS (Североамериканская система промышленной классификации), отражающая структуру разнообразных областей (таких как технология, строительство, сельское хозяйство и другие).

Существует ряд онтологий, которые объединяют несколько более мелких и служат своего рода каркасом для встраивания новых элементов на всех уровнях. Одной из таких онтологий является Suggested Upper Model Ontology (SUMO – (<http://www.ontologyportal.org/>)). Она и включенные в нее онтологии предметных областей представляют собой самую большую общедоступную онтологию, существующую на сегодняшний день. SUMO состоит из 20 000 элементов и 60 000 аксиом. В нее включаются сами разработки SUMO, онтология среднего уровня (MILO) и набор отраслевых онтологий (в сфере коммуникаций, транспорта, географии и многих других). Помимо этого она содержит большое количество аксиом, причем все ее элементы формально определены и это описание не зависит от конкретной системы осуществления выводов.

Онтология SENSUS также разрабатывалась как структура, в которую может быть включена дополнительная информация. Она содержит 70 000 записей и является расширением и реорганизацией WordNet; на верхнем уровне добавлены записи из Penman Upper Model и ветви WordNet перестроены.

Omega представляет собой среду для различных онтологий и ресурсов. Она содержит примерно 120 600 понятий и слов, в нее включены WordNet, Mikrokosmos, Penman Upper Model, а также дополнительные элементы. Это не одноязычная структура, в ней есть как английская, так и испанская лексика. Кроме того, в Omega содержится

множество различных ситуативных ролей (около 13 000), взятых из ресурсов Framenet, WordNet, LCS и PropBank. Эта среда обладает обширной базой отдельных экземпляров: 1,1 миллион персоналий, 765 000 фактов, 5,7 миллионов местоположений. Omega включает огромное количество (более 28 миллионов) утверждений о понятиях, отношениях и отдельных экземплярах. Она представляет собой ресурс с обширными возможностями построения, изменения и пополнения онтологий. Этот ресурс является библиотекой различных онтологий, позволяющей использовать объединять онтологии разных уровней для использования в какой-то прикладной задаче.

Это далеко не единственный ресурс, предоставляющий доступ к многочисленным отдельным онтологиям. Также существуют ресурсы, объединяющие онтологии, описанные с помощью какого-то языка представления, например, библиотека онтологий DAML (<http://www.daml.org/ontologies/>). Создатели онторедкторов (программ, упрощающих создание онтологий) предлагают своим пользователям самостоятельно формировать библиотеки онтологий, добавляя результаты своей работы прямо в специально созданный ресурс. Зачастую онтологии, содержащиеся в таких библиотеках, не отличаются общностью тематики. Однако такие ресурсы дают возможность не строить онтологии «с нуля», а использовать уже существующие наработки, находя все онтологии на одной ресурсе. Так, к таким библиотекам можно отнести библиотеку, сформированную пользователями онторедктора Protégé (ресурсы представлены на сайте -

http://protegewiki.stanford.edu/index.php/Protege_Ontology_Library). Также можно упомянуть Ontolingua (<http://www.ksl.stanford.edu/software/ontolingua/>), систему, разработанную в Стэнфордском университете, которая обеспечивает распределенную совместную среду для просмотра, создания, редактирования, модификации и использования онтологии.

К библиотекам онтологий относится и OntoSelect (<http://olp.dfki.de/ontoselect/>). Этот ресурс «отслеживает» появляющиеся онтологии в Сети, аннотирует их, предоставляя пользователям возможность в дальнейшем эффективно искать онтологии по интересующей тематике и в дальнейшем использовать их в своих целях. Также этот ресурс предоставляет возможность добавлять свою онтологию в библиотеку. На данном ресурсе можно найти ссылки на онтологии, описанные с помощью разных языков представления, так это и OWL, DAML, RDFS и другие. В OntoSelect представлены онтологии, созданные для разных языков, однако преимущественно все ресурсы разработаны для английского языка.

Вопросы для самоконтроля и текущего контроля

- 1) Что характеризует понятия концептуализация и формализация при определении проблемной области задачи?
- 2) Опишите структуру таксономической классификационной схемы.
- 3) Охарактеризуйте понятие и определите назначение онтологии предметной области.
- 4) Какова роль онтологий при поиске в сети Интернет?
- 5) Для чего предназначены модели знаний?
- 6) Для каких целей может потребоваться экспорт онтологий?

Задание на самостоятельную работу

- 1) Практическое задание

Выполнить создание спецификации знаний о предметной области в виде онтологии в редакторе Protege 5.5.0 согласно заданию практической работы №4.

- 2) Самостоятельная работа

Исследовать синтаксис и лексику современных языков создания онтологий.

Список рекомендуемой литературы¹:

[1] Сидоркина, Ирина Геннадьевна. Системы искусственного интеллекта [Текст] : [учеб. пособие для вузов по направлению 230100 "Информатика и вычисл. техника"] / И. Г. Сидоркина. М.: Кнорус, 2011. - 245 с.

Рассмотрены теоретические аспекты проектирования систем искусственного интеллекта и даны модели представления знаний, изложены материалы информационного, справочного, консультирующего характера по использованию инструментальных средств и технологическим особенностям разработки систем данного класса.

Подробнее: <https://www.labyrinth.ru/books/276217/>

[2] Богомолова, М. А. Экспертные системы (техника и технология проектирования) [Текст] : Методические указания к лабораторным работам / М. А. Богомолова. Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015. - 47 с.

Методические указания являются руководством при выполнении лабораторных занятий, проводимых по курсу «Экспертные системы (техника и технология проектирования)» с магистрами направления 09.04.03 - «Прикладная информатика» в терминальном классе. Лабораторные работы связаны с изучением принципов построения экспертных систем на основе различных моделей представления знаний, и могут быть полезны студентам других специальностей.

Подробнее: <https://www.iprbookshop.ru/71908.html>

[3] Советов, Борис Яковлевич. Представление знаний в информационных системах [Текст] : [учеб. для вузов по направлению подгот. "Информ. системы и технологии"] / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. М.: Академия, 2011.

Учебник создан в соответствии с Федеральным государственным образовательным стандартом по направлению бакалавриата «Информационные системы и технологии». Рассмотрены современное состояние теоретических и прикладных вопросов представления знаний в информационных системах, идеология построения интеллектуальных систем, математический аппарат представления знаний, возможности и пути использования искусственного интеллекта при проектировании информационных систем. Изложены новые аспекты представления знаний на основе искусственных нейронных сетей, расчетно-логических систем, генетических алгоритмов, мультиагентных систем. Приведены примеры практической реализации представления знаний на базе декларативного языка Пролог. Для студентов учреждений высшего профессионального образования. Может быть полезен разработчикам и пользователям информационных систем; преподавателям и научным сотрудникам, сферой интересов которых является интеллектуализация различных предметных областей; менеджерам и руководителям различного ранга, желающим самостоятельно ознакомиться с современным состоянием информационных технологий.

Подробнее: <https://lavkababuin.com/predstavlenie-znaniy-v-informacionnyh-sistemah-2-e-izd-ster-650564/>

[4] Остроух, А. В. Интеллектуальные информационные системы и технологии [Электронный ресурс] : монография / А. В. Остроух, А. Б. Николаев. Санкт-Петербург: Лань, 2019. - 308 с.

В монографии изложены концептуальные основы и методы представления знаний в интеллектуальных системах. Рассмотрены различные подходы, применяемые при проектировании и разработке интеллектуальных систем и технологий в транспортном комплексе, а также рассмотрены тенденции развития систем искусственного интеллекта.

Монография может быть использована для формирования профессиональной компетентности студентов высших учебных заведений, аспирантов и научных сотрудников обучающихся и ведущих научные исследования в области разработки и практического применения систем искусственного интеллекта по укрупнённой группе направлений подготовки "Информатика и вычислительная техника".

Подробнее: <https://www.labyrinth.ru/books/692861/>

Ссылки на открытые ресурсы (он-лайн-курсы, видеоуроки и т.д.)

[5]

ⁱСписок рекомендуемой литературы может быть более широким по сравнению со списком литературы, указанным в рабочей программе

Прикладные интеллектуальные системы

Конспект лекции 10. Semantic Web (Семантическая паутина). Поиск знаний в семантической паутине.

Аннотация: В лекции рассказывается о построении семантической паутины и её применении в практических задачах. Описан поиск знаний в семантической паутине.

План лекции:

1. Семантические технологии Web (Semantic Web). Определение и общее назначение технологии. Преимущества семантических сетей для интернета.
2. Технологии Semantic Web. Архитектура Semantic Web. Описание стека технологий и как они приведут к реализации Semantic Web.
3. Технологии Semantic Web. XML. XMLSchema.
4. Технологии Semantic Web. RDF.
5. Веб-сервисы (Web-Services). Определение, назначение подходы к созданию.
6. Особенности поисковых систем. Структура организации и алгоритм работы.
7. Язык запросов поисковых систем.
8. Семантическая поисковая система.
9. Онтологии в семантической паутине. Общее определение. Классификация онтологий.
10. Процесс разработки онтологий для семантической паутины.
11. Компоненты онтологического языка Web на основе OWL.
12. Применение онтологий. Значение онтологий для бизнеса.
13. Задания для самостоятельной работы.

Ключевые слова: Семантическая паутина, semantic web, онтологии, языки создания онтологий, автоматическое наполнение онтологий, XMLSchema, RDF, OWL

СОДЕРЖАНИЕ ЛЕКЦИИ:

1. Семантические технологии Web (Semantic Web). Определение и общее назначение технологии. Преимущества семантических сетей для интернета.

Общее определение понятия *семантика* - это изучение значений. Семантические технологии Web помогают выделять полезную информацию из данных, содержания документов или кодов приложений, опираясь на открытые стандарты. Если компьютер понимает *семантику* документа, то это не означает, что он просто интерпретирует набор символов, содержащихся в документе. Это значит, что компьютер понимает *смысл* документа.

При создании интернета (веба) предполагалось, что не только люди будут участвовать в обработке инф-ции, но и машины. Для машин трудно пригодна для обработки информация, представленная в вебе. У машинного робота на данный момент нет никакого алгоритма выделения семантики из текста.

Семантический веб - расширение для уже существующей сети, такое, что информация в ней снабжена точно определенным значением. Это позволяет человеку и машине более успешно взаимодействовать.

Каноническими примерами поисковых запросов в семантическом вебе явл.: «Где ближайшая библиотека?» С каждым человеком будет ассоциироваться свой собственный поисковый агент, который, опираясь на смысловое содержание страниц будет выполнять замысловатые запросы.

Семантический веб будет использоваться для:

- 1) Семантический поиск – при таком поиске будут выдаваться не те страницы, где встретилось искомое ключевое слово, а лишь те, где есть искомое понятие.
- 2) Объединение знаний – если будут существовать возможности представления знаний, то интернет превратится в глобальную БЗ.
- 3) Всепроникающие вычисления.

Семантические технологии Web очерчивают общие рамки, позволяющие осуществлять обмен данными и их многократное использование в различных приложениях, корпорациях и даже сообществах. Семантические технологии Web -это эффективный способ представления данных в интернете. Такую структуру также можно символически отождествить с базой данных, которая связана в глобальном масштабе с содержанием документов в интернете. Причем эта связь осуществляется способом, понятным компьютерам. Семантические технологии представляют значения с помощью онтологии и обеспечивают аргументацию, используя связи, правила, логику и условия, оговоренные в онтологии.

К семантическим технологиям Web относятся следующие:

1. Глобальная схема имен (URI);
2. Стандартный синтаксис описания данных (RDF);
3. Стандартные способы описания свойств данных (схема RDF);

Стандартные способы описания связей между объектами данных (онтология, определяемая с помощью онтологического языка Web (WebOntologyLanguage)).

Преимущества семантических сетей для интернета:

Интернет - это крупнейший из когда-либо существовавших информационных репозиторий, причем его содержание все время растет и представлено на самых разнообразных языках и практически во всех областях знаний. Но в конечном счете становится все труднее находить смысл во всем этом содержимом. Поисковые системы способны находить информацию, содержащую определенные слова, но эта информация не всегда оказывается именно той, что требуется. Какой-то элемент всегда оказывается упущенным. Поиск основан на содержании страниц, но не на семантическом значении этого содержания или информации о странице.

Как только будет создан семантический интернет, он даст возможность разметки всего содержания интернета, описания каждого элемента информации и обеспечения семантического значения этих элементов. Таким образом, поисковые системы становятся более эффективными, чем сейчас, а пользователи могут находить именно ту информацию, которая им необходима. Организации, оказывающие различные услуги, способны индексировать их с особым значением. А пользователи будут в состоянии оперативно находить эти услуги, используя программные средства на основе интернета, и использовать их для своей пользы или в сочетании с другими услугами.

2. Технологии Semantic Web. Архитектура Semantic Web. Описание стека технологий и как они приведут к реализации Semantic Web.

Основная идея создания семантического веба заключается в разработке специального языка, способного выражать информацию в понятной для машин форме. На таком языке можно описать как сами данные, так и отношения между этими данными и правила рассуждения. Это описание экспортируется в веб и становится пригодным для обработки машиной.

Любой другой язык имеет синтаксис и семантику.

Синтаксис определяет множество правил построения фраз языка, позволяющих отделить корректные от некорректных.

Семантика – система правил истолкования языковых конструкций. Семантика определяет смысл предложений языка.

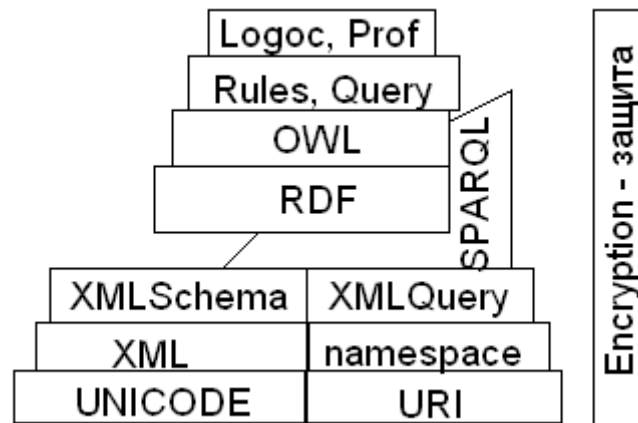
Примером языка с синтаксисом, но без семантики явл. XML. Пример языка с семантикой, но без синтаксиса – человеческая речь. Было предложено разрабатывать синтаксис и семантику языка отдельно.

Синтаксис – язык RDF – язык, отвечающий за синтаксис документов семантического веба. В нем широко используются ссылки на онтологии для определения смысла понятий.

OWL – язык описания онтологий.

Онтология- это предметная область, описание классов, объектов и их взаимосвязей для какой-то предметной области или домена.

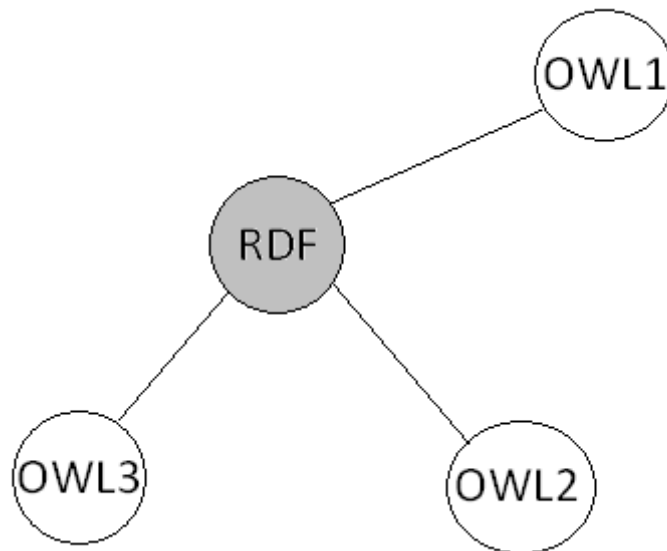
Стек технологий семантического веба:



URI описывает имя в виде ссылки куда-то. Множество URI – пр-во имен.

План реализации семантического веба:

1. Создать синтаксис для описания знаний, использующий ссылки на онтологии (уже разработан RDF);
2. Язык описания онтологий (OWL);



3. Язык для описания веб-сервисов (начато WSDL)\$
4. Инструменты чтения и разработки документов семантического веба (jena – веб-фреймворк, Protege – редактор для описания онтологий)
5. Язык запросов к знаниям, записанным на RDF (начат SPARQL – язык запросов к веб-документам)
6. Логический вывод знаний.
7. Семантическая поисковая система

Агенты семантического веба.

3.Технологии Semantic Web. XML. XMLSchema.

XML (англ.*eXtensibleMarkupLanguage* — расширяемый язык разметки; произносится [икс-эм-эль]) — рекомендованный Консорциумом Всемирной паутины язык разметки, фактически представляющий собой свод общих синтаксических правил. XML — текстовый формат, предназначенный для хранения структурированных данных (взамен существующих файлов баз данных), для обмена информацией между

программами, а также для создания на его основе более специализированных языков разметки (например, XHTML). XML является упрощённым подмножеством языка SGML. Теги в XML не заданы с самого начала, а определяются пользователем самостоятельно.

Ex: <node>

<from>Деканат</from>

<to>буриличев </to>

<data>Успеваемость </data>

</node>

XML не является заменой HTML, они создавались с различными целями.

XML фокусируется на том, чем являются данные, а HTML – как эти данные выглядят.

XML создавался для того, чтобы ничего не делать. В XML теги придумываются самим программистом. XML используется для отделения данных от представления, обмена данными, обмена фиксированной информацией, удобной организации совместной обработки данных. XML является основой для новых языков.

Одним из главных преимуществ XML является расширяемость. Она заключается в том, что если мы добавим новый тег или атрибут, то код программы, работающий со старым XML документом останется рабочим.

Для описания предварительной структуры XML документа используется технология DTD. DTD предназначена для задания разрешенных блоков XML документа.

XML Schema — язык описания структуры XML-документа. Спецификация XML Schema является рекомендацией W3C.

Как большинство языков описания XML, XML Schema была задумана для определения правил, которым должен подчиняться документ. Но, в отличие от других языков, XML Schema была разработана так, чтобы её можно было использовать в создании программного обеспечения для обработки документов XML.

После проверки документа на соответствие XML Schema, читающая программа может создать модель данных документа, которая включает:

- словарь (названия элементов и атрибутов);
- модель содержания (отношения между элементами и атрибутами и их структура);
- типы данных.

Каждый элемент в этой модели ассоциируется с определённым типом данных, позволяя строить в памяти объект, соответствующий структуре XML-документа.

Языкам объектно-ориентированного программирования гораздо легче иметь дело с таким объектом, чем с текстовым файлом.

Другим удобством XML Schema является то, что один словарь может ссылаться на другой, и, таким образом, разработчик может использовать уже существующие словари и легче устанавливать и распространять стандарты XML структуры для определённых задач (например, словарь протокола SOAP).

Файл, содержащий XML Schema, обычно имеет расширение «.xsd» (**XML Schema definition**).

Простой пример схемы на XML Schema, расположенной в файле "country.xsd" и описывающий данные о населении страны:

```
<?xmlversion="1.0"encoding="utf-8"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:elementname="страна"type="страна"/>
<xs:complexTypename="страна">
<xs:sequence>
<xs:elementname="название"type="xs:string"/>
<xs:elementname="население"type="xs:decimal"/>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

Пример документа, соответствующего этой схеме:

```
<?xmlversion="1.0"encoding="utf-8"?>
<страна
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="country.xsd">
<название>Франция</название>
<население>59.7</население>
</страна>
```

4. Технологии Semantic Web. RDF.

К семантическим технологиям Web относятся следующие:

- Глобальная схема имен (URI);
- Стандартный синтаксис описания данных (RDF);
- Стандартные способы описания свойств данных (схема RDF);

Стандартные способы описания связей между объектами данных (онтология, определяемая с помощью онтологического языка Web (WebOntologyLanguage)).

RDF это спецификация, которая определяет модель представления мира и синтаксис для сериализации и обмена этой модели. Консорциум всемирной сети (WorldWideWebConsortium, сокр. W3C) разработал XML-сериализацию для RDF. RDFXML - это стандартный формат обмена для RDF в семантической сети, хотя он не является единственным. Например, Notation3 - это отличная тестовая альтернативная сериализация.

RDF обеспечивает последовательный стандартный способ описания и работы практически с любыми интернет-ресурсами: от текстовых страниц и графиков до аудио-файлов и видео-клипов. Он предлагает синтаксические возможности для взаимодействия сетей и формирует базовый слой для создания семантической сети. RDF определяет управляемые графы связей, представленные тройками объект-атрибут-значение. Например, объект О имеет атрибут А со знач. V.

Листинг 1. Пример RDF XML

```
<?xm1 version="1.0"?>
```

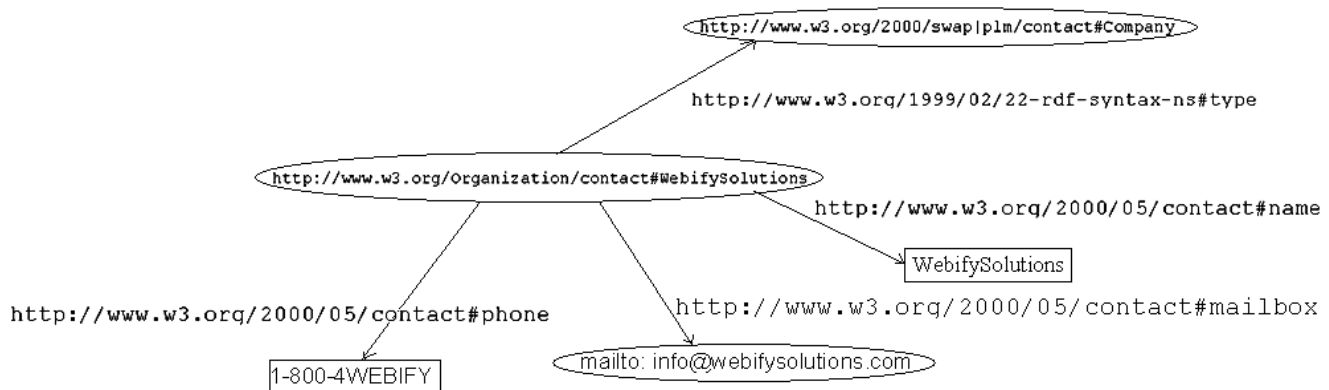
```
<rdf:RDFxmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:contact="http://www.w3.org/2000/05/contact#">
<contact:Companyrdf:about="http://www.w3.org/Organization/
contact#WebifySolutions">
<contact:name>Webify Solutions</contact:name>
<contact:mailboxrdf:resource="mailto:info@webifysolutions.com"/><contact:phone>1-
800-4WEBIFY</contact:phone>
</contact:Company>
</rdf:RDF>
```

Эл-т RDF в листинге 1 несет информацию о ресурсе, в данном сл. это компания <http://www.w3.org/Organization/contact#WebifySolutions>.

Компания может быть идентифицирована по URI <http://www.w3.org/Organization/contact#WebifySolutions> ее название - WebifySolutions, ее e-mail - info@webifysolutions.com, а номер телефона - 1-800-4WEBIFY.

На Рис. 1 показан управляемый граф связей, представляющий ту же информацию.

Рис.1. Граф RDF, опис. контактную инф. компании Webify Solutions

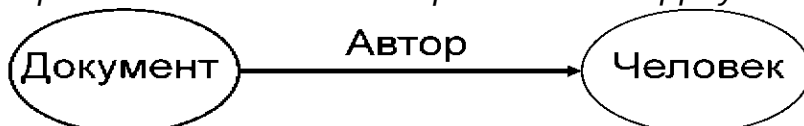


Стандартные способы описания свойств данных - схема RDF

Схема RDF- это семантическое расширение RDF. Она обеспечивает механизмы описания связанных ресурсов, а также, собственно, этих связей.

Система классов и свойств схемы RDF похожа на систему типов языков объектно-ориентированного программирования, таких, например, как Java, но отличается от многих других систем. Так, описательный язык словаря RDF определяет свойства в терминах того класса ресурсов, к которому эти свойства относятся. Другие системы же описывают класс в терминах свойств его элементов.

Определим свойство «автор» с доменом «Документ» и диапазоном «Человек».



a-la RDF (дескриптивная логика):

Класс («Документ»);

Класс («Человек»);

Свойство («автор», «Документ», «Человек»).

a-la Java:

Класс «Документ»

{

«Человек» «автор»

}

RDF и схема RDF основаны на XML и схеме XML. Существование стандартов для описания данных (RDF) и их атрибутов (схема RDF) позволяет создавать пакеты легко доступных инструментов для чтения и использования данных из многочисленных источников. То, насколько глубоко различные приложения могут обмениваться данными и использовать их, иногда называется *синтаксическим взаимодействием сетей* (syntactic interoperability). Чем более стандартизированными и распространёнными являются эти инструменты работы с данными, тем выше степень синтаксического взаимодействия сетей и тем легче и привлекательнее становится использование подхода на основе семантических сетей по сравнению с точечными интеграционными решениями.

5. Веб-сервисы (Web-Services). Определение, назначение подходы к созданию.

Веб-сервис (веб-служба) – это способ коммуникации между двумя электронными устройствами через сеть.

W3C: Веб-сервис – это программное обеспечение позволяющее производить общение между двумя устройствами через сеть. Веб-сервис имеет интерфейс описанный на машинно-понятном языке (обычно с помощью WebServicesDescriptionLanguage - WSDL).

Достоинства и недостатки:

Достоинства веб-служб:

- Веб-службы обеспечивают взаимодействие программных систем независимо от платформы
- Веб-службы основаны на базе открытых стандартов и протоколов. Благодаря использованию XML достигается простота разработки и отладки веб-служб

Использование интернет-протокола обеспечивает HTTP-взаимодействие программных систем через межсетевой экран

Недостатки веб-служб:

Меньшая производительность и больший размер сетевого трафика по сравнению с технологиями RMI, CORBA, DCOM за счёт использования текстовых XML-сообщений.

Используемые технологии:

1. HTTP (HyperTextTransferProtocol — «протокол передачи гипертекста»)
2. WSDL (WebServicesDescriptionLanguage — язык описания веб-сервисов, основанный на языке XML)
3. XML (eXtensible Markup Language — расширяемый язык разметки)
4. JSON (JavaScript Object Notation — текстовый формат обмена данными, основанный на JavaScript)

Виды веб-сервисов:

1. Remote procedure calls (RPC) - Вызов удалённых процедур

2. SimpleObjectAccessProtocol (SOAP) - Простой протокол доступа к объектам

3. Representational State Transfer (REST) - Передачасостоянияпредставления

Remoteprocedurecalls:

Класс технологий, позволяющих компьютерным программам вызывать функции или процедуры в другом адресном пространстве (как правило, на удалённых компьютерах)

Реализация RPC технологии включает в себя два компонента: сетевой протокол для обмена в режиме клиент-сервер и язык сериализации объектов.

Различные реализации RPC имеют очень отличающуюся друг от друга архитектуру и разнятся в своих возможностях: одни реализуют архитектуру SOA, другие CORBA или DCOM. На транспортном уровне RPC используют в основном протоколы TCP и UDP, однако, некоторые построены на основе HTTP.

SOAP:

Протокол обмена структурированными сообщениями в распределённой вычислительной среде.

SOAP может использоваться с любым протоколом прикладного уровня: SMTP, FTP, HTTP, HTTPS и др. Однако его взаимодействие с каждым из этих протоколов имеет свои особенности, которые должны быть определены отдельно. Чаще всего SOAP используется поверх HTTP.

Сообщение SOAP выглядит так:

SOAP-конверт

SOAP-заголовок

Элемент заголовка 1

Элемент заголовка 2

...

Элемент заголовка N

Тело SOAP

Элемент тела N

...

Элемент тела 2

Элемент тела 1

Недостатки:

Использование SOAP для передачи сообщений увеличивает их объём и снижает скорость обработки. В системах, где скорость важна, чаще используется пересылка XML-документов через HTTP напрямую, где параметры запроса передаются как обычные HTTP-параметры.

Хотя SOAP является стандартом, некоторые программы часто генерируют сообщения в несовместимом формате. Например, запрос, сгенерированный AXIS-клиентом, не будет понят сервером WebLogic.

REST:

Подход к архитектуре сетевых протоколов, обеспечивающих доступ к информационным ресурсам.

Данные должны передаваться в виде небольшого количества стандартных форматов (например HTML, XML, JSON).

6. Особенности поисковых систем. Структура организации и алгоритм работы.

1) Поисковый робот: Как браузер просматривает страницы, анализируя и выделяя существенные компоненты и отправляет их в специальном виде на сервера поисковика, далее по ссылкам переходит на другую страницу. Порядок обхода страниц, частота визитов и алгоритм выделения значимой информации зависит от конкретного языка.

Индексация – выделение наиболее значимых термов из документа и сохранение их в специальном виде (Под термом понимают все слова или понятия) $D \rightarrow \{t_1, t_2, \dots, t_n\}$

Обратная индексация – для многих термов ищутся документы, в которых они встречаются: $t_1 \rightarrow D_1, D_2, \dots, D_m$.

Ранжирование – упорядочивание по важности. Поиск определяет множество наиболее релевантных (соответствующих запросу) документов и выдает их в вершине поиска.

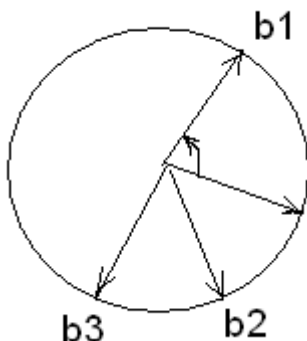
Наивный алгоритм ранжирования:

- 1) частота встречающегося терма
- 2) определение взвешенной суммы терма среди других слоев.

Модели поиска:

Подходы:

- 1) Логическая модель: $t_1 \text{ and } t_2 \text{ or } t_3 \text{ not } t_4$
- 2) векторная модель



- 3) Вероятностный поиск – Доказательство оценивается по вероятности пользователя встретить в нем свой поисковый запрос.

7.Язык запросов поисковых систем.

Язы́к за́просов — это искусственный язык, на котором делаются запросы к базам данных и другим информационным системам, особенно к информационно-поисковым системам.

Примеры:

- SQL — де-факто стандартный язык запросов к реляционным базам данных.
- Language Integrated Query — расширение для некоторых языков программирования в .NET Framework, добавляющее к ним SQL-подобный язык запросов.
- XQuery — язык запросов, разработанный для обработки данных в формате XML.
- XPath— язык запросов к элементам XML-документа.

Язык, на котором формулируются запросы к поисковым системам называется языком поисковых запросов или информационно-поисковым языком.

В различных поисковых системах язык запросов может различаться, однако обычно он является некоторым подобием языка регулярных выражений с дополнениями, связанными со спецификой работы той или иной поисковой системы. Знание и правильное применение языка запросов конкретной поисковой машины улучшает и упрощает пользователю процесс поиска информации.

Язык поисковых запросов состоит из логических операторов, префиксов обязательности, возможности учета расстояния между словами, морфологии языка, регистра слов, расширенных операторов, возможностей расширенного поиска, уточнения поиска.

Специальные операторы:

allintitle: чтобы результаты (терм-запрос) находились в заголовке

related: поиск в найденном

allinURL: ссылка с запросом

Ex: Related: цветы ~ полевые

^

|

синонимы

inURL: поиск найденного запроса в тексте гиперссылки.

site: поиск в каком-либо сайте

link: поиск по сайтам, кот. ссылаются на данный запрос. (link: адрес)

into: адрес – выдает информацию о данном адресе

define: слово – поиск толкования

safesearch: безопасный поиск

Filetype: тип функции

Scholar – поиск в научных статьях и документах

Cache: адрес|слово

allintext: поиск ключевых слов только в тексте страницы

allinlinus: слова – поиск по названию гиперссылки

8.Семантическая поисковая система.

Семантическая поисковая система:

Направлена на улучшение качества поиска и понимания поисковых запросов пользователя, используя конкретные значения понятий в пространстве поиска. Вместо использования ранжирования для определения релевантности страниц семантические поисковые системы используют семантику для определения релевантности запроса. Более того семантический поиск подразумевается как извлечение знаний из веб-онтологий.

Ключевая идея семантического поиска в том, что предполагает, что сайт, чья семантика будет описана лучше, будет иметь преимущества в выдаче поискового запроса.

Основные методологии семантического поиска:

1. RDFтраверс по RDFпутям
2. Miningключевых слов на концепте
3. Локализация – выдача по локальной позиции
4. Логический вывод на онтологиях
5. использование нечеткой логики и нечетких отношений

Примеры семантического поиска:

Freebase – полуоткрытая-полукоммерческая БЗ состоящая из множества онтологий

Nakia – полностью открытая БЗ

Powerbase – поиск по Википедии

9.Онтологии в семантической паутине. Общее определение. Классификация онтологий.

Термин «Онтология» используется в нескольких областях знаний по-разному:

1. О – философская дисциплина, изучающая бытие и сущности.
2. О – это стр-ра, описывающая значения элементов некоторой предметной области.
3. В агентно-ориентированном подходе О- теория того, какие сущности могут существовать у уже хорошо осведомленного агента.

Для создания онтологий используется процесс концептуализации (упрощенное видение мира, представленное для каких-то целей).

В общем виде стр-ра онтологии представляет собой набор из 4 элементов:

- 1) понятие
- 2) отношение
- 3) аксиома
- 4) отдельные экземпляры

Понятие рассматривается как концептуализация класса всех представителей некоторой сущности или явления. Классы явл. Общими категориями, которые могут быть упорядочены иерархически. Каждый класс описывает группы индивидуальных сущностей, кот. Объединены на основании общих свойств. Классы м. быть связаны разными отношениями. Самым ключевым отношением является отношение наследования.

Аксиомы задают условия соотнесения понятий и отношений. Они выражают какие-то утверждения, вводимые в онтологию в готовом виде, из которого могут быть выведены другие утверждения. Аксиомы позволяют выразить ту информацию, которая не может быть выражена в онтологии посредством построения иерархии понятий и установления отношений между ними.

Экземпляр- это отдельный представитель класса или категории.

У каждого класса есть значения, называемые слотами и на эти слоты накладываются ограничения, называемые аспектами или гранями.

Онтология – формальное явное описание понятий в рассмотренной предметной области, свойств каждого понятия, описывающий различные атрибуты и поведения (слоты) и ограничения, накладываемые на слоты.

Онтология вместе с конкретными экземплярами и составляет БЗ.

Критерии оценки Онтологии:

- **Ясность:** онтология должна быть ясной и легко передавать подразумеваемый смысл. Она должна быть объективной;
- **Последовательность:** в ней должны содержаться утверждения, которые не противоречат друг другу, иерархии понятий, связывающим их отношениям, экземплярам.
- **Возможность расширения:** наличие возможности введения новых элементов без пересмотра остальных элементов;
- **Минимальная степень специализации онтологии:** нежелательность полного подчинения онтологии конкретной задаче, что может осложнить ее последующее использование в других задачах.

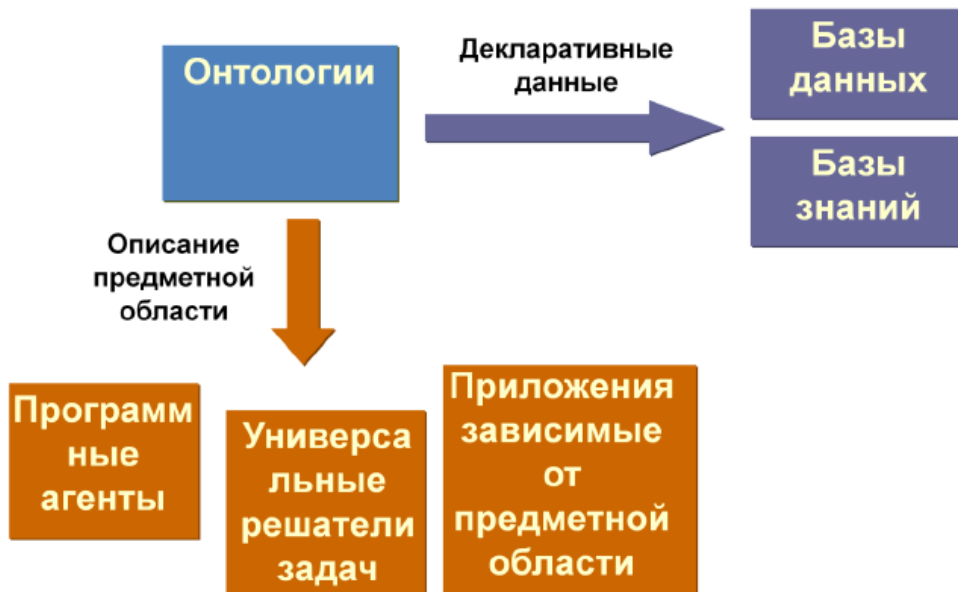
Классификация онтологий:

- 1) Онтологии верхнего уровня- выделяют только наиболее общие и абстрактные понятия (ex. Сущность, объект)
- 2) Онтологии средней зоны – представляют собой мир в целом и в какой-то его конкретной области. Как правило, содержат от 500 до 1000 понятий (концептов)

Онтологии нижней зоны – соотносятся с конкретной предметной областью и описывают ее специфику. Содержат большое количество аксиом данной предметной области, строятся как правило экспертами.

10. Процесс разработки онтологий для семантической паутины.

- Онтологии необходимо разрабатывать затем, чтобы:
- Разделять общее понимание о структуре информации между людьми и между программными агентами.
- Получить возможность повторного использования знания и той или иной предметной области. Это оберегает от изобретения велосипедов, а так же вводит стандарты в данной области, обеспечивающие совместимость.
- Чтобы явно записать, что мы знаем о какой-то области
- Чтобы отделить знания об области от процедур. Дает повторное использование знаний и повторное использование процедур.



Разработка онтологий находится на более высоком уровне абстракции, чем ООП проектирование.

Разработка онтологий	ООП проектирование
Отражает строение мира	Отражает построение кода и данных
фокусируется на структуре концептов	фокусируется на поведении (методы) и передаче сообщений между объектами
Физическое представление не важно	описывает физическоепредставл. данных (int, char, etc.)

Построение онтологий может быть в общих чертах описано при помощи следующего итеративного процесса, в котором каждые из нескольких этапов могут выполняться параллельно и по неск-ко раз.



Шаг 1. Определение области и масштаба онтологии

- Какую область будет охватывать онтология?
- Для чего мы собираемся использовать онтологию?
- На какие типы вопросов должна давать ответы инф-я в онтологии?
- Кто будет использовать и поддерживать онтологию?
- Онтология не должна содержать ВСЕ возможные знания о домене
она требуется уточнять или обобщать более, чем необходимо она требуется включать все возможные свойства классов

Шаг 2. Рассмотрение вариантов повторного использования существующих онтологий

- Библиотеки онтологий
 1. DAML ontology library
 2. Protege ontology library
- Онтологии верхнего уровня
 1. IEEE Standard Upper Ontology
 2. Cyc
- Общие онтологии
 1. DMOZ
 2. WordNet
- Специальные онтологии

1. UMLSemanticNet 2. GO (Gene Ontology)

- Библиотека онтологий Ontolingua
- Общедоступные коммерческие онтологии (например, UNSPSC, RosettaNet, DMOZ)

Шаг 3. Перечисление важных терминов в онтологии

- Какие термины являются ключевыми
- Каковы свойства этих понятий
- Что мы хотим сказать (выразить) этими терминами?
- неформально сгруппировать
- перефразировать и уточнить термины, чтобы получить неформальные определения концептов
- нарисовать неформальную диаграмму

Шаг 4. Определение классов и иерархии классов

- Нисходящая разработка - Начиная с общих понятий
- Восходящая разработка - Движение от конкретных классов
- Комбинированная разработка
- Организация классов в таксономию (установка отношений наследования)

а. Класс может иметь множество суперклассов (множ. насл.)

б. Все родственные понятия в иерархии классов должны быть на одном уровне иерархии

в. Имена классов должны быть или все в единственном числе или все в множественном числе (Animal is not a kind of Animals).

- Документация классов

Шаг 5. Определение свойств классов - слотов

- Подкласс наследует все слоты от всех суперклассов.

Шаг 6. Определение факторов слотов (ограничений на слоты)

- Тип значений

1. Строка 2. Число 3. Булево значение 4. Нумерованные слоты (один из множества)

о Слоты-экземпляры - ссылка на другой класс

- Количество значений (кардинальность)
- Свойства значений 1. Минимум, максимум 2. Значение по умолчанию.

Шаг 7. Создание экземпляров - 1. выбрать класс, 2. создать отдельный экземпляр этого класса 3. ввести значения слотов.

11. Технологии Semantic Web. OWL.

Семантические технологии Web - это эффективный способ представления данных в интернете.

К семантическим технологиям Web относятся следующие:

1. Глобальная схема имен (URI);
2. Стандартный синтаксис описания данных (RDF);
3. Стандартные способы описания свойств данных (схема RDF);
4. Стандартные способы описания связей между объектами данных (онтология, определяемая с помощью онтологического языка OWL (Ontology Web Language)).

Онтологический язык Web (WebOntologyLanguage), рекомендуемый консорциумом W3C, помогает в выражении онтологий. Рабочий OWL добавляет больше словарных возможностей для описания свойств и классов, чем RDF или схема RDF. В частности, он позволяет описывать связи между классами (например, неперекрываемость), мощность множества (например, "ровно один"), равенство, более богатую типологию свойств и их характеристики (например, симметрия).

Онтологический язык Web на основе OWL разработан для использования приложениями, которые должны работать с содержанием информации, а не просто предоставлять ее пользователю. OWL улучшает возможности автоматической интерпретации содержимого интернета по сравнению с теми, что могут обеспечить XML, RDF и схема RDF. Это происходит благодаря тому, что OWL предоставляет дополнительные словарные возможности наряду с формальной семантикой. OWL включает три подязыка: полный OWL (OWLFull), OWL DL и облегченный OWL (OWLLite) (перечислены в порядке убывания их выразительных возможностей).

Полная версия онтологического языка Web на основе OWL называется **owlFull**. Этот язык использует все базисные элементы языка OWL и позволяет комбинировать их случайным образом с RDF и схемой RDF. Полный OWL совместим "снизу вверх" с RDF, как синтаксически, так и семантически: любой разрешенный документ RDF является также разрешенным документом OWLFull. Маловероятно, что какие-либо интеллектуальные программные средства способны поддерживать все возможности OWLFull, поскольку этот язык предлагает максимум выразительных средств и синтаксической свободы RDF при отсутствии вычислительных гарантий.

OWL DL предназначен для тех пользователей, кому необходим максимум выразительных средств без потери вычислительных возможностей. OWL DL - это подязык конструкций языка OWLFull с некоторыми ограничениями, такими как *разделение типов* (typeseparation) (например, класс не может быть одновременно индивидуальным элементом или свойством, а свойство не может одновременно быть индивидуальным элементом или классом).

OWLLite предназначен для пользователей, которым необходима классификационная иерархия и простые ограничительные возможности. Преимуществом этого языка являются большая легкость его понимания и внедрения по сравнению с двумя другими. Но в то же время его выразительные возможности гораздо ниже. Например, хотя OWLLite и поддерживает ограничения мощности множества, единственными допустимыми значениями этого параметра являются 0 или 1

Примерами онтологий являются каталоги сайтов интерактивных покупок, таких как Amazon.com, стандартные терминологии той или иной области деятельности, например, UNSPSC - TheUnitedNationsStandardProductsandServicesCode (система стандартных продуктов и услуг ООН), или различные таксономические системы интернета, такие как категории сайта "MyYahoo".

12. Компоненты онтологического языка Web на основе OWL.

Основные компоненты OWL включают *классы, свойства и индивидуальные элементы*.

Классы - это основные блоки онтологии OWL. Класс - это концепция в домене. Классы обычно образуют таксономическую иерархию (т.е. систему подкласс-надкласс). Классы определяются с помощью элемента owl:Class. В языке OWL существует два заранее определенных класса: owl:Thing и owl:Nothing. Первый из них является наиболее общим и включает все, второй - это пустой класс. Любой класс, определяемый пользователем, является подклассом класса owl:Thing и надклассом класса owl:Nothing. Примеры классов в области банковского дела могут включать классы Account или Customer.

Листинг 2. Пример класса OWL:

