

# Лекция 1

1. ОБЩЕЕ ПРЕДСТАВЛЕНИЕ ОБ ИИ
2. ЦЕЛИ ПОСТРОЕНИЯ ИИ, ВОЗМОЖНО ЛИ РЕШЕНИЕ ПРОБЛЕМЫ ИИ
3. ИСТОРИЯ ПРОБЛЕМЫ
4. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ
  - ЕИ & ИИ. Соотношение, возможность установления соотношения.
  - Тест Тьюринга. Основные особенности.
  - Другие критерии ИИ.
  - ИИ и другие области научного знания.

## Общее представление об ИИ

Существуют различные подходы к определению понятия ИИ, целей и задач этой науки. Приведем несколько различных взглядов на проблематику ИИ, которые принадлежат ученым, стоявшим у истоков ИИ.

- Искусственный интеллект – это наука о концепциях, позволяющих вычислительным машинам делать такие вещи, которые у людей выглядят разумными. Центральные задачи ИИ состоят в том, чтобы сделать вычислительные машины более полезными и понять принципы, лежащие в основе интеллекта. (П. Уинстон)
- Искусственный интеллект ставит перед собой задачу построения теории интеллекта, базирующейся на обработке информации. (Н. Нильсон).
- Какова цель искусственного разума? По нашему мнению, эта цель состоит в создании таких программ для ЭВМ, поведение которых мы бы назвали разумным, если бы обнаружили его у людей. (Д. Фельдман).

И еще один взгляд на эту проблематику, принадлежащий Д. Люгеру:

- Искусственный интеллект можно определить, как область компьютерной науки (информатики), занимающуюся автоматизацией разумного поведения.

В приведенных определениях можно выделить один существенный недостаток: они **не конструктивны**. Из них никак не следует какими задачами нужно заниматься в ИИ, какой круг проблем интересует при решении таких задач и т.п. В силу не конструктивности даже не видно, что ИИ ассоциируется с какой-то наукой. Чтобы сделать представление об ИИ более конструктивным необходимо определить, как возникает само понятие ИИ.

Для этого обратимся к следующей схеме, на которой представление об интеллекте связывается с определенным уровнем сознания. Всего таких уровней существует три:



Уровня сознания не существует без носителя. С интеллектом в качестве его носителя связывают человека, а его интеллект называют **естественным** (ЕИ).

Какова роль ЕИ? Чтобы ответить на этот вопрос обратимся к следующей универсальной схеме объектного представления об окружающем мире (см. рис.1а):

Кругами на этой схеме обозначены объекты. Все объекты между собой связаны – напрямую или транзитивно. Интересно отметить, что если объект (со штриховкой) не связан ни с каким другим, то для других объектов его и не существует. Связь на данной схеме означает некоторое взаимодействие. **Любое такое взаимодействие может быть описано на информационном уровне**, хотя подчеркнем, что необязательно на информационном уровне оно должно быть организовано (объектами, природой, человеком). Это могут быть энергетические взаимодействия и проч. В качестве части объектов такой схемы при определенном масштабировании можно рассматривать носителей различной уровней сознания. В этом смысле схема может иметь вид (см. рис. 1б).