```
<owl:Class rdf:ID="SavingsAccount">  
<rdfs:subClassOf rdfs:resource="#Account"/>  
</owl:Class>
```

Код в листинге 2 указывает, что элемент SavingsAccount - это класс, являющийся подклассом класса Account.

OWL поддерживает шесть основных способов описания классов. Самый простой - это класс с именем (named). Другие типы - это классы пересечений (intersection), объединений (union), дополнений (complement), ограничений (restrictions) и классы перечислений (enumerated). В листинге 2 представлены два из этих способов описания классов: класс ограничений определяет SavingsAccount как подкласс класса с именем Account.

Свойства включают две основные категории:

свойства объекта (Object properties), которые связывают индивидуальные элементы между собой;

свойства типов данных (Datatype properties), которые связывают индивидуальные элементы со значениями типов данных, такими как целые числа, числа с плавающей запятой и строки. Для определения типов данных OWL использует схему XML. Свойство может включать домен и некоторую область, связанную с ним. Любое свойство попадает в одну из следующих категорий:

функциональная: для любого объекта свойство может принимать только одно значение (например, возраст, рост или вес человека);

обратно-функциональная: два различных индивидуальных элемента не могут иметь одно и то же значение. Например, у каждого человека свой уникальный номер банковского счета bankNumber или так называемый SSN (social security number);

симметричная: если свойство связывает элемент A с элементом B, то из этого можно сделать вывод, что оно также связывает элемент B с элементом A. Примеры симметричных свойств включают выражения типа "является братом (сестрой)" или "такой же, как";

транзитивная: если свойство связывает элемент A с элементом B, а элемент B с элементом C, то можно предположить, что оно также связывает элемент A с элементом C. Например, если A выше B, а B выше C, то A выше C.

К классам и свойствам могут применяться различные ограничения. Например, ограничения мощности множества указывают на число связей, в которых может участвовать класс или индивид. элемент.

Индивидуальные элементы - это элементы классов; свойства могут связывать их друг с другом. Например, индивидуальный элемент Smith может быть описан как элемент, принадлежащий классу Person (индивидуум).

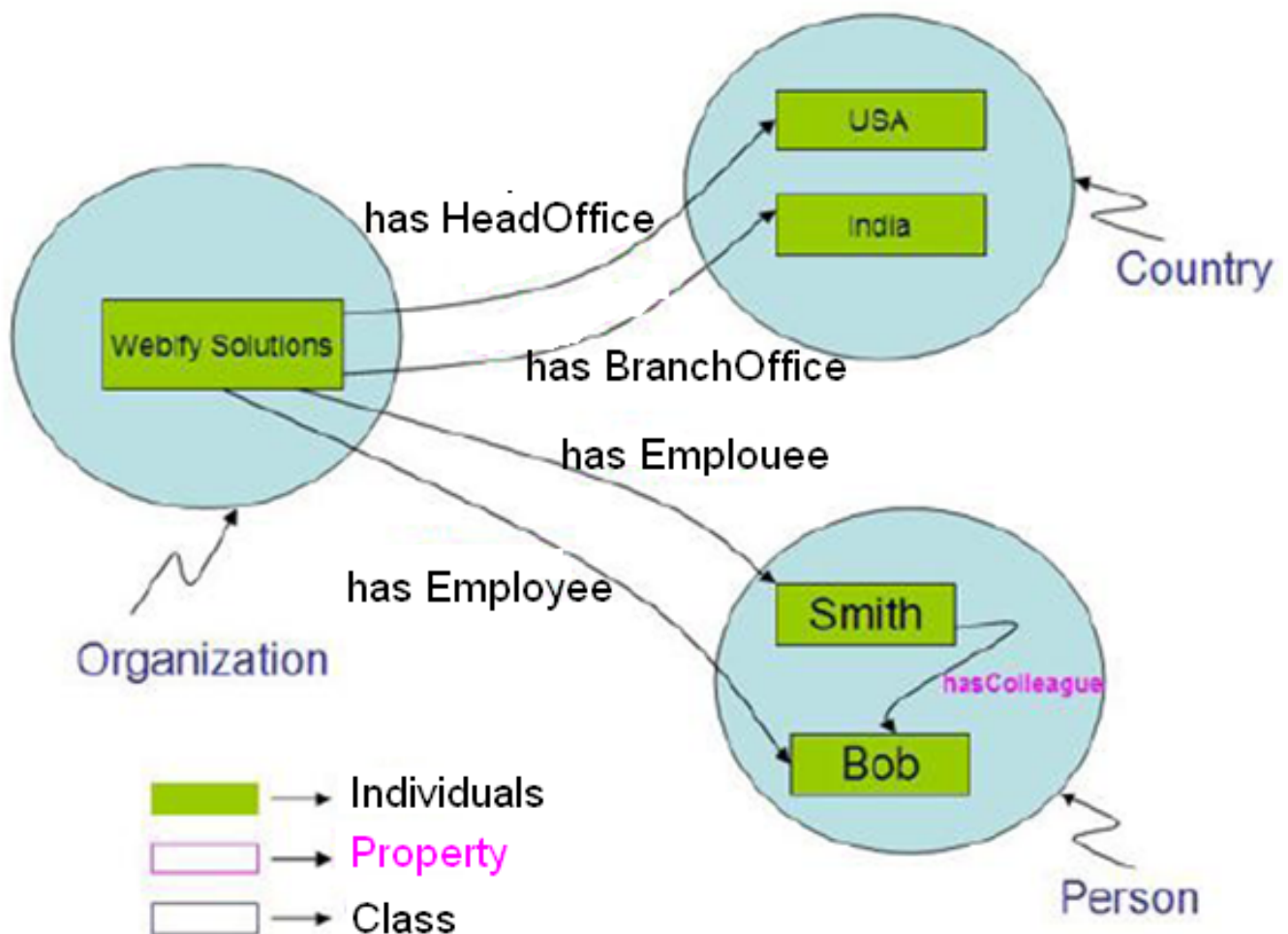
Свойство hasEmployer (имеет работодателя) может связывать его с другим индивидуальным элементом - WebifySolutions, указывая, таким образом, что Smith работает в компании WebifySolutions.

Листинг 3. Индивидуальный элемент OWL:

```
<owl:Thing rdf:about="SmithAccount">
<rdf:type="#Account"/>
</owl:Class>
```

Элемент rdf:type - это свойство RDF, которое связывает индивид.элемент с тем классом, к которому он принадлежит. Листинг 3 указывает, что элемент SmithAccount принадлежит к типу Account.

Рис. 2. Онтология OWL, описывающая организационную структуру компании WebifySolutions



13. Применение онтологий. Значение онтологий для бизнеса.

Онтологии могут быть использованы при решении следующих задач:

- ✓ для совместного применения людьми или программными агентами
- ✓ для возможности накопления и повторного использования знаний в предметной области
- ✓ для создания моделей и программ, оперирующих онтологиями.

- ✓ для "извлечения значимой информации из web-страниц при индексировании". В некоторой степени данная технология частично закрывает потребность в семантическом поиске.

Часто онтологии используют в качестве:

- *"Словаря предметной области"*. Онтология содержит общую терминологическую базу предметной области, поэтому разработчики программного обеспечения могут использовать термины из онтологии для документирования своего продукта и для формирования пользовательского интерфейса, в том числе и многоязычного.
- *"Отображения на базу данных"*. Онтология предоставляет набор базовых терминов предметной области, с которыми приходится иметь дело в любом процессе измерения. Поэтому онтология является удобным базисом для разработки схемы данных измерительной системы. Она не является полной, поскольку любой конкретный процесс измерения имеет частные особенности, не задаваемые на уровне онтологии. Тем не менее, она определяет базовые понятия, которые в той или иной форме присутствуют или должны присутствовать в любой схеме данных.
- *"Формата хранения метаданных"*. Свойства онтологических терминов определяют состав и формат представления метаданных, содержащихся в системе. Эффективная поддержка метаданных является одной из ключевых задач инженерии информационных систем. Привлечение онтологии позволяет повысить эффективность реализации различных средств обработки данных благодаря формированию богатых массивов метаинформации в машинно-читаемой форме.
- *"Формата обмена данными"*. Открытые форматы обмена данными с внешними системами, основанные на онтологии, существенно упрощают задачу интеграции систем, относящихся к различным областям либо созданных различными разработчиками.

Значение онтологий для бизнеса:

IT-системы организуют значения с помощью реляционных моделей данных, линейных файлов, объектно-ориентированных моделей или специально разработанных моделей данных. Время от времени, в связи с изменениями бизнес-требований, возникает необходимость добавления новых элементов и связей в реляционные модели данных или объектно-ориентированные модели.

Более того, если организация использует множественные приложения от различных поставщиков, то придется копировать одни и те же модели во все базы данных приложений. Например, банк предлагает набор различных продуктов для обслуживания разнообразных категорий клиентов. Корпоративному клиенту может потребоваться услуга по обнаружению мошенничества, а обычному потребителю окажется достаточно функциональных возможностей интерактивного осуществления банковских операций с помощью интернета. Обычно банк приобретает приложения у нескольких поставщиков, но каждое из них повторяет одну и ту же общую информацию - номера счетов, имена клиентов и т. д. - в своей базе данных. По мере того как

организация добавляет новые продукты для удовлетворения растущих запросов бизнеса, одна и та же избыточная информация распространяется по всей корпорации. Целый ряд услуг является общим для всех приложений, например, просмотр банковских транзакций и электронных переводов. Каждая из этих услуг также дублируется в формате, присущем тому или иному приложению, что ведет к необходимости осуществления точечной интеграции.

Если же банк принимает подход, основанный на онтологии, то он может собирать и представлять общую информацию о продуктах в нейтральной по отношению к языку форме и сохранять эту информацию в центральном репозитории. С помощью такой общей адаптированной онтологии организация может обеспечивать единое стандартизированное представление данных для всех приложений. Такое стандартизированное представление позволяет точно извлекать необходимую информацию и без проблем осуществлять корпоративную интеграцию, поскольку бизнес-процессы и различные источники данных могут быть связаны ("отображены") друг с другом с помощью общей мета-модели. Таким образом, общая онтология исключает необходимость в точечной интеграции и упрощает интеграцию приложений, сокращая избыточность данных и обеспечивая одно и то же семантическое значение для всех приложений, что облегчает поддержание функционирования банка и его обновление.

Вопросы для самоконтроля и текущего контроля

- 1) Дайте определение Semantic Web
- 2) Перечислите преимущества семантических сетей для интернета.
- 3) Опишите архитектуру Semantic Web
- 4) Опишите технологию XMLSchema
- 5) Опишите технологию RDF
- 6) Опишите технологию OWL
- 7) Дайте определение Web-Services
- 8) Перечислите особенности поисковых систем
- 9) Расскажите о назначении языков запросов поисковых систем.
- 10) Укажите в чём отличие семантической поисковой системы от обычной поисковой системы.
- 11) Опишите как применяются онтологии в семантической паутине.

Задание на самостоятельную работу

- 1) Практическое задание

Выполнить создание спецификации знаний о предметной области в виде онтологии в редакторе Protege 5.5.0 согласно заданию практической работы №4.

- 2) Самостоятельная работа

Исследовать синтаксис и лексику современных языков создания онтологий.

Список рекомендуемой литературы¹:

[1] Сидоркина, Ирина Геннадьевна. Системы искусственного интеллекта [Текст] : учеб. пособие для вузов по направлению 230100 "Информатика и вычисл. техника" / И. Г. Сидоркина. М.: Кнорус, 2011. - 245 с.

Рассмотрены теоретические аспекты проектирования систем искусственного интеллекта и даны модели представления знаний, изложены материалы информационного, справочного, консультирующего характера по использованию инструментальных средств и технологическим особенностям разработки систем данного класса.

Подробнее: <https://www.labirint.ru/books/276217/>

[2] Богомолова, М. А. Экспертные системы (техника и технология проектирования) [Текст] : Методические указания к лабораторным работам / М. А. Богомолова. Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015. - 47 с.

Методические указания являются руководством при выполнении лабораторных занятий, проводимых по курсу «Экспертные системы (техника и технология проектирования)» с магистрами направления 09.04.03 - «Прикладная информатика» в терминальном классе. Лабораторные работы связаны с изучением принципов построения экспертных систем на основе различных моделей представления знаний, и могут быть полезны студентам других специальностей.

Подробнее: <https://www.iprbookshop.ru/71908.html>

[3] Советов, Борис Яковлевич. *Представление знаний в информационных системах [Текст] : [учеб. для вузов по направлению подгот. "Информ. системы и технологии"]* / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. М.: Академия, 2011.

Учебник создан в соответствии с Федеральным государственным образовательным стандартом по направлению бакалавриата «Информационные системы и технологии». Рассмотрены современное состояние теоретических и прикладных вопросов представления знаний в информационных системах, идеология построения интеллектуальных систем, математический аппарат представления знаний, возможности и пути использования искусственного интеллекта при проектировании информационных систем. Изложены новые аспекты представления знаний на основе искусственных нейронных сетей, расчетно-логических систем, генетических алгоритмов, мультиагентных систем. Приведены примеры практической реализации представления знаний на базе декларативного языка Пролог. Для студентов учреждений высшего профессионального образования. Может быть полезен разработчикам и пользователям информационных систем; преподавателям и научным сотрудникам, сферой интересов которых является интеллектуализация различных предметных областей; менеджерам и руководителям различного ранга, желающим самостоятельно ознакомиться с современным состоянием информационных технологий.

Подробнее: <https://lavkababuin.com/predstavlenie-znaniy-v-informacionnyh-sistemah-2-e-izd-ster-650564/>

[4] Остроух, А. В. *Интеллектуальные информационные системы и технологии [Электронный ресурс] : монография* / А. В. Остроух, А. Б. Николаев. Санкт-Петербург: Лань, 2019. - 308 с.

В монографии изложены концептуальные основы и методы представления знаний в интеллектуальных системах. Рассмотрены различные подходы, применяемые при проектировании и разработке интеллектуальных систем и технологий в транспортном комплексе, а также рассмотрены тенденции развития систем искусственного интеллекта.

Монография может быть использована для формирования профессиональной компетентности студентов высших учебных заведений, аспирантов и научных сотрудников обучающихся и ведущих научные исследования в области разработки и практического применения систем искусственного интеллекта по укрупнённой группе направлений подготовки "Информатика и вычислительная техника".

Подробнее: <https://www.labirint.ru/books/692861/>

Ссылки на открытые ресурсы (он-лайн-курсы, видеоуроки и т.д.)

[5]

ⁱСписок рекомендуемой литературы может быть более широким по сравнению со списком литературы, указанным в рабочей программе

Прикладные интеллектуальные системы

Конспект лекции 11. Применение редактора Protege 5.5.0 для создания и наполнения онтологий

Аннотация: В лекции рассказывается о построении онтологий с помощью редактора онтологий Protege.

План лекции:

1. Зачем создавать онтологию?
2. Из чего состоит онтология?
3. Простая методология инженерии знаний
4. Определение классов и иерархии классов
5. Определение свойств
6. Имена понятий в онтологиях
7. Другие ресурсы для создания онтологий
8. Задания для самостоятельной работы.

Ключевые слова: Онтология, языки создания онтологий, система Protege, RDF, OWL, классы, понятия, свойства, отношения

СОДЕРЖАНИЕ ЛЕКЦИИ:

1. Зачем создавать онтологию?

В последние годы разработка онтологий - формальных явных описаний терминов предметной области и отношений между ними (Gruber 1993) – переходит из мира лабораторий по искусственному интеллекту на рабочие столы экспертов по предметным областям. Во всемирной паутине онтологии стали обычным явлением. 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). Консорциум WWW (W3C) разрабатывает RDF (Resource Description Framework) (Brickley and Guha 1999), язык кодирования знаний на веб-страницах, для того, чтобы сделать их понятными для электронных агентов, которые осуществляют поиск информации. Управление перспективных исследований и разработок министерства обороны США (The Defense Advanced Research Projects Agency, DARPA) в сотрудничестве с W3C разрабатывает Язык Разметки для Агентов DARPA (DARPA Agent Markup Language, DAML), расширяя RDF более выразительными конструкциями, предназначенными для облегчения взаимодействия агентов в сети (Hendler and McGuinness). Во многих дисциплинах сейчас разрабатываются стандартные онтологии, которые могут использоваться экспертами по предметным областям для совместного использования и аннотирования информации в своей области. Например, в области медицины созданы большие стандартные, структурированные словари, такие как snomed (Price and Spackman) и семантическая сеть Системы Унифицированного Медицинского Языка (the Unified Medical Language System) (Humphreys and Lindberg 1993). Также появляются обширные общецелевые онтологии. Например, Программа ООН по развитию (the United Nations Development Program) и компания Dun & Bradstreet объединили усилия для разработки онтологии UNSPSC, которая предоставляет терминологию товаров и услуг (<http://www.unspsc.org/>).

Онтология определяет общий словарь для ученых, которым нужно совместно использовать информацию в предметной области. Она включает машинно-интерпретируемые формулировки основных понятий предметной области и отношения между ними.

Почему возникает потребность в разработке онтологии? Вот некоторые причины:

- Для совместного использования людьми или программными агентами общего понимания структуры информации.
- Для возможности повторного использования знаний в предметной области.
- Для того чтобы сделать допущения в предметной области явными.
- Для отделения знаний в предметной области от оперативных знаний.
- Для анализа знаний в предметной области.

Совместное использование людьми или программными агентами общего понимания структуры информации является одной из наиболее общих целей разработки онтологий (Musen 1992; Gruber 1993). К примеру, пусть, несколько различных веб-сайтов содержат информацию по медицине или предоставляют информацию о платных

медицинских услугах, оплачиваемых через Интернет. Если эти веб-сайты совместно используют и публикуют одну и ту же базовую онтологию терминов, которыми они все пользуются, то компьютерные агенты могут извлекать информацию из этих различных сайтов и накапливать ее. Агенты могут использовать накопленную информацию для ответов на запросы пользователей или как входные данные для других приложений. Обеспечение возможности использования знаний предметной области стало одной из движущих сил недавнего всплеска в изучении онтологий. Например, для моделей многих различных предметных областей необходимо сформулировать понятие времени. Это представление включает понятие временных интервалов, моментов времени, относительных мер времени и т.д. Если одна группа ученых детально разработает такую онтологию, то другие могут просто повторно использовать ее в своих предметных областях. Кроме того, если нам нужно создать большую онтологию, мы можем интегрировать несколько существующих онтологий, описывающих части большой предметной области. Мы также можем повторно использовать основную онтологию, такую как UNSPSC, и расширить ее для описания интересующей нас предметной области.

Создание явных допущений в предметной области, лежащих в основе реализации, дает возможность легко изменить эти допущения при изменении наших знаний о предметной области. Жесткое кодирование предположений о мире на языке программирования приводит к тому, что эти предположения не только сложно найти и понять, но и также сложно изменить, особенно непрограммисту. Кроме того, явные спецификации знаний в предметной области полезны для новых пользователей, которые должны узнать значения терминов предметной области.

Отделение знаний предметной области от оперативных знаний – это еще один вариант общего применения онтологий. Мы можем описать задачу конфигурирования продукта из его компонентов в соответствии с требуемой спецификацией и внедрить программу, которая делает эту конфигурацию независимой от продукта и самих компонентов (McGuinness and Wright 1998). После этого мы можем разработать онтологию компонентов и характеристик ЭВМ и применить этот алгоритм для конфигурирования нестандартных ЭВМ. Мы также можем использовать тот же алгоритм для конфигурирования лифтов, если мы предоставим ему онтологию компонентов лифта (Rothenfluh et al. 1996).

Анализ знаний в предметной области возможен, когда имеется декларативная спецификация терминов. Формальный анализ терминов чрезвычайно ценен как при попытке повторного использования существующих онтологий, так и при их расширении (McGuinness et al.).

Часто онтология предметной области сама по себе не является целью. Разработка онтологии сродни определению набора данных и их структуры для использования другими программами. Методы решения задач, доменно-независимые приложения и программные агенты используют в качестве данных онтологии и базы знаний, построенные на основе этих онтологий. К примеру, в этой статье мы разрабатываем онтологию вин и еды, а также подходящие комбинации вин и блюд. Затем эту онтологию можно будет использовать как основу для приложений в наборе

инструментов для управления рестораном: Одно приложение могло бы составлять список вин для меню на текущий день или отвечать на запросы официантов и посетителей. Другое приложение могло бы анализировать инвентарный перечень винного погреба и предлагать категории вин для пополнения и конкретные вина для закупки к следующим меню или для поваренных книг.

Пример вина и еды, который мы используем на протяжении всей статьи, сделан на основе примерной базы знаний, которая представлена в статье, описывающей CLASSIC – систему представления знаний, основанную на описательно-логическом подходе (Brachman et al. 1991). В учебном пособии по CLASSIC (McGuinness et al. 1994) этот пример получил дальнейшее развитие. Protege- и другие фреймовые системы описывают онтологии декларативным образом, определяя явным образом, какова классовая иерархия и к каким классам принадлежат индивидуальные концепты. Некоторые идеи по разработке онтологий в этом руководстве берут свое начало в литературе по объектно-ориентированному проектированию (Rumbaugh et al. 1991; Booch et al. 1997). Однако разработка онтологий отличается от проектирования классов и отношений в объектно-ориентированном программировании. Объектно-ориентированное программирование сосредотачивается главным образом на методах классов – программист принимает проектные решения, основанные на *операторных* свойствах класса, тогда как разработчик онтологии принимает эти решения, основываясь на *структурных* свойствах класса. В результате структура класса и отношения между классами в онтологии отличаются от структуры подобной предметной области в объектно-ориентированной программе.

Невозможно охватить все трудности, которые, возможно, придется преодолеть разработчику онтологии, и в этом руководстве мы не пытаемся затронуть их всех. Вместо этого мы пытаемся дать отправную точку, исходное руководство, которое могло бы помочь неопытному проектировщику онтологий в их разработке. В конце мы предлагаем источники, в которых можно посмотреть пояснения к более сложным структурам и механизмам разработки, если они потребуются для предметной области.

В конечном счете, единственной правильной методологии разработки онтологий не существует, и мы не пытались определить таковую. Представленные здесь идеи мы сочли полезными, исходя из нашего опыта разработки онтологий. В конце этого руководства мы предлагаем список ссылок на альтернативные методологии.

2. Из чего состоит онтология?

В литературе по искусственному интеллекту содержится много определений понятия онтологии, многие из которых противоречат друг другу. В этой статье **онтология** – формальное явное описание понятий в рассматриваемой предметной области (**классов** (иногда их называют **понятиями**)), свойств каждого понятия, описывающих различные свойства и атрибуты понятия (**слотов** (иногда их называют **ролями** или **свойствами**)), и ограничений, наложенных на слоты (**фацетов** (иногда их называют **ограничениями ролей**)). Онтология вместе с набором индивидуальных **экземпляров** классов образует **базу знаний**. В действительности, трудно определить, где

кончается онтология и где начинается база знаний.

В центре большинства онтологий находятся классы. Классы описывают понятия предметной области. Например, класс вин представляет все вина. Конкретные вина – экземпляры этого класса. Вино Bordeaux в бокале перед вами, когда вы читаете этот документ, – это экземпляр класса вин Bordeaux. Класс может иметь **подклассы**, которые представляют более конкретные понятия, чем надкласс. Например, мы можем разделить класс всех вин на красные, белые и розовые вина. В качестве альтернативы мы можем разделить класс всех вин на игристые и не игристые вина. Слоты описывают свойства классов и экземпляров: вино Chateau Lafite Rothschild Pauillac - крепкое, оно производится на винном заводе Chateau Lafite Rothschild. У нас есть два слота, которые описывают вино в этом примере: слот крепость со значением «крепкое» и слот производитель со значением «винный завод Chateau Lafite Rothschild». Мы можем сказать, что на уровне класса у экземпляров класса Вино есть слоты, которые описывают вкус, крепость, уровень сахара, производителя вина и т.д. [1]

Все экземпляры класса Вино и его подкласс Pauillac имеют слот производитель, значение которого является экземпляром класса Винный завод (Рис. 1). Все экземпляры класса Винный завод имеют слот производит, относящийся ко всем винам (экземплярам класса Вино и его подклассов), которые производятся на этом заводе. На практике разработка онтологии включает:

- определение классов в онтологии;
- расположение классов в таксономическую иерархию (подкласс – надкласс);
- определение слотов и описание допускаемых значений этих слотов;
- заполнение значений слотов экземпляров.

После этого мы можем создать базу знаний, определив отдельные экземпляры этих классов, введя в определенный слот значение и дополнительные ограничения для слота.

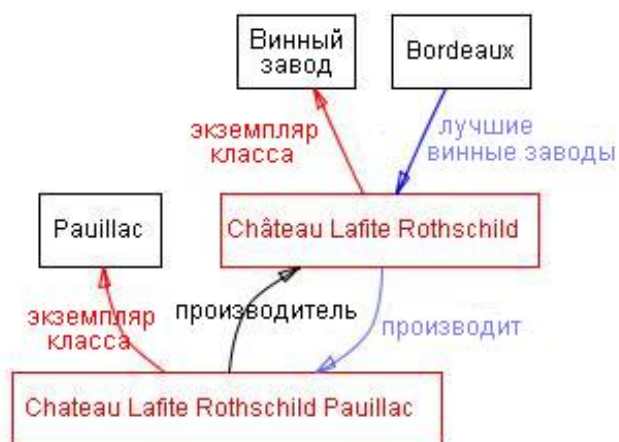


Рис. 1. Некоторые классы в области вин, экземпляры и отношения между ними. Черным мы обозначили классы, а красным – экземпляры. Прямые связи обозначают слоты и внутренние связи, такие как «экземпляр [класса]» и «подкласс [класса]».

3. Простая методология инженерии знаний

Как мы сказали выше, не существует единственного «правильного» способа или методологии разработки онтологий. Здесь мы обсуждаем общие моменты, которые нужно учитывать, и предлагаем один из возможных способов разработки онтологии. Мы описываем итеративный подход к разработке онтологии: мы начинаем с первого чернового просмотра онтологии. Затем мы проверяем и уточняем получаемую онтологию и добавляем детали. Попутно мы обсуждаем решения, касающиеся моделирования, которые должен принять разработчик, а также «за» и «против» и результаты принятия различных решений.

Во-первых, мы бы хотели выделить некоторые фундаментальные правила разработки онтологии, к которым мы будем неоднократно обращаться. Эти правила могут показаться довольно категоричными. Тем не менее, во многих случаях они могут помочь принять проектные решения.

- 1) *Не существует единственного правильного способа моделирования предметной области – всегда существуют жизнеспособные альтернативы. 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. Скорее всего, это существительные (объекты) или глаголы (отношения) в предложениях, которые описывают вашу предметную область.*

То есть, знание того, для чего вы собираетесь использовать онтологию и насколько детальной или общей она будет, повлияет на многие решения, касающиеся моделирования. Среди нескольких жизнеспособных альтернатив нам нужно определить, какая поможет лучше решить поставленную задачу и будет более наглядной, более расширяемой и более простой в обслуживании. Нам также нужно помнить, что онтология – это модель реального мира и понятия в онтологии должны отражать эту реальность. После того, как мы определим начальную версию онтологии, мы можем оценить и отладить ее, используя ее в приложениях или в методах решения задач и/или обсудив ее с экспертами предметной области. В результате почти наверняка нам нужно будет пересмотреть начальную онтологию. Этот процесс итеративного проектирования, вероятно, будет продолжаться в течение всего жизненного цикла онтологии.

Шаг 1. Определение области и масштаба онтологии

Мы предлагаем начать разработку онтологии с определения ее области и масштаба. То есть, ответим на несколько основных вопросов:

1. Какую область будет охватывать онтология?
2. Для чего мы собираемся использовать онтологию?
3. На какие типы вопросов должна давать ответы информация в онтологии?
4. 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.

Рассмотрим онтологию вина и еды, которую мы представили ранее. Область нашей онтологии – представление еды и вин. Мы собираемся использовать эту онтологию для приложений, которые будут предлагать хорошие сочетания вин и еды.

Конечно, в нашу онтологию будут включены понятия, описывающие различные типы вин, основные виды еды, понятие хорошего и плохого сочетания вина и еды. В то же время, маловероятно, что онтология будет включать понятия для управления инвентарем на винном заводе или служащими в ресторане, даже хотя эти понятия отчасти связаны с понятиями вина и еды.

Если онтология, которую мы проектируем, будет использоваться для помощи при обработке естественного языка статей в журналах о винах, то, возможно, понадобится включить в онтологию синонимов понятий и информации о частях речи. Если онтология будет использоваться для того, чтобы помочь посетителям ресторана решить, какое вино заказать, нам нужно будет включить информацию о розничных ценах. Если она будет использоваться для помощи покупателям вина в создании запасов в винном погребе, то могут понадобиться сведения об оптовых ценах и о наличии вин. Если люди, которые будут поддерживать онтологию, опишут предметную область языком, отличающимся от языка пользователей онтологии, то нам может потребоваться предоставить таблицу соответствий между языками.

Вопросы для проверки компетентности

Один из способов определить масштаб онтологии – это набросать список вопросов, на которые должна ответить база знаний, основанная на онтологии, т.е. **вопросы для проверки компетентности** (Gruninger and Fox 1995). Эти вопросы будут служить лакмусовой бумажкой: Содержит ли онтология достаточно информации для ответа на эти типы вопросов? Требуется ли для ответов особый уровень детализации или представление определенной области? Эти вопросы для проверки компетентности являются всего лишь формальными и не должны быть исчерпывающими.

В области вина и еды возможны следующие вопросы для проверки компетентности:

1. Какие характеристики вина мне следует учитывать при выборе вина?
2. Вино Bordeaux красное или белое?
3. Хорошо ли сочетается Cabernet Sauvignon с морскими продуктами?
4. Какое вино лучше всего подойдет к жареному мясу?
5. Какие характеристики вина влияют на его сочетаемость с блюдом?
6. Влияет ли с [год производства вина](#) на его букет или крепость?
7. Какие урожаи Napa Zinfandel были хорошими?

Судя по этому списку вопросов, онтология будет включать информацию о различных характеристиках вина и типах вин, годах производства вин (хороших и плохих), классификациях еды, которые нужно учесть при выборе подходящего вина, рекомендуемых сочетаниях вина и еды.

Шаг 2. Рассмотрение вариантов повторного использования существующих онтологий
Почти всегда стоит учесть, что сделал кто-то еще, и проверить, можем ли мы улучшить и расширить существующие источники для нашей конкретной предметной области и задачи. Повторное использование существующих онтологий может быть

необходимым, если нашей системе нужно взаимодействовать с другими приложениями, которые уже вошли в отдельные онтологии или контролируемые словари. Многие онтологии уже доступны в электронном виде и могут быть импортированы в используемую Вами среду проектирования онтологии. Формализм онтологии часто не имеет значения, т.к. многие системы представления знаний могут импортировать и экспортировать онтологии. Даже если система представления знаний не может работать напрямую с отдельным формализмом, задача перевода онтологии из одного формализма в другой обычно не является сложной.

В литературе и всемирной паутине существуют библиотеки повторно используемых онтологий. Например, мы можем использовать библиотеку онтологий Ontolingua (<http://www.ksl.stanford.edu/software/ontolingua/>) или библиотеку онтологий DAML (<http://www.daml.org/ontologies/>). Существует также ряд общедоступных коммерческих онтологий (например, UNSPSC (www.unspsc.org), RosettaNet (www.rosettanet.org), DMOZ (www.dmoz.org)).

К примеру, база знаний по французским винам уже может существовать. Если мы можем импортировать эту базу знаний и онтологию, на которой она основана, то у нас будет не только классификация французских вин, но и первый шаг к классификации характеристик вин, использующихся для разделения и описания вин. Списки свойств вина уже могут быть доступны на коммерческих веб-сайтах, таких как <http://www.wines.com/>, которые клиенты используют при покупке вин.

Тем не менее, в этом руководстве мы будем считать, что соответствующих онтологий еще не существует, и начнем разрабатывать онтологию с нуля.

Шаг 3. Перечисление важных терминов в онтологии

Полезно составить список всех терминов, о которых мы хотели бы сказать что-либо или которые хотели бы объяснить пользователю. Какие термины мы бы хотели рассмотреть? Какие свойства имеют эти термины? Что бы мы хотели сказать об этих терминах? Например, в число важных терминов, связанных с винами, входят вино, виноград, винный завод, местоположение, цвет вина, его крепость, вкус и содержание сахара; различные виды еды, такие как рыба и черное мясо; типы вина, такие как белое вино и т.д. В начале важно получить полный список терминов, не беспокоясь о пересечении понятий, которые они представляют, об отношениях между терминами, о возможных свойствах понятий или о том, чем являются понятия – классами или слотами.

Следующие два шага – разработка иерархии классов и определение свойств понятий (слотов) – тесно переплетены. Сложно выполнить сначала один из них, а потом – другой. Обычно в иерархии мы даем несколько формулировок понятий и затем описываем свойства этих понятий и т.д. Также эти два шага – самые важные шаги в процессе проектирования онтологии. Здесь мы опишем их вкратце, а затем в следующих двух главах рассмотрим более сложные проблемы, которые необходимо принять во внимание, часто встречающиеся трудности, решения, которые нужно принять, и т.д.

Шаг 4. Определение классов и иерархии классов

Существует несколько возможных подходов для разработки иерархии классов (Uschold and Gruninger 1996):

- Процесс **нисходящей** разработки начинается с определения самых общих понятий предметной области с последующей конкретизацией понятий. Например, мы можем начать с создания классов для общих понятий Вино и Еда. Затем мы конкретизируем класс Вино, создавая его подклассы: Белое вино, Красное вино, Розовое вино. Мы можем еще дальше категоризировать класс Красное Вино, например, в Syrah, Red Burgundy, Cabernet Sauvignon и т.д.

- Процесс **восходящей** разработки начинается с определения самых конкретных классов, листьев иерархии, с последующей группировкой этих классов в более общие понятия. Например, сначала мы определяем классы для вин Pauillac и Margaux. Затем мы создаем общий надкласс для двух этих классов – Medoc, который, в свою очередь является подклассом Bordeaux.

- Процесс **комбинированной** разработки – это сочетание нисходящего и восходящего подходов: Сначала мы определяем более заметные понятия, а затем соответствующим образом обобщаем и ограничиваем их. Мы могли бы начать с нескольких понятий высшего уровня, таких как Вино, и нескольких конкретных понятий, таких как Margaux. Затем мы можем соотнести их с понятием среднего уровня, таким как Medoc. После этого нам может понадобиться сформировать все классы вин из области Франции, формируя таким образом ряд понятий среднего уровня.

На рис. 2 показано возможное деление на различные уровни обобщения.

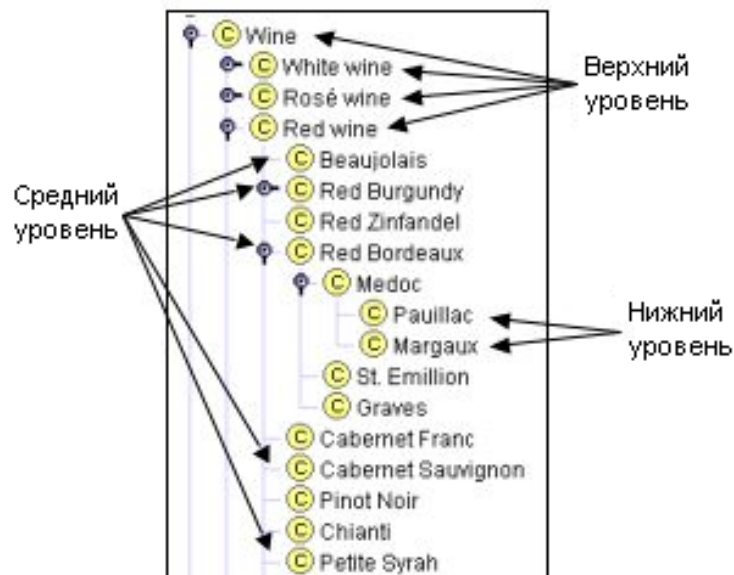


Рис. 2. Различные уровни таксономии Вино: Вино, Красное вино, Белое вино, Розовое вино – более общие понятия, верхний уровень. Pauillac и Margaux – самые конкретные классы в иерархии, нижний уровень.

Ни один из этих трех методов не лучше других по своей сути. Выбор подхода в большой степени зависит от личного взгляда на предметную область. Если разработчик склонен к рассмотрению предметной области сверху вниз, то ему,

возможно, больше подойдет нисходящий метод. Часто для многих разработчиков онтологий самым простым является комбинированный метод, т.к. понятия, находящиеся «посередине», имеют тенденцию быть самыми наглядными понятиями в предметной области (Rosch 1978).

Если вы склонны делать сначала самую общую классификацию вин, то вам больше подойдет нисходящий метод. Если вы бы начали приводить конкретные примеры, то более подходящим является восходящий метод.

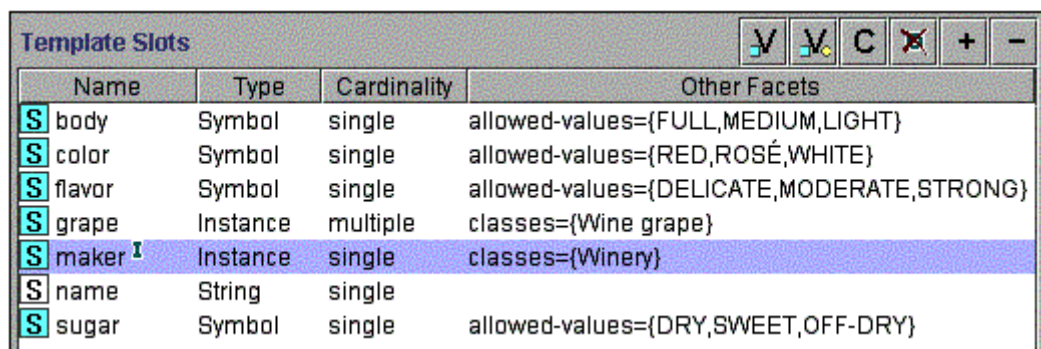
Какой метод мы бы ни избрали, обычно мы начинаем с определения классов. Из списка, составленного в Шаге 3, мы выбираем термины, которые описывают объекты, существующие независимо, а не термины, которые описывают эти объекты. В онтологии эти термины будут классами и станут точками привязки в иерархии классов^[2]. Мы организуем классы в иерархическую таксономию, задавая вопрос: если объект является экземпляром одного класса, будет ли он обязательно (т.е. по определению) экземпляром некоторого другого класса?

Если класс А – надкласс класса В, то каждый экземпляр В также является экземпляром А.

Другими словами, класс В представляет собой понятие, которое является «разновидностью» А.

Например, каждое вино Pinot Noir – обязательно красное вино. Поэтому класс Pinot Noir – подкласс класса Красное вино.

На рис. 2 показана часть иерархии классов онтологии по винам. В 4-й главе детально рассмотрено, что нужно искать при определении иерархии классов.



Name	Type	Cardinality	Other Facets
body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
grape	Instance	multiple	classes={Wine grape}
maker ^I	Instance	single	classes={Winery}
name	String	single	
sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

Рис. 3. Слоты класса Вино и facets этих слотов. Значок “I” рядом со слотом производитель указывает, что у слота есть обратный слот (Глава 5.1.).

Шаг 5. Определение свойств классов – слотов

Классы сами по себе не предоставляют достаточно информации для ответа на вопросы проверки компетентности из Шага 1. После определения некоторого количества классов мы должны описать внутреннюю структуру понятий.

Мы уже выбрали классы из списка терминов, который мы создали на Шаге 3.

Большинство оставшихся терминов, вероятно, будут свойствами этих классов. Эти термины включают, к примеру, цвет вина, его крепость, вкус и содержание сахара, а также местоположение винного завода.

Для каждого свойства из списка мы должны определить, какой класс оно описывает. Эти свойства станут слотами, привязанными к классам. Таким образом, у класса Вино будут следующие слоты: цвет, крепость, вкус и сахар. А у класса Винный завод будет слот местоположение.

Вообще, в онтологии слотами могут стать несколько типов свойств объектов:

- «внутренние» свойства, такие как вкус вина;
- «внешние» свойства, такие как название вина и область, в которой оно было произведено;
- части, если объект имеет структуру; они могут быть как физическими, так и абстрактными «частями» (например, блюда, входящие в обед);
- отношения с другими индивидуальными концептами; это отношения между отдельными членами класса и другими элементами (например, производитель вина, представляющий отношение между вином и винным заводом, и виноград, из которого произведено вино).

Таким образом, в дополнение к ранее определенным свойствам, к классу Вино нам нужно добавить следующие слоты: название, область, производитель, виноград. На рис. 3 показаны слоты класса Вино.

Все подклассы класса **наследуют** слот этого класса. Например, все слоты класса Вино будут унаследованы всеми подклассами этого класса, включая Красное Вино и Белое Вино. К классу Красное Вино мы добавим дополнительный слот уровень танина (низкий, средний или высокий). Слот уровень танина будет унаследован всеми классами, представляющими красные вина (такие как Bordeaux и Beaujolais).

Слот должен быть привязан к самому общему классу, у которого может быть данное свойство. Например, крепость и цвет вина нужно будет привязать к классу Вино, т.к. это самый общий класс, чьи экземпляры будут иметь крепость и цвет.

Шаг 6. 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. Например, значение слота название (как в «название вина») – одна строка. То есть, название – это слот с типом значения Строка. Слот производит (как в выражении «винный завод производит эти вина») может иметь множественные значения, которые являются экземплярами класса Вино. То есть, производит – это слот с типом значения Экземпляр, и разрешенным классом является Вино.

Сейчас мы опишем несколько общих фацетов.

Мощность слота

Мощность слота определяет, сколько значений может иметь слот. В некоторых системах различаются только единичная мощность (возможно только одно значение) и множественная мощность (возможно любое число значений). Крепость вина будет слотом единичной мощности (вино может иметь только одну крепость). Вина, производимые на конкретном заводе, заполняют слот множественной мощности производит класса Винный завод.

Некоторые системы позволяют определить минимальную и максимальную мощность для того, чтобы более точно описать количество значений слота. Минимальная мощность N означает, что слот должен иметь не менее N значений. Например, слот виноград класса Вино имеет минимальную мощность 1: каждое вино делается, как минимум, из одного сорта винограда. Максимальная мощность M означает, что слот может иметь максимум M значений. Максимальная мощность слота виноград для вин из одного сорта винограда равняется 1. Иногда полезно установить максимальную мощность в 0. Эта установка будет означать, что для определенного подкласса слот не может иметь значений.

Тип значения слота

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.
- **Число** (иногда используются более конкретные типы значений: Float (Число с плавающей запятой) и Integer (Целое число)) описывает слоты числовыми значениями. Например, стоимость вина может иметь тип Float.
- **Булевы** слоты – это простые флаги «да - нет». Например, если мы не будем представлять игристые вина как отдельный класс, то принадлежность к игристым винам может быть показана значением булевого слота: если значение «истина» («да»), то вино игристое, а если значение «ложь» («нет»), то вино не игристое.
- **Нумерованные** слоты определяют список конкретных разрешенных значений слота. Например, мы можем установить, что слот вкус может принять одно из трех возможных значений: сильный, умеренный и мягкий. В Protege-нумерованные слоты имеют тип Символ.
- **Слоты-экземпляры** позволяют определить отношения между индивидуальными концептами. Слоты с типом значения Экземпляр также должны определять список разрешенных классов, экземпляры которых можно использовать. Например, слот производит класса Винный завод в качестве значений может иметь экземпляры класса Вино^[3].

На рис. 4 показано определение слота производит класса Винный завод.

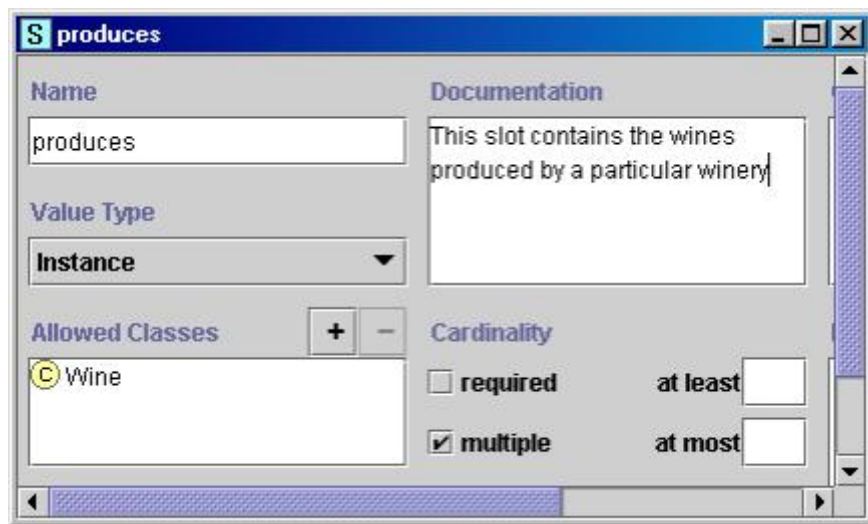


Рис. 4. Определение слота производит, который описывает вина, производимые на винном заводе. Слот имеет множественную мощность и значение типа Экземпляр. Разрешенным классом для значений этого слота является класс Вино.

Домен слота и диапазон значений слота

Разрешенные классы для слотов типа Экземпляр часто называют **диапазоном значений** слота. В примере на рис. 4 класс Вино является диапазоном значений слота производит. Некоторые системы позволяют ограничить диапазон значений слота, если слот привязан к определенному классу.

Классы, к которым слот привязан, или классы, свойство которых слот описывает, называются **доменом** слота. Класс Винный завод – домен слота производит. В системах, где мы *привязываем* слоты к классам, домен слота обычно составляют классы, к которым привязан слот. Нет необходимости отдельно определять домен. Основные правила определения домена слота и диапазона значений слота схожи друг с другом:

При определении домена или диапазона значений слота найдите наиболее общие классы или класс, которые могут быть соответственно доменом или диапазоном значений слотов.

*С другой стороны, не определяйте 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. Не выбирайте слишком общий класс для диапазона значений (то есть, вы не захотите делать **THING** диапазоном значений, а захотите выбрать класс, который охватит все заполнители.)*

Вместо того чтобы перечислить все возможные подклассы класса Вино для диапазона значений слота производит, просто внесите в список класс Вино. В то же время, нам не нужно определять диапазон значений слота как **THING** (самый общий класс в онтологии).

Конкретнее:

Если список классов, определяющих диапазон значений слота или домен слота, включает класс и его подкласс, удалите подкласс.

Если диапазон значений слота содержит и класс Вино, и класс Красное Вино, мы можем удалить Красное Вино из диапазона значений, т.к. он не добавляет новую

информацию: Красное Вино – это подкласс класса Вино, и поэтому диапазон значений слота уже неявно включает его, также как и все другие подклассы класса Вино.

Если список классов, определяющих диапазон значений слота или домен слота, включает все подклассы класса А, но не включает сам класс А, то в диапазон значений должен входить только класс А, а не его подклассы.

Вместо указания того, что диапазон значений слота включает Красное Вино, Белое Вино и Розовое Вино (перечисление всех прямых подклассов класса Вино), мы можем ограничить диапазон значений самим классом Вино.

Если список классов, определяющих диапазон значений слота или домен слота, включает почти все подклассы класса А, подумайте, может, для определения диапазона значений лучше подойдет класс А.

В системах, где привязка слота к классу равнозначна добавлению класса к домену слота, к привязке слота применяются те же правила: С одной стороны, нам нужно постараться сделать его как можно более общим. С другой стороны, мы должны гарантировать, что каждый класс, к которому мы привязываем слот, на самом деле имеет свойство, которое представляет слот. Мы можем привязать слот уровень танина к каждому классу, представляющему красные вина (например, Bordeaux, Merlot, Beaujolais и т.д.). Однако, т.к. все красные вина имеют свойство «уровень танина», то вместо этого нам нужно прикрепить этот слот к более общему классу Красные Вина. Будет неправильно дальше обобщать домен слота уровень танина (привязка его к классу Вино), т.к. мы не используем уровень танина для описания, к примеру, белых вин.

Шаг 7. Создание экземпляров

Последний шаг – это создание отдельных экземпляров классов в иерархии. Для определения отдельного экземпляра класса требуется (1) выбрать класс, (2) создать отдельный экземпляр этого класса и (3) ввести значения слотов. Например, мы можем создать отдельный экземпляр Chateau-Morgon-Beaujolais для представления определенного типа вина Beaujolais. Chateau-Morgon-Beaujolais – это экземпляр класса Beaujolais, представляющего все вина Beaujolais. У этого экземпляра определены следующие значения слотов (рис. 5):

- Крепость: Легкое
- Цвет: Красный
- Вкус: Мягкий
- Уровень танина: Низкий
- Виноград: Gamay (экземпляр класса Виноград для изготовления вин)
- Производитель: Chateau-Morgon (экземпляр класса Винный завод)
- Область: Beaujolais (экземпляр класса Винная область)
- Сахар: Сухое

Chateau Morgon Beaujolais (Beaujolais)	
Name	Chateau Morgon Beaujolais
Area	Beaujolais region
Body	LIGHT
Color	RED
Maker	Chateau Morgon
Flavor	DELICATE
Sugar	DRY
Grape	Gamay grape
Tannin Level	LOW

Рис. 5. Определение экземпляра класса Beaujolais. Экземпляром является вино Chateau Morgon Beaujolais из области Beaujolais, произведенное из винограда Gamay на заводе Chateau Morgon. Оно легкое, с мягким вкусом, красное, с низким уровнем танина. Это сухое вино.

4. Определение классов и иерархии классов

В этой главе говорится о том, за чем нужно следить, и об ошибках, которые легко сделать при определении классов и иерархии классов (Шаг 4 из Главы 3). Как мы уже говорили ранее, для любой предметной области не существует единственной правильной иерархии классов. Иерархия зависит от возможных способов применения онтологии, уровня детализации, необходимого для приложения, личных предпочтений и иногда от требований по совместимости с другими моделями. Тем не менее, мы рассматриваем несколько руководящих принципов, которые нужно учитывать при разработке иерархии классов. После определения значительного количества новых классов полезно остановиться и проверить, соответствует ли возникающая иерархия этим руководящим принципам.

4.1. Обеспечение правильности иерархии классов

Отношение “is-a”¹

Иерархия классов представляет отношение “is-a”: класс А – это подкласс В, если каждый экземпляр В также является экземпляром А. Например, Chardonnay – подкласс класса Белое Вино. Другой способ подхода к таксономическому отношению – это отношение “kind-of”²: Chardonnay – вид Белого вина. Реактивный лайнер – вид самолета. Мясо – вид еды.

Подкласс класса представляет понятие, которое является «разновидностью» понятия, представляемого надклассом.

Отдельно взятое вино не является подклассом всех вин

¹ Буквально «является».

² Буквально «вид, разновидность [чего-то]».

Распространенная ошибка при моделировании – это включение в иерархию варианта одного и того же понятия как в единственном, так и во множественном числе, сделав первое подклассом второго. Например, будет неправильно определить класс Вина и класс Вино как подкласс класса Вина. Как только вы начинаете считать, что иерархия представляет собой отношение “kind-of”, то ошибка при моделировании становится очевидной: отдельное Вино не является **видом** Вин. Лучший способ избежать таких ошибок – всегда использовать имена классов или в единственном, или во множественном числе (присваивание имен понятиям подробно рассмотрено в Главе 6).

Транзитивность иерархических отношений

Отношение подкласса транзитивно:

Если В – это подкласс А, а С – подкласс В, то С – подкласс А.

Например, мы можем определить класс Вино, а потом определить класс Белое вино как подкласс класса Вино. Затем мы определяем класс Chardonnay как подкласс класса Белое Вино. Транзитивность отношения подкласса означает, что класс Chardonnay также является подклассом класса Вино. Иногда мы различаем прямые и косвенные подклассы. **Прямой подкласс** – самый близкий подкласс класса: в иерархии между классом и его прямым подклассом нет других классов. То есть, между классом и его прямым надклассом в иерархии нет других классов. В нашем примере Chardonnay – это прямой подкласс класса Белое вино и не прямой подкласс класса Вино.

Развитие иерархии классов

Поддержание последовательной иерархии классов может вызывать сложности по мере того, как развиваются предметные области. Например, много лет все вина Zinfandel были красными. Поэтому мы определяем класс вин Zinfandel как подкласс класса Красное вино. Тем не менее, производители вин иногда начали выжимать виноград и сразу удалять цветообразующие вещества из винограда, изменяя таким образом цвет получаемого вина. Так мы получаем «белое Zinfandel» розового цвета. Теперь нам нужно разбить класс Zinfandel на 2 класса zinfandel – Белое zinfandel и Красное zinfandel – и классифицировать их как подклассы классов Розовое вино и Красное вино соответственно.

Классы и их имена

Важно различать класс и его имя:

Классы представляют понятия предметной области, а не слова, которые обозначают эти понятия.

Имя класса может измениться, если мы выберем другую терминологию, но сам термин представляет объективную реальность мира. Например, мы можем создать класс Shrimps, а потом переименовать его в Prawns – класс представляет все то же понятие. Вина, которые подходили к блюдам из shrimp, должны подходить к блюдам из prawn³.

³ «Shrimp» и «prawn» - это синонимы, которые переводятся одинаково – «креветка».

В действительности нужно все время соблюдать правило:

Синонимы одного и того же понятия не представляют различные классы.

Синонимы – всего лишь разные имена понятия или термина. Следовательно, у нас не должно быть класса с именем Shrimp и одновременно с этим класса с именем Prawn, а также класса с именем Crevette⁴. Предпочтительнее будет иметь один класс с именем Shrimp или Prawn. Многие системы позволяют ассоциировать с классом список синонимов, переводов или имен представления. Если система не позволяет осуществлять такие ассоциации, то синонимы всегда можно перечислить в документации к классу.

Избежание циклов классов

Нам следует избегать **циклов** в иерархии классов. Мы говорим, что в иерархии есть цикл, когда у некоторого класса А есть подкласс В и в то же время В – это надкласс А. Создание такого цикла в иерархии равнозначен объявлению того, что классы А и В эквивалентны: все экземпляры А – это экземпляры В, а все экземпляры В также являются экземплярами А. Действительно, поскольку В – подкласс А, то все экземпляры В должны быть экземплярами класса А. Поскольку А – подкласс В, то все экземпляры А также должны быть экземплярами класса В.

4.2. Анализ узлов-братьев в иерархии классов

Узлы-братья в иерархии классов

Узлы-братья в иерархии – это классы, которые являются прямыми подклассами одного и того же класса (см. Раздел 4.1).

Все узлы-братья в иерархии (кроме тех, что находятся в корне) должны располагаться на одном уровне обобщения.

Например, Белое вино и Chardonnay не должны быть подклассами одного и того же класса (скажем, класса Вино). Белое вино – более общее понятие, чем Chardonnay. Узлы-братья должны представлять понятия, которые находятся «на одной линии», так же как разделы одного уровня в книге должны находиться на одном уровне обобщения. В этом смысле требования к иерархии классов похожи на требования к структуре книги.

Однако понятия, которые находятся в корне иерархии (и которые всегда представлены как прямые подклассы некоторого самого общего класса, такого как Thing), представляют основные деления в предметной области и не должны быть схожими понятиями.

Что называть «слишком много», а что – «слишком мало»?

Не существует жестких правил относительно того, сколько прямых подклассов должен иметь класс. Тем не менее, во многих онтологиях с четкой структурой имеется от двух до дюжины прямых подклассов. Отсюда два руководящих принципа:

Если класс имеет только один прямой подкласс, то, возможно, при моделировании допущена ошибка или онтология неполная.

⁴ Также переводится как «креветка».

Если у данного класса есть более дюжины подклассов, то, возможно, необходимы дополнительные промежуточные категории.

Первое правило похоже на правило набора текста, которое гласит, что у маркированного текста никогда не должна быть лишь одна маркированная точка. Например, большинство красных бургундских вин – это вина Cotes d'Or. Предположим, что мы хотели представить только этот основной тип бургундских вин. Мы могли создать класс Red Burgundy и затем единственный подкласс Cotes d'Or (рис. 6а). Тем не менее, если в нашем представлении красные бургундские вина и вина Cotes d'Or, по существу, эквивалентны (все красные бургундские вина являются винами Cotes d'Or и все вина Cotes d'Or – это красные бургундские вина), то нет необходимости создавать класс Cotes d'Or, и он не добавит в представление новую информацию. Если бы нам нужно было включить вина Cotes Chalonnaise (более дешевые бургундские вина из области к югу от Cotes d'Or), то мы бы создали два подкласса класса Burgundy: Cotes d'Or и Cotes Chalonnaise (рис. 6б).

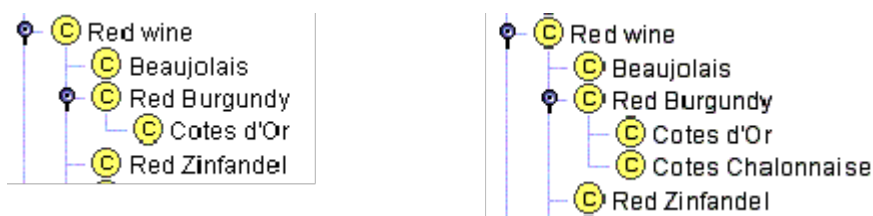


Рис. 6. Подклассы класса Red Burgundy.

Наличие у класса только одного подкласса указывает на ошибку при моделировании.

Теперь предположим, что мы перечислили все вина как прямые подклассы класса Вино. Тогда этот список будет включать такие более общие сорта, как Beaujolais и Bordeaux, так же как и более конкретные вина, как Paulliac и Margaux (рис. 7а). У класса Вино слишком много прямых подклассов и, действительно, для того чтобы онтология отражала различные сорта вин более четко, Medoc должно быть подклассом Bordeaux, а Cotes d'Or должно быть подклассом Burgundy. Кроме того, наличие таких промежуточных категорий как Красное вино и Белое Вино также будет отражать ту понятийную модель предметной области вин, которая есть у многих людей (рис. 7б).

Тем не менее, если не существует естественных классов для группировки понятий в длинный список узлов-братьев, то не нужно создавать искусственные классы – оставьте все, как есть. В конце концов, онтология – это отражение реального мира, и если в действительности категоризации нет, то онтология должна это отражать.

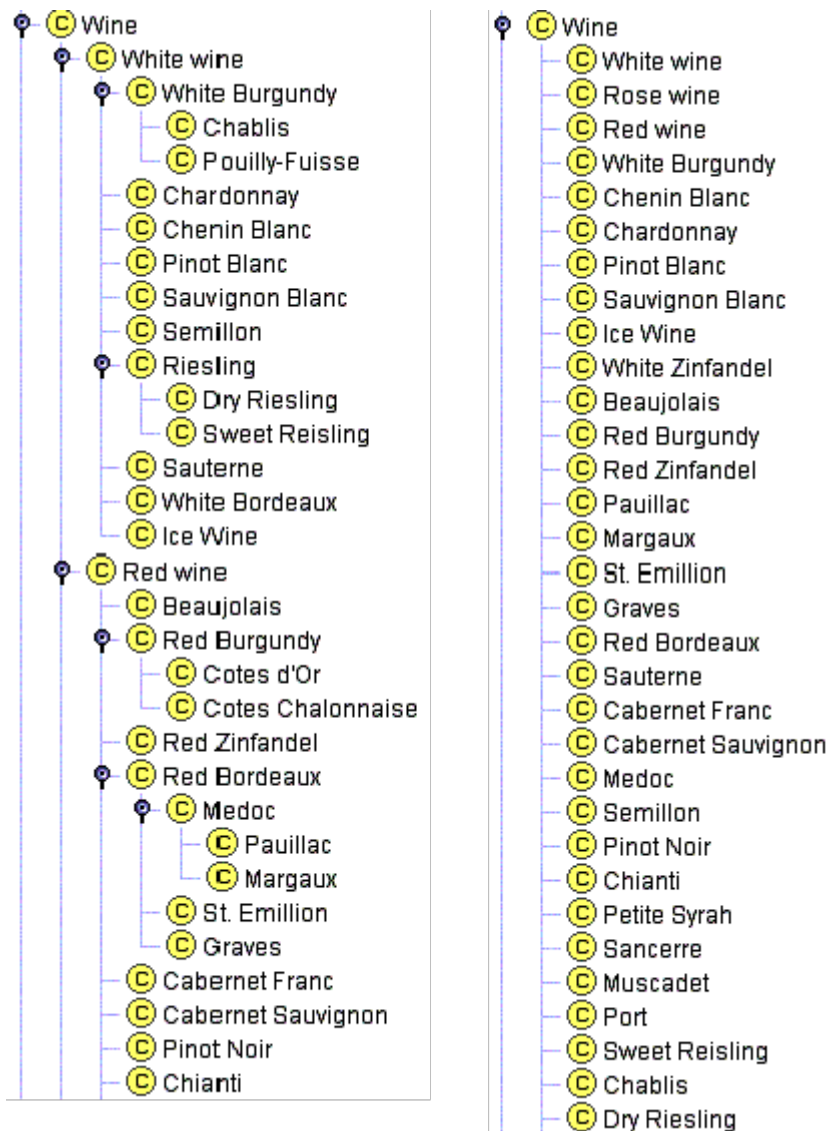


Рис. 7. Категоризация вин. Простое перечисление всех вин против нескольких уровней категоризации.

4.3. Множественное наследование.

Большинство систем представления знаний позволяют осуществлять **множественное наследование** в иерархии классов: класс может быть подклассом нескольких классов. Предположим, что мы хотим создать отдельный класс десертных вин - класс Десертное вино. Вино Port является и красным, и десертным вином ^[4]. Следовательно, мы определяем, что у класса Port есть 2 надкласса: Красное вино и Десертное вино. Все экземпляры класса Port будут экземплярами как класса Красное вино, так и класса Десертное вино. Класс Port унаследует слоты и фацеты от обоих родителей. Таким образом, он унаследует значение СЛАДКОЕ из слота Сахар класса Десертное вино, а также слот уровень танина и значение слота цвета класса Красное вино.

4.4. Когда вводить (или не вводить) новый класс