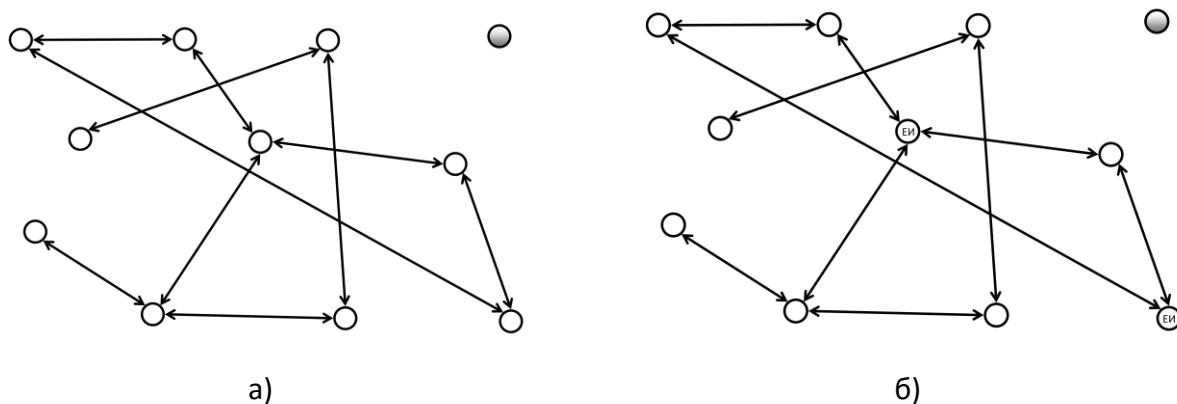
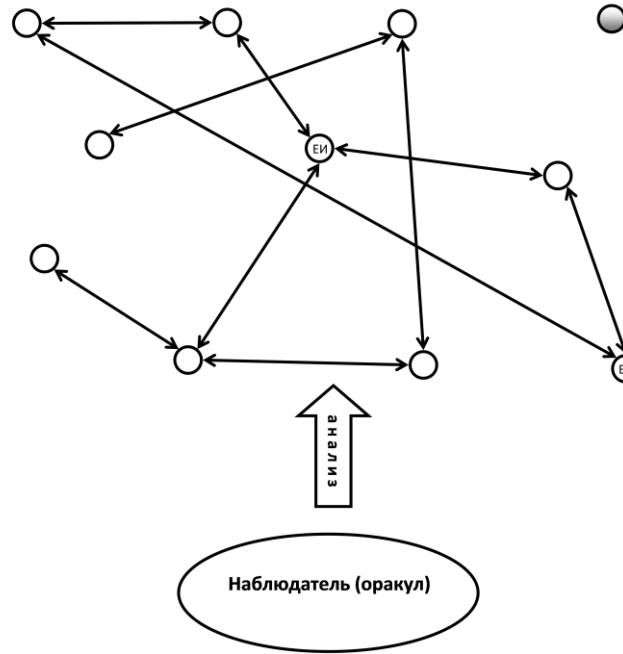


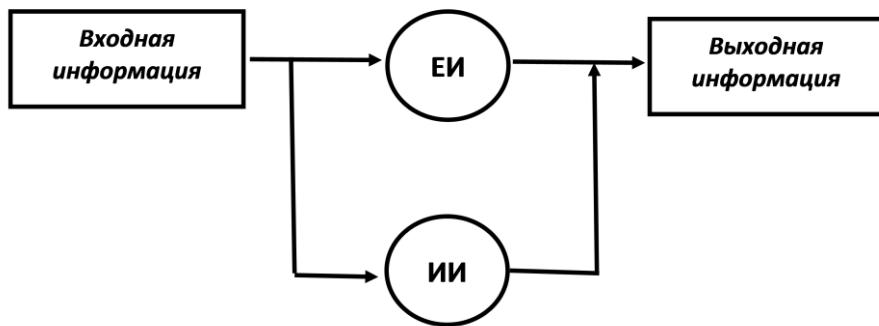
Рисунок 1 – объектная схема реальной действительности

Т.е. ЕИ участвует во взаимодействии с другими объектами реального мира и характер этого взаимодействия может быть различным. Но обязательно в этом взаимодействии должны быть черты, присущие всему уровню сознания (интеллект) в целом.

Можно ли отличить объекты, которые являются носителями ЕИ, от других объектов? Да, но для этого необходим сторонний наблюдатель (оракул), который по результатам анализа взаимодействия конкретного объекта с другими может прийти к заключению, что данный объект является носителем ЕИ. Наблюдатель, в общем-то, должен быть отделен от приведенной схемы (не видеть объекты непосредственно). Эта простая идея лежит в основе известного теста Тьюринга, который используется при анализе интеллектуальности программ, роботов и проч. Очень условно эту схему можно представить следующим образом:



В целом картина не поменяется, если некоторые объекты ЕИ на схеме заместить искусственными (т.е. созданными с участием ЕИ и других объектов). Если наблюдатель не заметит изменения характера взаимодействия, то можно сказать, что такие объекты наделены **искусственным интеллектом**. С информационной точки зрения, такие объекты должны быть моделью объектов ЕИ. В информационном смысле схему построения модели можно представить в виде:



Подводя черту, можно сказать, что ИИ – это модель ЕИ. В силу специфики специальности в дальнейшем будем рассматривать информационные аспекты модели. Характер, свойства этой модели определяются исходя из универсальной модели взаимодействия объектов в окружающей нас действительности. С точки зрения наблюдателя подмена объекта ЕИ его моделью не должна приводить к изменениям общей картины взаимодействий.