Одно из самых сложных решений, которое нужно принять во время моделирования, - это определить, когда ввести новый класс или когда

сформулировать различие с помощью разных значений свойств. Сложно ориентироваться как в иерархии с очень большой степенью вложенности и множеством посторонних классов, так и в очень плоской иерархии, где очень мало классов, но в их слотах закодировано слишком много информации. Найти подходящий баланс нелегко.

Существует несколько практических способов определения того, когда в иерархию следует ввести новые классы:

Обычно подклассы класса (1) имеют дополнительные свойства, которых нет у надкласса, или (2) ограничения, отличные от тех, которые есть у надкласса, или (3) состоят в других отношениях, нежели надклассы.

У красных вин разные уровни танина, тогда как это свойство не используется для описания вин в общем. Слот сахар класса Десертное вино имеет значение СЛАДКОЕ, тогда как для надкласса класса Десертное вино это не так. Вина Pinot Noir могут хорошо сочетаться с морскими продуктами, тогда как другие красные вина – нет. Другими словами, мы обычно вводим в иерархию новый класс только тогда, когда мы можем сказать про этот класс что-то такое, чего мы не можем сказать о надклассе.

На практике к каждому подклассу нужно добавить новые слоты или определить у него новые значения слотов и переопределить некоторые фацеты наследованных слотов.

Однако иногда может быть полезно создать новые классы, даже если они не вводят никаких новых свойств.

Классы в иерархиях терминов не обязаны новые свойства.

Например, некоторые онтологии включают большие иерархии ссылок обычных терминов, используемых в предметной области. К примеру, онтология, которая лежит в основе электронной системы записи медицинской информации, может включать классификацию различных болезней. Эта классификация может быть как раз такой - иерархией терминов, без свойств (или с одним и тем же набором свойств). В этом случае по-прежнему полезно организовать термины в иерархию, а не в линейный список, потому что она (1) облегчит изучение и навигацию и (2) позволит врачу легко выбрать подходящий уровень общности термина.

Другая причина введения новых классов без новых свойств – это моделирование понятий, среди которых эксперты в предметной области обычно проводят разграничение, даже несмотря на то, что мы могли принять решение не моделировать само разграничение. Так как мы используем онтологии для содействия коммуникации между экспертами в предметной области, а также между экспертами и системами, основанными на знаниях, то в онтологии мы бы хотели отразить точку зрения эксперта на предметную область.

В конце концов, нам не следует создавать подклассы класса для каждого дополнительного ограничения. К примеру, мы ввели классы Красное вино, Белое вино и Розовое вино, так как это разделение является естественным в мире вин. Мы не ввели классы для изысканного вина, для вин с умеренным вкусом и т.д. Когда мы определяем иерархию классов, наша цель – добиться равновесия между созданием

классов, которые полезны для организации классов, и созданием слишком большого числа классов.

4.5. Новый класс или значение свойства?

При моделировании предметной области нам часто нужно решать, моделировать ли определенное разграничение (такое как белое, красное или розовое вино) как значение свойства или как набор классов, что опять же зависит от масштаба предметной области и от решаемой задачи.

Создать ли нам класс Белое вино или просто создать класс Вино и ввести различные значения слота цвет? Ответ всегда зависит от того масштаба, который мы определили для онтологии. Насколько важно понятие Белое вино в нашей предметной области? Если важность вин в предметной области незначительна и то, белое это вино или нет, не особо влияет на его отношения с другими объектами, то нам не нужно вводить отдельный класс белых вин. Для модели предметной области, используемой на предприятии, где производятся винные этикетки, правила для этикеток вин различных цветов одни и те же и разделение не столь важно. И наоборот, для представления вина, еды и их подходящих сочетаний, красное вино сильно отличается от белого: оно сочетается с другой едой, у него другие свойства и т.д. Также, цвет вина важен для базы знаний по винам, которую мы можем использовать для определения последовательности дегустации вин. Поэтому мы создаем отдельный класс Белое вино.

Если понятия с разными значениями слота становятся ограничениями для различных слотов в других классах, то для разделения нам следует создать новый класс. В противном случае разделение представляется в значении слота.

Подобным образом, в нашей онтологии вин есть такие классы как Красное Merlot и Белое Merlot, а не единый класс для всех вин Merlot: красные Merlot и белые Merlot – это действительно разные вина (сделанные из одного винограда), и если мы разрабатываем подробную онтологию, то это разделение представляет важность. *Если в предметной области разграничение представляет важность и объекты с другими значениями разграничения мы считаем другими типами объектов, то для осуществления разграничения нам нужно создать новый класс.*

При принятии решения о том, вводить новый класс или нет, также может быть полезно рассмотреть потенциальные отдельные экземпляры класса.

Класс, к которому принадлежит отдельный экземпляр, не должен часто меняться.

Обычно, когда для определения различий между классами мы используем внешние, а не внутренние свойства понятий, экземпляры этих классов должны часто будут перемещаться из одного класса в другой. Например, Охлажденное вино не должно являться классом в онтологии, описывающей бутылки вин в ресторане. Свойство охлажденное должно быть просто атрибутом вина в бутылке, так как экземпляр класса Охлажденное вино может легко перестать быть экземпляром этого класса, а затем снова стать его экземпляром.

Обычно числа, цвета, местоположения являются значениями слотов и не приводят к созданию новых классов. Тем не менее, вино – особое исключение, так как цвет вина имеет первостепенную важность для его описания.

В качестве другого примера рассмотрим онтологию человеческой анатомии. Когда мы представляем ребра, создаем ли мы класс для «1-го левого ребра», потом класс для «2-го левого ребра» и т.д.? Или у нас есть класс Ребро со слотами для последовательности и стороны (левое - правое)?^[5] Если информация о каждом из ребер, которые мы представляем в онтологии, существенно отличается, то нам, действительно, нужно создать класс для каждого ребра. То есть, если мы хотим подробно представить информацию о смежности и положении (которые различны для всех ребер), а также особые функции, которые выполняет каждое ребро, и какие органы оно защищает, то нам нужно создать классы. Если мы моделируем анатомию на чуть более низком уровне обобщения и для наших потенциальных приложений все ребра очень похожи (мы говорим всего лишь о том, какое ребро сломано, судя по рентгеновскому снимку, безотносительно к другим частям тела), то нам потребуется упростить иерархию и оставить только класс Ребро с двумя слотами: сторона и порядковый номер.

4. 6. Экземпляр или класс?

Определение того, чем является определенное понятие - классом в онтологии или отдельным экземпляром - зависит от потенциальных приложений онтологии. Определение того, где заканчиваются классы и начинаются отдельные экземпляры, начинается с определения нужной глубины детализации в представлении. Глубина детализации, в свою очередь, определяется потенциальным приложением онтологии. Другими словами, какие самые конкретные элементы будут представлены в базе знаний? Если возвратиться к вопросам для проверки компетентности, которые мы определили в Шаге 1 Главы 3, самые конкретные понятия, которые будут ответами на эти вопросы, лучше всего подойдут на роль индивидуальных концептов в базе знаний. *Отдельные экземпляры - самые конкретные понятия, представленные в базе знаний.*

Например, если мы собираемся говорить только о подборе сочетаний вина и еды, то нас не будут интересовать конкретные материальные бутылки вина. Поэтому такие термины как Sterling Vineyards Merlot, вероятно, будут самыми конкретными используемыми нами терминами. Следовательно, Sterling Vineyards Merlot будет экземпляром в базе знаний.

С другой стороны, если бы мы хотели поддерживать в ресторане ассортимент вин в дополнение к базе знаний хороших сочетаний «вино-еда», то отдельными экземплярами в нашей базе знаний могли бы стать отдельные бутылки каждого вина.

Аналогично, если бы мы хотели записать различные качества для каждого конкретного урожая Sterling Vineyards Merlot, то экземпляром в базе знаний стал бы конкретный урожай вина, а Sterling Vineyards Merlot стал бы классом, содержащим экземпляры всех его урожаев.

Другое правило может «переместить» некоторые отдельные экземпляры в разряд классов:

Если понятия формируют естественную иерархию, то нам нужно представить их, как классы.

Рассмотрим винные области. В начале мы можем определить основные винные регионы, такие как Франция, США, Германия и т.д. как классы, а конкретные винные области внутри этих регионов как экземпляры. Например, область Bourgogne – это экземпляр класса регион Франция. Однако мы бы также хотели отметить, что область Cotes d'Or – это область Bourgogne. Поэтому область Bourgogne должна быть классом (чтобы иметь подклассы или экземпляры). Однако представление области Bourgogne как класса, а области Cotes d'Or как экземпляра области Bourgogne кажется произвольным: очень сложно четко разграничить, какие области являются классами, а какие – экземплярами. Поэтому мы определяем все винные области как классы. Protege- дает возможность пользователю определить некоторые классы как Абстрактные, показывая, что у класса не может быть прямых экземпляров. В нашем случае, все классы областей являются абстрактными (рис. 8).

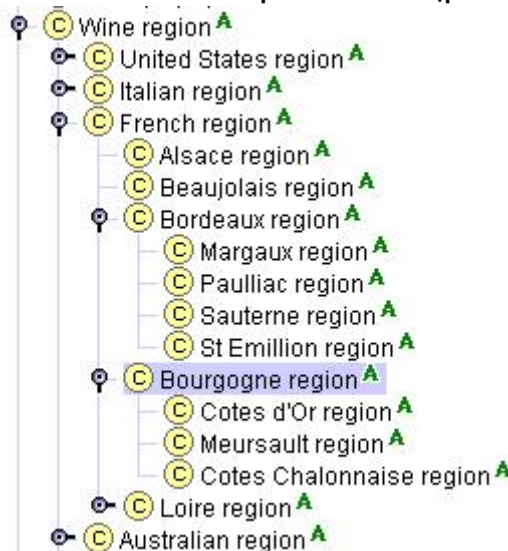


Рис. 8. Иерархия винных областей. Иконка «А» рядом с именами классов указывает на то, что это абстрактные классы и у них не может быть прямых экземпляров.

Так же иерархия классов была бы неправильной, если бы мы пропустили в имени класса слово «область». Мы не можем сказать, что класс Alsace – это подкласс класса Франция: Alsace – это не разновидность Франции. Тем не менее, область Alsace – разновидность региона Франция.

В виде иерархии можно представить только классы – в системах представления знаний нет категории «подэкземпляр». Поэтому, если среди терминов существует естественная иерархия, такая как в иерархиях терминов в Разделе 4.2, то нам нужно охарактеризовать эти термины как классы, даже если они могут и не иметь своих экземпляров.

4.7. Ограничение масштаба

В качестве последнего замечания по составлению иерархии классов, следующий набор правил всегда поможет при ответе на вопрос, закончено ли

определение онтологии:

Онтология не должна содержать всю возможную информацию о предметной области: вам не нужно конкретизировать или обобщать больше, чем вам нужно для вашего приложения (не более 1 дополнительного уровня в каждую сторону).

Для нашего примера с вином и едой нам не требуется знать, из какой бумаги делаются этикетки, или как готовятся блюда из креветок.

Также онтология не должна содержать все возможные свойства классов и различия между классами в иерархии.

В нашу онтологию мы, естественно, не включаем все свойства, которые могли бы иметь вино или еда. В нашей онтологии мы представили самые значимые свойства классов элементов. Даже несмотря на то, что в книгах по винам можно узнать о размере виноградин, мы не включили эту информацию. Так же, в нашу систему мы не включили все возможные отношения между всеми терминами. К примеру, мы не включаем в онтологию такие отношения, как любимое вино или любимая еда, просто для того, чтобы сделать более полное представление обо всех взаимосвязях между определенными нами терминами более полным.

Последние правила также применимы для установления отношений между понятиями, которые мы уже включили в онтологию. Рассмотрим онтологию, описывающую биологические эксперименты. Эта онтология, вероятно, будет содержать понятие Биологические организмы. Она также будет содержать понятие Экспериментатор (включающее имя, место работы и т.д.), который проводит эксперимент. Верно то, что экспериментатор как человек также является биологическим организмом. Тем не менее, мы, наверное, не должны отражать эту особенность в онтологии: в целях этого представления экспериментатор не является биологическим организмом, и мы, наверное, никогда не будем проводить эксперименты на самих экспериментаторах. Если бы в онтологии мы делали представление всего того, что мы можем сказать о классах, то Экспериментатор бы стал подклассом класса Биологический организм. Однако нет необходимости включать это знание для возможных приложений. Фактически, включение этой дополнительной классификации существующих классов действительно мешает: теперь у экземпляра Экспериментатора будут слоты для веса, возраста, вида и других данных, которые относятся к биологическому организму, но совершенно неуместны в контексте описания эксперимента. Тем не менее, для пользователей, которые будут иметь дело с этой онтологией и которые могут не знать о задуманном нами приложении, нам нужно отразить это проектное решение в документации.

4.8. Дизъюнктивные подклассы

Многие системы позволяют нам явным образом задать, что несколько классов являются **дизъюнктивными**. Классы дизъюнктивные, если у них не может быть общих экземпляров. Например, в нашей онтологии классы Десертное вино и Белое вино не являются дизъюнктивными: существует множество вин, являющихся экземплярами обоих классов. Одним из таких примеров является Rothermel Trochenbierenauslese Riesling, экземпляр класса Сладкое Riesling. В то же время, классы Красное вино и Белое вино

дизъюнктивны: ни одно вино не может быть одновременно и белым, и красным. Определение классов как дизъюнктивных позволяет системе лучше проверять правильность онтологии. Если мы объявим классы Красное вино и Белое вино дизъюнктивными и затем создадим класс, который будет подклассом и Riesling (подкласс Белого вина) и Port (подкласс Красного вина), то система может показать, что имеется ошибка в моделировании.

5. Определение свойств

В этой главе мы затронем еще несколько деталей, которые нужно иметь при определении слотов в онтологии (Шаги 5 и 6 в Главе 3). В основном, мы обсуждаем обратные слоты и значения слота по умолчанию.

5.1. Обратные слоты

Значение слота может зависеть от значения другого слота. Например, если вино было произведено на винном заводе, то винный завод производит это вино. Эти два отношения, производитель и производит, называются **обратными отношениями**. Излишне хранить информацию и о том, и о другом. Когда мы знаем, что вино производится на винном заводе, то приложение, которое использует базу знаний, всегда может вывести значение для обратного отношения: винный завод производит вино. Тем не менее, с точки зрения приобретения знаний удобно иметь оба блока информации доступными в явном виде. Этот подход позволяет пользователям указать вино в одном случае и винный завод в другом. После это система приобретения знаний может автоматически заполнить значение для обратного отношения, обеспечивая согласованность базы знаний.

В нашем примере есть пара обратных слотов: слот производитель класса Вино и слот производит класса Винный завод. Когда пользователь создает экземпляр класса Вино и заполняет значение слота производитель, система автоматически добавляет вновь созданный экземпляр к слоту производит соответствующего экземпляра класса Винный завод. Например, когда мы говорим, что Sterling Merlot производится на заводе Sterling Vineyard, система автоматически добавляет Sterling Merlot к списку вин, которые производит завод Sterling Vineyard (рис. 9).

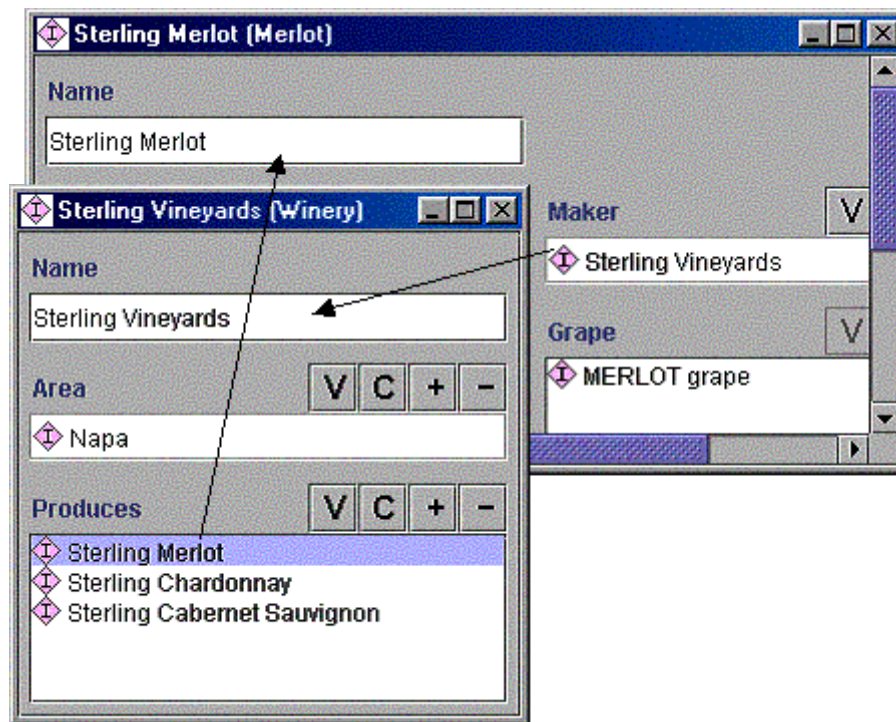


Рис. 9. Экземпляры с обратными слотами. Слот производит класса Винный завод является обратным для слота производитель класса Вино. Заполнение одного из слотов приводит к автоматическому обновлению другого.

5.2. Значения по умолчанию

Многие фреймовые системы позволяют определить для слотов значения по умолчанию. Если значение определенного слота одинаково для большинства экземпляров класса, то мы можем определить это значение как **значение слота по умолчанию**. Затем, когда создается каждый экземпляр класса, имеющего этот слот, система автоматически заполняет значение по умолчанию. После этого мы можем изменить это значение на любое другое, которое позволят фацеты. То есть, значения по умолчанию созданы для удобства: в любом случае они не накладывают какие-либо ограничения на модель или никак ее не меняют.

Например, если большинство вин, о которых мы собираемся говорить, являются крепкими, то значение крепости вина мы можем сделать «крепкое» по умолчанию. Тогда все вина, которые мы определяем, будут крепкими, если мы не укажем иное.

Обратите внимание, что это отличается от **значений слота**. Значения слота не могут быть изменены. Например, мы можем сказать, что слот сахар класса Десертное вино имеет значение «СЛАДКОЕ». Тогда у всех подклассов и экземпляров класса Десертное вино значение слота сахар будет «СЛАДКОЕ». Для всех подклассов или экземпляров этого класса это значение изменить нельзя.

6. Имена понятий в онтологиях

Определение единых правил присваивания имен понятиям в онтологии, а затем строгое соблюдение этих правил не только делает онтологию более простой для понимания, но также помогает избежать некоторых общих ошибок при моделировании. Существует много вариантов присваивания имен понятиям. Обычно нет особой

причины для выбора того или иного варианта. Тем не менее, нам нужно *Определить единые правила присваивания имен классам и слотам и придерживаться их.*

На выбор правил присваивания имен влияют следующие особенности системы представления знаний:

- Имеет ли система одно и то же пространство имен классов, слотов и экземпляров? То есть, позволяет ли система иметь класс и слот с одинаковым именем (как, например, класс винный завод и слот винный завод)?
- Различает ли система регистр букв? То есть, считает ли система разными имена, которые отличаются только регистром (как Винный завод и винный завод)?
- Какие разделители в именах позволяет использовать данная система? То есть, могут ли имена содержать пробелы, запятые, звездочки и т.д.?

К примеру, Protege- имеет единое пространство имен для всех своих фреймов. Она различает регистр букв. Таким образом, у нас не может быть класса винный завод и слота винный завод. Однако у нас может быть класс Винный завод (не прописные буквы) и слот винный завод. С другой стороны, CLASSIC не различает регистр букв и имеет разные пространства имен для классов, слотов и индивидуальных концептов. Таким образом, с точки зрения системы, мы можем с легкостью присвоить имя Винный завод и классу, и слоту.

6.1. Заглавные буквы и разделители

Во-первых, мы можем значительно улучшить читаемость онтологии, если мы все время будем писать названия понятий с большой буквы. Например, общепринято начинать имена классов с большой буквы, а имена слотов – с маленькой (предполагая, что система различает регистр букв).

Когда имя понятия содержит больше одного слова (как в Винный завод), нам нужно разделить слова. Вот возможные варианты:

- Использовать пробел: Винный завод (многие системы, включая Protege, позволяют использовать пробелы в именах понятий).
- Соединить слова вместе и каждое слово написать с большой буквы: ВинныйЗавод.
- Использовать в имени подчеркивание или тире, или другой разделитель: Винный_Завод, Винный_завод, Винный-Завод, Винный-завод (если вы используете разделитель, вам также нужно решить, писать каждое слово с большой буквы или нет).

Если система представления знаний позволяет использовать пробелы в именах, то для многих разработчиков онтологий пробелы могут быть самым естественным решением. Однако важно учитывать другие системы, с которыми может взаимодействовать ваша система. Если в этих системах не используются пробелы или ваше средство представления не очень хорошо обрабатывает пробелы, то может быть лучше использовать другой метод.

6.2. Единственное или множественное число

Имя класса представляет набор объектов. Например, класс Вино в действительности представляет все вина. Поэтому для многих разработчиков было бы естественнее дать классу имя Вина, а не Вино. Ни один из вариантов не лучше и не хуже другого (хотя на практике для имен классов чаще используется единственное число). Тем не менее, каким бы ни был выбор, его следует придерживаться на протяжении всей онтологии. Некоторые системы даже требуют от своих пользователей заранее объявить, какое число (единственное или множественное) они будут использовать в именах классов, и не дают им отклоняться от своего выбора.

Использование все время одной и той же формы также предотвращает такие ошибки разработчика при моделировании, как создание класса Вина, а затем создание класса Вино как его подкласса (см. Раздел 4.1).

6.3. Договоренность в отношении использования префиксов и суффиксов

Некоторые методологии по базам знаний советуют придерживаться договоренности в отношении использования префиксов и суффиксов в именах для того, чтобы различать классы и слоты. Существует две распространенных традиции: добавлять к именам слотов *has-* или предлог *-of*. Таким образом, наши слоты меняются на *его-производитель* и *его-винный_завод*, если мы выберем использование *его-*. Слоты меняются на *maker-of* и *winery-of*, если мы выберем использование *of-*. Этот подход позволяет любому, кто посмотрит на термин, сразу же определить, что это: класс или слот. Однако имена терминов становятся немного длиннее.

6.4. Другие соображения по присваиванию имен

Еще несколько моментов, которые нужно иметь в виду при определении правил присваивания имен:

- Не добавляйте к именам понятий такие строки как «класс», «свойство», «слот» и т.д.

Из контекста всегда ясно, что это, к примеру, класс или слот. В дополнение к тому, что для классов и слотов вы используете разные правила присваивания имен (скажем, пишете их с большой и с маленькой буквы соответственно), само имя будет показывать, чем является это понятие.

- Обычно лучше не сокращать имена понятий (то есть, используйте Cabernet Sauvignon, а не Cab).
- Имя надкласса должно входить или во все имена прямых подклассов, или ни в одно из них. Например, если мы создаем два подкласса класса Вино для представления красных и белых вин, то подклассы должны называться или Красное Вино и Белое Вино, или Красное и Белое, но не Красное Вино и Белое.

7. Другие ресурсы для создания онтологий

В наших примерах в качестве среды разработки онтологий мы использовали Protege-. Duineveld с коллегами (Duineveld et al.) описывает и сравнивает ряд других сред для разработки онтологий.

Мы постарались рассказать о самом основном о разработке онтологий и не коснулись многих углубленных тем или альтернативных методологий разработки онтологий. Gyzmez-Рйrez (Gyzmez-Рйrez 1998) и Uschold (Uschold and Gruninger 1996) представляют альтернативные методологии разработки онтологий. В руководстве по Ontolingua (Farquhar 1997) говорится о некоторых формальных аспектах моделирования знаний.

В настоящее время исследователи придают особое значение не только разработке онтологий, но также и анализу онтологий. Чем больше онтологий будет создаваться и повторно использоваться, тем больше будет инструментальных средств для анализа онтологий. К примеру, Chimaera (McGuinness et al.) предоставляет диагностические инструментальные средства для анализа онтологий. Анализ, который осуществляет Chimaera, включает как проверку логической верности онтологии, так и диагностику типичных ошибок при проектировании онтологий. Разработчик онтологий может провести диагностику разрабатываемой онтологии с помощью Chimaera, чтобы определить соответствие общим способам моделирования онтологий.

Вопросы для самоконтроля и текущего контроля

- 1) Дайте определение онтологии
- 2) Объясните зачем необходимо создавать онтологии?
- 3) Перечислите из чего состоит онтология?
- 4) Опишите методологию инженерии знаний
- 5) Дайте определение классу и понятию в онтологии
- 6) Опишите назначение иерархии классов в онтологии
- 7) Объясните для чего применяются свойства в онтологиях
- 8) Перечислите рекомендации именования понятий в онтологиях
- 9) Укажите какие есть ресурсы для создания онтологий помимо Protege.
- 10) Опишите как применяются онтологии при решении прикладных задач.

Задание на самостоятельную работу

- 1) Практическое задание

Выполнить создание спецификации знаний о предметной области в виде онтологии в редакторе Protege 5.5.0 согласно заданию практической работы №4.

- 2) Самостоятельная работа

Исследовать синтаксис и лексику современных языков создания онтологий.

Список рекомендуемой литературы¹:

[1] Сидоркина, Ирина Геннадьевна. Системы искусственного интеллекта [Текст] : [учеб. пособие для вузов по направлению 230100 "Информатика и вычисл. техника"] / И. Г. Сидоркина. М.: Кнорус, 2011. - 245 с.

Рассмотрены теоретические аспекты проектирования систем искусственного интеллекта и даны модели представления знаний, изложены материалы информационного, справочного, консультирующего характера по использованию инструментальных средств и технологическим особенностям разработки систем данного класса.

Подробнее: <https://www.labirint.ru/books/276217/>

[2] Богомолова, М. А. Экспертные системы (техника и технология проектирования) [Текст] : Методические указания к лабораторным работам / М. А. Богомолова. Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015. - 47 с.

Методические указания являются руководством при выполнении лабораторных занятий, проводимых по курсу «Экспертные системы (техника и технология проектирования)» с магистрами направления 09.04.03 - «Прикладная информатика» в терминальном классе. Лабораторные работы связаны с изучением принципов построения экспертных систем на основе различных моделей представления знаний, и могут быть полезны студентам других специальностей.

Подробнее: <https://www.iprbookshop.ru/71908.html>

[3] Советов, Борис Яковлевич. Представление знаний в информационных системах [Текст] : [учеб. для вузов по направлению подгот. "Информ. системы и технологии"] / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. М.: Академия, 2011.

Учебник создан в соответствии с Федеральным государственным образовательным стандартом по направлению бакалавриата «Информационные системы и технологии». Рассмотрены современное состояние теоретических и прикладных вопросов представления знаний в информационных системах, идеология построения интеллектуальных систем, математический аппарат представления знаний, возможности и пути использования искусственного интеллекта при проектировании информационных систем. Изложены новые аспекты представления знаний на основе искусственных нейронных сетей, расчетно-логических систем, генетических алгоритмов, мультиагентных систем. Приведены примеры практической реализации представления знаний на базе декларативного языка Пролог. Для студентов учреждений высшего профессионального образования. Может быть полезен разработчикам и пользователям информационных систем; преподавателям и научным сотрудникам, сферой интересов которых является интеллектуализация различных предметных областей; менеджерам и руководителям различного ранга, желающим самостоятельно ознакомиться с современным состоянием информационных технологий.

Подробнее: <https://lavkababuin.com/predstavlenie-znaniy-v-informacionnyh-sistemah-2-e-izd-ster-650564/>

[4] Остроух, А. В. Интеллектуальные информационные системы и технологии [Электронный ресурс] : монография / А. В. Остроух, А. Б. Николаев. Санкт-Петербург: Лань, 2019. - 308 с.

В монографии изложены концептуальные основы и методы представления знаний в интеллектуальных системах. Рассмотрены различные подходы, применяемые при проектировании и разработке интеллектуальных систем и технологий в транспортном комплексе, а также рассмотрены тенденции развития систем искусственного интеллекта.

Монография может быть использована для формирования профессиональной компетентности студентов высших учебных заведений, аспирантов и научных сотрудников обучающихся и ведущих научные исследования в области разработки и практического применения систем искусственного интеллекта по укрупнённой группе направлений подготовки "Информатика и вычислительная техника".

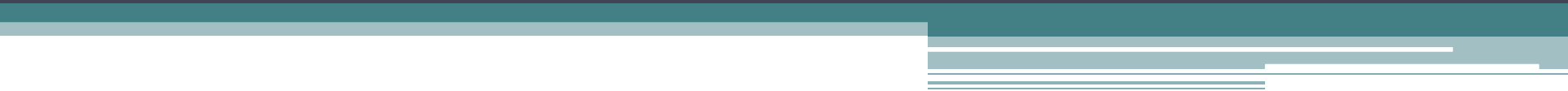
Подробнее: <https://www.labyrinth.ru/books/692861/>

Ссылки на открытые ресурсы (он-лайн-курсы, видеоуроки и т.д.)

[5]

¹Список рекомендуемой литературы может быть более широким по сравнению со списком литературы, указанным в рабочей программе

История искусственного интеллекта

A series of horizontal lines in teal and light blue colors, with varying lengths and thicknesses, extending from the left edge of the slide towards the right.

Сейчас технологии развиваются с немыслимой скоростью. Ранее те возможности, что, казалось бы, были доступны только профессиональным ученым, в современной жизни доступны каждому. Один из подобных прорывов – искусственный интеллект(ИИ), прочно обосновавшийся во многих сферах человеческой жизни.

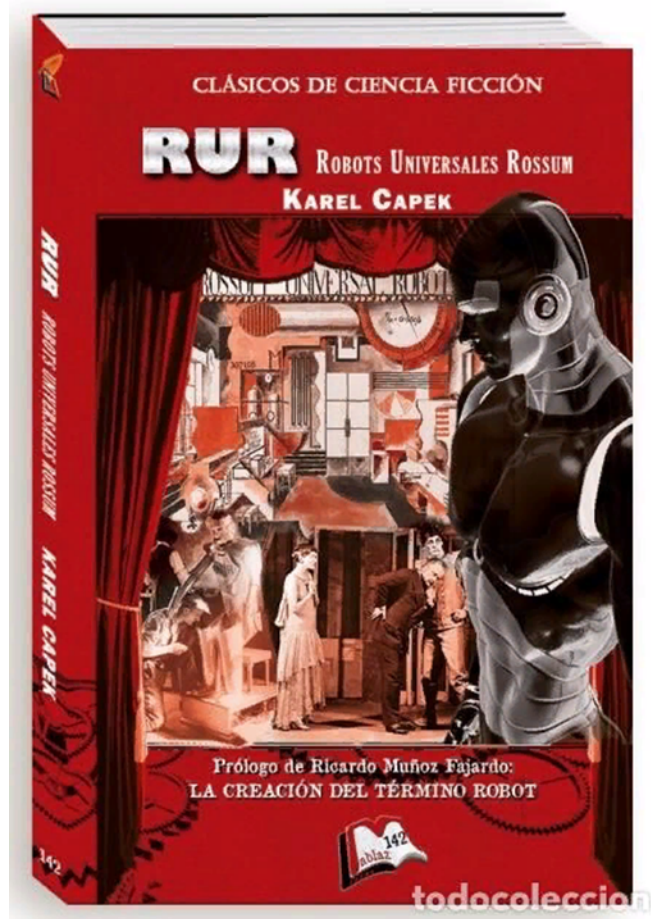
Так что же такое ИИ, как он возник, где применяется, а также чем он отличается от человеческого разума?

Предпосылки появления ИИ

Идея искусственного интеллекта давно волнует человечество. Автоматам уделялось много внимания в мифах Древней Греции, и наиболее известным примером является искусственная женщина Пандора, созданная Зевсом. В 17 веке некоторые философы размышляли над возможностью «вложить разум» в неодушевлённые предметы. Было выдвинуто множество теорий. К примеру, Рене Декарт верил в дуализм разума и тела. У Лейбница были другие взгляды. Он полагал, что вся человеческая мысль может быть представлена математически с помощью элементарных символов. Искусственный интеллект и автоматы фигурируют в художественной литературе с незапамятных времён. В качестве примера можно привести «Франкенштейн, или Современный Прометей» Мэри Шелли и «Россумские универсальные роботы» Карела Чапека.



"Рисунок 1 – книга «Франкенштейн» Мери Шелли"



" Рисунок 2 – книга «Россумские универсальные роботы» Карела Чепека "

Это были философские предпосылки для появления ИИ. Теперь давайте поговорим о конкретных учёных и их трудах, благодаря которым искусственный интеллект из выдумки стал реальностью. Относительно конкретная и подкреплённая фактами история началась относительно недавно. Знаменитый тест Тьюринга впервые был проведён в 1950, но лишь спустя шесть лет, в 1956 году, ИИ начал формироваться как отдельная дисциплина. Давайте посмотрим, как это было.

Разница между искусственным и естественным интеллектом

Сравнивать искусственный и естественный интеллект можно лишь по некоторым общим параметрам. Например, человеческий мозг и компьютер работают по примерно схожему принципу, включающему четыре этапа – кодирование, хранение данных, анализ и предоставление результатов. И естественный, и искусственный разум склонны к самообучению, они решают те или иные задачи и проблемы, используя специальные алгоритмы.

Помимо общих умственных способностей к рассуждению, обучению и решению проблем, человеческое мышление также имеет эмоциональную окраску и сильно зависит от влияния социума. Искусственный интеллект не имеет никакого эмоционального характера и не ориентирован социально.

Если говорить об IQ – большинство ученых склонны считать, что сей параметр оценки никак не связан с искусственным интеллектом. С одной стороны, это действительно так, ведь стандартные IQ-тесты направлены на измерение «качества» человеческого мышления и связаны с развитием интеллекта на разных возрастных этапах.

С другой стороны, для ИИ создан собственный «IQ-тест», названный в честь Тьюринга. Он помогает определить, насколько хорошо машина обучилась и способна ли она уподобиться в общении человеку. Это своего рода планка для ИИ, установленная людьми. А ведь все больше ученых склоняется к тому, что скоро компьютеры обгонят человечество по всем параметрам... Развитие технологий идет по непредсказуемому сценарию, и вполне допустимо, что так и будет.

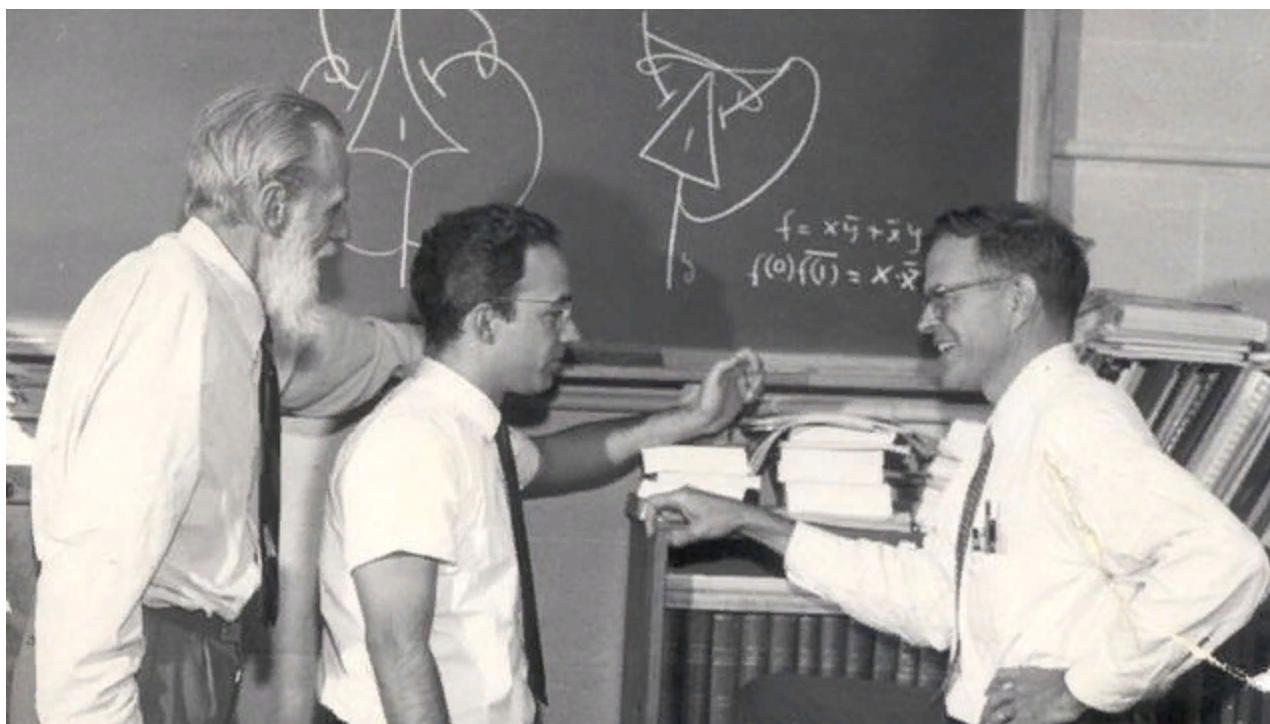
Первые шаги

Количество научных работ, связанных с искусственным интеллектом, резко выросло в 1950-60-х годах. Но и ранее встречались исследования, затрагивающие данную тему. Бертран Рассел и Альфред Норт Уайтхед опубликовали «Принципы математики» в 1913 году. Примерно в то же время Джордж Буль выдвинул свои законы мысли. Так были заложены основы математической логики.

Возможно, всё началось в тот момент, когда непонятный 15-летний мальчишка ворвался в кабинет Рудольфа Карнапа. Карнап тогда уже стал влиятельным философом и преподавал в Чикагском университете. Он опубликовал «Логический синтаксис языка». Мальчик пришел без разрешения, и указал на ошибки в этой работе. Рудольф был потрясен. Посетитель был необычным. Кроме того, он даже не представился, и сразу убежал. После нескольких месяцев поисков Рудольф наконец нашел своего визитёра в одном из местных университетов. Им оказался Уолтер Гарри Питтс.

За три года до этого Уолтер написал письмо Бертрону Расселу, указав на проблемы, найденные в вышеупомянутых «Принципах». Рассел оказался настолько впечатлён, что пригласил мальчика учиться в аспирантуре Кембриджского университета в Великобритании. Уолтер, хотя и рос в неблагополучной семье, не решился на переезд из Детройта. Однако, когда Рассел приехал с лекциями в Чикаго, Питтс сбежал из дома, чтобы учиться у него. Студентом Чикагского университета он не стал, но прилежно ходил на лекции.

В 1942 году Уолтер Питтс встретился с Уорреном МакКаллоком. Маккаллок пригласил Питтса остаться у него дома. Они были убеждены в правильности теории Лейбница, которая предполагала возможность «механизации» человеческой мысли. И пытались создать модель для нейробиологии нервной системы человека. Они опубликовали свою основную статью об этом же в 1943 году, назвав её «Логическое исчисление идей, относящихся к нервной активности». Этот документ внёс неоценимый вклад в область искусственного интеллекта. Учёные предложили простую модель, известную как математический нейрон Маккаллока — Питтса. Она до сих пор изучается на курсах машинного обучения. Идеи, которые предложили учёные, являются основой почти всех современных ИИ.

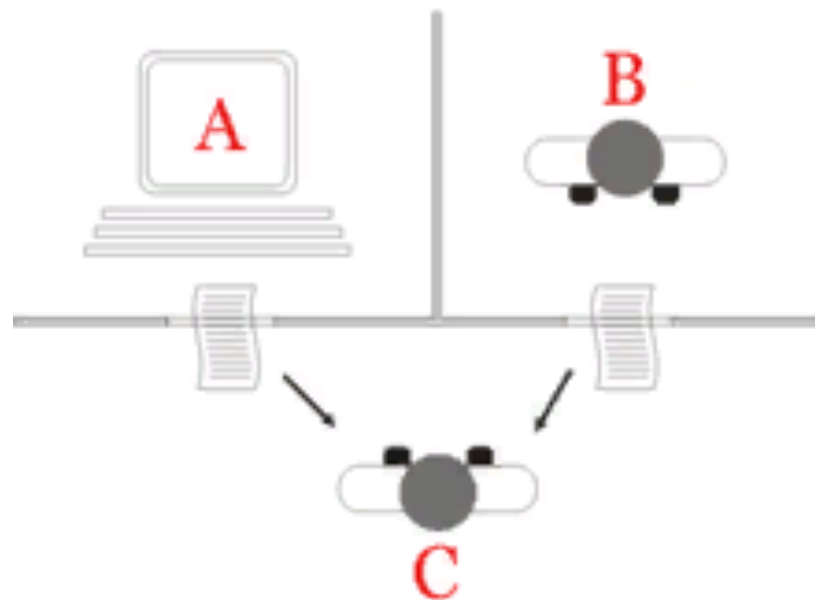


" Рисунок 3- Уоррен МакКаллок и Уолтер Питтс "

Кибернетика Норберта Вейнера и теория информации Клода Шеннона были опубликованы в 1948 году. Кибернетика — это исследование «управления и коммуникации в живом организме и в машине». Теория информации — это измерение количества информации, её хранения и передачи. Обе работы сильно повлияли на сферу ИИ.

Кибернетика обеспечила непосредственное исследование биологического и механического интеллекта. А теория информации повлияла на базовую математику.

Спустя пару лет Алан Тьюринг выступил с тестом Тьюринга. Он описал способ определения, является ли машина «умной». Упрощённо тест выглядит следующим образом: человек общается с одним компьютером и одним человеком. На основании их ответов на вопросы он должен определить, с кем он разговаривает: с человеком или компьютерной программой. Задача компьютерной программы — ввести человека в заблуждение, заставив сделать неверный выбор. Хотя тест Тьюринга слишком ограничен для тестирования современных «умных» систем, в то время это был настоящий прорыв. Имя Алана Тьюринга попало в прессу, сделав отрасль более популярной.



" Рисунок 4 – тест Тьюринга "

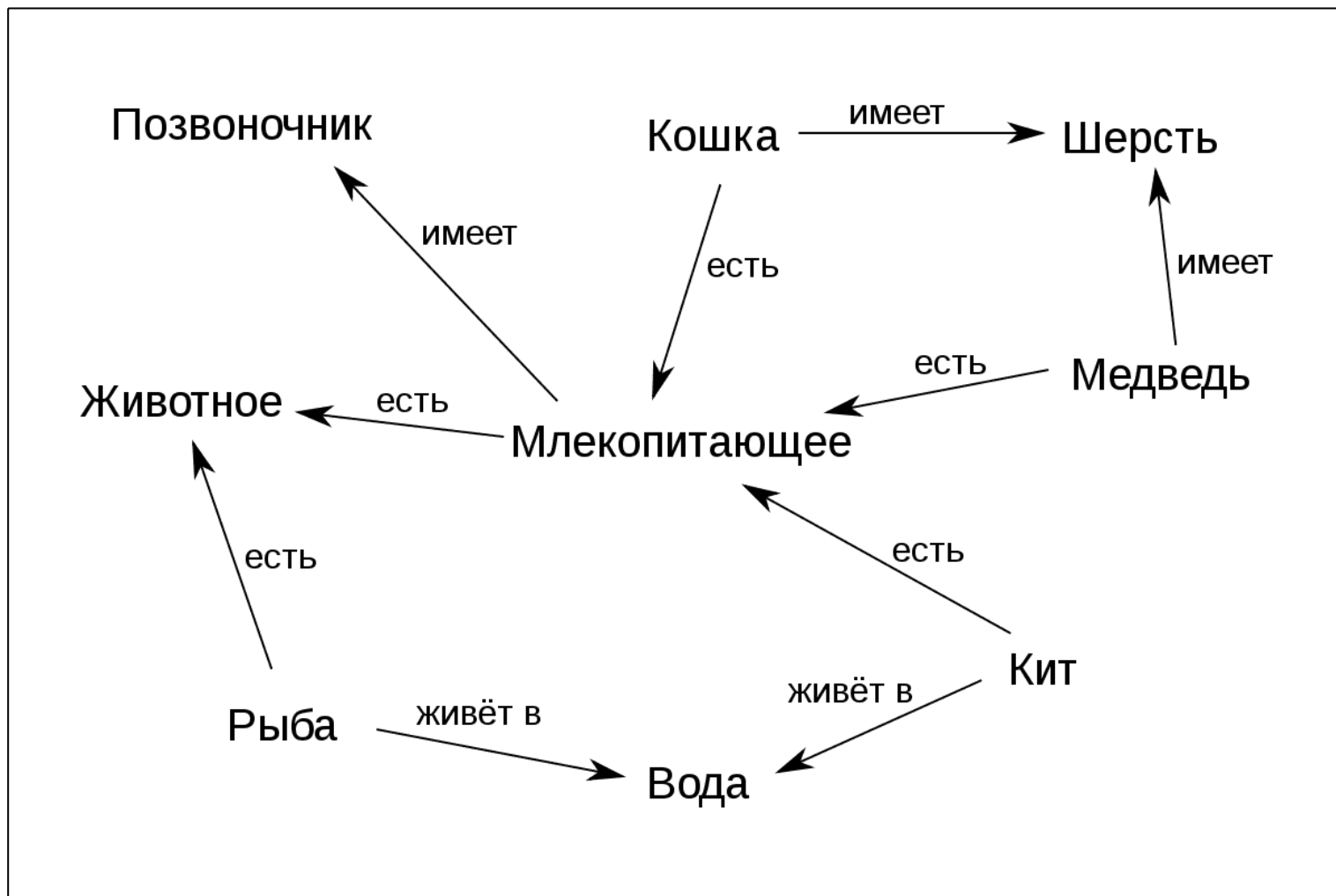
В 1956 в Дартмуте прошла конференция, посвящённая теме изучения концепций «механизации» интеллекта. Позже участники этой конференции станут выдающимися личностями в области ИИ. Наиболее известным был Марвин Мински, который в 1951 году создал первую машину нейронных сетей, SNARC. Он станет самым известным именем в мире искусственного интеллекта на последующие десятилетия.

Присутствовал на конференции и Клод Шеннон. А будущий нобелевский лауреат Герберт А. Саймон и Аллен Ньюэлл дебютировали со своей «Логикой-теоретикой». Впоследствии он решит 38 из первых 52 теорем в «Принципах математики» Рассела. Джон Маккарти — также один из пионеров искусственного интеллекта — придумал название «Искусственный интеллект». Участники согласились с этим термином. Это было рождение ИИ.

Развитие отрасли (1956–1974)

Благодаря этой конференции интерес к ИИ вырос, появилось множество интересных разработок. В 1959 году Ньюэлл и Саймон создали «Решатель общих проблем», который теоретически мог решить любую формализованную проблему. А Джеймс Слэйгл создал SAINT эвристическую программу, решавшую проблемы символической интеграции в исчислении. Эти программы были впечатляющими.

После создания теста Тьюринга естественность языка стала важной областью ИИ. Программа Даниэля Боброу STUDENT смогла решить проблемы со словом в средней школе. Чуть позже появилось понятие семантической сети — карты различных понятий и отношений между ними (как на рисунке). Несколько успешных программ были построены на этой сети. В 1966 году Джозеф Вайзенбаум создал ELIZA. Виртуальный собеседник мог вести реалистичные беседы с людьми.



" Рисунок 5 – семантическая сеть "

В связи со столь активным развитием отрасли ИИ учёные выдвигали крайне смелые заявления:

- Ньюэлл и Саймон, 1958 год: «Через 10 лет цифровой компьютер станет чемпионом мира по шахматам». И «в течение 10 лет цифровой компьютер обнаружит и докажет важную новую математическую теорему».
- Саймон Г.А., 1965 г.: «Машины будут способны в течение 20 лет выполнять любую работу, которую может выполнять человек».
- Марвин Мински, 1967 г.: «Через поколение... проблема создания искусственного интеллекта »будет существенно решена».
- Марвин Мински, 1970 г.: «Через три-восемь лет у нас будет машина с общим интеллектом обычного человека».

Так или иначе, в отрасль пришли деньги. Управление перспективных исследовательских проектов выделило команде MIT 2,2 миллиона долларов. Тогдашний президент DARPA решил, что они должны «финансировать людей, а не проекты», и создал свободную исследовательскую культуру. Это позволило исследователям осуществлять любые проекты, которые они считали правильными.

Временное похолодание (1974–1980)

В 1969 году Марвин Мински и Сеймур Паперт опубликовали свою книгу «Перцептроны». В ней они показали принципиальные ограничения перцептронов и выдвинули на первый план неспособность перцептронов управлять элементарной схемой XOR. Это привело к смещению интереса исследователей искусственного интеллекта в противоположную от нейросетей область символьных вычислений.

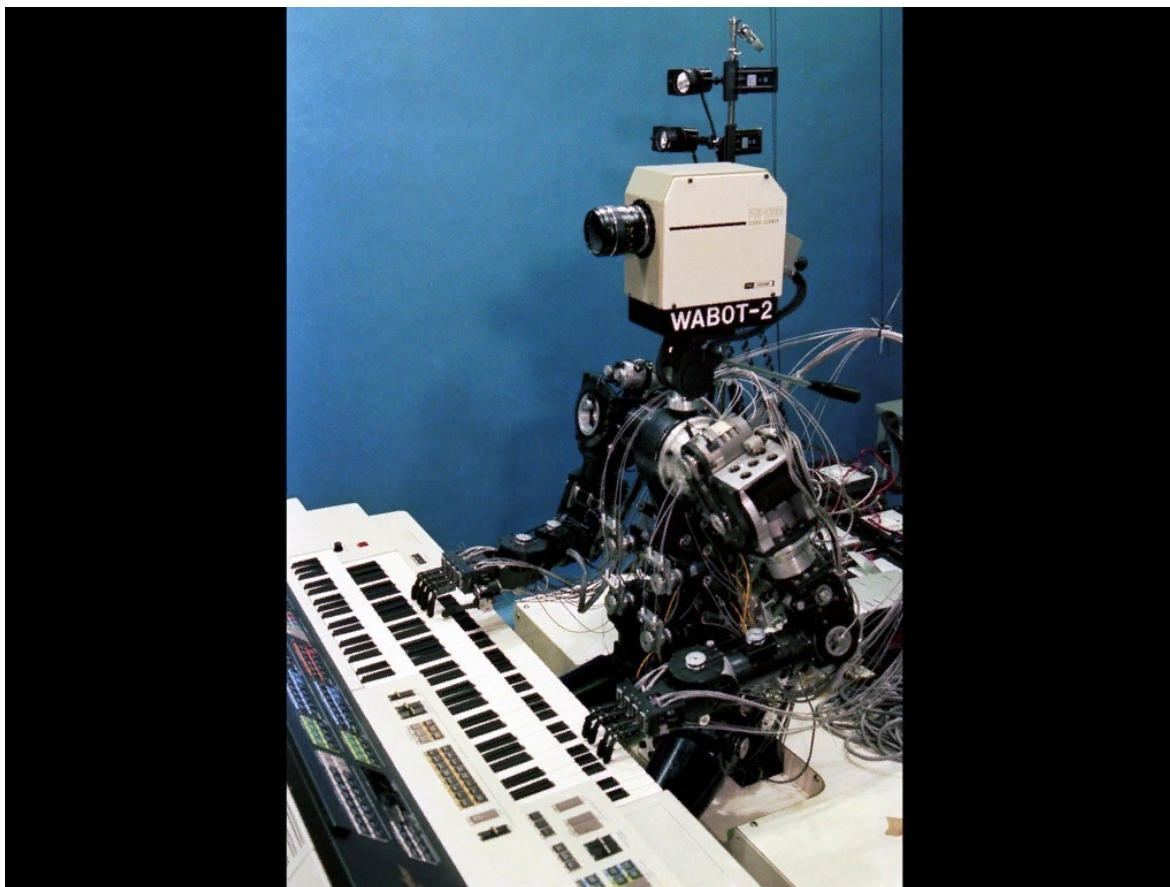
Альтернативный подход символьного ИИ получил взрывной рост. Но этот подход не дал каких-либо существенных результатов. В 1970-х годах стало ясно, что исследователи ИИ чрезмерно оптимистичны в отношении ИИ. Цели, которые они обещали, ещё не были достигнуты и их достижение казалось делом весьма отдалённого будущего.

Исследователи поняли, что упираются в стену. ИИ применялся к простым задачам. Но реальные сценарии оказались слишком сложными для этих систем. Число возможностей, которые алгоритмы должны были исследовать, получалось астрономическим. Это привело к проблеме комбинаторного взрыва. А затем возник классический вопрос «как сделать компьютер умным». Это была проблема здравого смысла.

Всё это привело к тому, что инвесторы разочаровались в технологии. Финансирование ИИ, таким образом, исчезло, а исследования прекратились. DARPA тоже больше не могла поддерживать хакерскую культуру исследований из-за изменений в законодательстве. Спонсоры заморозили финансирование исследований в области искусственного интеллекта. Поэтому период 1974–1980 гг. называют «зимой ИИ».

Возрождение

В 1981 году правительство Японии начало серьёзно вкладываться в исследования в области искусственного интеллекта. Страна выделила 850 млн долларов на проект «Компьютер пятого поколения». Он был нацелен на развитие ИИ. Предполагаемые компьютеры должны были общаться, переводить другие языки, распознавать картинки. Ожидалось, что компьютеры станут базой для создания устройств, способных имитировать мышление.



" Рисунок 6 – Wabot-2 "

Исследования в Японии казались многообещающими. В 1980 году в университете Васэда был разработан робот Wabot-2. Он мог общаться с людьми, читать музыкальные партитуры и играть на электронном органе. Успех японцев заставил правительства других стран и частный бизнес снова обратить свой взгляд в сторону ИИ.

В этот момент возник вопрос коннекционизма. В 1982 году Хопфилд создал новую форму нейронной сети, которая способна изучать и обрабатывать информацию.

Джеффри Хинтон и Дэвид Румелхарт популяризировали авторазличие в обратном режиме. Эта технология была феноменальной и остается жизненно важной в машинном обучении и по сей день. ИИ получил коммерческий успех в виде «экспертных систем». Эти системы обладали глубоким знанием конкретной темы. Университет Карнеги — Меллона (CMU) в 1980 году выпустил экспертную систему под названием XCON. Она использовалась в корпорации Digital Equipment. До 1986 года компания ежегодно экономила 40 млн долларов.

Вторая зима

В конце 1980-х успешный рост отрасли ИИ омрачила «компьютерная революция». Apple и IBM производили все более и более мощные компьютеры одновременно. Настольные компьютеры стали дешевле и мощнее, чем Lisp-машины на базе искусственного интеллекта. Целая индустрия стоимостью полмиллиарда долларов была разрушена за одну ночь. Успехи ИИ в таких экспертных системах, как машина XCON, оказались слишком дорогими.



" Рисунок 7 -ПК революция изменила курс развития ИИ "

Возникли проблемы с экспертными системами. Они не могли учиться. Они были «хрупкими» (то есть могли совершать серьёзные ошибки, если им давали необычные данные). Практическая область применения экспертных систем стала ограниченной. Новое руководство DARPA решило, что ИИ не является «следующей волной». И перенаправило инвестиции на проекты, которые, с их точки зрения, должны были принести немедленные результаты.

К 1991 году цели компьютерного проекта Японии так и не были достигнуты. Учёные недооценили те трудности, с которыми им пришлось столкнуться. Свыше 300 ИИ-компаний закрылись, обанкротились или были куплены к концу 1993 года. Это был фактически конец первой волны коммерческого использования искусственного интеллекта.

Свежая струя

Когда отголоски компьютерной революции начали затихать, у людей появилась возможность пользоваться большими вычислительными мощностями. С повсеместным распространением компьютеров стало расти и количество разнообразных баз данных. Это всё было очень здорово и с точки зрения развития ИИ.

Новые технологии решали проблемы и снимали барьеры, мешавшие учёным. Пользуясь возрастающей вычислительной мощностью, исследователи раздвигали границы возможного. Из баз данных, ставших Big Data, удавалось извлечь всё больше и больше знаний. Возможности практического применения ИИ стали более очевидными.

Новая концепция под названием «интеллектуальный агент» утвердилась в 1990-х годах. Интеллектуальный агент (ИА) — это система, которая самостоятельно выполняет задание, выданное пользователем, в течение длительных промежутков времени. Появилась надежда, что однажды мы сможем научить ИА взаимодействовать друг с другом. Это привело бы к созданию универсальных и более «умных» систем.

В сообществе ИИ бродили разные мнения об использовании математики в сфере искусственного интеллекта. Одни считали, что интеллект слишком сложен для того, чтобы описывать его математическими символами. По их мнению, люди редко руководствуются логикой при принятии решений. Их оппоненты возражали, что логические цепочки — это путь вперёд.



" Рисунок 8 - Г. Каспаров играет с Deep Blue "

Очень скоро, в 1997 году, суперкомпьютер IBM Deep Blue победил Гарри Каспарова. В тот момент Каспаров был чемпионом мира по шахматам. То, что, по словам Ньюэлла и Саймона, должно было произойти к 1968 году, наконец-то свершилось в 1997 году.

Современные шахматные компьютеры намного сильнее любого человека. Самый высокий рейтинг Эло, когда-либо достигнутый человеком, составляет 2882. Для компьютеров вполне обычный показатель — 3000 Эло. Самый высокий показатель за всю историю был более 3350.

В 2005 году Стэнфорд разработал робота для автономного вождения. Он выиграл DARPA Grand Challenge, проехав 131 милю (211 км) по неизученной пустынной тропе.

В феврале 2011 года IBM решила протестировать свой IBM Watson в викторине «Jeopardy». Компьютер смог победить двух величайших чемпионов Jeopardy со значительным отрывом.

С лавинообразным развитием интернета и соцсетей росло и количество информации. ИТ-компаниям необходимо было что-то делать с получаемыми данными. И применение ИИ стало необходимостью, а не развлечением.



" Рисунок 9 - Jeopardy "

Теперь Google сортирует выдачу, используя системы машинного обучения. YouTube подбирает рекомендуемые видео с помощью алгоритмов ML, по той же схеме рекомендует товары Amazon. Новостная лента Facebook генерируется «умным» компьютером. И даже Tinder находит людей, используя алгоритмы ML. Благодаря компьютерной революции ИИ-технологии стали нашими незаменимыми помощниками. И теперь человечество смотрит вперёд, ожидая создание супермашин. И хотя последствия достижений искусственного интеллекта порой спорны, дальнейшее развитие этой технологии неизбежно.

Заключение

Не так давно, казалось бы, ученые ввели понятие «искусственный интеллект», а чуть больше полвека спустя технология уже находит широкий спрос в самых различных сферах. Сейчас искусственный разум, можно сказать, находится в шаговой доступности для любого человека — компьютер и ноутбук, смартфон и электронные часы, даже многие простейшие приложения

Контрольные вопросы:

В каком году был впервые проведен тест Тьюринга?

Что такое кибернетика?

Кто создал первую машину нейронных сетей?

Кто такой Джон Маккарти?

Кто и в каком году создал «Решатель общих проблем»?

Литература

- <https://timeweb.com/ru/community/articles/chtotakoe-iskusstvennyy-intellekt>
- <https://habr.com/ru/company/cloud4y/blog/469447/>
- <https://wiki.programstore.ru/istoriya-iskusstvennogo-intellekta/>

асибо за вниман

Искусственный интеллект

Пушкарёв Виктор Николаевич

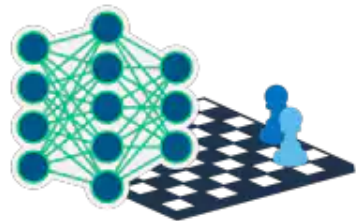
Что это такое и почему это так важно

Искусственный интеллект (ИИ) позволяет компьютерам обучаться на собственном опыте и выполнять те задачи, которые раньше были под силу только человеку. В большинстве случаев реализации ИИ - от компьютерных шахматистов до беспилотных автомобилей - крайне важна возможность глубокого обучения и обработки естественного языка. Благодаря этим технологиям компьютеры можно «научить» выполнению задач с помощью обработки большого объема данных и выявления в них закономерностей.

История развития искусственного интеллекта

Термин «искусственный интеллект» появился в 1956 году. Первые исследования в области ИИ в 50-х годах прошлого века были направлены на решение проблем и разработку систем символьных вычислений. В 60-х годах это направление привлекло интерес Министерства обороны США: американские военные начали обучать компьютеры имитировать мыслительную деятельность человека. Эти работы стали основой для принципов автоматизации и формальной логики рассуждений, которые используются в современных компьютерах.

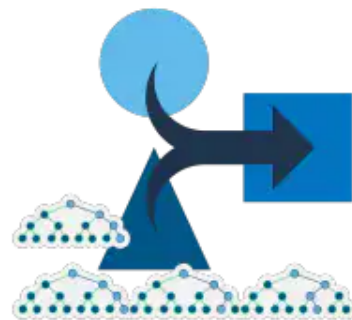
Этапы развития искусственного интеллекта



1950s-1970-e

Нейросети

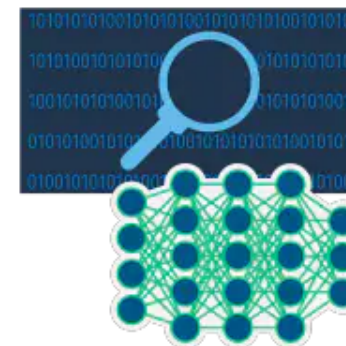
Первые разработки в области нейросетей вызвали ажиотаж в связи с возможностью создания «мыслящих» машин.



1980s-2010-e

Машинное обучение

Становятся популярными технологии машинного обучения.



Настоящее время

Глубокое обучение

Прорывы в сфере глубокого обучения привели к расцвету технологий ИИ.

В чем заключается важность искусственного интеллекта?

ИИ позволяет автоматизировать повторяющиеся процессы обучения и поиска за счет использования данных. Однако ИИ отличается от роботизации, в основе которой лежит применение аппаратных средств. Цель ИИ — не автоматизация ручного труда, а надежное и непрерывное выполнение многочисленных крупномасштабных компьютеризированных задач. Такая автоматизация требует участия человека для первоначальной настройки системы и правильной постановки вопросов.

ИИ делает существующие продукты интеллектуальными. Как правило, технология ИИ не реализуется как отдельное приложение. Функционал ИИ интегрируется в имеющиеся продукты, позволяя усовершенствовать их, точно так же, как технология Siri была добавлена в устройства Apple нового поколения. Автоматизация, платформы для общения, боты и «умные» компьютеры в сочетании с большими объемами данных могут улучшить различные технологии, которые используются дома и в офисах: от систем анализа данных о безопасности до инструментов инвестиционного анализа.

ИИ адаптируется благодаря алгоритмам прогрессивного обучения, чтобы дальнейшее программирование осуществлялось на основе данных. ИИ обнаруживает в данных структуры и закономерности, которые позволяют алгоритму освоить определенный навык: алгоритм становится классификатором или предикатором. Таким образом, по тому же принципу, по которому алгоритм осваивает игру в шахматы, он может научиться предлагать подходящие продукты онлайн. При этом модели адаптируются по мере поступления новых данных. Обратное распространение — это метод, который обеспечивает корректировку модели посредством обучения на базе новых данных, если первоначальный ответ оказывается неверным

ИИ осуществляет более глубокий анализ больших объемов данных с помощью нейросетей со множеством скрытых уровней. Несколько лет назад создание системы обнаружения мошенничества с пятью скрытыми уровнями было практически невозможным. Все изменилось с колоссальным ростом вычислительных мощностей и появлением «больших данных». Для моделей глубокого обучения необходимо огромное количество данных, так как именно на их основе они и обучаются. Поэтому чем больше данных, тем точнее модели.

Глубинные нейросети позволяют ИИ достичь беспрецедентного уровня точности. К примеру, работа с Alexa, поисковой системой Google Search и сервисом Google Photos осуществляется на базе глубокого обучения, и чем чаще мы используем эти инструменты, тем эффективнее они становятся. В области здравоохранения диагностика раковых опухолей на снимках МРТ с помощью технологий ИИ (глубокое обучение, классификация изображений, распознавание объектов) по точности не уступает заключениям высококвалифицированных рентгенологов.

ИИ позволяет извлечь максимальную пользу из данных. С появлением самообучающихся алгоритмов сами данные становятся объектом интеллектуальной собственности. Данные содержат в себе нужные ответы — нужно лишь найти их при помощи технологий ИИ. Поскольку сейчас данные играют гораздо более важную роль, чем когда-либо ранее, они могут обеспечить конкурентное преимущество. При использовании одинаковых технологий в конкурентной среде выиграет тот, у кого наиболее точные данные.

Как используется искусственный интеллект?

Функционал ИИ широко востребован во всех отраслях, особенно это касается вопросно-ответных систем, которые могут применяться при оказании правовой помощи, поиске патентов, оповещении о рисках и в медицинских исследованиях. Прочие возможности применения ИИ представлены ниже.

Здравоохранение

Технологии ИИ могут применяться в персонализированной медицине и при расшифровке рентгеновских снимков. Персональные медицинские помощники могут напоминать пользователям, что нужно принять лекарство, выполнить физические упражнения или перейти на более здоровый режим питания.

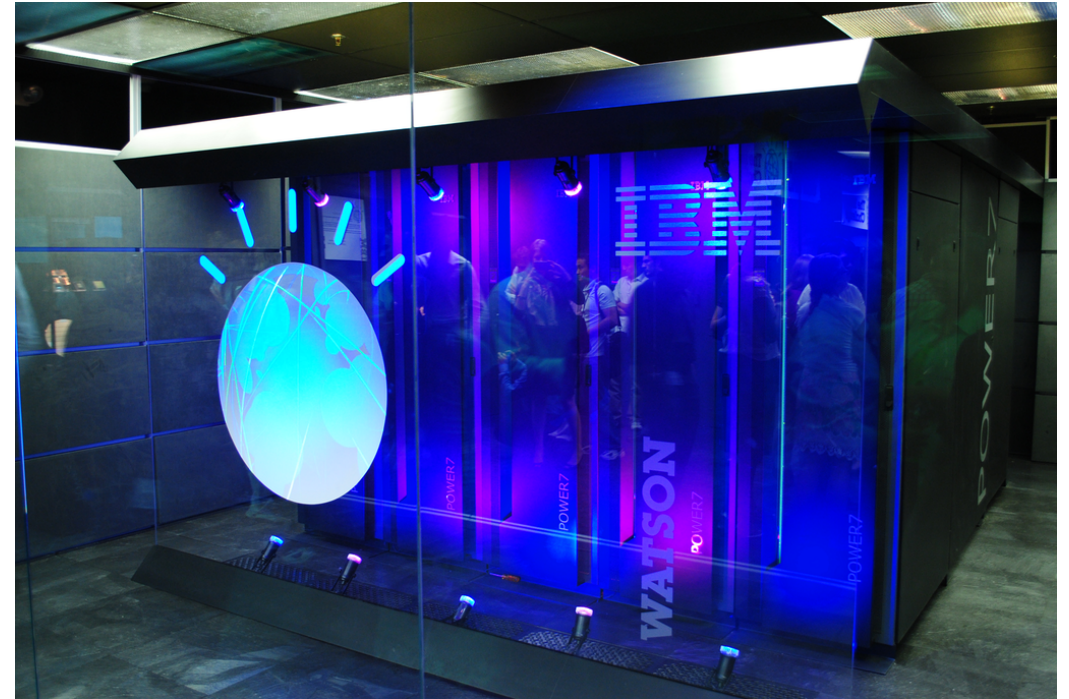


Рисунок 1 – ИИ для распознавания онкологии IBM Watson

Промышленность

ИИ может анализировать данные IoT с производственного участка, получаемые от подключенного оборудования, и прогнозировать загрузку и спрос с помощью рекуррентных сетей - особого вида сетей глубокого обучения, используемых для работы с последовательными данными.

Ритейл

ИИ помогает совершать покупки онлайн с индивидуально подобранными рекомендациями, а также дает возможность продавцам обсуждать покупки с клиентами. Кроме того, технологии ИИ могут оптимизировать процессы управления товарными запасами и размещения товара.

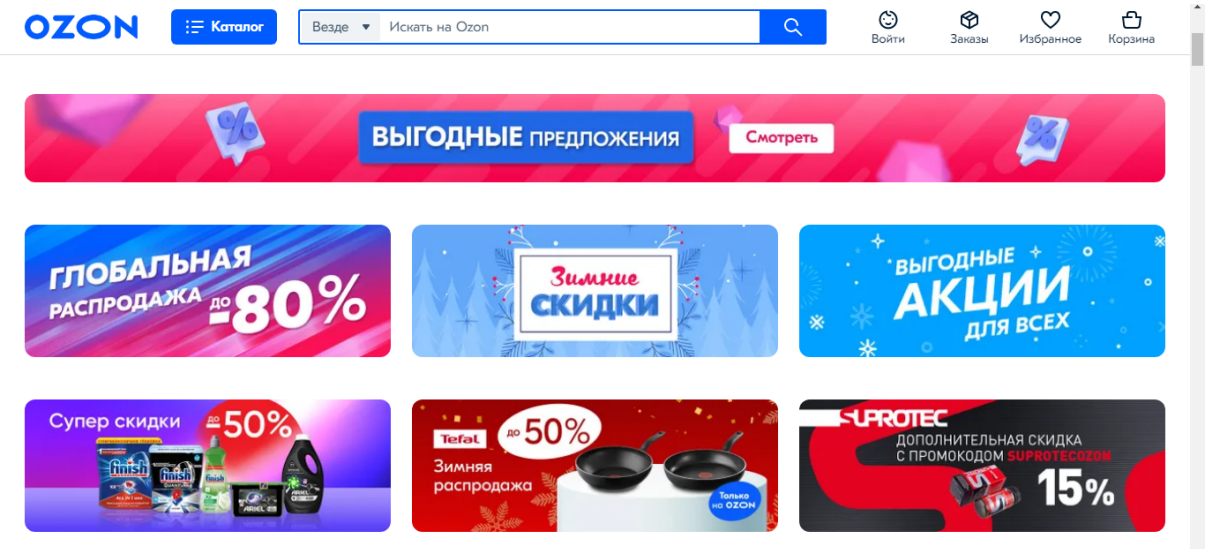


Рисунок 2 – пример сайта, использующего ИИ для создания рекомендаций

Спорт

Тренеры получают отчеты со снимками с камер и показателями датчиков о том, как лучше организовать игру, в том числе как оптимизировать расстановку игроков и стратегию.

Каковы проблемы в сфере применения искусственного интеллекта?

Главное ограничение ИИ заключается в том, что обучение возможно только на основе данных, другими способами - невозможно. Это означает, что любые неточности в данных отразятся на результатах. А новые уровни прогнозирования или анализа необходимо добавлять отдельно.

Современные системы ИИ заточены под выполнение четко определенных задач. Система, предназначенная для игры в покер, не сможет раскладывать пасьянсы или играть в шахматы. Система, настроенная на выявление мошенничества, не сможет водить машину или предоставлять правовую помощь. Более того, система ИИ, предназначенная для выявления мошенничества в сфере здравоохранения, не сможет с той же степенью точности выявлять махинации с налогами или претензиями по гарантиям.

Другими словами, эти системы характеризуются очень узкой специализацией. Они предназначены для выполнения одной конкретной задачи, и им далеко до многозадачности человека.

Кроме того, самообучающиеся системы не являются автономными. Образы технологий ИИ, которые мы видим на экранах телевизоров и кинотеатров, по-прежнему являются элементами фантастики. Тем не менее компьютеры, способные анализировать сложные данные для освоения и совершенствования конкретных навыков, уже не редкость

Принцип работы искусственного интеллекта

Принцип работы ИИ заключается в сочетании большого объема данных с возможностями быстрой, итеративной обработки и интеллектуальными алгоритмами, что позволяет программам автоматически обучаться на базе закономерностей и признаков, содержащихся в данных. ИИ представляет собой комплексную дисциплину со множеством теорий, методик и технологий. Ее главными направлениями являются следующие:

- **Машинное обучение**
- **Нейросеть**
- **Глубокое изучение**
- **Когнитивные вычисления**
- **Компьютерное зрение**
- **Обработка естественного языка**

Машинное обучение — это область знаний, исследующая алгоритмы, которые обучаются на данных с целью найти закономерности. В нем используются методы нейросетей, статистики, исследования операций и т.п. для выявления скрытой полезной информации в данных; при этом явно не программируются инструкции, указывающие, где искать данные и как делать выводы.

Нейросеть — это один из методов машинного обучения. Это математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма.

В глубоком обучении используются сложные нейросети со множеством нейронов и слоев. Для обучения этих глубоких нейросетей, а также для обнаружения сложных закономерностей в огромных массивах данных используются повышенные вычислительные мощности и усовершенствованные методики. Распространенные области применения: распознавание изображений и речи.

Когнитивные вычисления — направление ИИ, задачей которого является обеспечение процесса естественного взаимодействия человека с компьютером, аналогичного взаимодействию между людьми. Конечная цель ИИ и когнитивных вычислений — имитация когнитивных процессов человека компьютером благодаря интерпретации изображений и речи с выдачей соответствующей ответной реакции.

Компьютерное зрение опирается на распознавание шаблонов и на глубокое обучение для распознавания изображений и видео. Машины уже умеют обрабатывать, анализировать и понимать изображения, а также снимать фото или видео и интерпретировать окружающую обстановку.

Обработка естественного языка — это способность компьютеров анализировать, понимать и синтезировать человеческий язык, включая устную речь. Сейчас мы уже можем управлять компьютерами с помощью обычного языка, используемого в повседневном обиходе. Например, используя Siri или Google assistant.

Кроме того, функционирование ИИ обеспечивают следующие технологии:

- Существование ИИ невозможно без **графических процессоров**, так как они предоставляют вычислительные мощности, необходимые для итеративной обработки данных. Для обучения нейросетей необходимы «большие данные» и вычислительные ресурсы.
- **Интернет вещей** собирает колоссальные объемы данных от подключенных устройств. Большая часть этих данных не проанализирована. Автоматизация моделей с помощью ИИ позволит использовать больше таких данных.
- Разрабатываются и по-новому комбинируются более **совершенные алгоритмы**, которые позволяют быстрее анализировать большой объем данных сразу на нескольких уровнях. Такая интеллектуальная обработка — ключ к выявлению и прогнозированию редких событий, пониманию сложных систем и оптимизации уникальных сценариев.

- **API (программные интерфейсы приложений)** представляют собой переносимые пакеты кода, благодаря которым функционал ИИ может быть интегрирован в существующие продукты и пакеты программ. С помощью API можно добавить функцию распознавания изображений в домашнюю систему безопасности или вопросно-ответные функции для описания данных, создания титров и заголовков, обнаружения в данных интересных закономерностей и иной полезной информации.



Рисунок 3 – одни из самых популярных API

Цель ИИ - обеспечение работы программных продуктов, способных к анализу входных данных и интерпретации полученных результатов. Искусственный интеллект — средство, обеспечивающее более интуитивный процесс взаимодействия человека с программами и помощь при принятии решений в рамках определенных задач. ИИ не замена человеку, и в обозримом будущем таковой не станет.

Какой бывает ИИ?

Искусственный интеллект обычно разделяют на две большие категории:

- Слабый ИИ [Weak AI]: этот вид искусственного интеллекта, который иногда называют «узким ИИ» [Narrow AI], работает в ограниченном контексте и является имитацией человеческого интеллекта. Слабый ИИ часто ориентирован на очень хорошее выполнение одной задачи. И, хотя эти машины могут показаться умными, они работают с большими ограничениями.
- Общий искусственный интеллект (AGI): AGI, иногда называемый «сильным ИИ» [Strong AI], – вид искусственного интеллекта, который мы видим в фильмах, например, роботов из «Мира Дикого Запада» или голограмму Джой из «Бегущего по лезвию 2049». AGI – это машина с общим интеллектом, которая, как и человек, может применять его для решения любой проблемы.

Что такое слабый искусственный интеллект?

Искусственный интеллект обычно разделяют на две большие категории:

Слабый ИИ окружает нас повсюду, и на сегодняшний день это самая успешная реализация искусственного интеллекта.

Ориентируясь на выполнение конкретных задач, за последнее десятилетие он совершил множество прорывов, которые принесли «значительные общественные выгоды и внесли вклад в экономическую жизнеспособность нации», согласно отчету «Подготовка к будущему искусственного интеллекта», опубликованному администрацией Обамы в 2016 году.

Вот несколько примеров слабого ИИ:

- поиск Google;
- ПО для распознавания изображений;
- Siri, Alexa и другие голосовые помощники;
- Беспилотные автомобили;
- Рекомендательные системы Netflix и Spotify;
- IBM Watson;

Как работает слабый ИИ?

Большая часть слабого ИИ основана на достижениях в области машинного обучения и глубокого обучения. Схожесть этих понятий может сбивать с толку, однако их следует различать. Венчурный капиталист Фрэнк Чен предложил следующее определение: «Искусственный интеллект — это набор алгоритмов, которые пытаются имитировать человеческий интеллект. Машинное обучение является одним из них, а глубокое обучение — одним из методов машинного обучения».

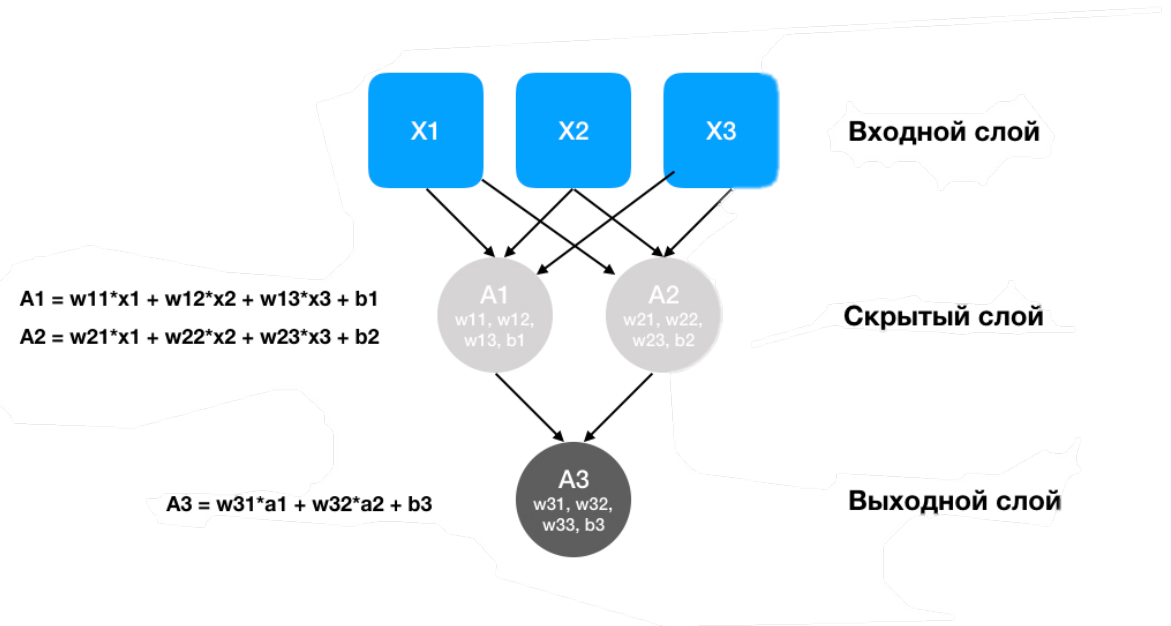


Рисунок 4 – пример машинного обучения

Другими словами, машинное обучение снабжает компьютер данными и использует статистические методы, чтобы помочь ему научиться выполнять задачи, не будучи специально запрограммированным для них, что устраняет необходимость в миллионах строк написанного кода. Популярными видами машинного обучения являются обучение с учителем (с использованием помеченных наборов данных), обучение без учителя (с использованием немаркированных наборов данных), и обучение с подкреплением.

Глубокое обучение — это тип машинного обучения, при котором вводимые данные обрабатываются через архитектуру нейронной сети, основанной на биологических принципах.

Нейронные сети содержат ряд скрытых слоев, через которые обрабатываются данные, что позволяет машине «углубиться» в свое обучение, устанавливать связи и взвешивать ввод для достижения наилучших результатов.

Что такое машинное обучение?

Искусственный интеллект и машинное обучение – не одно и то же. Машинное обучение является лишь одним из подразделов ИИ.

Наиболее распространенными типами машинного обучения — с учителем, без учителя и с подкреплением.



Рисунок 5 – схема разновидностей машинного обучения

Обучение с учителем используют когда у разработчиков имеется набор данных и они знают какие именно признаки должен искать алгоритм.

Как правило, оно делится на две категории: классификация и регрессия.

Классификация применяется когда необходимо отнести объекты в заранее известные классы. Данный тип обучения используется в спам-фильтрах, определении языка или выявлении подозрительных транзакций.

Регрессию используют, когда необходимо соотнести объект с временной линией, например — для прогнозирования стоимости ценных бумаг, спроса на товар или постановки медицинских диагнозов.

Обучение без учителя — менее популярный вид МО из-за его непредсказуемости. Алгоритмы обучаются на неразмеченных данных и им необходимо самостоятельно найти признаки и закономерности. Часто используется для кластеризации, уменьшения размерности и поиска ассоциаций.

Кластеризация — это как классификация, но без известных классов. Алгоритм должен сам найти признаки схожести в объектах и объединить их в кластеры. Используется для анализа и разметки новых данных, сжатия изображений или объединения меток на карте.

Уменьшение размерности — обобщает конкретные признаки в абстракции более высокого уровня. Часто используется для определения тематики текстов или создания рекомендательных систем.

Ассоциации нашли свое применение в маркетинге, например — при составлении акций и распродаж или анализа поведения пользователей на сайте. Также может служить для создания рекомендательной системы.

Обучение с подкреплением — это обучение агента выживать в среде, в которой он существует. Средой может быть все что угодно: от видеоигры до реального мира. 29

Обучение с подкреплением предусматривает награду для агента за правильное действие и наказание за ошибки. Алгоритму не обязательно запоминать весь свой предыдущий опыт и просчитывать все возможные варианты развития событий. Он должен научиться действовать по ситуации.

Например, существуют алгоритмы, которые играют в Супер-Марио не хуже людей, а в реальном мире автопилот в машинах Tesla или робот-пылесос делают все, чтобы объезжать препятствия на своем пути.



Рисунок 6 – ИИ во время прямой трансляции самостоятельно проходит Марио

Что такое нейронные сети и глубокое обучение?

Концепция искусственных нейросетей не нова. Впервые это понятие сформулировали американские ученые Уоррен Мак-Каллок и Уолтер Питтс в 1943 году.

Любая нейронная сеть состоит из нейронов и связей. Нейрон – это функция, имеющая множество входов и один выход. Они обмениваются информацией между собой по каналам связи, каждый из которых имеет определенный вес.

Вес – это параметр, определяющий прочность связи между нейронами. Сам нейрон не разбирается, что он посылает, поэтому вес необходим для того, чтобы регулировать, на какие входы реагировать, а на какие нет.

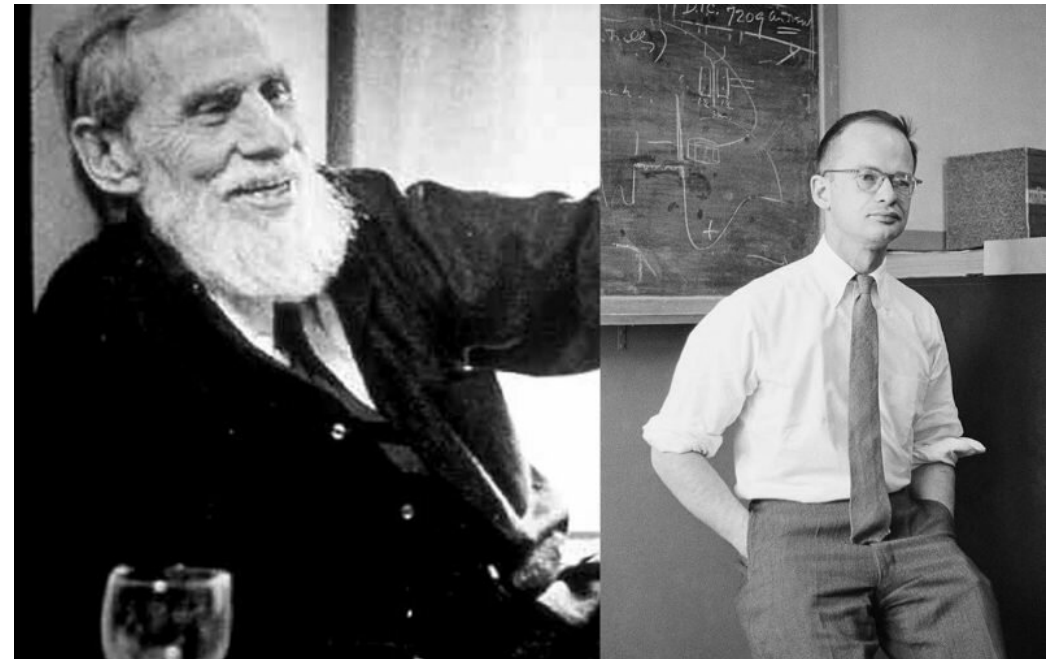


Рисунок 7 – Уоррен Мак-Каллок и Уолтер Питтс

По мере усложнения архитектуры нейросетей, нейроны решили связывать не как угодно, а по слоям. Внутри слоя нейроны никак не взаимодействуют между собой, а получают и передают информацию из предыдущего слоя в следующий.

Как правило, чем больше слоев в нейросети – тем сложнее и точнее модель. Но тогда, 50 лет назад, исследователи уперлись в ограничения вычислительных мощностей. В итоге технология оказалась разочарованием и о ней забыли на долгие годы.

Глубокое обучение отличается от нейронных сетей лишь в методах обучения сетей больших размеров. На практике, как правило, разработчики не выясняют, какую сеть можно считать глубокой, а какую нет. Сегодня даже для построения сетей на пять слоев разработчики пользуются «глубокими» библиотеками, такими как Keras, TensorFlow или PyTorch.

На сегодняшний день самыми популярными сетями являются сверточные нейросети (CNN) и рекуррентные нейросети (RNN).

CNN часто используется для распознавания лиц, поиска объектов на фотографиях и видео, улучшения качества изображений и прочих задач. Рекуррентным сетям нашли применение в машинном переводе текста и синтезе речи. Например, с 2016 года Google Translate работает на основе RNN-архитектуры.

Также популярность нашли генеративно-состязательные сети (GAN). В ее основе лежат две нейросети, одна из которых генерирует данные, например — изображение, а вторая пытается отличить правильные образцы от неправильных. Так как две сети соревнуются между собой, между ними возникает антагонистическая игра.

GAN часто используется для создания фотореалистичных фотографий. Например, репозиторий изображений This Person Does Not Exist состоит из портретных фото «людей», созданных генеративной нейросетью.

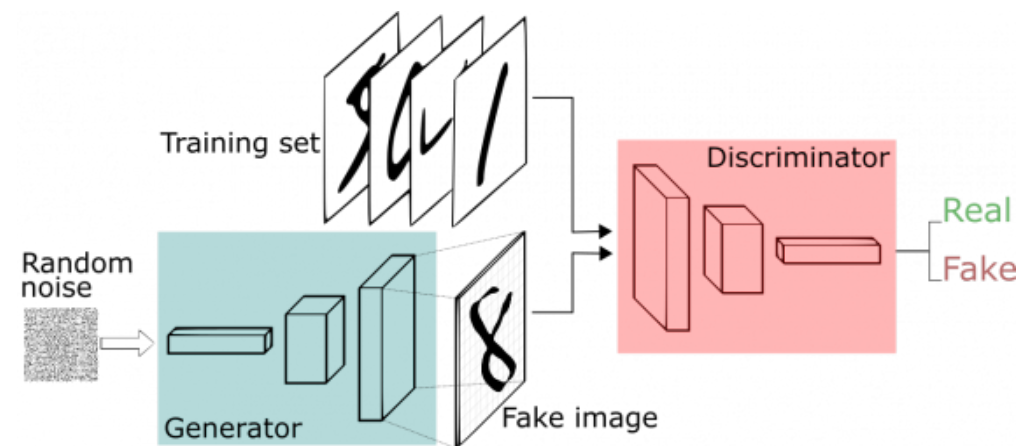


Рисунок 8 – пример обработки изображения ИИ

Что такое общий искусственный интеллект?

Создание машины с интеллектом человеческого уровня, которая может быть применена к любой задаче, является Святым Граалем для многих исследователей ИИ, но поиски AGI сопряжены с некоторыми трудностями.

Общий ИИ уже давно является музой антиутопической научной фантастики, в которой сверхразумные роботы наводняют человечество, но эксперты сходятся во мнении, что это не то, о чем нам нужно беспокоиться в ближайшее время.

Что такое нейронные сети и глубокое обучение?

Американский изобретатель и футуролог Рэй Курцвейл предсказал, что общий ИИ появится уже к 2029 году. Его коллега Родни Брукс не столь оптимистичен, и уверен, что переломный момент развития технологий машинного разума произойдет к 2300 году.

Стюарт Рассел, один из авторов учебника «Искусственный интеллект: современный подход», предполагает, что изобретение AGI станет случайным, как, например, открытие ядерной энергии в 1933 году. Ученый считает, что это яркий пример того, как бессмысленно давать какие-либо прогнозы в развитии столь непредсказуемой технологии, которая до конца еще не изучена.



Рисунок 9 – Рэй Курцвейл

Развитие и потенциал

В настоящий момент ИИ имеет широкий спектр применения, в том числе в России.

При этом в России по мнению аналитиков ожидается удвоение количества случаев использования ИИ в различных компаниях в течении 5 ближайших лет.

Но почвы для разработки технологий на уровне лидеров отрасли в данный момент нет, так как отсутствует должное финансирование.

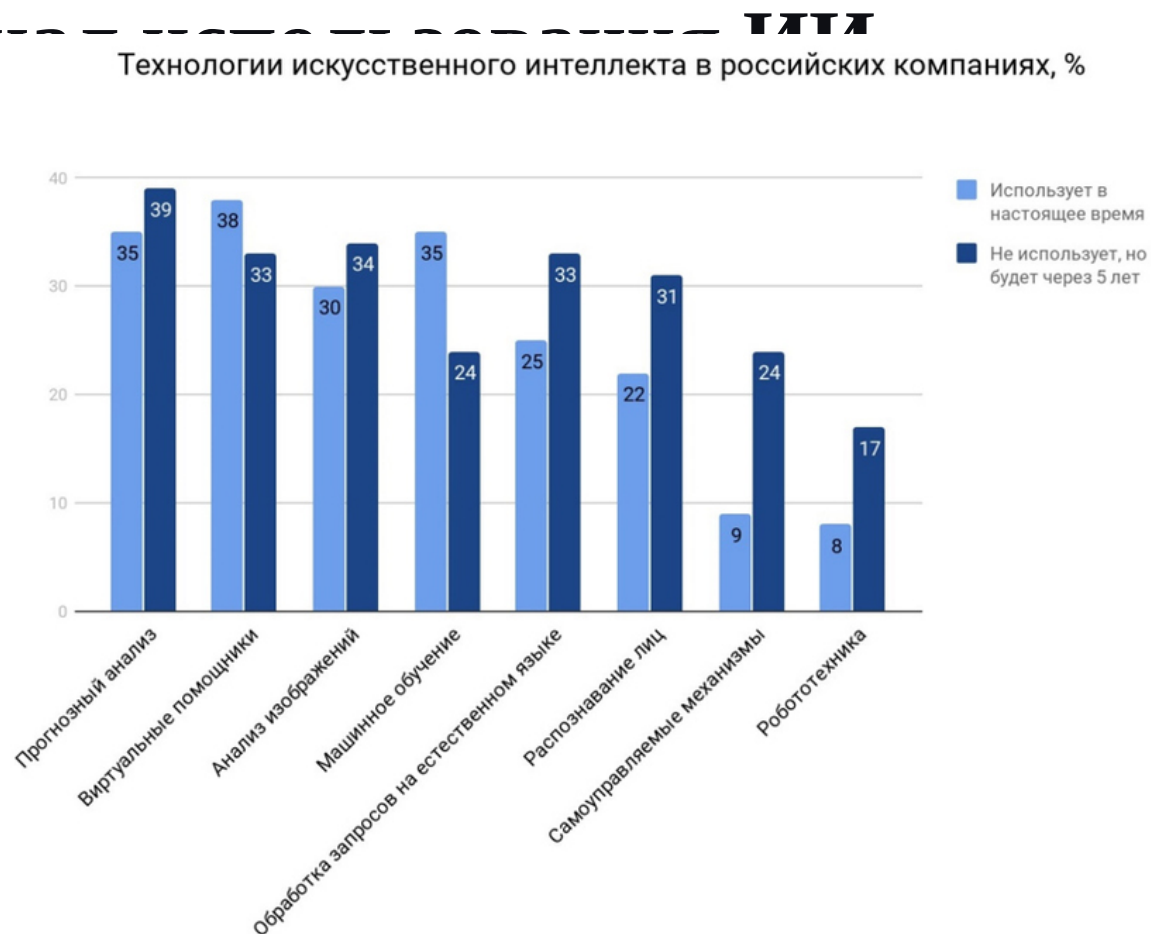


Рисунок 10 – использование ИИ в российских компаниях

Компьютер больше не слеп

С момента первого использования систем глубокого обучения на конкурсе по распознаванию образов в 2012 году, такие системы стали гораздо реже ошибаться.

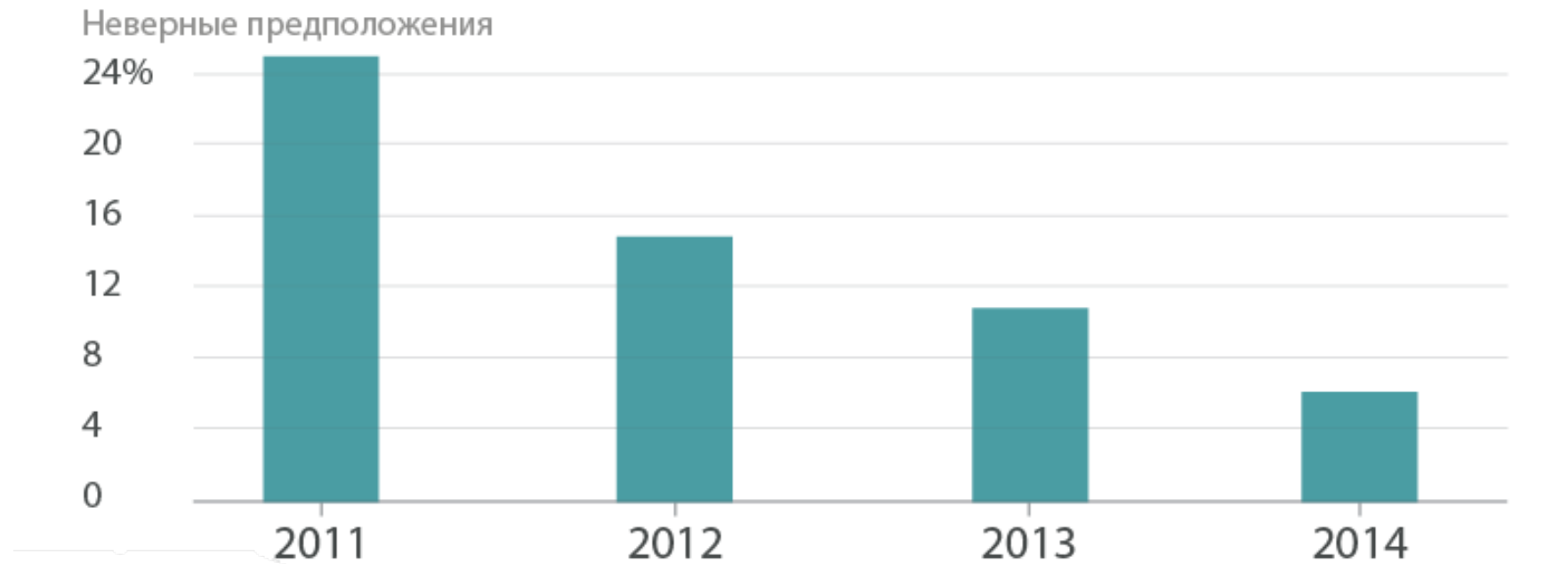


Рисунок 11 – диаграмма точности ИИ

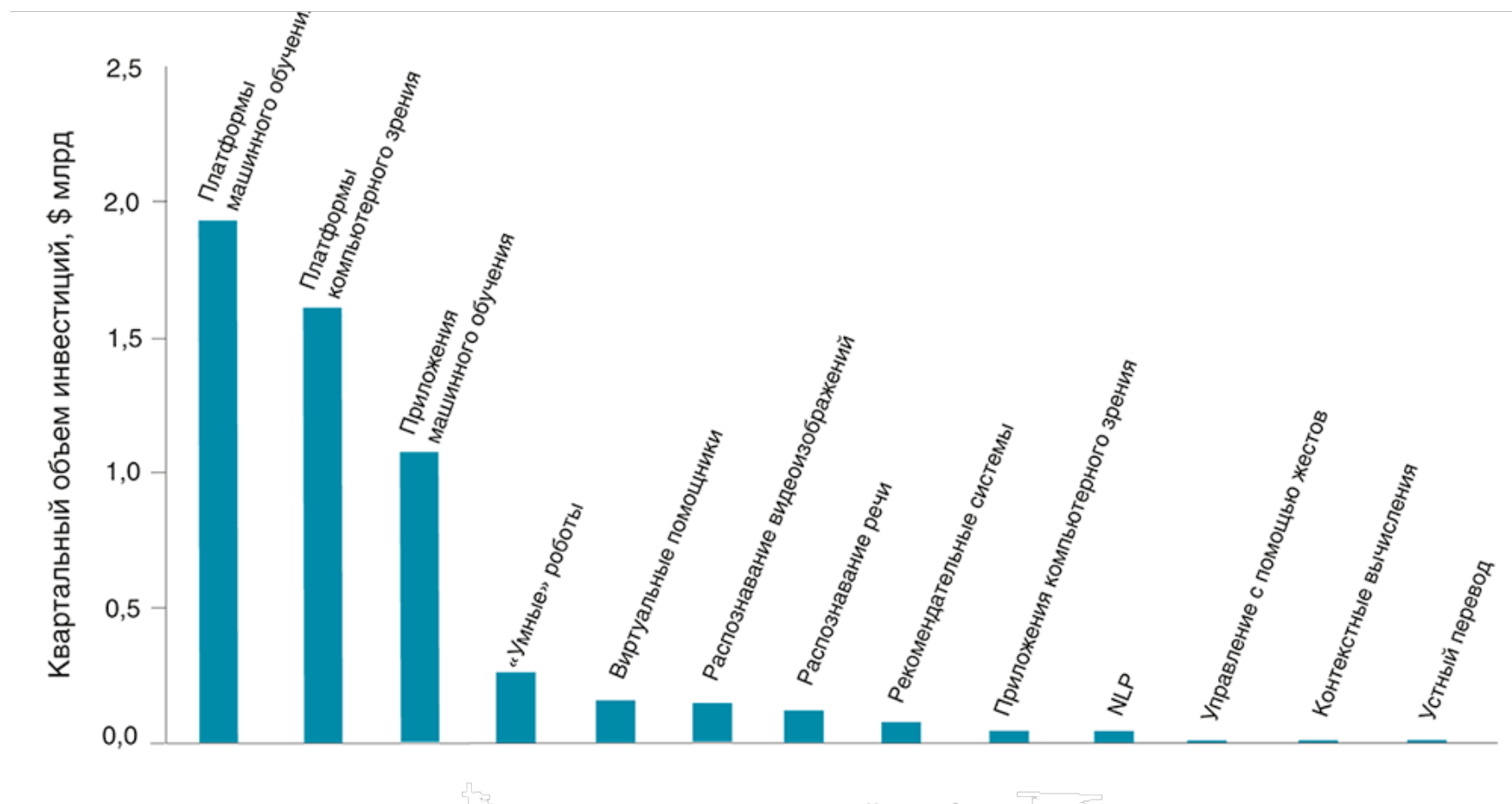


Рисунок 12 – инвестиции в отрасли развития ИИ

Вывод

Очевидно, что ИИ всё больше внедряется в нашу жизнь, и в будущем этот процесс будет только ускоряться. Уже получены разрешения на производство автопилота 3 уровня. А автопилоты 2 уровня действуют на дорогах общего пользования. Мы часто даже не подозреваем, где можем столкнуться со столь прогрессивными технологиями.

До сих пор происходит нормирование того, каким должен и может быть ИИ в плане безопасности, скорости действия, его возможностей и близости к человеческому мышлению.

Экспертные системы

Экспертная система

Экспертная система— это информационная система, назначение которой частично или полностью заменить эксперта в той или иной предметной области.

Подобные интеллектуальные системы эффективно применяются в таких областях, как логистика, управление воздушными полетами, управление театром военных действий. Основной направленной деятельностью предсказание, прогнозирование в рамках определенного аспекта в предметной области.

Архитектура экспертных систем

- **Архитектура экспертных систем**, как правило, состоит из нескольких компонентов, присутствие каждого из которых обеспечивает работу системы в целом. Эти компоненты важны не столько по отдельности, сколько в слаженной взаимосвязанной работе, т.к. все они играют важную роль в решении задач, для которых и предназначена экспертная система.

Архитектура экспертной системы



Рис.1. Архитектура экспертной системы

Область применения

1. Медицина;
2. Финансы;
3. Нефтяная и газовая промышленность;
4. Энергетика;
5. Фармацевтическая промышленность;
6. Metallургия и машиностроение;
7. Транспорт.

Основные типы задач, решаемых с помощью экспертных систем

1. **Интерпретация** — это процедура анализа данных с целью определения их смысла.
2. **Планирование**, или процедура составления планов. Планирующие системы предназначены для достижения конкретных целей при решении задач с большим числом переменных.
3. **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.

Основные типы задач, решаемых с помощью экспертных систем

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.

Диагностика — это процесс поиска неисправностей в системе или определение стадии заболевания в медицине, основанный на интерпретации данных (возможно, зашумленных).

Основные типы задач, решаемых с помощью экспертных систем

7. **Обучение** рассматривается в ЭС в двух аспектах : обучение пользователя и самообучение системы, причем как на этапе приобретения знаний, так и в процессе работы ИП.
8. **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.

История ЭС

В 1965 году в Стэнфордском университете Эдвард Фейгенбаум, Джошуа Ледерберг начали работы по созданию первой экспертной системы **DENDRAL**.

Задача — создать компьютерного помощника, который мог бы определять путем расчета молекулярную структуру химических соединений

Проблемы :

- 1) построение гибкой программы, оперирующей с многочисленными знаниями и работающей по правилам логики ("если - то");
- 2) создание базы данных, включающую знания многих специалистов в органической химии;
- 3) отделение механизма логического вывода от базы знаний.

История ЭС

В 1970-е ЭС стали ведущим направлением в области искусственного интеллекта. создано множество разнообразных экспертных и диагностических систем, большая часть которых действует и сегодня. Самыми известными из них являются **MYCIN**, служащая для диагностики и лечения инфекционных заболеваний, и **PROSPECTOR**, предназначенная для геологической разведки месторождений полезных ископаемых.

Примеры известных прикладных ЭС

- **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;
- **XCO N** — ЭС конфигурирования оборудования системы VAX;
- **PALLADIO** — ЭС проектирования и тестирования СБИС;
- **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 ;

Преимущества ЭС

Постоянство и стабильность;

Простота передачи информации;

Устойчивость получаемых результатов;

Низкая стоимость разработки и эксплуатации ЭС.

Недостатки ЭС

- 1. Предназначены для узкой предметной области;**
- 2. Качество работы зависит от качества БЗ;**
- 3. Не способны к самообучению;**
- 4. Необходимость обновления программных средств.**

Роли создателей ЭС

Эксперт определяет знания (данные и правила), характеризующие проблемную область, обеспечивает полноту и правильность введенных в ЭС знаний.

Инженер по знаниям помогает эксперту выявить и структурировать знания, необходимые для работы ЭС; осуществляет выбор того ИС, которое наиболее подходит для данной проблемной области, и определяет способ представления знаний в этом ИС; выделяет и программирует стандартные функции

Программист разрабатывает ИС (если ИС разрабатывается заново), содержащее в пределе все основные компоненты ЭС, и осуществляет его сопряжение с той средой, в которой оно будет использовано.

Структура ЭС

База знаний

Знания — это правила, законы, закономерности получены в результате профессиональной деятельности в пределах предметной области.

База знаний — база данных содержащая правила вывода и информацию о человеческом опыте и знаниях в некоторой предметной области. Другими словами, это набор таких закономерностей, которые устанавливают связи между вводимой и выводимой информацией.

2. Данные

Данные — это совокупность фактов и идей представленных в формализованном виде.

Собственно на данных основываются закономерности для предсказания, прогнозирования. Продвинутые интеллектуальные системы способные учиться на основе этих данных, добавляя новые знания в базу знаний.

Структура ЭС

Модель представления знаний

Модель представления знаний — это способ задания знаний для хранения, удобного доступа и взаимодействия с ними, который подходит под задачу интеллектуальной системы.

Механизм логического вывода

Механизм логического вывода данных выполняет анализ и проделявает работу по получению новых знаний исходя из сопоставления исходных данных из базы данных и правил из базы знаний. Механизм логического вывода в структуре интеллектуальной системы занимает наиболее важное место.

Структура ЭС

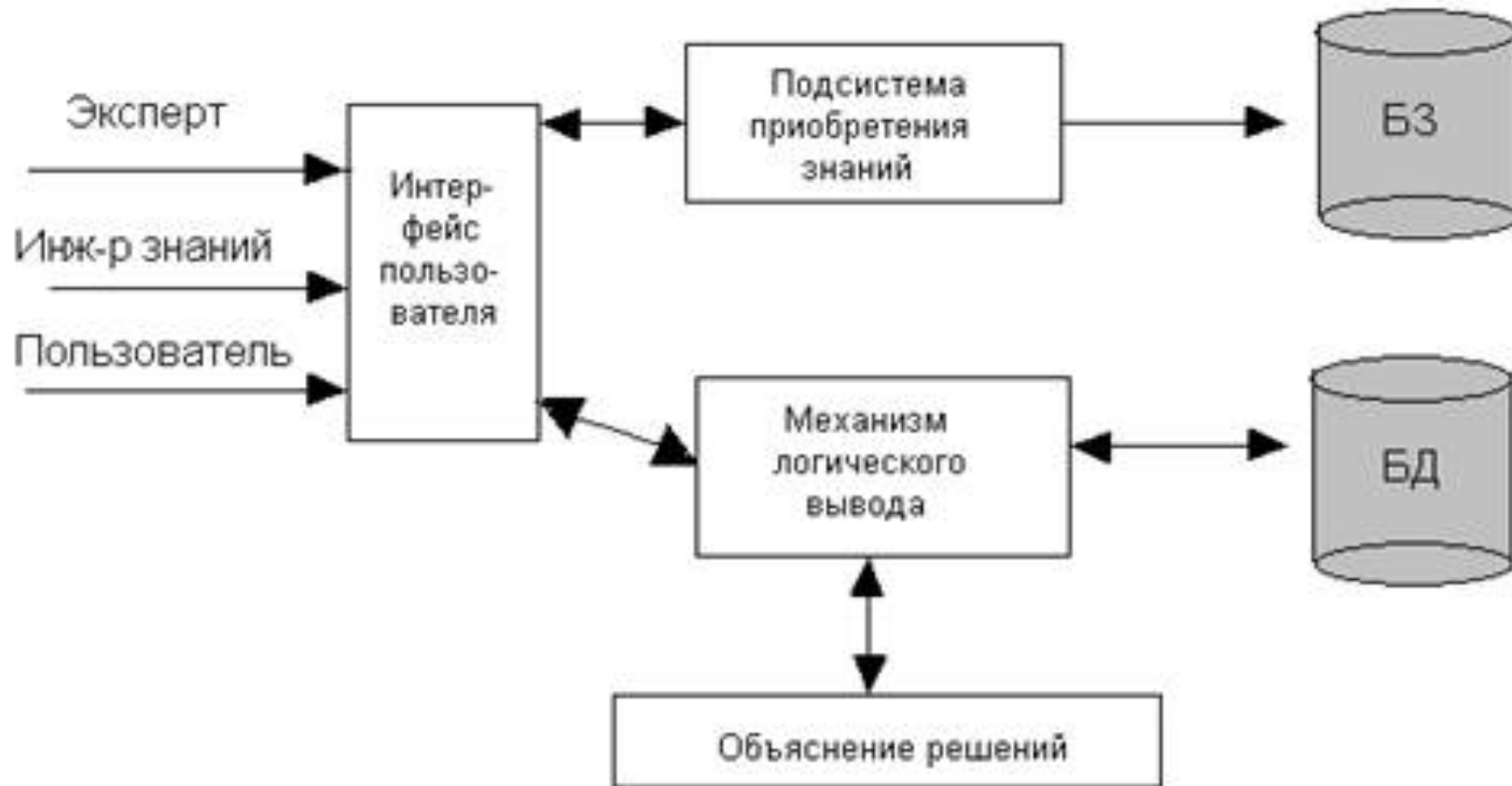


Рис.2. Структура ЭС

Модели представления знаний

Распространены четыре основных МПЗ:

Продукционная МПЗ

Семантическая сеть МПЗ

Фреймовая МПЗ

Формально логическая МПЗ

Продукционная модель

Продукционная модель – это модель, основанная на правилах, позволяющая представить знание в виде предложений типа:

“Если условие, ТО действие”.

Продукционная модель обладает тем недостатком, что при накоплении достаточно большого числа (порядка нескольких сотен) продукций они начинают противоречить друг другу.

Семантическая сеть

Однозначное определение семантической сети в настоящее время отсутствует. В инженерии знаний под ней подразумевается **граф**, отображающий смысл целостного образа. Узлы графа соответствуют понятиям и объектам, а дуги – отношениям между объектами.

Формально сеть можно задать в следующем виде:

$$H = \langle I, C, J \rangle$$

I - множество информационных единиц;

C - множество типов связей между информационными единицами;

G - отображение, задающее конкретные отношения из имеющихся типов **C** между элементами **I**.

Фреймовая модель

Фреймовая модель представляет собой систематизированную психологическую модель памяти человека и его сознания. Информация, относящаяся к фрейму, содержится в составляющих его слотах.

Фреймы образуют иерархию. Иерархия во фреймовых моделях порождает единую многоуровневую структуру.

Формально фрейм – это тип данных вида:

$$F = \langle N, S1, S2, S3 \rangle$$

N - имя объекта;

S1 - множество слотов, содержащих факты, определяющие декларативную семантику фрейма;

S2 - множество слотов, обеспечивающих связи с другими фреймами (каузальные, семантические и т. д.);

S3 - множество слотов, обеспечивающих преобразования, определяющие процедурную семантику фрейма.

Логическая модель

Основная идея при построении логических моделей знаний – вся информация, необходимая для решения прикладных задач, рассматривается как совокупность фактов и утверждений, которые представляются как формулы в некоторой логике.

В основе логических моделей знаний лежит понятие формальной теории, задаваемое **кортежем**:

$$S = \langle A, F, Ax, R \rangle$$

A - счетное множество базовых символов (алфавит);

F - множество, называемое формулами;

Ax - выделенное подмножество априори истинных формул (аксиом);

R - конечное множество отношений между формулами, называемое правилами вывода.

Реализация

Реализация ЭС предполагает выбор инструментальных средств.

Для реализации ЭС используются как традиционные языки программирования, язык и обработки списков , логического программирования, процедурные языки, так и программные оболочки (среды) для создания экспертных систем, экспертные системы и системы приобретения знаний.

Применяются следующие технологии:

- **объектно-ориентированный подход** к разработке экспертных систем;
- **системы с распределенной архитектурой** (клиент-сервер);
- **технология реинжиниринга** или моделирования процессов.

Вид экспертных систем



Рис.3.Вид экспертных систем

Статические системы

По степени соответствия реальному масштабу времени различают **статические** и **динамические** экспертные системы.

Статические системы используют постоянную информацию из базы данных и неизменные правила в базе знаний. Данные и знания не могут быть добавлены в процессе работы ИП. ЭС данного типа разрабатываются, как правило, для приложений, в которых нет необходимости учитывать влияние окружающей среды.

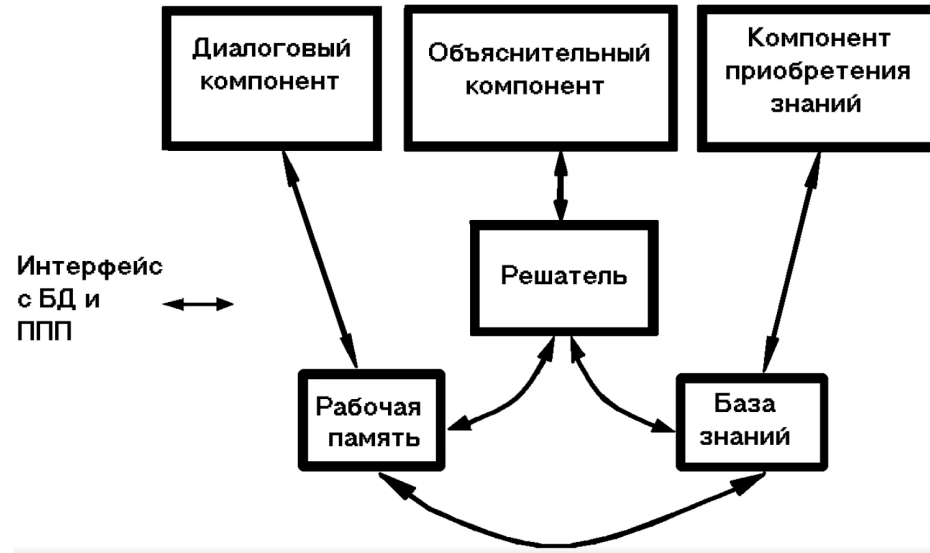


Рис.4.Статистические системы.

Динамические системы

Динамические системы, напротив, ориентированы на работу с постоянно меняющейся информацией в БД и в БЗ системы или ИП.

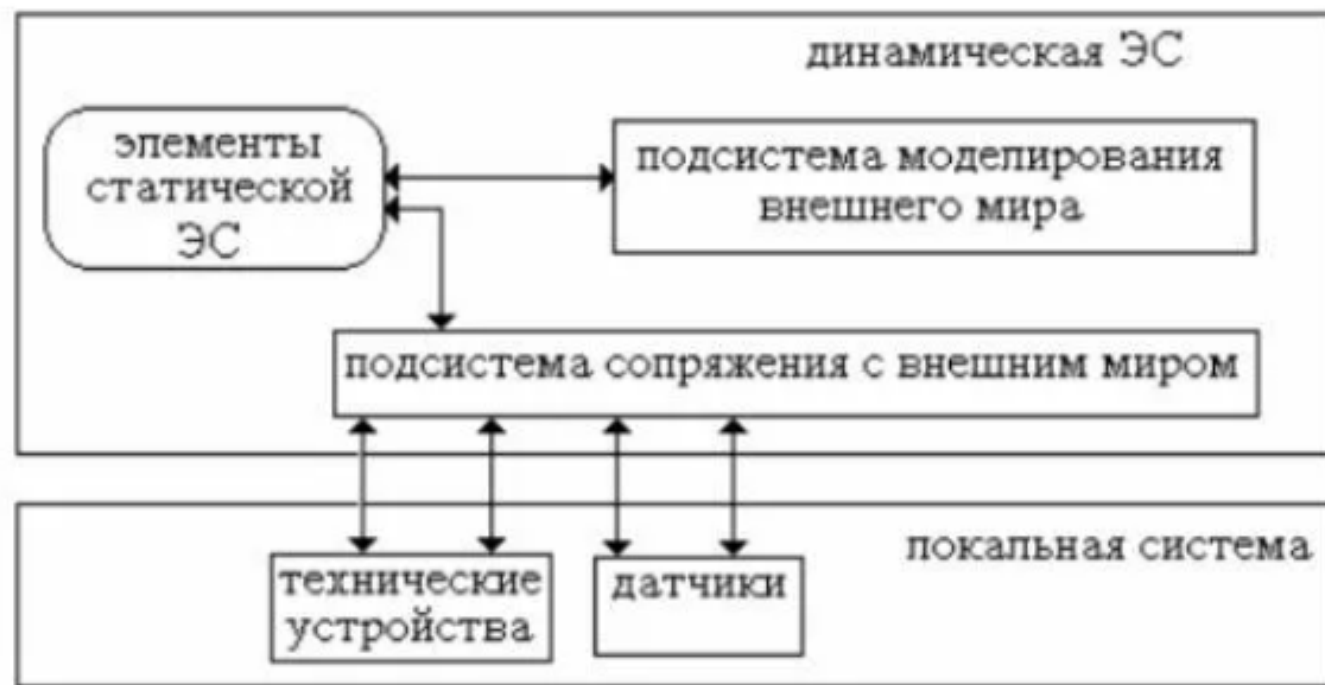


Рис.5.Динамические системы

Современные ЭС

Современные ЭС обычно сочетают в себе оба указанных класса, реализованных в виде соответствующих режимов работы. Например, в зависимости от вида работы, архитектура экспертной системы будет просто включать различные функциональные элементы.

Преимущества ЭС

Существует ряд преимуществ экспертных систем как перед человеком-оператором, так и перед обычными алгоритмическими базами данных:

- интегрируемость. Существуют инструментальные средства, легко входящие в состав других информационных технологий и средств
- открытость и переносимость: у них нет предубеждений и они устойчивы к различным помехам;
- отсутствие поспешных выводов;
- выдача оптимального решения
- неограниченные размеры базы знаний.
- постоянное хранение данных: эксперт может что-то забыть, машина — никогда.

Этапы разработки ЭС

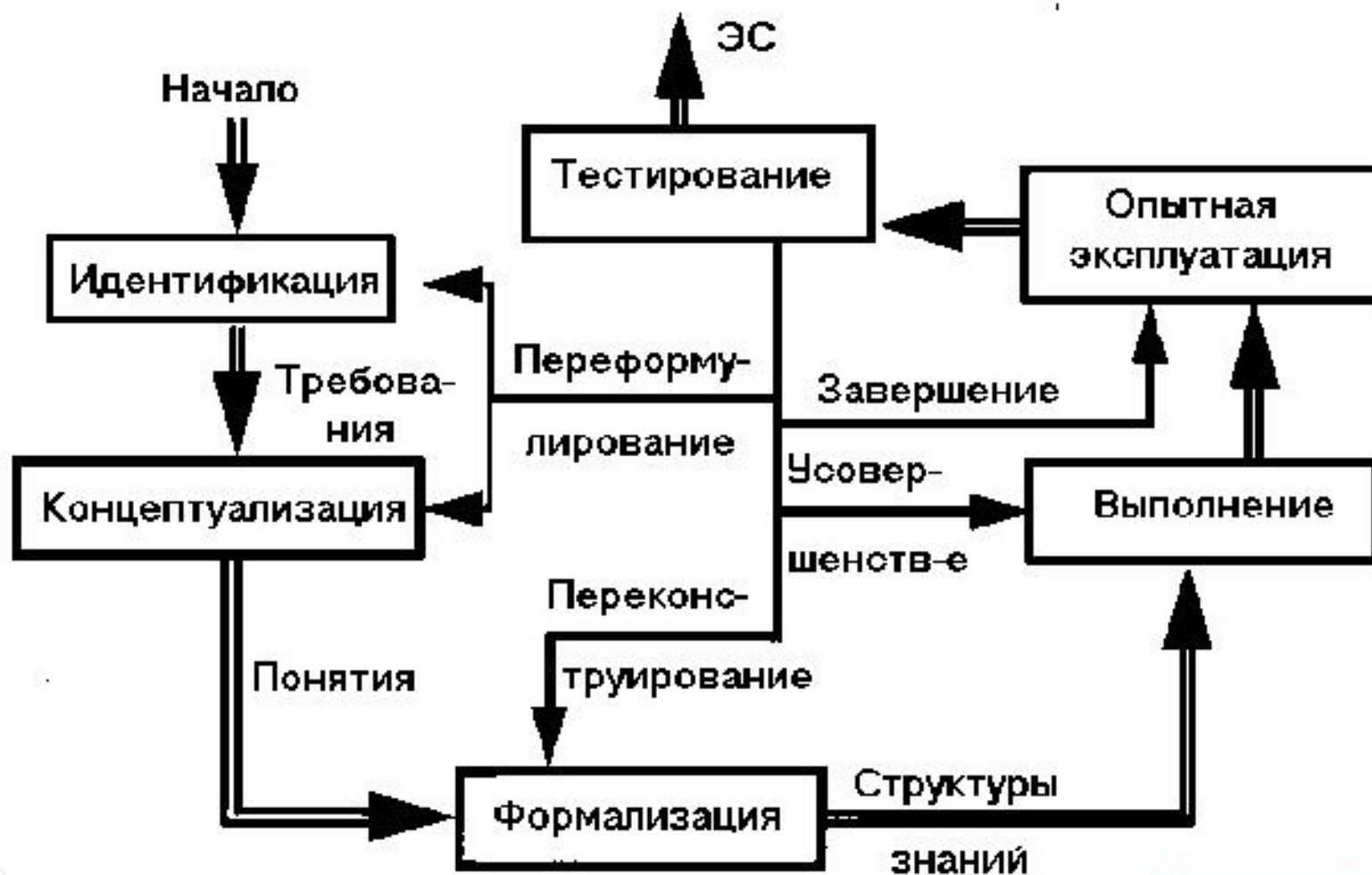


Рис.6.Этапы разработки ЭС

Этапы разработки ЭС

- **Этап идентификации проблем** — определяются задачи, которые подлежат решению, выявляются цели разработки, определяются эксперты и типы пользователей.
- **Этап извлечения знаний** — проводится содержательный анализ проблемной области, выявляются используемые понятия и их взаимосвязи, определяются методы решения задач.
- **Этап структурирования знаний** — выбираются ИС и определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, моделируется работа системы, оценивается адекватность целям системы зафиксированных понятий, методов решений, средств представления и манипулирования знаниями.

Этапы разработки ЭС

- **Этап формализации** — осуществляется наполнение экспертом базы знаний. В связи с тем, что основой ЭС являются знания, данный этап является наиболее важным и наиболее трудоёмким этапом разработки ЭС. Процесс приобретения знаний осуществляется инженером по знаниям на основе анализа деятельности эксперта по решению реальных задач.
- **Реализация ЭС** — создаётся один или несколько прототипов ЭС, решающие требуемые задачи.
- **Этап тестирования** — производится оценка выбранного способа представления знаний в ЭС в целом.

Механизм принятия решения

Механизм принятия решения на основе логического вывода в экспертных системах называется интерпретатором.



Рис. 7. Механизм принятия решения

Иерархия классов

Данные, описывающие текущую проблемную информацию, и правила, хранящиеся в базе знаний, совмещаются в интерпретаторе. Затем на основе логического вывода и с учетом выбранной стратегии рассуждений осуществляется принятие решения. Организация вывода или принятия решения выполняется в четыре этапа: выборка, сопоставление, разрешение конфликтов и выполнение.



Рис. 8. Механизм вывода

Классификации оболочек ЭС

Статические оболочки для решения задач с данными, неизменными в процессе решения.

Статические оболочки для решения задач анализа и синтеза в режиме с разделением времени.

Оболочки для проектирования динамических систем.

Оболочки для разработки динамических систем (решения задач анализа и синтеза в реальном масштабе времени).

Уровни реализации экспертных систем

Экспериментальный прототип. Система ориентирована на правила общего вида, релевантные решаемой задаче.

Промышленный образец. Среднее количество используемых правил в таких системах также не более 50, но находится в диапазоне от 20 до 50.

Промышленная система. Стабильно работающая система , содержащая не менее 100 правил общего и частного вида.

Коммерческая система. Стабильно работающая система , используемая в конкретной области человеческой деятельности и предназначенная для продажи.

Трудности при разработке экспертных систем

Трудности при разработке экспертных систем

Проблема извлечения знаний экспертов;

Проблема формализации знаний экспертов;

Проблема нехватки времени у эксперта;

Недостаток ресурсов;

Неадекватность инструментальных средств решаемой задаче.

Особенность ЭС

Экспертные системы и системы искусственного интеллекта отличаются от систем обработки данных тем, что в них в основном используются символьный (а не числовой) способ представления, символьный вывод и эвристический поиск решения (а не исполнение известного алгоритма).

Решения экспертных систем обладают "прозрачностью", т.е. могут быть объяснены пользователю на качественном уровне.

Это качество экспертных систем обеспечивается их способностью рассуждать о своих знаниях и умозаключениях.

Экспертные системы способны пополнять свои знания в ходе взаимодействия с экспертом.

Характерные черты ЭС

- Наличие четко ограниченной предметной области;
- Возможность принятия решений в условиях неопределенности;
- Способность объяснять мотивы и результат принимаемого решения;
- Четкое разделение фактов и выводов;
- Возможность постоянного пополнения базы знаний и наращивания системы, а также её обучения в процессе эксплуатации;
- Вывод результата в форме рекомендаций по перечню конкретных действий;
- Ориентация на решение трудно формализуемых и неформализуемых задач принятия решений;
- Отсутствие заранее заданного алгоритма принятия решений.

Особенности работы ЭС

ЭС 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?".

Требования к задачам решаемым ЭС

Задача может быть естественным образом решена посредством манипуляции с символами.

Задача должна иметь эвристическую, а не алгоритмическую природу, т.е. ее решение должно требовать применения эвристических правил.

Задача должна быть достаточно сложна, чтобы оправдать затраты на разработку ЭС. Однако она не должна быть чрезмерно сложной (решение занимает у эксперта часы, а не недели), чтобы ЭС могла ее решать;

Задача должна быть достаточно узкой, чтобы решаться методами ЭС, и практически значимой.

Необходимые условия создания ЭС

Существуют эксперты в данной области, которые решают задачу значительно лучше, чем начинающие специалисты;

Эксперты сходятся в оценке предлагаемого решения, иначе нельзя будет оценить качество разработанной ЭС;

Эксперты способны вербализовать (выразить на естественном языке) и объяснить используемые ими методы.

Решение задачи требует только рассуждений, а не действий;

Задача не должна быть слишком трудной.

Необходимые условия создания ЭС

Задача хотя и не должна быть выражена в формальном виде, но все же должна относиться к достаточно "понятной" и структурированной области, Решение задачи не должно в значительной степени использовать "здоровый смысл" (т.е. широкий спектр общих сведений о мире и о способе его функционирования, которые знает и умеет использовать любой нормальный человек).

Контрольные вопросы

Дайте определение «экспертная система».

Назовите области применения экспертных систем.

В каком веке были открыты экспертные системы?

Как называется первая экспертная система?

Назовите достоинства и недостатки ЭС.

Назовите 4 модели представления знаний.

Назовите преимущества ЭС над человеком

Назовите 2 ЭС по степени соответствия реальному масштабу времени.

Перечислите этапы разработки ЭС.

Состав и взаимодействие участников построения и эксплуатации экспертных систем

Применение оболочки для создания экспертных систем CLIPS 6.31

Содержание

• Экспертные системы.....	3
• Структура ЭС.....	4
• CLIPS.....	6
• Факты.....	7
• Правила.....	9
• Версия CLIPS.....	11
• Машина логического вывода.....	12
• Язык программирования CLIPS.....	13

Экспертная система

- Экспертная система (далее по тексту — ЭС) — это информационная система, назначение которой частично или полностью заменить эксперта в той или иной предметной области. Подобные интеллектуальные системы эффективно применяются в таких областях, как логистика, управление воздушными полетами, управление театром военных действий. Основную направленную деятельностью предсказание, прогнозирование в рамках определенного аспекта в предметной области.

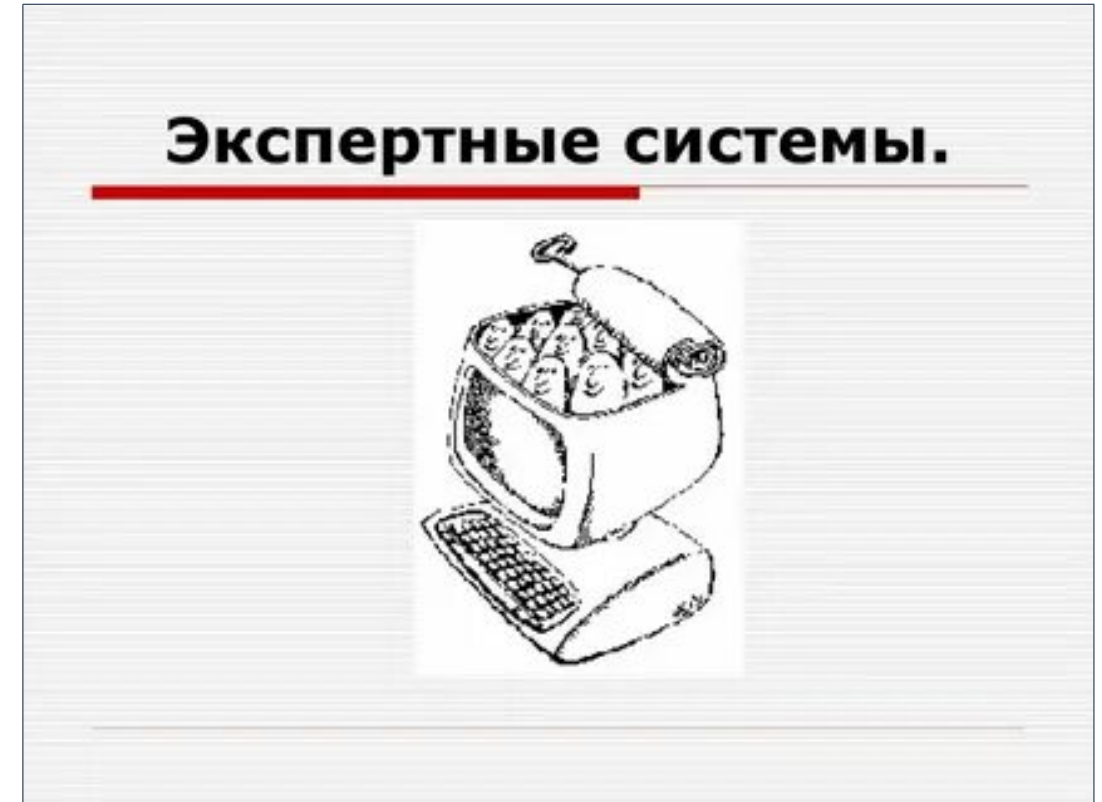


Рис. 1

Структура ЭС

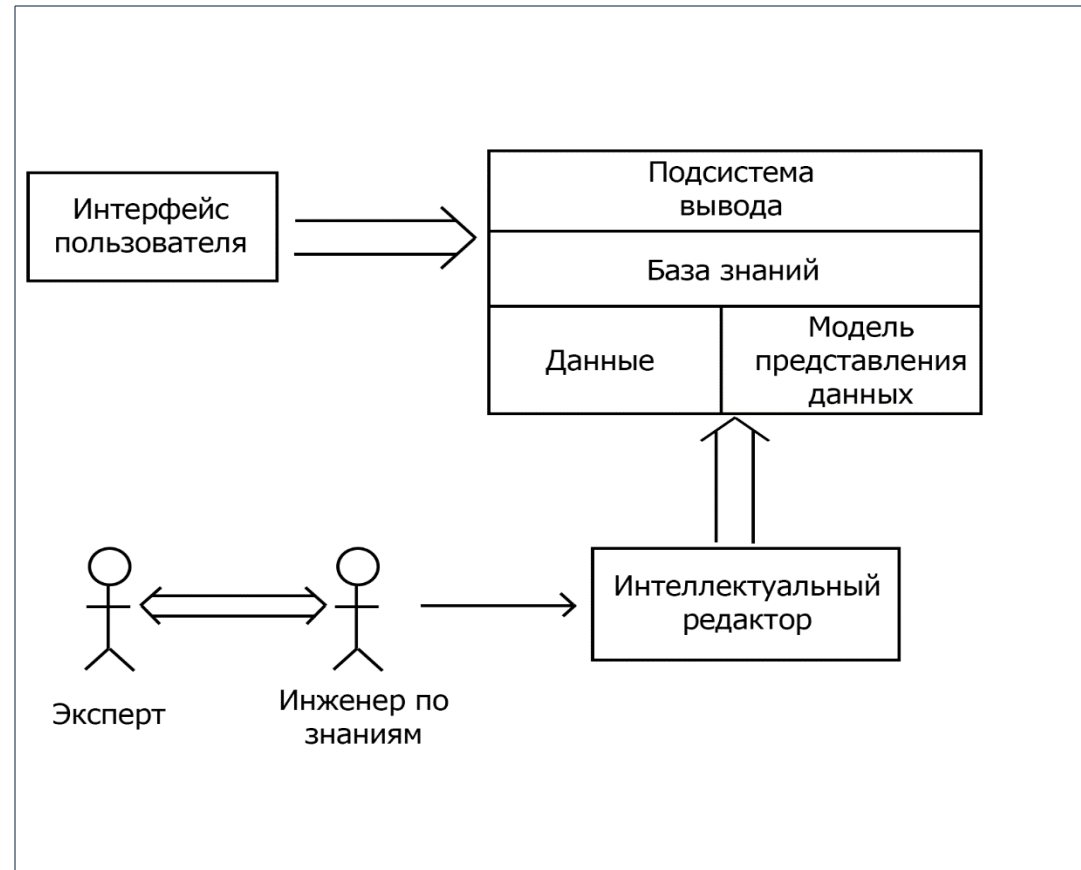


Рис. 2

Структура ЭС

1. **База знаний** **Знания** — это правила, законы, закономерности получены в результате профессиональной деятельности в пределах предметной области. База знаний — база данных содержащая правила вывода и информацию о человеческом опыте и знаниях в некоторой предметной области. Другими словами, это набор таких закономерностей, которые устанавливают связи между вводимой и выводимой информацией.

2. **Данные** — это совокупность фактов и идей представленных в формализованном виде. Собственно на данных основываются закономерности для предсказания, прогнозирования. Продвинутое интеллектуальные системы способные учиться на основе этих данных, добавляя новые знания в базу знаний

3. **Модель представления данных** самая интересная часть экспертной системы. Модель представления знаний (далее по тексту — МПЗ) — это способ задания знаний для хранения, удобного доступа и взаимодействия с ними, который подходит под задачу интеллектуальной системы

CLIPS 6.31

CLIPS, (от англ. C Language Integrated Production System) — программная среда для разработки экспертных систем. Синтаксис и название предложены Чарльзом Форги (Charles Forgy) в OPS (Official Production System). Первые версии CLIPS разрабатывались с 1984 года в Космическом центре Джонсона NASA, как альтернатива существовавшей тогда системе ART*Inference, пока в начале 1990-х не было приостановлено финансирование, и NASA вынудили купить коммерческие продукты. CLIPS является продукционной системой. Реализация вывода использует алгоритм Rete.

Основная идея состоит в представлении знаний в виде такой формы:

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

Рис.3

Факты

Как и другие экспертные системы, CLIPS имеет дело с правилами и фактами.

Факты.

Информация, на основании которой экспертная система делает логический вывод, называется **фактами**. В CLIPS есть 2 вида фактов: **упорядоченные** и **шаблонные**. Шаблонные факты имеют шаблон, задаваемый конструкцией **deftemplate**. Упорядоченные не имеют явной конструкции **deftemplate**, однако она подразумевается. Шаблонный факт напоминает структуру в языке C или запись в языке Pascal, поля называются слотами и объявляются конструкцией **slot**. Например, следующий шаблон объявляет шаблон с именем *cars* и полями: *model*, *color* и *number*.

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

Рис.4

ФАКТЫ

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

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

Рис.5

добавит в рабочую память **упорядоченный** факт *cars*.

Следующая команда поместит шаблонный факт с тремя атрибутами

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

Рис.6

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

Правила

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

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

Рис.7

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

Правила объявляются с помощью команды **defrule**. Пример правила:

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

Рис.8

Правила

- Правила объявляются с помощью команды **defrule**. Пример правила:

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

Рис.9

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

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

Чтобы активированные правила выполнились нужно выполнить команду (run).

Версии CLIPS

- Потомками CLIPS являются следующие языки и среды:
- Jess — коммерческий последователь CLIPS, доступен бесплатно для академического использования, переписан на Java, позже развился в других направлениях. Поддерживает обратный вывод.
- HaleyRules Eclipse — язык правил системы HaleyRules компании Haley Systems. Поддерживает обратный вывод. Проект закрыт после поглощения Haley Systems корпорацией Oracle в декабре 2009 в пользу Oracle Policy Automation.
- FuzzyCLIPS — CLIPS с добавлением концепции значимости (*relevancy*) в язык. Не поддерживает обратного вывода.
- EHSIS — испанский CLIPS, разработан в группе Erabaki Факультета Информатики Университета Страны Басков. Поддерживает нечеткую логику как FuzzyCLIPS.
- CLIPS/R2 — новая версия от компании Production Systems Technologies, обратно совместимая со стандартным CLIPS версий 6.x, но с добавлением алгоритма Rete II, поддержкой обратного вывода и прозрачной поддержкой структур языка Си и классов C++.
- Последняя версия — CLIPS 6.3, выпущена 17 марта 2015.

Машина логического вывода

Процессом помещения правил в рабочий список и их выполнением управляет машина логического вывода(МЛВ). МЛВ реагирует на определенные события:

Событие	Действие
ПОМЕЩЕНИЕ ФАКТОВ В РАБОЧУЮ ПАМЯТЬ	1) Сопоставление фактов с правилами из базы знаний 2) Сопоставление фактов с правилами из рабочего списка правил
УДАЛЕНИЕ ФАКТОВ ИЗ РАБОЧЕЙ ПАМЯТИ	1) Сопоставление фактов с правилами из базы знаний 2) Сопоставление фактов с правилами из рабочего списка правил
ПРИ СОПОСТАВЛЕНИИ НАЙДЕНЫ ПРАВИЛА СООТВЕТСТВУЮЩИЕ ФАКТАМ ИЗ РАБОЧЕЙ ПАМЯТИ	Помещение найденных правил в рабочий список правил
В РАБОЧИЙ СПИСОК ПРАВИЛ ДОБАВЛЕНЫ НОВЫЕ ПРАВИЛА	рабочий список правил сортируется согласно выбранной стратегии разрешения конфликтов
ПРИ СОПОСТАВЛЕНИИ ФАКТОВ С РАБОЧИМ СПИСОКОМ ПРАВИЛ ОБНАРУЖЕНЫ НЕАКТУАЛЬНЫЕ ПРАВИЛА	Неактуальные правила (условия не удовлетворяют фактам) удаляются из рабочего списка
ВЫПОЛНЯЕТСЯ КОМАНДА (RUN)	Выполняются действия (правая часть) первого в рабочем списке правила.
РАБОЧИЙ СПИСОК ПРАВИЛ СТАЛ ПУСТ	Останавливается выполнение правил из рабочего списка

Рис.10

Язык программирования CLIPS

- Язык программирования CLIPS•CLIPS –среда (система) разработки ЭС.
- Определяет язык программирования, описанный в Clips Basic Programming Guide.
- Поддерживает программирование на правилах (с использованием встроенного алгоритма вывода –RETE), процедурное, ООП.
- Преемник OPS, начало разработки –1984.
- Актуальная версия CLIPS 6.3, выпущена 17 марта 2015.•Не поддерживает обратный вывод (в оригинале).

CLIPS стратегия решение конфликтов

- Свойство salience
- Свойство auto-focus
- Стратегия разращение проблемы
- Стратегия глубины
- Стратегия ширины
- LEX стратегия
- МЕА стратегия
- Стратегия упрощения
- Стратегия усложнения



Рис 11 CLIPS

CLIPS стратегия решение конфликтов

- **Стратегия salience**

Свойство правила salience позволяет пользователю назначать приоритет для своих правил. Объявляемый приоритет должен быть выражением, имеющим целочисленное значение из диапазона от —10 000 до +10 000. Выражение, представляющее приоритет правила, может использовать глобальные переменные и функции. Однако старайтесь не указывать в этом выражении функций, имеющих побочное действие. В случае если приоритет правила явно не задан, ему присваивается значение по умолчанию — 0.

Значение приоритета может быть вычислено в одном из трех случаев: при добавлении нового правила, при активации правила и на каждом шаге основного цикла выполнения правил. Два последних варианта называются динамическим приоритетом (dynamic salience). По умолчанию значение приоритета вычисляется только во время добавления правила. Для изменения этой установки можно использовать команду set-salience-evaluation.

Кроме того, пользователи Windows-версии среды CLIPS могут изменить эту настройку с помощью диалогового окна **Execution Options**. Для этого выберите пункт **Options** в меню **Execution**, в появившемся диалоговом окне укажите необходимый режим вычисления приоритета с помощью раскрывающегося списка **Salience Evaluation**.

CLIPS стратегия решение конфликтов

- Стратегия salience

Кроме того, пользователи Windows-версии среды CLIPS могут изменить эту настройку с помощью диалогового окна **Execution Options**. Для этого выберите пункт **Options** в меню **Execution**, в появившемся диалоговом окне укажите необходимый режим вычисления приоритета с помощью раскрывающегося списка **Salience Evaluation**.

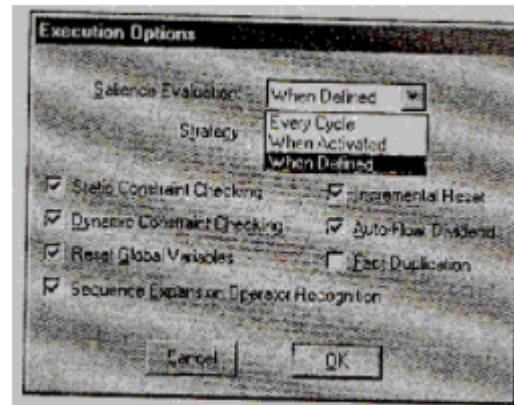


Рис.12

CLIPS стратегия решение конфликтов

- **Свойство auto-focus**

Свойство auto-focus позволяет автоматически выполняться команде focus при каждой активации правила. Если свойство auto-focus установлено в значение true, то команда focus в модуле, в котором определено данное правило, автоматически выполняется всякий раз при активации правила. Если свойству auto-focus присвоено значение false, то при активации правила не происходит никаких действий. По умолчанию это свойство установлено в false

CLIPS стратегия решение конфликтов

- **Стратегия разращение проблемы**

План решения задачи — это список всех правил, имеющих удовлетворенные условия при некотором, текущим состоянием списка фактов и объектов (и которые еще не были выполнены). Каждый модуль имеет свой собствен-ный план решения задачи. Выполнение плана подобно стеку (верхнее пра-вило плана всегда будет выполнено первым). Когда активируется новое правило, оно размещается в плане решения задачи руководствуяс

- Только что активированное правило помещается выше всех правил с меньшим приоритетом и ниже всех правил с большим приоритетом.
- Среди правил с одинаковым приоритетом используется текущая страте-гия разрешения конфликтов для определения размещения среди других правил с одинаковым приоритетом.
- Если правило активировано вместе с несколькими другими правилами, добавлением или исключением некоторого факта и с помощью шагов 1 и 2 нельзя определить порядок правила в плане решения задачи, то пра-вило произвольным образом упорядочиваются вместе с другими прави-лами, которые были активированы. Заметьте, что в этом случае порядок, в котором правила были добавлены в систему, оказывает произвольный эффект на разрешения конфликта. Старайтесь не использовать произволь-ное упорядочивание правил при решении задач, в которых требуются точные результаты или объяснения полученных решений.

CLIPS стратегия решение конфликтов

- **Стратегия разращение проблемы**

CLIPS поддерживает семь различных стратегий разрешения конфликтов: стратегия глубины (depth strategy), стратегия ширины (breadth strategy), стратегия упрощения (simplicity strategy), стратегия усложнения (complexity strategy), LEX (LEX strategy), MEA (MEA strategy) и случайная стратегия (random strategy). По умолчанию в CLIPS установлена стратегия глубины. Текущая стратегия может быть установлена командой set-strategy (которая переупорядочит текущий план решения задачи, базируясь на новой страте-гии). Кроме того, пользователи Windows-версии среды CLIPS могут указать необходимую стратегию поиска с помощью диалогового окна **Execution Options** (см. рис. 6.3). Для этого выберите пункт **Options** в меню **Execution**, в появившемся диалоговом окне выберите необходимую стратегию с помощью раскрывающегося списка **Strategy**.

CLIPS стратегия решение конфликтов

- **Стратегия глубины**

Только что активированное правило помещается выше всех правил с таким же приоритетом. Например, допустим, что факт-А активировал правило-1 и правило-2 и факт-Б активировал правило-3 и правило-4, тогда, если факт-А добавлен перед фактом-Б, в плане решения задачи правило-3 и правило-4 будут располагаться выше, чем правило-1 и правило-2. Однако позиция правила-1 относительно правила-2 и правила-3 относительно правила-4 будет произвольной.

Это воплощение стратегии новизны данных по отношению к правилам, имеющим одинаковый класс выпуклости. Правила, выбранные в список заявок на основании данных, которые были включены в рабочую память сравнительно недавно, располагаются в этом списке раньше правил, при выборе которых использованы более старые данные. Таким образом, предпочтение отдается принципу поиска в глубину в пространстве состояний проблемы, т.е. правила, которые являются следствием более поздних изменений состояния системы, имеют определенный приоритет. В системе CLIPS 6.0 эта стратегия реализуется по умолчанию.

CLIPS стратегия решение конфликтов

- Стратегия глубины

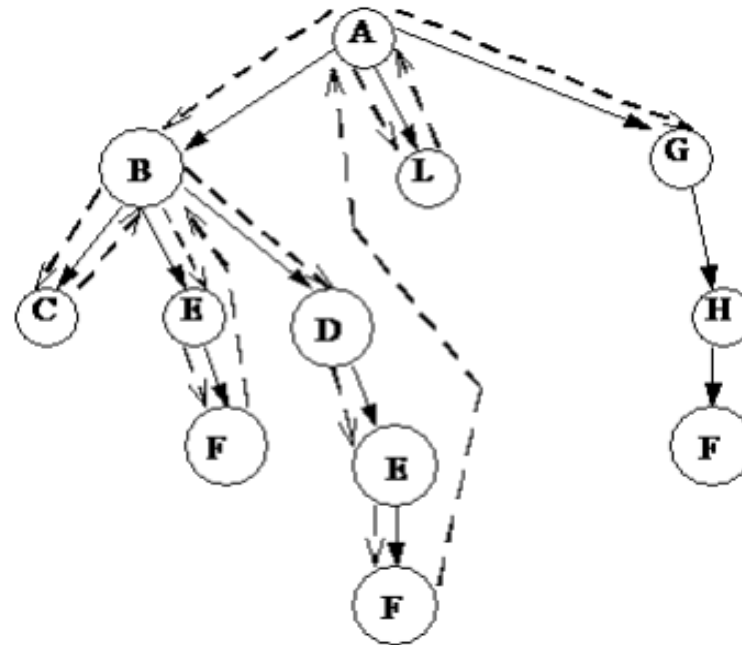


Рис.13 Стратегия поиска в глубину

CLIPS стратегия решение конфликтов

- **Стратегия ширины**

Эта стратегия обратна рассмотренной выше стратегии глубины и предназначена для реализации поиска в ширину в пространстве состояний проблемы. Правила, выбранные в список заявок на основании данных, которые были включены в рабочую память сравнительно давно, располагаются в этом, списке раньше правил, при выборе которых использованы более свежие данные.

Только что активированное правило помещается ниже всех правил с таким же приоритетом. Например, допустим, что факт-А активировал правило-1 и правило-2 и факт-Б активировал правило-3 и правило-4, тогда, если факт-А добавлен перед фактом-В, в плане решения задачи правило-1 и правило-2 будут располагаться выше, чем правило-3 и правило-4. Однако позиция правила-1 относительно правила-2 и правила-3 относительно правила-4 будет произвольной.

CLIPS стратегия решение конфликтов

- Стратегия ширины

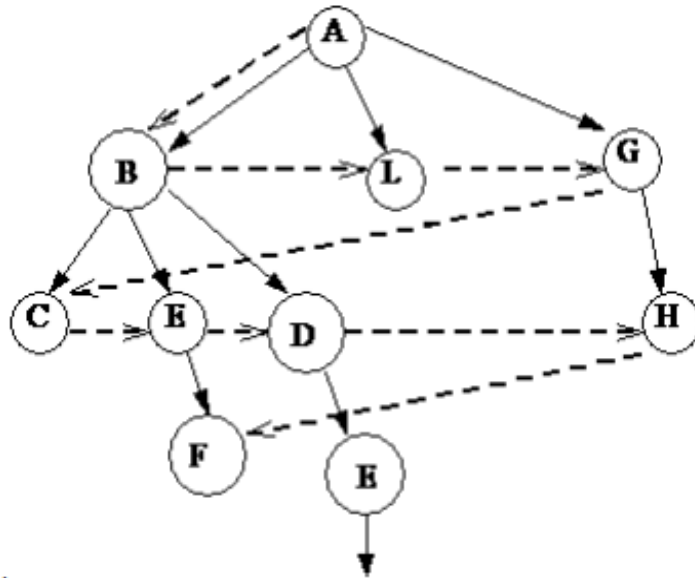


Рис.14 Стратегия поиска в ширину.

CLIPS стратегия решение конфликтов

- **LEX стратегия**

Между правилами с одинаковым приоритетом только что активированные правила размещаются с использованием одноименной стратегии, впервые использованной в системе OPS5. Для определения места активированного правила в плане решения задачи используется "новизна" образца, который активировал правило. CLIPS маркирует каждый факт или объект временным тегом для отображения относительной новизны каждого факта или объекта в системе. Образцы, ассоциированные с каждой активацией правила, сортируются по убыванию тегов для определения местоположения правила. Активация правила, выполненная более новыми образцами, располагается перед активацией, осуществленной более поздними образцами. Для определения порядка размещения двух активаций правил, поодиночке сравниваются отсортированные временные теги для этих двух активаций, начиная с наибольшего временного тега. Сравнение продолжается до тех пор, пока не останется одна активация с наибольшим временным тегом. Эта активация размещается выше всех остальных в плане решения задачи.

Если активация некоторого правила выполнена большим числом образцов, чем активация другого правила и все сравниваемые временные теги одинаковы, то активация с большим числом временных тегов помещает перед активацией с меньшим.

CLIPS стратегия решение конфликтов

- **LEX стратегия**

Если две активации имеют одинаковое количество временных тегов и их значения равны, то правило с большей определенностью помещается перед активацией с меньшей. В отличие от системы OPS5, условный элемент `not` в CLIPS имеет псевдовременной тег, который также используется в данной стратегии разрешения конфликтов. Временной тег условного элемента `not` всегда меньше, чем временной тег образца.

В качестве примера рассмотрим следующие шесть активаций правил, приведенные в LEX-порядке (запятая в конце строки активации означает наличие логического элемента `not`). Учтите, что временные теги фактов не обязательно равны индексу, но если индекс факта больше, то больше и его временной тег. Для данного примера примем, что временные теги равны индексам.

CLIPS стратегия решение конфликтов

- Пример 1: **Правила, отсортированные стратегией LEX**

rule-6: f-1,f-4

rule-5: f-1,f-2,f-3,

rule-1: f-1,f-2,f-3

rule-2: f-3, f-1

rule-4: f-1,f-2

rule-3: f-2, f-1

В примере 2 показаны те же активации с индексами фактов в том поряд-ке, в котором они сравниваются стратегией LEX.

- Пример 2: **Порядок сравнения стратегией LEX**

rule-6: f-4,f-1

+rule-5: f-3,f-2,f-1,

rule-1: f-3,f-2,f-1

rule-2: f-3,f-1

rule-4: f-2,f-1,

rule-3: f-2,f-1

CLIPS стратегия решение конфликтов

- **МЕА стратегия**

Между правилами с одинаковым приоритетом только что активированные правила размещаются с использованием одноименной стратегии, впервые использованной в системе OPS5. Основное отличие стратегии МЕА от LEX в том, что в стратегии МЕА не производится сортировка образцов, активировавших правило. Сравниваются только временные теги первых образцов двух активаций. Активация с большим тегом помещается в план решения задачи перед активацией с меньшим. Если обе активации имеют одинаковые временные теги, ассоциированные с первым образцом, то для определения размещения активации в плане решения задачи используется стратегия LEX. Так же, как и в стратегии LEX, условный элемент `not` имеет псевдовременной тег.

В качестве примера рассмотрим следующие шесть активаций, приведенные в МЕА-порядке (запятая на конце активации означает наличие логического элемента `not`).

CLIPS стратегия решение конфликтов

- **МЕА стратегия**

Пример 1: **Правила, отсортированные стратегией МЕА**

rule-2: f-3,f-1

rule-3: f-2,f-1

rule-6: f-1,f-4

rule-5: f-1,f-2,f-3,

rule-1: f-1,f-2,f-3

rule-4: f-1, f-2,

CLIPS стратегия решение конфликтов

- **Стратегия упрощения**

Между всеми правилами с одинаковым приоритетом только что активированные правила размещаются выше всех активированных правил с равной или большей определенностью (specificity). Определенность правила вычисляется по числу сопоставлений, которые нужно сделать в левой части правила. Каждое сопоставление с константой или заранее связанной с фактом переменной добавляет к определенности единицу. Каждый вызов функции в левой части правила, являющийся частью условных элементов :, = или test, также добавляет к определенности единицу. Логические функции and, or и not не увеличивают определенность правила, но их аргументы могут сделать это. Вызовы функций, сделанные внутри функций, не увеличивают определенность правила.

Например, следующее правило имеет определенность, равную 5

CLIPS стратегия решение конфликтов

- **Стратегия упрощения**
- **Пример 1: Вычисление определенности правила**

(defrule example

(item ?x ?y ?x)

(test (and (numberp ?x) (> ?x (+ 10 ?y)) (< ?x 100))))

=>)

И сравнение заранее связанной переменной ?x с константой, и вызовы функций numberp, < и > добавляют единицу к определенности правила. В итоге получаем определенность, равную 5. Вызовы функций and и + не увеличивают определенность правила.

CLIPS стратегия решение конфликтов

- **Стратегия усложнения**

Между правилами с одинаковым приоритетом, только что активированные правила размещаются выше всех активированных правил с равной или меньшей определенностью



Рис.14 Организация процедуры поиска

Объектно-ориентированное программирование в CLIPS

Использование объектно-ориентированных средств в **CLIPS** позволяет значительно упростить программирование правил, поскольку для обновления данных можно применять механизм передачи и обработки сообщений методами классов. В этом разделе мы продемонстрируем, как это делается на примере, который моделирует правила обращения с полуавтоматическим пистолетом.

Первым делом определим класс **pistol**, в котором будут перечислены свойства, необходимые для моделирования.

- (defclass pistol
- (is-a USER)
- (role concrete)
- (pattern-match reactive)
- (slot safety (type SYMBOL) (create-accessor read-write))
- (slot slide (type SYMBOL) (create-accessor read-write))
- (slot hammer (type SYMBOL) (create-accessor read-write))
- (slot chamber (type INTEGER) (create-accessor read-write))
- (slot magazine (type SYMBOL) (create-accessor read-write))
- (slot rounds (type INTEGER) (create-accessor read-write)))

Объектно-ориентированное программирование в CLIPS

Первые три слота — системные. Они нужны объектно-ориентированной надстройке CLIPS (COOL — CLIPS object-oriented language). Эти слоты COOL извещают о том, что

pistol — это пользовательский класс;

pistol является конкретным классом, т.е. возможно создание экземпляров этого класса (альтернативный тип — абстрактный класс, который играет ту же роль, что и виртуальный класс в C++);

экземпляры класса pistol могут быть использованы в качестве объектов данных, которые можно сопоставлять с условиями в правилах и использовать в действиях, определенных правилами.

Объектно-ориентированное программирование в CLIPS

Следующие пять слотов представляют свойства и члены данных класса:

1. слот `safety` (предохранитель) может содержать символ `on` или `off`;
2. слот `slide` (затвор) может содержать значение `forward` или `back`, т.е. хранит информацию о положении затвора;
3. слот `hammer` (курок) содержит информацию о состоянии курка, `back` или `down`;
4. слот `chamber` (патронник) содержит значение 1 или 0, в зависимости от того, есть ли патрон в патроннике;
5. слот `magazine` (обойма) может содержать значение `in` или `out`, в зависимости от того, вставлена ли обойма;
6. слот `rounds` (патроны) содержит текущее количество патронов в обойме.

Объектно-ориентированное программирование в CLIPS

Для того чтобы иметь возможность записывать в слот новое значение или считывать текущее, нужно разрешить формирование соответствующих функций доступа через фацет `create-accessor`. Теперь сформируем экземпляр класса `pistol` с помощью следующего выражения:

```
(definstances pistols (PPK of pistol (safety on)
(slide forward) (hammer down) (chamber 0) (magazine out) (rounds 6)
```

Этот экземпляр, РПК, правильно уложен — обойма вынута из рукоятки, пистолет установлен на предохранитель, затвор в переднем положении, курок опущен, а патронник пуст. В обойме имеется 6 патронов

- Теперь, имея в программе определение класса и сформировав экземпляр класса, разработаем правила и обработчики сообщений, с помощью которых можно описать отдельные операции обращения с пистолетом и стрельбы из него. Для этого сначала разработаем шаблон задачи. Желательно отслеживать две вещи:
- есть ли патрон в патроннике;
- произведен ли выстрел.

Объектно-ориентированное программирование в CLIPS

Для этого можно использовать следующий шаблон:

1. (deftemplate range-test
2. (field check (type SYMBOL) (default no))
3. (field fired (type SYMBOL) (default no)))

Первое правило будет устанавливать в рабочую память программы задачу range-test.

1. (defrule start
2. (initial-fact) =>
3. (assert (range-test)))

При активизации этого правила в рабочую память будет добавлено (range-test (check no) (fired no))

Следующие три правила будут проверять, правильно ли снаряжен пистолет.

(defrule check

1. (object (name [ППК]) (safety on) (magazine out)
2. ?T <- (range-test (check no)) =>
3. (send [ППК] clear)
4. (modify ?T (check yes))

Объектно-ориентированное программирование в CLIPS

Правило `check` заключается в том, что если пистолет стоит на предохранителе (`safety on`), обойма вынута (`magazine out`) и пистолет не был проверен, то нужно очистить патронник и проверить, нет ли в нем патрона. Обработчик сообщения `clear` для класса `pistol` будет выглядеть следующим образом:

1. `(defmessage-handler pistol clear ( )`
2. `(dynamic-put chamber 0)`
3. `(ppinstance) )`

В первой строке объявляется, что `clear` является обработчиком сообщения для класса `pistol`, причем этот обработчик не требует передачи аргументов. Оператор во второй строке "очищает" патронник. Присвоение выполняется независимо от того, какое текущее значение имеет слот `chamber`, — 0 или 1 . Оператор в третьей строке требует, чтобы экземпляр распечатал информацию о текущем состоянии своих слотов.

Объектно-ориентированное программирование в CLIPS

- В следующих двух правилах обрабатываются ситуации, когда пистолет снаряжен неправильно, — не установлен на предохранитель или в него вставлена обойма. Правило `correct1` устанавливает пистолет на предохранитель, а правило `correct2` извлекает из него обойму.

1. `(defrule correct1`
2. `(object (name [PPK]) (safety off) )`
3. `(range-test (check no)) =>`
4. `(send [PPK] safety on)`
5. `)`
6. `(defrule correct2`
7. `(object (name [PPK]) (safety on) (magazine in))`
8. `(range-test (check no)) =>`
9. `(send [PPK] drop) )`
10. Как и при разработке предыдущего правила, нам понадобятся обработчики сообщений `safety` и `drop`.
11. `(defmessage-handler pistol safety (?on-off)`
12. `(dynamic-put safety ?on-off)`
13. `(if (eq ?on-off on)`
14. `then (dynamic-put hammer down)`
15. `))`

Объектно-ориентированное программирование в CLIPS

Обработчик сообщения `safety` принимает единственный аргумент, который может иметь только два символических значения `on` или `off`. В противном случае нам пришлось бы разработать два обработчика: один для сообщения `safety-on`, а другой — для сообщения `safety-off`. Учтите, что в некоторых моделях, например в Walther PPK, при установке пистолета на предохранитель патронник очищается автоматически.

Обработчик сообщения `drop` просто извлекает обойму из пистолета.

1. `(defmessage-handler pistol drop ()`
2. `(dynamic-put magazine out) )`
3. Теперь, когда обеспечено правильное исходное снаряжение пистолета, можно приступить к стрельбе. Следующее правило обеспечивает вставку обоймы в пистолет перед стрельбой:
4. `(defrule mag-in`
5. `(object (name [PPK]) (safety on) (magazine out))`
6. `(range-test (fired no) (check yes)) =>`
7. `(send [PPK] seat) )`

Объектно-ориентированное программирование в CLIPS

Обработчик сообщения `seat` выполняет действия, противоположные тем, которые выполняет обработчик `drop`.

1. `(defmessage-handler pistol seat (`
2. `dynamic-put magazine in) )`
3. Можно было бы, конечно, включить в программу и следующее правило `mag-in`:
4. `(defrule mag-in`
5. `?gun <- (object (name [PPK]) (safety on)`
6. `(magazine out)) (range-test (fired no) (check yes)) =>`
7. `(modify ?gun (magazine in) )`

но это противоречит одному из принципов объектно-ориентированного программирования, который гласит, что объект должен самостоятельно обрабатывать содержащиеся в нем данные. Следующее правило обеспечивает снаряжение обоймы патронами:

1. `(defrule load`
2. `(object (name [PPK]) (magazine in) (chamber 0)) =>`
3. `(send [PPK] rack) )`

Объектно-ориентированное программирование в CLIPS

На примере обработчика сообщения `rack` вы можете убедиться в справедливости нашего замечания о том, что обработку данных внутри объекта нужно поручать методам этого объекта, а не включать прямо в правило.

1. `(defmessage-handler pistol rack ()`
2. `(if (> (dynamic-get rounds) 0) then (dynamic-put chamber 1)`
3. `(dynamic-put rounds (- (dynamic-get rounds) 1))`
4. `(dynamic-put slide forward) else (dynamic-put chamber 0)`
5. `(dynamic-put slide back)`

Объектно-ориентированное программирование в CLIPS

В этом обработчике обеспечивается досылка патрона в патронник в том случае, если в обойме имеются патроны. Следующее правило подготавливает пистолет к стрельбе, снимая его с предохранителя. Обратите внимание на то, что в нем повторно используется сообщение `safety`, но на этот раз с аргументом `off`.

1. `(defrule ready`
2. `(object (name [PPK]) (chamber 1)) =>`
3. `(send [PPK] safety off) )`
- Правило `fire` выполняет стрельбу.
4. `(defrule fire`
5. `(object (name [PPK]) (safety off);`
6. `?T <- (range-test (fired no)) =>`
7. `(if (eq (send [PPK] fire) TRUE)`
8. `then (modify ?T (fired yes)))) )`

Объектно-ориентированное программирование в CLIPS

- Обратите внимание, что в данном правиле используется обработчик сообщения, которое возвращает значение. Анализируя его, можно выяснить, произведен ли выстрел, т.е. выполнена ли в действительности та операция, которая "закреплена" за этим сообщением. Если в патроннике был патрон и пистолет был снят с предохранителя, то обработчик сообщения вернет значение TRUE (после того, как выведет на экран BANG !). В противном случае он вернет FALSE (после того, как выведет на экран click).

1. (def message-handler pistol fire () (if (and
2. (eq (dynamic-get chamber) 1) (eq (dynamic-get safety) off)
3.)
4. then (printout t crlf "BANG!" t crlf)
5. TRUE
6. else (printout t crlf "click" t crlf) FALSE
7.)

Объектно-ориентированное программирование в CLIPS

- Пусть вас не смущает, что в обработчике сообщения анализируется условие, которое уже было проанализировано правилом, отославшим сообщение (в данном случае речь идет об условии `safety off`). Дело в том, что одно и то же сообщение может отсылаться разными правилами и нет никакой гарантии, что в каждом из них будет проверяться это условие.
- После завершения стрельбы пистолет нужно вновь вернуть в положение "по-походному". Начинается это с того, что пистолет устанавливается на предохранитель, для чего используется ранее разработанный обработчик сообщения `safety`.
 1. `(defrule unread`
 2. `(object (name [ППК]) (safety off))`
 3. `(range-test (fired yes)) =>`
 4. `(send [ППК] safety on) )`
- Следующая операция — вынуть обойму. Обратите внимание, что в нем мы вновь обращаемся к обработчику сообщения `drop`.
 5. `(defrule drop`
 6. `(object (name [ППК]) (safety on))`
 7. `(range-test (fired yes)) =>`
 8. `(send [ППК] drop) )`

Объектно-ориентированное программирование в CLIPS

Последнее правило выбрасывает патрон из патронника, вызывая обработчик сообщения clear.

1. (defrule unload
2. (object (name [PPK]) (safety on) (magazine out))
3. (range-test (fired yes)) =>
4. (send [PPK] clear))

В этом примере было продемонстрировано, как в рамках единой CLIPS программы "уживаются" правила и объекты. Правила управляют ходом вычислений, но некоторые операции объекты выполняют и самостоятельно, получив "указание" (сообщение) от правил. Объекты не являются резидентами рабочей памяти, но члены левой части правил (условий) могут быть сопоставлены с содержимым их слотов. Состояние объектов может измениться и вследствие побочных эффектов активизации правил, но я считаю, что лучше предоставить объектам возможность самостоятельно выполнять манипуляции с хранящимися в них данными в ответ на поступающие от правил сообщения. Объекты не могут самостоятельно активизировать правила, но их обработчики сообщения могут возвращать определенную информацию о результатах, которая используется для управления логикой выполнения действий в правой части правил.

Спасибо за внимание

«Онтологические модели представления знаний о предметной области»

Что такое онтология?

Онтологии используются для описания знаний о некоторой предметной области. Онтология описывает понятия предметной области, а также отношения, которые имеются между этими понятиями.

Онтология, как модель, выражает определенный взгляд (разработчика) на некоторую предметную область и формально может быть описана следующим набором множеств:

$$O = \langle X, R, \Phi \rangle$$

где: X – конечное множество концептов (понятий, терминов) предметной области, которую представляет онтология O . R – конечное множество отношений между концептами (понятиями, терминами) заданной предметной области. Φ – конечное множество функций интерпретации, заданных на концептах и/или отношениях онтологии O .

Из всего множества отношений в онтологии выделяется специальный класс – простая таксономия:

$$O = T^o = \langle X, \{is\_a\}, \{\} \rangle$$

Под таксономической структурой понимается иерархическая система понятий, связанных между собой отношением *is_a* («быть элементом класса» или «быть подклассом класса»). Это отношение (*is_a*) позволяет организовать структуру понятий онтологии в виде

дерева. Отношение *is_a* имеет фиксированную заранее семантику. Предложение «Элемент *A* является подклассом $\xrightarrow{\quad}$ класса *B*», описывается простой логической формулой (импликацией): $A \rightarrow B$ – «Если *A*, то *B*». Получаем формальную таксономию.

Таким образом, из определения онтологии следует, что основными ее компонентами являются: **Классы** (Концепты) или понятия, **Индивидуальности** (экземпляры), **Отношения**, **Функции**, **Аксиомы**.

Концептуальная схема онтологии



Причины потребности в разработке онтологий

- Совместное использование людьми или программными агентами общего понимания структуры информации;
- Разработка и управление терминологией;
- Возможность повторного использования знаний в предметной области;
- Получение надежного семантического базиса в определении содержания;
- Отделение знаний в предметной области от оперативных знаний;
- Получение логической теории, которая состоит из словаря и набора утверждений на некотором языке логики, что позволяет на основе этой теории получать вывод новых знаний, явно не заложенных в онтологии;
- Возможность использования онтологий для поддержки функционирования и роста нового вида цифровых библиотек.

Процесс создания онтологий

1. Определение классов в онтологии;
2. Организация классов в некоторую иерархию (подкласс → класс);
3. Определение отношений (связей) между классами, между элементами классов;
4. Определение свойств (характеристик, атрибутов) элементов класса;
5. Определение экземпляров классов и задание значений их свойств.

При **построении онтологии** нужны определенные ориентиры для принятия решений. Целый ряд решений может приниматься на основе практической цели построения онтологии. Однако можно сформулировать и универсальные требования к онтологиям, не зависящие от конкретной задачи. Так, общая структура онтологии должна быть понятной и должна существовать возможность ее многократного использования. Онтология должна быть когнитивно прозрачной. Важно учитывать следующие требования:

- **Ясность:** онтология должна быть ясной и легко передавать подразумеваемый смысл. Она должна быть объективной;
- **Последовательность:** в ней должны содержаться утверждения, которые не противоречат друг другу, иерархии понятий, связывающим их отношениям, экземплярам.
- **Возможность расширения:** наличие возможности введения новых элементов без пересмотра остальных элементов;
- **Минимальная степень специализации онтологии:** нежелательность полного подчинения онтологии конкретной задаче, что может осложнить ее последующее использование в других задачах.

В качестве примера рассмотрим простую и понятную для студентов вуза предметную область, где основными концептами являются понятия «Группа», «Студент», «Факультет» и т.д.

На рис. 1 приведена онтология предметной области «Учебная деятельность преподавателя». Для этого примера приведены логические формулы, описывающие классы, и формулы, создающие новые отношения.

- Описание классов:

Студент_получит_степень → Бакалавр **or** Магистр **or** Специалист

- Описание отношений в виде цепочки отношений:

лаб_раб_дисциплина <- лаб_раб_тема о тема_дисциплина

студент_ дисциплина <- студент_группа о группа_дисциплина

студент_ факультет <- студент_группа о группа_факультет.

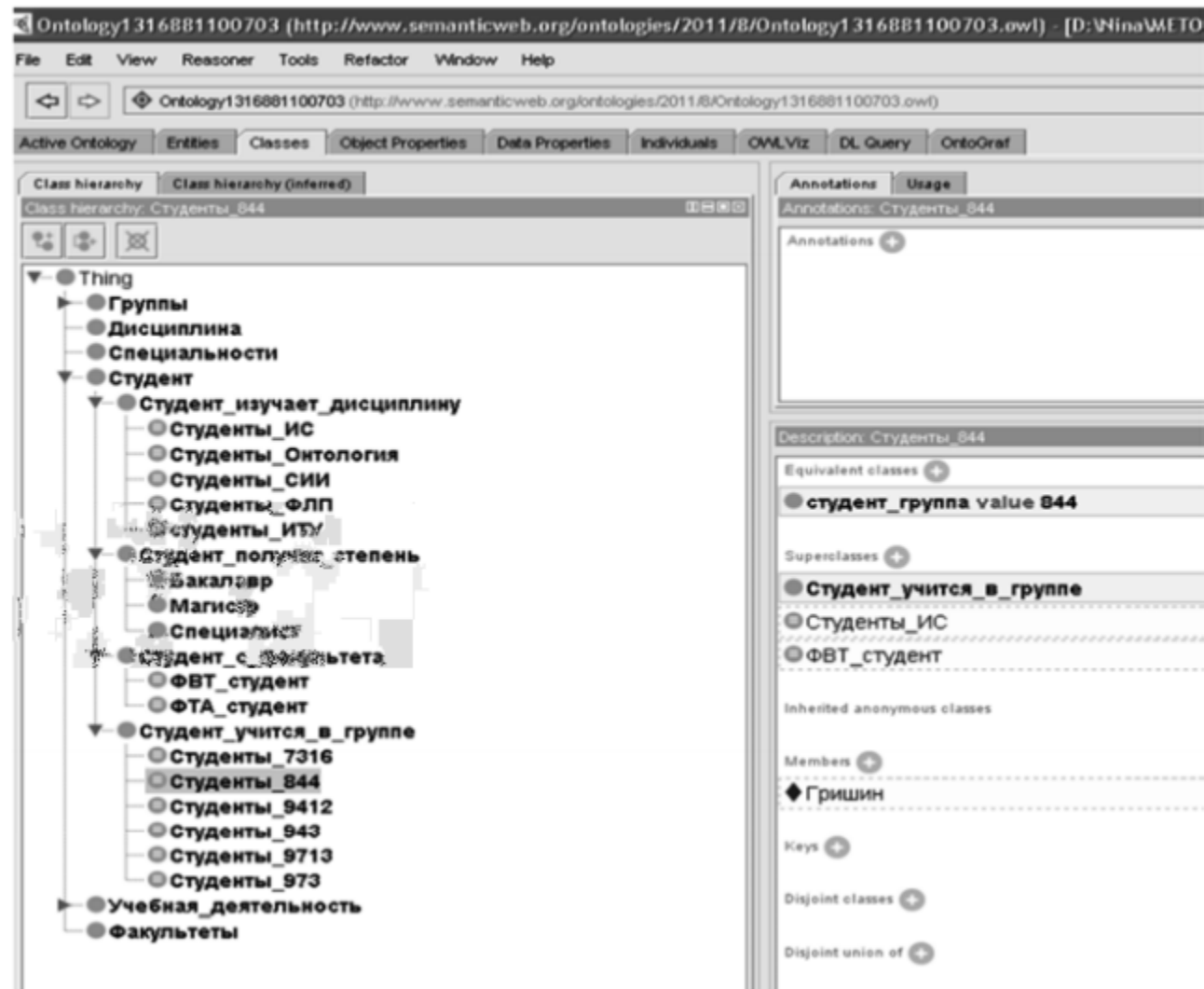


Рисунок 1 – Онтология предметной области
«Учебная деятельность преподавателя»

Компоненты онтологии

Онтология состоит из индивидуальностей, свойств и классов.

Индивидуальности представляют собой объекты интересующей исследователя области. OWL не использует понятие уникального имени объекта. Это означает, что различные имена могут указывать на один и тот же объект. В OWL надо обязательно заявить, индивидуальности идентичны или отличны друг от друга. Индивидуальности рассматриваются как экземпляры классов.

На рис. 2 показано представление индивидуальностей в некоторой области в виде ромбиков или вершин.

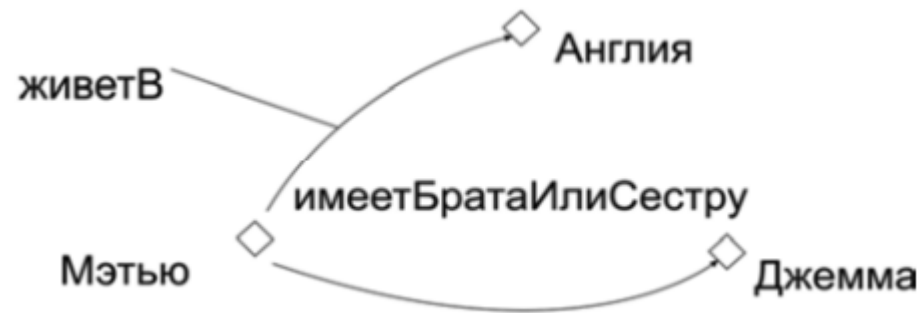


Рисунок 2 – Представление индивидуальностей и свойств

Свойство — это бинарное отношение (связь) между двумя индивидуальностями. Так, например, свойство «Иметь брата или сестру» связывает личность Мэтью с личностью Джеммой, свойство «Иметь ребенка» может связывать индивидуальность Петер с индивидуальностью Мэтью. Свойства могут быть инверсными.

Например, фраза «сущность имеет собственника» является инверсной по смыслу фразе «некто является собственником» некоторой сущности. Рис.2 показывает, как свойства (отношения) связывают индивидуальности между собой. В некоторых формализмах свойства называются атрибутами или реквизитами.

Классы интерпретируются как множества, которые содержат индивидуальности. Сами классы подразделяются на подклассы, а те, в свою очередь, делятся на еще более мелкие классы.

Классы отображаются в виде кругов или овалов (рис. 3), как множества в диаграммах Венна. Концепт и класс при описании Онтологий рассматриваются как синонимы.

Описания классов строятся в виде условий (утверждений), которым должны удовлетворять индивидуальности для того, чтобы быть членами класса.

Индивидуальности могут принадлежать более чем к одному классу.

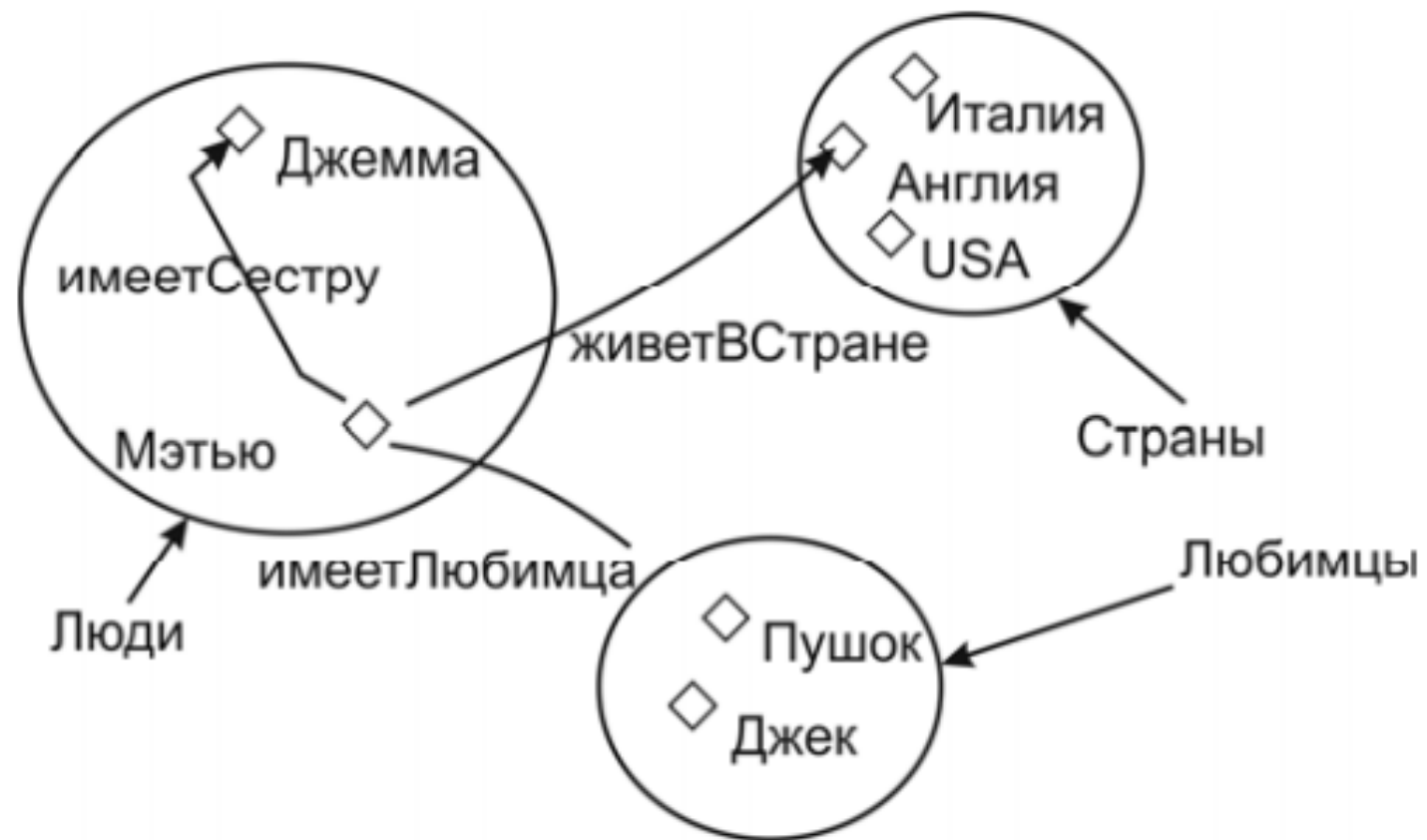


Рисунок 3 – Представление классов, содержащих индивидуальности

Обратные (инверсные) свойства

Каждое свойство объекта может иметь соответствующее **обратное свойство**. Если некоторая сущность *а* связана с сущностью *б*, то сущность *б* так же имеет обратную связь с сущностью *а*. Например, на рис. 4 показаны свойства *имеетРодителя* и его обратное свойство *имеетРебенка* – если *Мэтью* *имеетРодителя* *Жан*, то из обратного свойства мы можем сделать вывод, что *Жан* *имеетРебенка* *Мэтью*.



Рисунок 4 – Пример обратного свойства: свойство *имеетРебенка* является обратным к свойству *имеетРодителя*

Характеристики свойств объекта

- Функциональные свойства
- Обратные Функциональные свойства
- Транзитивные Свойства
- Симметричные свойства
- Асимметричные свойства
- Рефлексивные свойства
- Иррефлексивные свойства
- Цепочки свойств (отношений)

Функциональные свойства

Если свойство **функционально**, то для данного индивида не может быть более одного объекта, с которым он связан через данное свойство. Рис. 5 показывает пример функционального свойства *имеетБиологическуюМаму*. У каждого индивида биологическая мама одна. Если мы говорим, что *Жан имеетБиологическуюМаму Пеггии* мы также говорим, что *Жан имеетБиологическуюМаму Маргарет*, то, поскольку *имеетБиологическуюМаму* – функциональное свойство, мы можем заключить, что *Пеггии Маргарет* должны быть одним и тем же человеком. Следует отметить, однако, что если *Пеггии Маргарет* были явно заданы как различные лица, то вышеуказанное заявление приведет к противоречию.

Обратные Функциональные свойства

Если свойство **обратно-функциональное**, то это означает, что обратное свойство также представляет собой функцию. Для данного индивида не может быть более одного объекта, связанного с этим индивидом через свойство. Можно сказать, что обратно-функциональное свойство соответствует степени связи 1:1. Рис. 6 показывает пример обратного функционального свойства *является Биологической Матерью*.

Это обратное свойство к свойству *имеет Биологическую Мать*. Свойство *является Биологической Матерью* – обратное функциональное. Его смысл: *Пегги* или *Маргарет* – мама только одного ребенка.

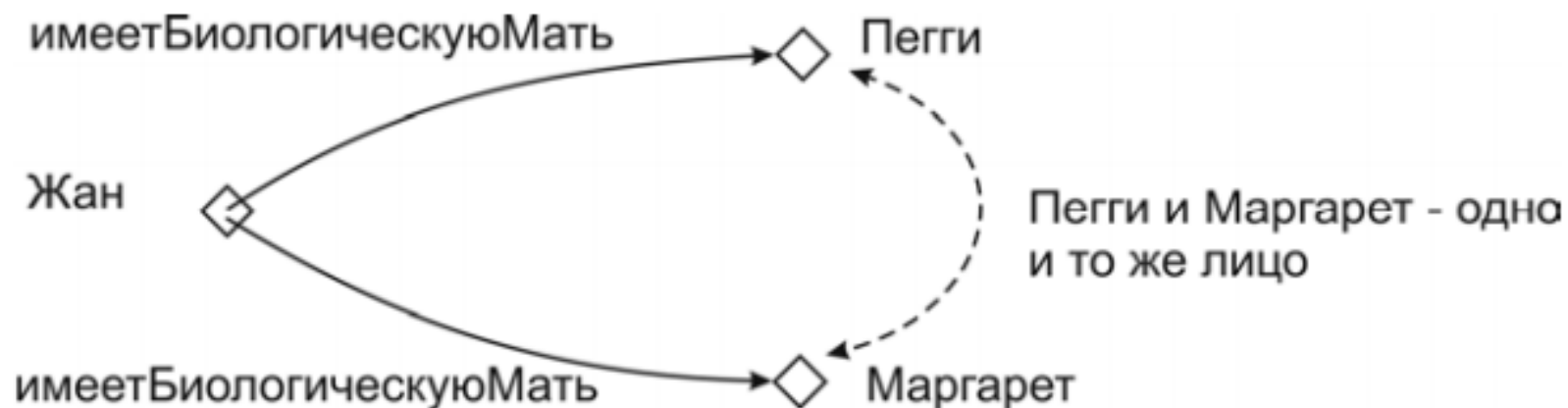


Рисунок 5 – Пример функционального свойства:
имеетБиологическуюМаму

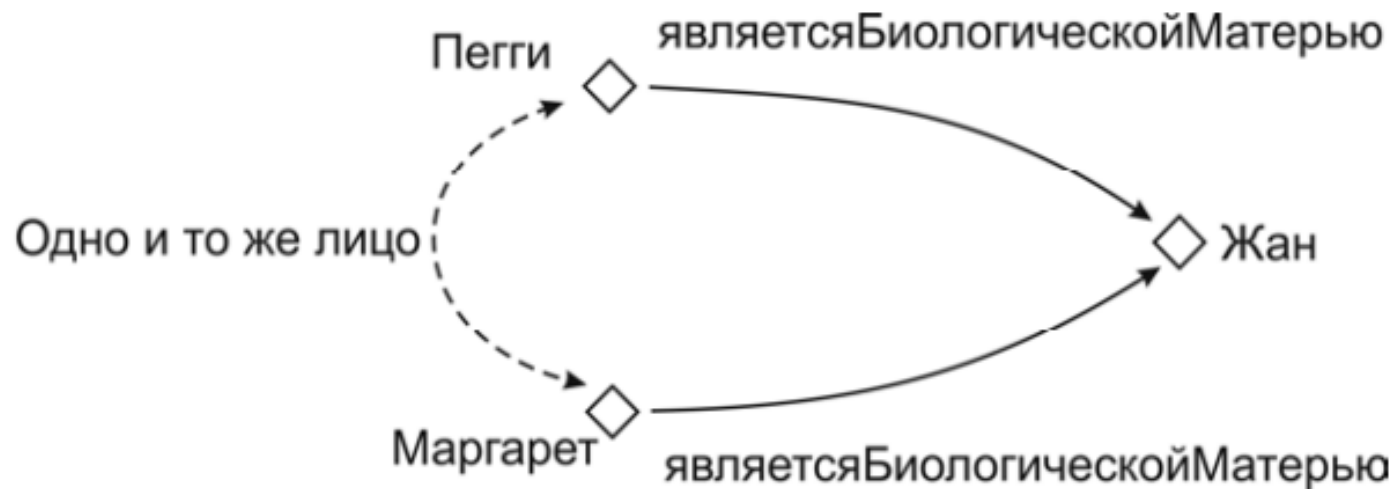


Рисунок 6 – Обратное функциональное свойство
являетсяБиологическойМатерью

Транзитивные Свойства

Если свойство **транзитивно** и связывает отдельные индивиды *a* и *б*, а также индивид *б* с индивидом *с*, то мы можем сделать вывод, что индивид *a* связан этим же свойством с индивидом *с*. Например, рис. 7 показывает пример транзитивного свойства *имеетПредка*. Если *Мэтью* является предком *Петра* и *Петр* – предок *Уильяма*, то можно сделать вывод, что *Мэтью* – предок *Уильяма*, – на это указывает пунктирная линия на рисунке 7.

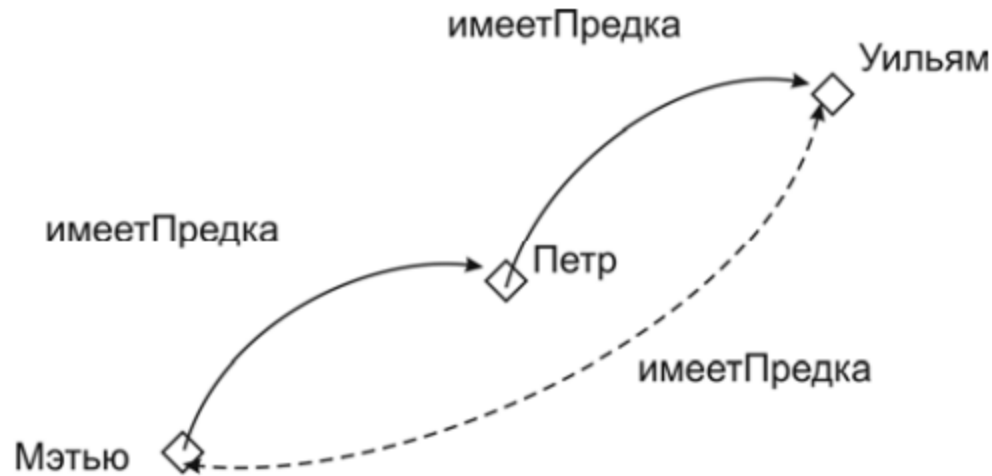


Рис. 7 – Пример транзитивного свойства: имеетПредка

Симметричные свойства

Если свойство P является **симметричным** и свойство связывает индивида a с индивидом b , то b также связан с индивидом a с помощью того же свойства P . Рис. 8 показывает пример симметричного свойства. Если индивид *Мэтью* связан с индивидом *Джеммой* через свойство *имеетБратаИлиСестру*, то мы можем заключить, что *Джемма* также должна быть связана с *Мэтью* через свойство *имеетБратаИлиСестру*.



Рис. 8 – Пример симметричного свойства: *имеетБратаИлиСестру*

Асимметричные свойства

Если свойство P является **асимметричным** и связывает индивида a с индивидом b , то индивид b не может быть связан с индивидом a через свойство P . Рис.9 показывает пример асимметричного свойства. Если индивид *Жан* связан с индивидом *Мэтью* через свойство *являетсяРебенком*, то можно утверждать, что *Мэтью* не может являться ребенком *Жана*, а потому не может быть связан с *Жаном* свойством *являетсяРебенком*. Это, однако, не мешает *Мэтью* быть связанным с другим лицом *Биллом* через свойство *являетсяРебенком*.



Рис. 9 – Пример асимметричного свойства являетсяРебенком

Рефлексивные свойства

Свойство P называется **рефлексивным**, когда это свойство связывает индивида с самим собой. На рис.10 мы видим пример рефлексивного свойства *знает*, *Джордж знает Джорджа*, т.е. самого себя. Однако, кроме того, *Джордж* может знать других людей, поэтому *Джордж* может быть связан отношением *знает* с индивидом *Симоном*.



Рисунок 10– Пример рефлексивного свойства *знает*



Рисунок 11– Пример иррефлексивного свойства *являетсяМамой*

Иррефлексивные свойства

Если свойство **иррефлексивное**, то оно может быть описано как свойство, которое связывает индивида *a* с индивидом *b*, только когда эти индивиды различны. Как пример рассмотрим свойство *являетсяМамой*: индивид *Алиса* может быть связан с индивидом *Бобом* свойством *являетсяМамой*, но *Алиса* не может быть мамой самой себе (рис. 11).

Домены и диапазоны свойств

Свойство описывает бинарное отношение. **Доменом** называется область определения отношения, а диапазоном – область значений отношения. Свойства (отношения) связывают индивидов из **домена** с индивидами **диапазона**.

Применение онтологий

Ряд способов использования онтологий:

- для совместного использования людьми или программными агентами общего понимания структуры информации;
- для возможности повторного использования знаний в предметной области;
- для того чтобы сделать допущения в предметной области явными;
- для отделения знаний в предметной области от оперативных знаний;
- для анализа знаний в предметной области.

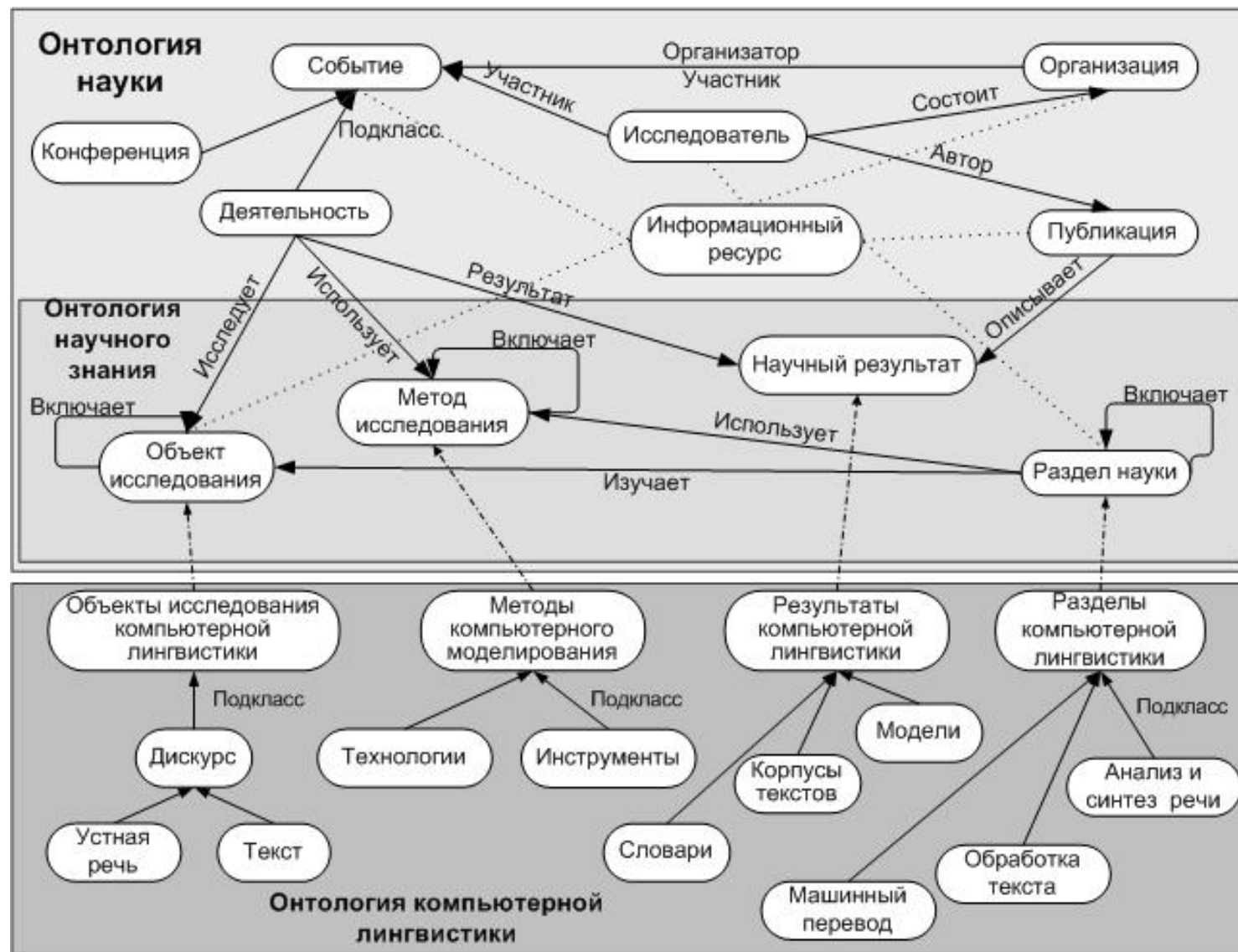


Рисунок 12 – Пример онтологии как научного знания и компьютерной лингвистики

Построение онтологии часто не является само по себе конечной целью, обычно онтологии далее используются другими программами для решения практических целей. На данном этапе развития науки существует ряд задач, где применение онтологий может дать хорошие результаты. Однако сейчас лишь малое количество приложений на естественном языке включают в себя онтологические базы, откуда черпаются знания об окружающей действительности. Использование онтологий возможно в:

- машинном переводе;
- вопросно-ответных системах;
- информационном поиске;
- системах извлечения знаний;
- общих системах ведения диалога между компьютером и человеком;
- системах понимания языка (автоматическое реферирование текста, рубрикация и др.)

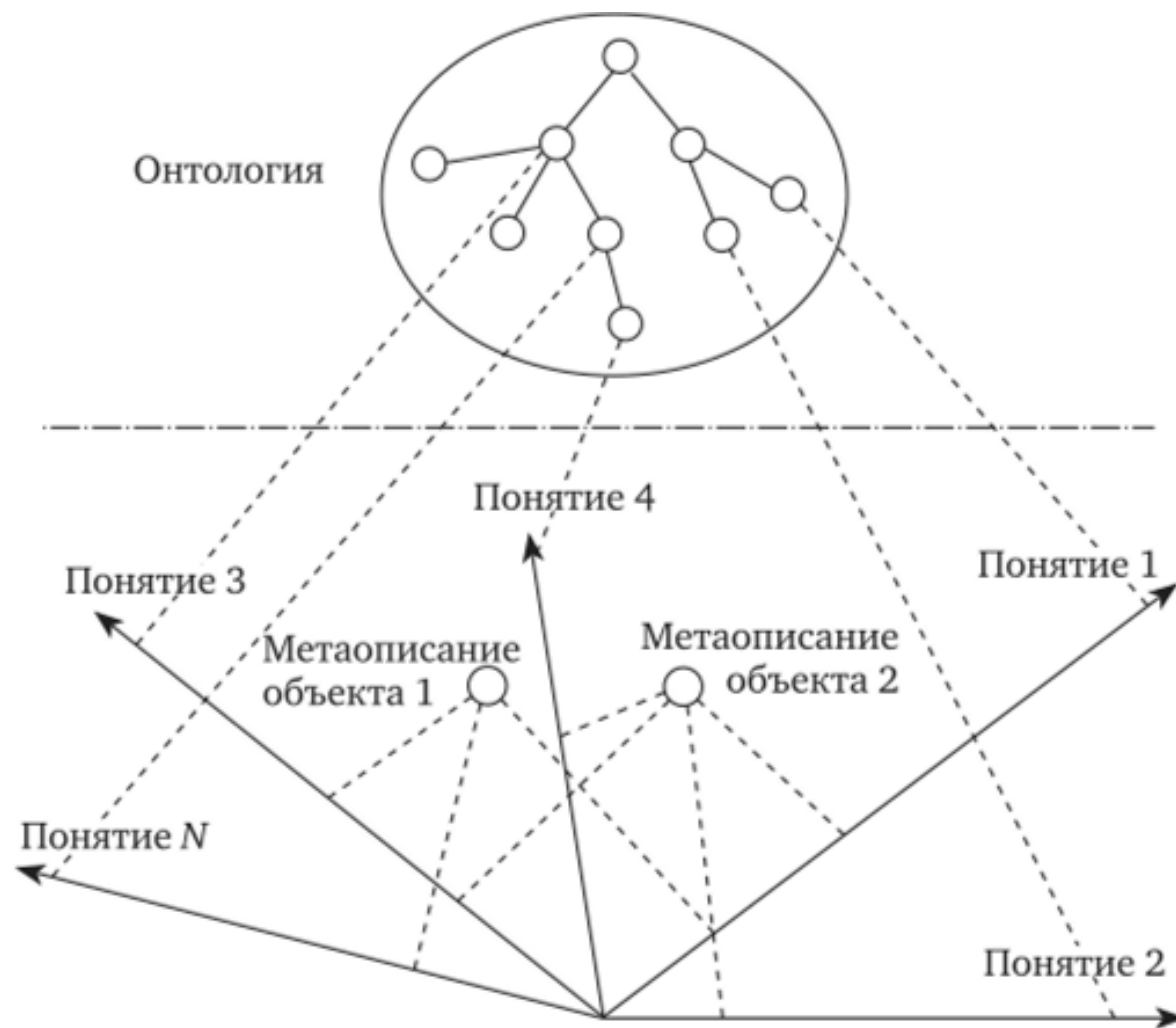


Рисунок 13 – Онтология как модель знаний.
Схема интеллектуального пространства

В **искусственном интеллекте** онтологии используются для формальной спецификации понятий и отношений, которые характеризуют определенную область знаний. Поскольку компьютер не может понимать, как человек, положение вещей в мире, ему необходимо представление всей информации в формальном виде.

Таким образом, онтологии служат своеобразной **моделью окружающего мира**, а их структура такова, что легко поддаются машинной обработке и анализу. Онтологии снабжают систему сведениями о хорошо описанной семантике заданных слов и указывают иерархическое строение области, взаимосвязь элементов. Все это позволяет компьютерным программам при помощи онтологий делать умозаключения из представленной информации и манипулировать ими.

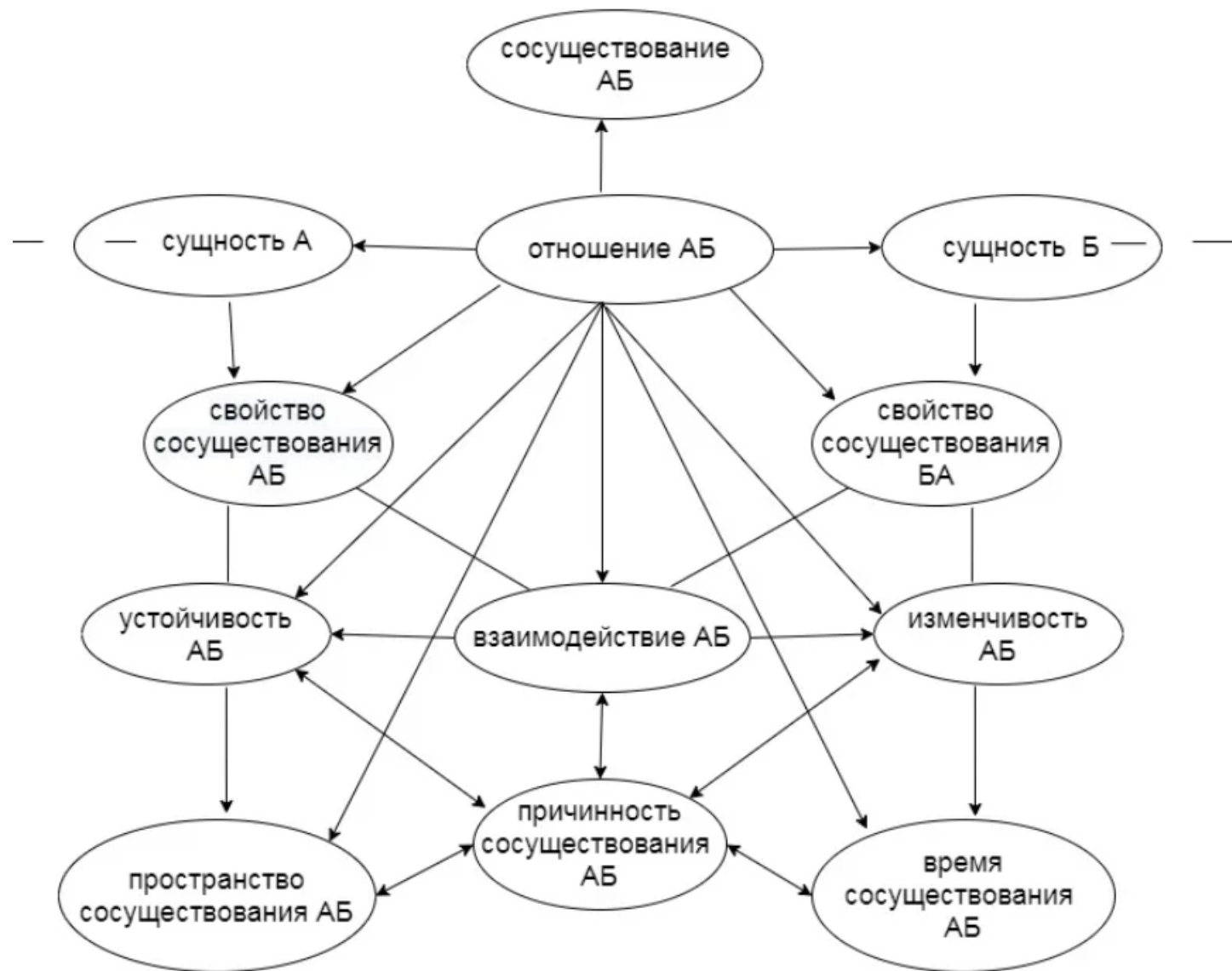


Рисунок 14 – Онтология в искусственном интеллекте

Онтологии используются также при **построении корпуса определений**, служащего **справочным материалом**. В дальнейшем результаты этой работы могут использоваться для сложных процедур обработки естественного языка, например, в снятии омонимии на основе контекста. Онтологии могут использоваться для **вывода умозаключений**, необходимых для понимания текстов на глубинно-семантическом уровне, и что требуется для высококачественного машинного перевода и может служить базой для расширения и уточнения информационного поиска. Глубокий анализ текста необходим и для систем автоматического реферирования. Стоит упомянуть, что также онтологии могут способствовать **систематизации понятий**. На базе онтологий может осуществляться автоматическое аннотирование и разбор текстов, которое в дальнейшем может использоваться в первую очередь в информационном поиске, а также при различных видах анализа информации.

■ EngMath

■ PhysSys

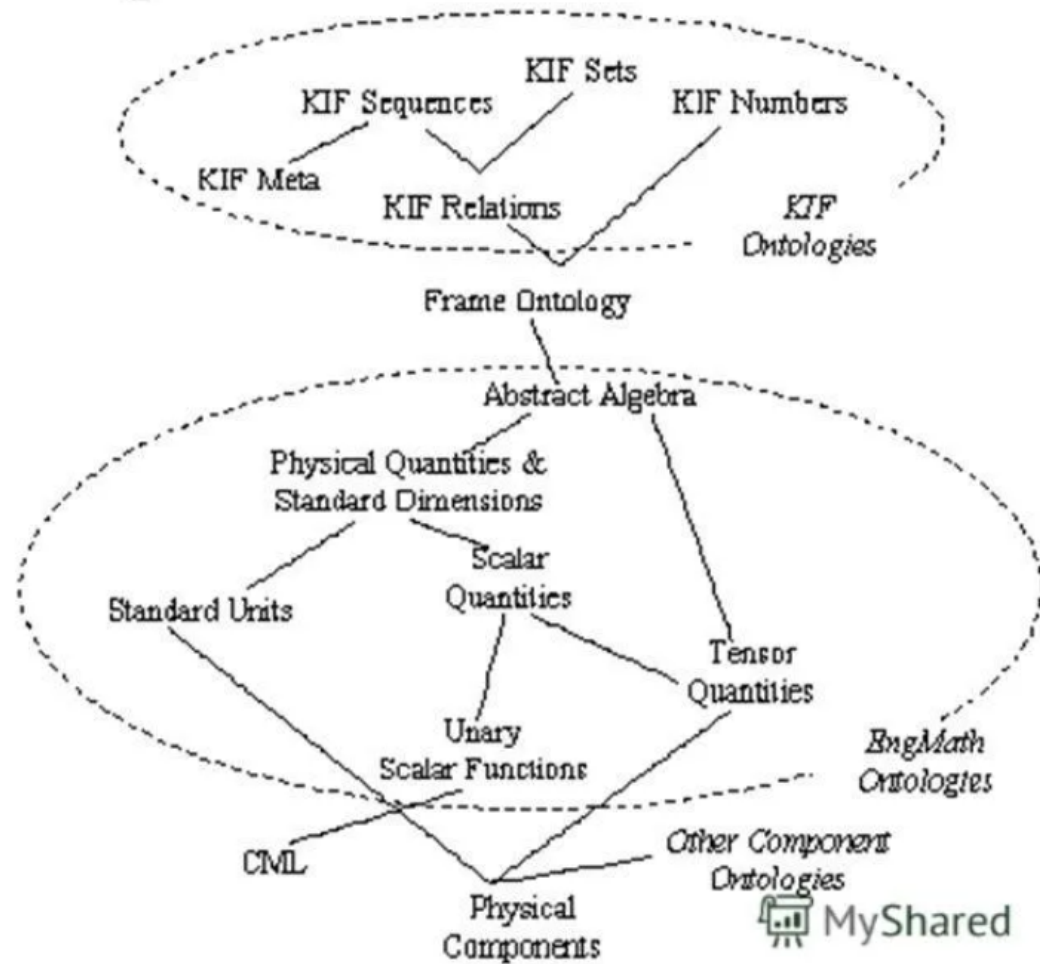


Рисунок 15 – Онтологии предметных областей. Инженерные онтологии

Примеры существующих систем, содержащих онтологические приложения

1. Европейский исследовательский проект под названием CROSSMARC
2. Система OntoLearn (используемая при переводе многословных терминов с английского языка на итальянский)
3. Вопросно-ответная система YAWA (по запросу выдает информацию о главах государств и правительствах стран мира)
4. Система понимания языка, разработанная в НПЦ «Интелтек Плюс» (производит проверки правильности расшифровки стенограмм и проверки редакций различных типов документов на предмет их соответствия эталонным словарям и базам данных)
5. InfoMap

Существующие онтологические ресурсы

На данный момент существует ряд разнообразных онтологических ресурсов. Эти онтологии, противопоставляемые по направленности, отличаются друг от друга по форме и содержанию. В качестве примера можно привести онтологическую структуры, созданной под руководством Э.Хови, которая включает 400 записей и описывает наиболее общее разбиение понятий по категориям. Известной является также Penman Upper Model, состоящая всего лишь из 300 элементов. В ней нет лексических единиц и аксиом. Ее записи представляют понятийно-грамматические классы, выступающие как связующее звено между языком и реальным миром. Между классами верхней модели и синтаксическими типами установлены связи. Эта модель, протестированная для грамматик разных языков и машинного перевода, может быть использована при построении и координации онтологий областей знаний.

Онтология Дж. Совы

Онтология, разработанная Дж. Совой (**Sowa's KR Ontology**), относится к категории структур верхней зоны. Эта онтология базируется на структурах, разработанных философами. Главной целью, преследуемой ее автором, являлось создание основы, в которую могут быть включены все остальные онтологии или модели их верхних зон. Структурная решетка этой онтологии представляет собой 12 понятий, являющихся всеми комбинациями трех отличительных признаков:

- первичность/вторичность/третий порядок
- материальный/абстрактный
- длительный/мгновенный (continuant/occurrent)(объект/процесс)

В онтологии Дж. Совы используются также некоторые стандартные **ситуативные роли**:

- детерминирующий участник (участник, определяющий направление процесса, то есть инициатор или цель);
- неотъемлемый участник (присутствует на протяжении всего процесса, но активно не контролирует происходящее);
- источник (должен присутствовать в начале процесса, но не обязан принимать участие во всем процессе);
- продукт (может появляться в конце процесса, но не обязан принимать участие во всем процессе);

Для того чтобы сохранить открытость, онтология, по мнению Совы, должна быть основана не на фиксированной иерархии концептов, а на каркасе, описывающем различия, по которым иерархия генерируется автоматически. В любом конкретном приложении "концепты не определяются рисованием линий на диаграмме, а задаются выбором подходящего множества различий".

	Физический		Абстрактный	
	Континуальный	Происходящий	Континуальный	Происходящий
Независимый	Объект	Процесс	Схема	Скрипт
Относительный	Слияние	Участие	Описание	История
Опосредованный	Структура	Ситуация	Причина	Цель

Таблица 1 – Онтологические категории верхнего уровня, предложенные Джоном Совой

Кроме приведенных в таблице категорий в онтологии есть еще два понятия. Сущность не определяет никаких отличительных признаков или различий и является над типом для всех других концептов. Второе понятие - Абсурдный тип, наследующий все возможные, в том числе противоречащие, различия. Ни один экземпляр не может иметь этот тип.

SUMO

SUMO (Standard Upper Merged Ontology) - онтология верхнего уровня, разработанная в рамках проекта IEEE SUO (IEEE Standard Upper Ontology) и Teknowledge. Проект претендует на статус стандарта для онтологий верхнего уровня.

Онтология SUMO содержит наиболее общие и самые абстрактные концепты, имеет исчерпывающую иерархию фундаментальных понятий (около 1 тыс.), а также набор аксиом (примерно 4 тыс.), определяющих эти понятия. Назначение SUMO - содействовать улучшению *интероперабельности* данных, извлечения и поиска информации, автоматического вывода и обработки естественного языка. Онтология охватывает следующие области знания: общие виды процессов и объектов, абстракции (теория множеств, атрибуты, отношения), числа и единицы измерения, временные понятия, части и целое, агенты и намерения.



Рисунок 16 – Иерархия классов SUMO

Основными концептами, как во многих онтологиях верхнего уровня, являются Сущность и ее категории - **Физический** и **Абстрактный**. Первая категория включает всё, что имеет положение в пространстве-времени, а вторая - все остальное. Физический делится на **Объект** и **Процесс**, что соответствует подходу, реализованному в *DOLCE*. Непосредственно под концептом Объект находятся два непересекающихся понятия: *СвязныйОбъект* и *Коллекция*. Первое обозначает любой объект, все части которого непосредственно или косвенно связаны друг с другом. Концепт *СвязныйОбъект* разделен на два концепта: *НепрерывныйОбъект* и *ДискретныйОбъект*. *НепрерывныйОбъект* характеризуется тем, что все его части (вплоть до самого низкого уровня деления) имеют такие же свойства, как и целое. Такие субстанции, как вода и глина могут быть подклассами концепта *НепрерывныйОбъект*, так же как и поверхности и географические области. Ниже на диалекте SUO-KIF языка KIF записаны формальные аксиомы, определяющие различие между концептами *НепрерывныйОбъект* и *ДискретныйОбъект*.

A1. ($\Rightarrow$
(and
 (subclass-of ?OBJECTTYPE НепрерывныйОбъект)
 (instance-of ?OBJECT ?OBJECTTYPE)
 (part-of ?PART ?OBJECT))
(instance-of ?PART ?OBJECTTYPE))

A2. (equal ДискретныйОбъект (ComplementFn НепрерывныйОбъект))

Аксиома **A1** формализует утверждение "Если PART является частью объекта OBJECT, являющегося, в свою очередь, экземпляром некоторого подкласса OBJECTTYPE класса НепрерывныйОбъект, то PART, как и OBJECT, является экземпляром класса OBJECTTYPE ". Аксиома **A2** постулирует факт, что классы ДискретныйОбъект и НепрерывныйОбъект являются взаимодополняющими.

Контрольные вопросы

1. Что такое онтология?
2. Назовите характеристики свойств объекта.
3. Что такое домен и диапазон свойства?
4. Перечислите компоненты онтологии.
5. Где онтология может быть использована?
6. Назовите причины потребности в разработке онтологий.
7. Что такое обратное свойство?
8. Какие вы знаете примеры существующих систем, содержащих онтологические приложения?
9. К какой категории структур относится онтология Дж.Совы?
10. Что такое SUMO?

Глоссарий

Онтология — формальная спецификация разделяемой концептуальной модели.

Индивидуальности — объекты интересующей исследователя области.

Свойство — это бинарное отношение (связь) между двумя индивидуальностями.

Классы — множества, которые содержат индивидуальности.

Домен — область определения отношения.

Диапазон — область значений отношения.

Классификация — типы, объединённые в иерархию.

Свойства (отношения) – связывают индивидов из домена с индивидами диапазона.

Искусственный интеллект — свойство интеллектуальных систем выполнять творческие функции, которые традиционно считаются прерогативой человека

SUMO (Standard Upper Merged Ontology) - онтология верхнего уровня, разработанная в рамках проекта IEEE SUO (IEEE Standard Upper Ontology) и Teknowledge.

Онтология Дж.Совы – онтология, разработанная Дж. Совой (Sowa's KR Ontology), относится к категории структур верхней зоны.

Объект— в самом широком смысле то, на что направлено индивидуальное или коллективное сознание.

Semantic Web. Поиск знаний в семантической паутине.

Введение

Существующая сеть WWW представляет собой гигантское количество информации в формате, приспособленном для человеческого восприятия.

Пользователь может переходить с одной ссылки на другую, давать запросы различным поисковым системам или же находить сайты, просто вводя их адреса. И хотя веб-страницы весьма привлекательны для человека, для компьютерной программы, обрабатывающей их содержимое, они не более чем строчки из случайных СИМВОЛОВ.

Введение

Всемирная паутина представляет собой информационный ресурс с практически неограниченным потенциалом. Однако, этот потенциал является относительно неиспользованным, поскольку он является сложным для машиной обработки, а интегрировать эту информацию необходимо. В последнее время исследователи приступили к изучению возможности привлечения Веб-содержимого в целях создания Semantic Web. Вместо того чтобы полагаться на обработку естественного языка, для извлечения смысла из имеющихся документов, такой подход требует, чтобы авторы описывали документы, используя знания языка представления.

Семантическая паутина

Семантическая паутина (Semantic Web) — это направление развития Всемирной паутины, целью которого является представление информации в виде, пригодном для машинной обработки.

В обычной Паутине, основанной на HTML-страницах, информация заложена в тексте страниц и извлекается человеком с помощью браузера. Семантическая паутина предполагает запись информации в виде семантической сети с помощью онтологий.

Т.е. программа-клиент может непосредственно извлекать из паутины факты и делать из них логические заключения.

Семантическая паутина

Семантическая паутина работает параллельно с обычной Паутиной и на её основе, используя протокол HTTP и идентификаторы ресурсов URI (Uniform Resource Identifier).

Термин «семантическая паутина» был впервые введён сэром Тимом Бернерсом-Ли (изобретателем Всемирной паутины) в мае 2001 года в журнале «Scientific American», и называется им «следующим шагом в развитии Всемирной паутины».

Основные сферы применения—моделирование бизнес-процессов, семантическая паутина, искусственный интеллект.

Семантическая паутина



Рисунок 1 –Тим Бернес Ли

Семантическая паутина

Главная трудность при построении семантической сети заключается в том, чтобы найти способ унифицированного представления всех типов данных.

Для этого предлагается использовать три основные составляющие - механизм описания ресурсов RDF (Resource Definition Framework), язык онтологий OWL (Web Ontology Language) и язык запросов SPARQL.

Работы над первыми двумя составляющими были завершены более десяти лет назад, а недавно консорциум World Wide Web(W3C) обнародовал и спецификации SPARQL. Таким образом, эра семантической сети может наступить быстрее, чем кажется.

История и мотивация

Семантический Веб — новая концепция развития Веба и сети Интернет, принятая и продвигаемая W3C (World Wide Web Consortium). Эта организация разрабатывает и внедряет технологические стандарты для Всемирной паутины. Когда сейчас слово «Интернет» употребляется в обиходе, то чаще всего имеется в виду Всемирная паутина и доступная через нее информация, а не сама физическая сеть компьютеров. Интернет—всемирная система объединенных компьютерных сетей, построенная на использовании протоколов (TCP/IP) для связи и маршрутизации пакетов данных.

История и мотивация

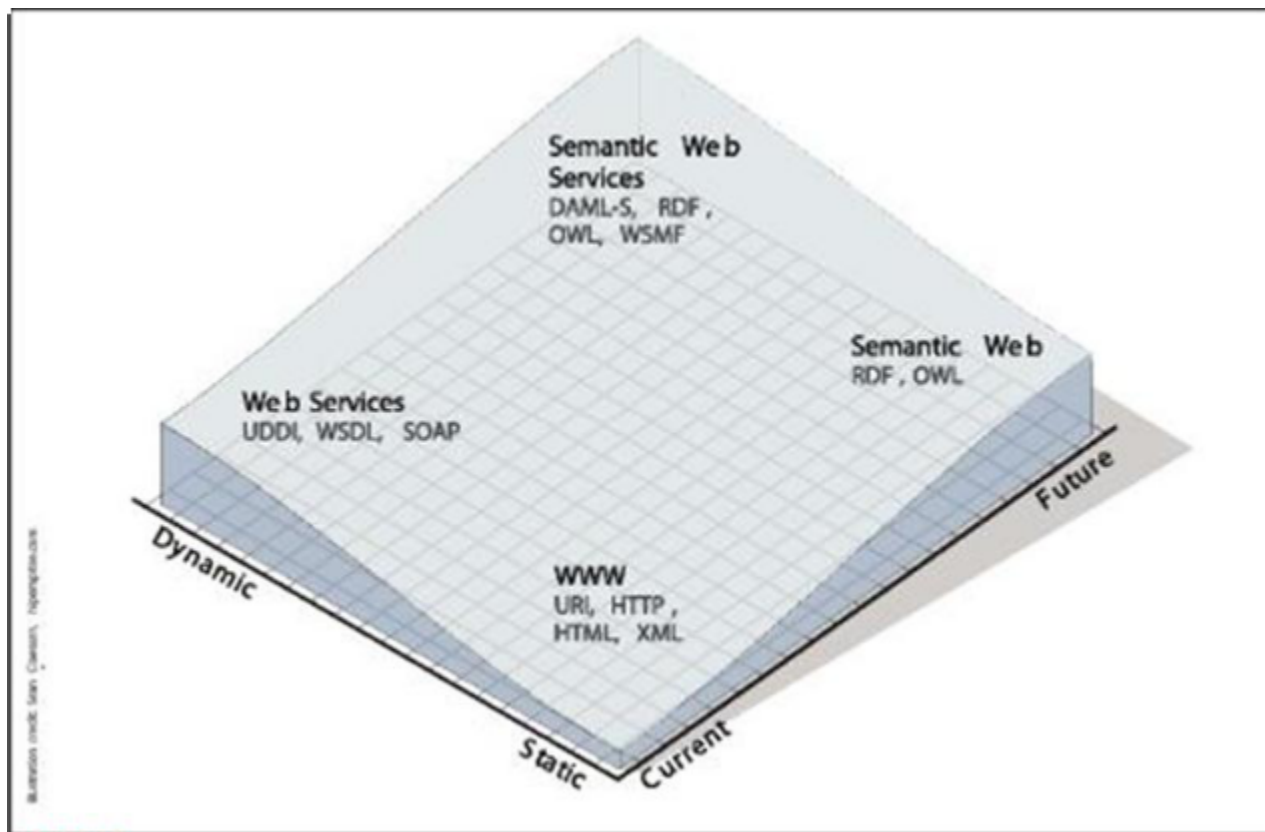


Рисунок 2 – Эволюция сети WWW

История и мотивация

Веб — глобальное информационное пространство, основанное на физической инфраструктуре Интернета и протоколе передачи данных HTTP.

При его создании в 1989 Тимом Бернерсом-Ли предполагалась не только человеческая взаимосвязь, но и участие компьютеров в осмысленном манипулировании информацией.

История и мотивация

Основной помехой для этого является тот факт, что большинство данных в Вебе хранится в форме, рассчитанной на восприятие человеком. Таким образом, даже при учете того, что информация получена из какой-либо базы, ее структура не очевидна роботу.

Компьютер способен умело воспроизвести веб-страницу, но, вообще говоря, у него нет надежного способа для извлечения семантики документа. Семантический Веб как раз имеет своей целью наверстать это упущение.

История и мотивация

Семантический Веб — это не какая-то отдельная сеть, а расширение уже существующей, такое, что в ней информация снабжена точно определенным значением, что позволит человеку и машине успешней взаимодействовать. Первые этапы на пути к «вплетению» Семантической Сети в структуру имеющейся Сети уже осуществляются полным ходом. В ближайшем будущем данные разработки возвестят о новых значительных функциональных возможностях, когда машины станут намного лучше обрабатывать (понимать) те данные, которые сейчас они просто показывают на экране.

История и мотивация

Можно выделить несколько вариантов использования Семантического Веба:

1. Семантический поиск.
2. Объединение знаний (интеграция баз данных).
3. Всепроникающие вычисления (ubiquitous computing).

История и мотивация

1. Семантический поиск

Поисковая система сможет выдавать только те сайты, где упоминается в точности искомое понятие, а не произвольные страницы, в тексте которых встретилось данное многозначное ключевое слово.

Сегодняшние поисковые системы зачастую выдают бесчисленное множество совершенно не относящихся к запросу страниц, обрекая пользователя на длительный ручной отбор материала.

История и мотивация

2. Объединение знаний (интеграция баз данных)

Мы уже говорили, что отсутствие семантики ведет к тому, что компьютеру не ясно, как поступить с информацией из базы данных даже при условии того, что он знает названия всех столбцов полученной таблицы.

Семантический Веб, именуя всякое понятие просто с помощью URI-идентификатора, даст возможность каждому выражать новые понятия, которые он изобретает, с минимальными усилиями. Его универсальный логический язык позволит постепенно связать все эти понятия в универсальную Сеть.

История и мотивация

3. Всепроникающие вычисления (ubiquitous computing)

На следующем этапе своего развития Семантическая Сеть уже вырвется из виртуальной области и расширит сферу своего влияния на физический мир, поскольку URI-идентификаторы могут указывать на что угодно, в частности, и на физические объекты, такие как сотовый телефон или телевизор.

Эти устройства смогут рекламировать свои функциональные возможности (что они умеют делать и каким образом ими можно управлять) практически так же, как это делают программные агенты.

Хронология

- 1994: Создание W3C. Консорциум разработал стандарты: HTML, URL, XML, HTTP, PNG, SVG, CSS.
- 1998: Тим Бернерс-Ли публикует план Семантического Веба (SemanticWeb Roadmap).
- 1999: W3C создает группы по проектированию Семантического Веба, публикуется первая версия RDF.
- 2000: Американские военные начинают исследования по описанию онтологий (DAML+OIL project).

Хронология

- 2001: В журнале Scientific American публикуется описание Семантического Веба.
- 2004: Выпущена новая версия RDF, представлен язык описания онтологий OWL.
- 2006: Представлена версия языка запросов SPARQL (candidate recommendation).

В разработке проекта участвуют: HP, Mozilla, IBM, MIT, Stanford, ...

Архитектура Семантического Веба

Для реализации нужно следующее:

1. Разработать язык огромной выразительной силы, на котором можно описать все знания человека и который был бы понятен компьютерам.
2. Перевести все сайты на этот язык.
3. Написать программы, работающие со знаниями на этом языке (обработка запросов, логический вывод).

Первые два пункта представляются невыполнимыми. Рассмотрим более тонкое решение, предложенное Тимом Бернерсом-Ли.

Общая архитектура

В отличие от подхода, использующего искусственный интеллект для обучения компьютеров поведению людей, его идея заключается в поэтапной и распределенной разработке языка, способного выражать информацию в понятной машине форме.

Т.е. цель Семантического Веба—создание языка, на котором можно будет описать как данные, так и правила рассуждений об этих данных, так что правила вывода, существующие в какой-либо системе представления знаний, можно будет экспортировать в Веб. Чтобы определить язык, необходимо задать его синтаксис и семантику.

Общая архитектура

Синтаксис—набор правил построения фраз языка, позволяющий определить корректные предложения в этом языке. Основным инструментом синтаксиса является наличие правил проверки, позволяющих судить о том, удовлетворяет ли текст синтаксису или нет.

Семантика—система правил истолкования отдельных языковых конструкций. Семантика определяет смысловое значение предложений языка.

Примером языка с синтаксисом, но без семантики, является XML. Примером семантики без синтаксиса — человеческая речь. Поэтому программам так трудно разобратся, где что.

Общая архитектура

Тим Бернерс-Ли предложил отдельно разрабатывать синтаксис и семантику языка описания всех знаний человечества:

- RDF (Resource Description Framework) — язык, отвечающий за синтаксис документов Семантического Веба. В нем широко используются ссылки на онтологии для определения смысла слов.
- OWL (Ontology Web Language) — язык описания онтологий. Онтология — описание классов объектов, их свойств и взаимоотношений для какой-то предметной области (домена).

Общая архитектура

Т.е. план Тима Бернерса-Ли требует последовательно разработать:

1. Синтаксис для представления знаний, использующий ссылки на онтологии (сделано: RDF).
2. Язык описания онтологий (сделано: OWL).
3. Язык описания веб-сервисов (начато: WSDL, OWL-S). В настоящее время уже существует множество автоматизированных веб-сервисов безо всякой семантики. Но у других программ, таких как агенты, нет никакого способа разыскать в сети подобную программу, выполняющую ту или иную функцию.

Общая архитектура

4. Инструменты чтения и разработки документов Семантического Веба (начато: Jena, Haystack, Protege). Главный минус концепции Семантического Веба— сложность внедрения. Формат RDF был разработан людьми с академическим образованием и изначально не был рассчитан на применение рядовыми пользователями Интернета. Даже многим веб-мастерами программистам бывает сложно освоить RDF и OWL.

Несмотря на это, Тим Бернерс-Ли утверждает, что в будущем никаких специальных знаний для создания страниц не потребуется.

Общая архитектура

5. Язык запросов к знаниям, записанным в RDF (начато: SPARQL). SPARQL —новый язык запросов для быстрого доступа к данным RDF.

Технология SPARQL позволяет извлекать данные из распределенных источников и может служить в качестве средства интеграции разнородной информации. Используя обычный протокол и язык SPARQL, приложения могут анализировать RDF-описания ресурсов и получать из сети нужную информацию.

Общая архитектура

6. Логический вывод знаний (не сделано).
7. Семантическая поисковая система (начата разработка проекта SHOE (Simple HTML Ontology Extensions). Это XML-совместимый язык для представления знаний для Web-среды. Это - небольшое расширение для HTML, которое позволяет автору веб-страницы представить его веб-документ в виде машиночитаемых знаний.
8. Агенты Семантического Веба (не сделано).

Общая архитектура

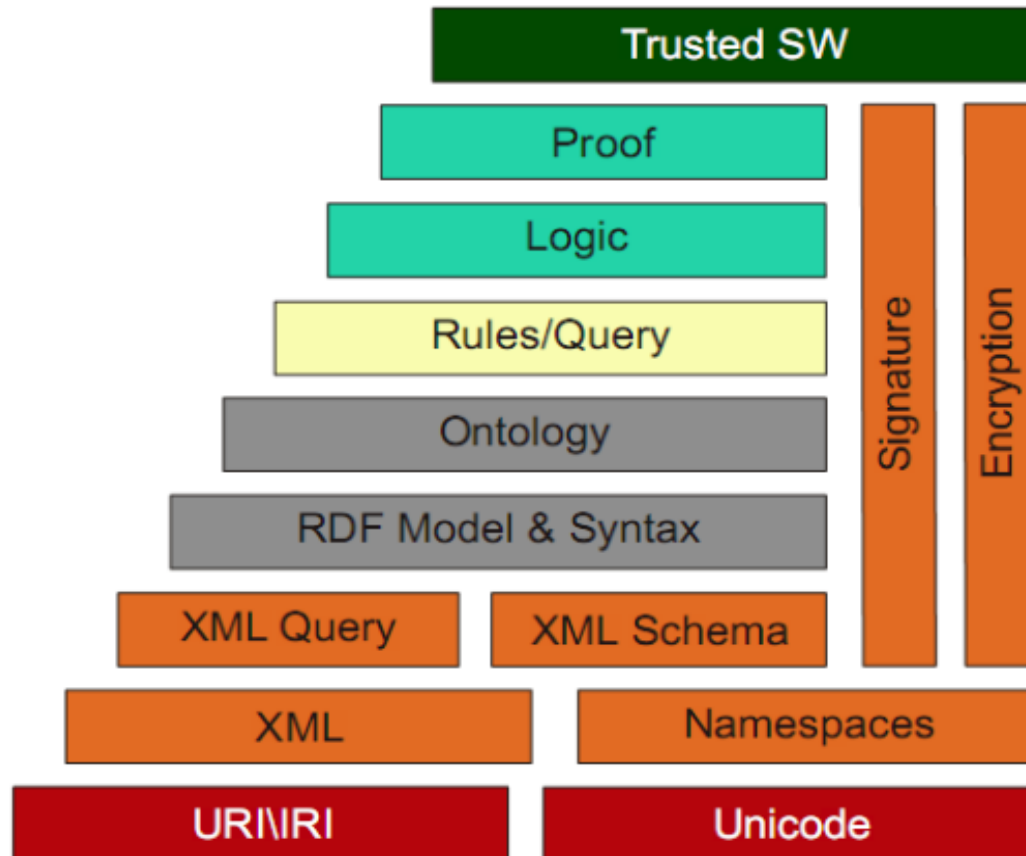


Рисунок 3 - Пирог уровней Семантического Веба.
(Тим Бернерс-Ли, 2000)

Общая архитектура

XML используется в качестве синтаксиса документов. Совместно с пространством имен может использоваться для выражения информации и обмена ею между программами. XML Schema используется для выражения набора правил, которым должен удовлетворять XML-документ, чтобы быть признанным корректным. Протокол HTTP и язык HTML лежат где-то в области этого уровня, но воспринимаемые человеком документы и их передача в значительной степени отличаются от материала, ориентированного преимущественно на машины. Поэтому на этой диаграмме они не показаны.

Общая архитектура

- Логический вывод (Logic) используется для обеспечения связности и корректности информации и для получения новых данных.
- Доказательства (Proof) отслеживают и объясняют шаги логического вывода. Заслуживающий доверия
- Семантический Веб (TrustedSW) — средства, выполняющие аутентификацию, проверку достоверности информации, надежности сервисов и агентов.

RDF: синтаксис

Семантического Веба

Язык XML позволяет пользователям создавать документы произвольной структуры. Но сам язык ничего не говорит о том, что означает эта структура. Смысл выражается посредством языка RDF, который кодирует его с помощью деревьев глубины три (Notation3). Здесь каждое дерево состоит из субъекта(подлежащее), предиката (свойства) (сказуемое) и объекта(дополнение). Объект можно назвать функцией свойства от субъекта. Например, утверждение «Небо голубого цвета» в RDF-терминологии можно представить следующим образом: субъект —«небо», свойство —«иметь цвет», объект —«голубой». Для идентификации субъектов, свойств и объектов используются Uniform Resource Identifiers (URI).

RDF: синтаксис Семантического Веба

В начале любого RDF документа идет список ссылок на онтологии. Таким образом, каждая вершина может быть задана строкой или ссылкой на объект из какой-либо онтологии. Вершины могут иметь дополнительные уточняющие квалификаторы. Частным случаем формата RDF является формат RSS. RSS — семейство XML-форматов, предназначенных для описания лент новостей, анонсов статей, изменений в блогах и т.п. Информация из различных источников, представленная в формате RSS, может быть собрана, обработана и представлена пользователю в удобном для него виде специальными программами-агрегаторами.

Пример документа в формате RDF

```
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#  
xmlns:dc=http://purl.org/dc/elements/1.1/  
xmlns:geo=http://www.w3.org/2003/01/geo/wgs84_pos#  
xmlns:edu="http://www.example.org/">  
<rdf:Description rdf:about="http://www.princeton.edu">  
<geo:lat>40.35</geo:lat>  
<geo:long>-74.66</geo:long>  
<edu:hasDept rdf:resource=http://www.cs.princeton.edu  
dc:title="Department of Computer Science"/>  
</rdf:Description>  
</rdf:RDF>
```


RDFS и парадигма ООП

ООП

- В классе описываются его атрибуты.
- Атрибуты, их характеристики и ограничения определены локально в контексте каждого класса.
- В двух разных классах можно встретить 2 атрибута с одинаковыми названиями, но разными смыслами, характеристиками ограничениями.

RDFS 1.0

- Определение свойства «глобально».
- Смысл и характеристики (включая domain, range) свойства не зависят оттого, в контексте какого класса этот атрибут появляется.
- В каждом классе может появиться заданный атрибут, который указывается в описании свойства с помощью domain, а не в описании класса.

OWL

Выразительные возможности RDF и RDF –схемы намеренно очень ограничены: RDF (грубо говоря) ограничивается бинарными предикатами, а RDF-схема (грубо говоря) ограничивается иерархией подклассов и иерархией свойств, с определением областей определения и значений этих свойств.

OWL -расширяет словарь описания свойств и классов, добавляя возможности выражения:

- отношений между классами (напр., disjoint With),
- мощности (напр., minCardinality1)
- равенства классов (напр., equivalent Class)
- характеристики свойств (напр., Functional Property)

OWL: язык описания онтологий

Основные компоненты OWL включают классы, свойства и индивидуальные элементы.

Класс—это концепция в онтологии. Классы являются основными блоками OWL и обычно образуют таксономическую иерархию (т.е. систему подкласс-надкласс).

OWL поддерживает шесть основных способов описания классов. Самый простой — класс с именем (named). Другие типы: классы пересечений (intersection), объединений (union), дополнений (complement), ограничений (restrictions) и классы перечислений (enumerated).

OWL: язык описания онтологий

Индивидуальные элементы —это элементы классов.

В RDF они будут объектами и субъектами. Мир классов и индивидов был бы совершенно неинтересным, если бы мы могли только определять таксономии. Свойства позволяют нам утверждать общие факты о членах классов и особые факты об индивидах. Они включают две основные категории: свойства-объекты, которые связывают индивидуальные элементы между собой и свойства-значения (data type properties), которые связывают индивидуальные элементы со значениями типов данных. Для определения типов данных OWL использует схему XML.

OWL: язык описания онтологий

Характеристики свойств:

- Симметричность: $\forall x, y$ и свойства $R: R(x, y) \Rightarrow R(y, x)$.
- Транзитивность: $\forall x, y, z$ и свойства $R: R(x, y) \wedge R(y, z) \Rightarrow R(x, z)$.
- Функциональность: $\forall x, y, z$ и свойства $R: R(x, y) \wedge R(x, z) \Rightarrow y = z$.

Свойством функциональности обладает дата рождения человека, у каждого человека она единственна.

$\forall x, y$ «Для любых», «Для всех» $\Rightarrow$ «влечёт»

Рисунок 4 – характеристика свойств OWL

Проекты Семантического Веба

Дублинское ядро (Dublin Core) появилось раньше RDF. Теперь это просто аннотации (метаданные) к любым объектам, записанным на RDF с помощью онтологии DC.

Цель DC —установить единый формат метаданных для облегчения поиска по автору, названию, году выпуска и т. д.

Множество метаданных Дублинского ядра (Dublin Core Metadata Element Set (DCMES)) состоит из 15 элементов: Title, Creator, Subject, Description, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage, Rights.

Дублинское ядро (Dublin Core)

Dublin Core Metadata Element Set "simple Dublin Core"<http://purl.org/dc/elements/1.1/>.

Contributor, coverage, creator, date, description, format, identifier, language, publisher, relation, rights, source, subject, title, type.

Квалифицированный Dublin Core включал три дополнительных элемента (аудитория, происхождение и правообладатель), а также группу уточнений элементов (также называемых квалификаторами), которые могли бы уточнить семантику элементов таким образом, чтобы быть полезными в обнаружение ресурсов.

Дублинское ядро (Dublin Core)

С 2012 года оба (простой и квалифицированный DC) были включены в состав DCMИ метаданных как единый набор терминов с использованием Resource Description Framework (RDF).

Полный набор элементов находится в пространстве имен <http://purl.org/dc/terms/>.

Поскольку определение терминов часто содержит области определения и значений, которые могут не быть совместимы с до-RDF определениями, используемыми для исходных 15 элементов Dublin Core, сохранилось и старое пространство имен.

Пример

```
<?xml version="1.0"?>
<metadata
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xmlns:dc=http://purl.org/dc/elements/1.1/
xmlns:dcterms="http://purl.org/dc/terms/">
  <dc:title>Automated Theorem Proving</dc:title>
  <dc:creator>Mantsivoda Andrei</dc:creator>
  <dc:subject xsi:type=dcterms:UDC>681.3</dc:subject>
  <dc:date> 2004-04-04</dc:date>
  <dc:type>Article</dc:type>
  <dc:identifier>http://andrei.baikal.ru/atp2004</dc:identifier>
</metadata>
```

Выводы

Семантический Веб—снабжение Интернет страниц описаниями, которые понятны компьютерам. Описания пишутся на языке RDF со ссылками на онтологии, построенные с помощью языка OWL.

Отдельные подпроекты Семантического Веба имеют самостоятельное значение: FOAF, DC, RSS.

В полную силу Семантический Веб будет реализован тогда, когда люди создадут множество программ, которые, знакомясь с содержимым Сети из различных источников, обрабатывают полученную информацию и обмениваются результатами с другими программами.

Выводы

Эффективность таких программных агентов будет расти экспоненциально по мере увеличения количества доступного машинно-воспринимаемого веб-контента и автоматизированных сервисов (включая других агентов).

Выводы

Общее определение целей будущего веба выглядит очень заманчиво и амбициозно. Семантический веб-это не технология, а идеология. Для Бернерса-Ли Семантический веб - это следующая ступень эволюции Интернета. Нынешний веб нацелен исключительно на предоставление информации людям и может эффективно читаться только людьми. "Вместо этого Семантический веб разрабатывает языки для выражения информации в форме, доступной для машинной обработки". То есть Семантический веб - это веб не для людей, а для машин, это комплекс технологий, позволяющих машинам лучше разбираться в Интернете, что, в свою очередь, сделает веб более полезным для людей.

Контрольные вопросы

1. Кем впервые был введён термин «семантическая паутина»?
2. Что такое семантическая паутина (Semantic Web)?
3. Что такое Веб?
4. В чем заключается главная трудность при построении семантической сети?
5. Какие варианты в использовании Семантического Веба можно выделить?
6. Что такое синтаксис, семантика?

Контрольные вопросы

7. Чем являются классы OWL и что они обычно образуют?
8. OWL поддерживает шесть основных способов описания классов. Назовите их.
9. Назовите характеристики свойств OWL.
10. Что является основной целью дублинского ядра?
11. Когда в полную силу будет реализован Семантический Веб?
12. От чего зависит эффективность?

Источники

1. Тим Бернерс-Ли, Джеймс Хендлер и Ора Лассила.
Семантический Веб
http://ezolin.pisem.net/logic/semantic_web_rus.html
2. Joshua Tauberer. What Is RDF?
<http://www.xml.com/pub/a/2001/01/24/rdf.html>
3. Рекомендация W3C. Перевод Дмитрия Щербины. OWL,
язык веб-онтологий. Руководство
http://sherdim.rsu.ru/pts/semantic_web/REC-owl-guide-20040210_ru.html
4. А.В. Манцивода. Система мета описаний Dublin Core
<http://teacode.com/concept/eor/dc.html>
5. Страница курса <http://logic.pdmi.ras.ru/yura/internet.html>

АВТОМАТИЗИРОВА ННАЯ ОБРАБОТКА ЕСТЕСТВЕННОГО ЗЫКА.

NLP: начало

- возникло в конце 60 -х гг;
- развивалось в рамках дисциплины «искусственный интеллект» ;
- АОЕЯ — разработка методов, технологий и конкретных систем, обеспечивающих общение человека с ЭВМ на естественном или ограниченном естественном языке.

Пути решения проблемы организации взаимодействия

- адаптация языков программирования и операционных систем к конечному пользователю;
- разработка систем взаимодействия с ЭВМ на естественном языке или каком-то его ограниченном варианте.

Компьютерная лингвистика

- междисциплинарная область, которая возникла на стыке таких наук, как лингвистика, математика, информатика и искусственный интеллект.

Области лингвистики

- фонология — изучает звуки речи и правила их соединения при формировании речи;
- морфология — занимается внутренней структурой и внешней формой слов речи, включая части речи и их категории;
- синтаксис — изучает структуру предложений, правила сочетаемости и порядка следования слов в предложении, а также общие его свойства как единицы языка. · создания словарей.

Области лингвистики

- семантика и прагматика — тесно связанные области: семантика занимается смыслом слов, предложений и других единиц речи, а прагматика — особенностями выражения этого смысла в связи с конкретными целями общения;
- лексикография описывает лексикон конкретного ЕЯ — его отдельные слова, их грамматические и семантические свойства, а также методы создания словарей.

ИСКУССТВЕННЫЙ ИНТЕЛЕКТ

- задача ИИ – компьютерное моделирование интеллектуальных функций;
- первая известная программа ИИ по обработке ЕЯ – Система Т. Винограда (70-е годы).

ЕЯ

- ЕЯ – сложная система знаков, возникшая для обмена информацией в процессе человеческой деятельности и постоянно изменяющаяся вместе с ней

Особенности ЕЯ

- разнообразие языков и языковые универсалии;
- две стороны знака: означаемое – означающее;
- сложность системы ЕЯ;
- взаимосвязь всех уровней;
- нестандартная сочетаемость (синтактика) единиц ЕЯ на всех уровнях;
- большая системность (число уровней);
- асимметрия связи единиц и выражаемых ими смыслов: полисемия, синонимия, омонимия.

Сложности ЕЯ

- комбинаторная система яз. знаков;
- многоуровневость системы ЕЯ;
- каждый уровень (подсистема) – правила сочетания знаков;
- взаимосвязь уровней.

Уровни ЕЯ

1.Фонологический: звуки (фонемы)/ буквы – незначащие единицы , средство различения др. единиц.

2.Морфологический – слова (словоформы).

- подуровень морфем.

3.Синтаксический – предложения (фразы) ЕЯ.

- подуровень словосочетаний;

- надуровень сверхфразовых единств ($\approx$ абзацев) – предложений, объединяющихся по смыслу.

Задача морфологического синтеза

- морфологический анализ — это получение леммы или основы (псевдоосновы) заданного токена, а при необходимости, морфологических параметров;
- морфологический синтез — это генерация нужной словоформы слова или всей его парадигмы по нормальной форме (или основе) и морфологическим характеристикам.

ДОПОЛНИТЕЛЬНЫЕ УРОВНИ ЕЯ

- семантический : набор элементарных единиц – сем;
- лексический : множество лексем (лексикон);
- дискурсивный (уровень связного текста): схематические структуры текстов (патентные формулы, деловые письма и т.п.).

Сложность ЕЯ

- графематический анализ;
- морфологический анализ;
- постморфологический анализ:
разрешение морфологической омонимии.

Различие морфологических модулей

- результату работы — лемма или основа с набором морфологических характеристик (род, число, падеж, вид, лицо и т.п.) заданной словоформы;
- методу анализа — с опорой на словарь словоформ языка или на словарь основ, либо же бессловарный метод;
- возможности обработки словоформы лексемы, не включенной в словарь.

Функциональные возможности морфологических процессов

- выполняющие только лемматизацию или стемминг;
- определяющие часть речи;
- осуществляющие полный морфологический анализ, т. е. лемматизацию и определение всех морфологических характеристик словоформы;
- осуществляющие морфемный анализ, выделяющие морфы, входящие в состав слова.

Виды систем

- Система Диалинг–АОТ;
- Система TreeTagger;
- Система Pymorphy2;
- Система Snowball;
- Система MyStem.

Виды систем

1. Система Диалинг–АОТ

- включает в себя практически все этапы автоматического анализа текстов на ЕЯ, в том числе и морфологический;
- лексической основой служит словарь Зализняка;
- включая нахождение леммы и морфологических характеристик словоформы, а также синтез словоформ.

Виды систем

2. Система TreeTagger

- система для определения частей речи слов с возможностью настройки на любой естественный язык при наличии словаря и размеченного корпуса;
- позволяет определять части речи слов и другие морфологические характеристики, а также их нормальную форму;
- базируется на словарной морфологии и использует словарь английского языка из проекта Penn TreeBank.

Виды систем

3. Система Rymorphy2

- морфологический процессор с открытым исходным кодом, предоставляет все функции полного морфологического анализа и синтеза словоформ;
- базируется на словарной морфологии и использует словарные данные проекта OpenCorpora;
- в процессе морфологического синтеза, по исходной словоформе и тегам выполняется поиск нормальной формы слова, а затем перебор всех возможных пар в найденной лексеме, пока не будет найдена пара с заданными морфологическими тегами.

Виды систем

4. Система Snowball

- использует систему суффиксов и окончаний для предсказания части речи и грамматических параметров;
- реализуется на языке программирования в виде большого количества условных операторов, анализирующих самый длинный постфикс и его контекст.

Виды систем

5. Система MyStem

- морфологический анализатор, разработанный компанией Яндекс;
- базируется на словаре НКРЯ;
- производит разрешение морфологической омонимии и делает разбор несловарных словоформ.

Сравнение систем

Система	АОТ	MyStem	TreeTagger	Pymorphy2
Открытые исходные коды	да	нет	нет	да
Скорость слов в секунду	60-90 тыс.	100-120 тыс.	20-25 тыс.	80-100 тыс.
Подключение словарей	нет	да	да	нет
Объем словаря, тыс.слов	160	>250	210	250

"Таблица 1 - Характеристики морфологических процессоров"

Сложность ЕЯ

- предсинтаксис - сегментация текста на предложения;
- синтаксический анализ предложений;
- семантический и прагматический анализ.

Формализмы для предоставления семантики

- формулы исчисления предикатов, выражающие свойства, состояния, процессы, действия и отношения;
- семантические сети — размеченные графы, в которых вершины соответствуют понятиям, а дуги — отношениям между ними.

Особенности именно моделей КЛ

- формальность и, в конечном счете, алгоритмизируемость;
- функциональность (воспроизведение функций языка как «черного ящика», без построения точной модели синтеза и анализа речи человеком);
- опора на лингвистические ресурсы; • экспериментальная обоснованность, предполагающая тестирование модели на разных текстах.

Сложность при обработке текстов

- полисемия — наличие у одной единицы языка нескольких связанных между собой значений;
- синонимия — полное или частичное совпадение значений разных единиц;
- омонимия — совпадение по форме двух разных по смыслу единиц (в отличие от полисемии нет смысловой связи между совпавшими по форме единицами).

Сложность при обработке текстов

Виды омонимии

- лексическая омонимия означает одинаково звучащие и пишущиеся слова, не имеющие общих элементов смысла;
- морфологическая омонимия — совпадение форм одного и того же;
- лексико-морфологическая омонимия возникает при совпадении словоформ двух разных лексем;
- синтаксическая омонимия означает неоднозначность синтаксической структуры, что приводит к нескольким интерпретациям.

Сложность при обработке текстов

Классы омонимии

- неоднозначные по параметрам — в анализе присутствуют словоформы с различными множествами грамматических параметров, но совпадающей леммой и частью речи;
- неоднозначные по части речи — в анализе присутствуют словоформы, совпадающие по лемме, но отличающиеся по части речи. Так как части речи не совпадают, то наборы параметров у словоформ также будут отличаться. В связи с этим сравнение параметров проводиться не может.

Сложность при обработке текстов

Классы омонимии

- неоднозначные по лемме — в анализе присутствуют словоформы, отличающиеся по лемме, но имеющие одинаковую часть речи. Здесь параметры могут как совпадать, так и отличаться;
- неоднозначные по части речи и лемме — в анализе присутствуют словоформы, отличающиеся как по лемме, так и по части речи. Сравнение параметров здесь также проводиться не может.

Подходы для создания лингвистических процессов

- основанный на правилах (rulebased);
- инженерный, и основанный на машинном обучении (machine learning).

Признаковой модели

- BOW (bag of words — мешок слов);
- статистическая языковая модель (Language Model).

Архитектура систем NLP

- блок анализа речевого сообщения пользователя;
- блок интерпретации сообщения;
- блок порождения смысла ответа;
- блок синтеза поверхностной структуры высказывания;
- диалоговый компонент.

Архитектура систем NLP

Блок анализа речевого сообщения пользователя

- определение информации, которую следует передать пользователю;
- предполагаемое членение информации на «порции», соответствующие предложению;
- определение последовательности «порций» смысла;
- построение семантического представления отдельных предложений.

Архитектура систем NLP

Блок синтеза поверхностной структуры высказывания

- упаковка семантического представления высказывания в синтаксические структуры предложения;
- здесь играют существенную роль категории коммуникативной организации смысла высказывания — тема, рема, данное, новое.

Архитектура систем NLP

Задачи обработки текстов

- информационный поиск;
- извлечение информации;
- вопросно-ответные системы;
- диалоговые системы;
- машинный перевод;
- классификация и кластеризация.

Прикладные задачи NLP

- определение языка сообщений;
- идентификация диктора;
- определение эмоционального и физического состояния человека по его голосу;
- шумоочистка;
- разделение дикторов ;
- music spotting.

Современные речевые технологии

- распознавание речи;
- синтез речи по тексту;
- выделение ключевых слов в слитной речи.

Современные речевые технологии



"Рисунок 1 – Голосовой помощник Siri"

Siri

- англ. Speech Interpretation and Recognition Interface;
- персональный помощник и вопросно-ответная система, адаптированная для iPhone OS;
- использует обработку естественной речи, чтобы отвечать на вопросы и давать рекомендации;
- приспособливается к каждому пользователю индивидуально, изучая его предпочтения в течение долгого времени.

Диалоговые системы

Пользователь: Мне нужен самолет из СПб до Бостона
прибытие в 10 утра.

Компьютер: На какое число?

Пользователь: На завтра.

Компьютер: Доступные рейсы на завтра...

<список рейсов>

"Рисунок 2 – Пример работы диалоговой системы"

Вопросы по презентации

1. Когда возникло NLP?
2. Что такое компьютерная лингвистика?
3. Назовите несколько особенностей естественного языка?
4. Назовите классы омонимии?
5. Назовите виды омонимии?
6. Назовите задачи морфологического синтеза?
7. Назовите подходы для создания лингвистических процессов?
8. Архитектура систем NLP?
9. Назовите несколько задач для обработки текстов?
10. Назовите современные речевые технологии?

Спасибо за внимание