### **Цели построения ИИ, возможно ли решение проблемы ИИ**

В проблематике ИИ принято выделять следующие цели, которые определяют и направление возможного развития ИИ в целом.

- первую из них часто называют **информационной**, или **эвристической**. Эта цель заключается в создании программ для ЭВМ, с помощью которых удалось бы автоматизировать такие виды человеческой деятельности, которые традиционно считаются интеллектуальными. При этом, как будут устроены подобные программы, насколько будут близки те

способы, которыми они достигают поставленной цели по сравнению с человеческими, абсолютно не интересно. Важен лишь конечный результат, его совпадение с тем, который получает человек при решении той же задачи;

- вторая цель называется **бионической**. Состоит в использовании программ ИИ для объяснения процессов, протекающих у человека, когда он решает какие-либо задачи. В этом случае программы должны имитировать процесс получения результата человеком, помогать постигать эти процессы;
- третья цель называется **эволюционной**. Смысл ее состоит в том, что программы только тогда станут по-настоящему интеллектуальными, когда приобретут способность обучаться тому, чего раньше они абсолютно не умели делать.

Во всех случаях речь идет о создании программ. Поэтому представляет интерес методология создания таких программ, приемы их построения. Если такие программы содержат в себе некоторые знания о том, как решается конкретная задача, то должны существовать знания более высокого уровня (**метазнания**) о том, как строить программы, умеющие работать с конкретными знаниями о задаче.

Общие принципы построения подобных программ обсуждались и формулировались не только специалистами по решению задач на ЭВМ (прикладная математика, информатика). Но и психологами, нейрофизиологами, лингвистами, философами, логиками. Соответственно, существуют различные аспекты проблематики ИИ.

При рассмотрении проблемы ИИ возникает вопрос: а может ли быть в принципе построена искомая программа (модель, технология)? Существует ли возможность моделирования мышления человека. Ответы на этот вопрос могут быть разные, в зависимости от многих обстоятельств.

Далее приводятся несколько соображений, которые могут быть положены в основу положительного ответа.

- *Первое соображение является холастическим*, и основывается на непротиворечивости ИИ и Библии. По-видимому, даже люди далекие от религии, знают слова священного писания: "И создал Господь человека по образу и подобию своему ...". Исходя из этих слов, мы можем заключить, что, поскольку Господь, во-первых, создал нас, а во-вторых, мы по своей сути подобны ему, то мы вполне можем создать кого-то по образу и подобию человека.
- *Второе соображение основано на биологии человека*, возможности создания нового разума биологическим путем. Наблюдая за детьми, мы видим, что большую часть знаний они приобретают путем обучения, а не как заложенную в них заранее программу.
- *Третье соображение имеет отношение к области достижений ИИ*. То, что раньше казалось вершиной человеческого творчества — игра в шахматы, шашки, распознавание зрительных и звуковых образов, синтез новых технических решений, на практике оказалось не таким уж сложным делом. Теперь речь идет не о возможности либо невозможности реализации перечисленных выше задач, а о построении оптимальных алгоритмов. Зачастую указанные задачи даже не относят к проблемам ИИ.
- *Четвертое соображение также имеет отношение к биологии*. Дело в том, что с проблемой воспроизведения мышления тесно связана проблема самовоспроизведения. Способность к самовоспроизведению долгое время считалась прерогативой живых организмов. Однако некоторые явления, происходящие в неживой природе (например, рост кристаллов, синтез сложных молекул копированием), очень похожи на самовоспроизведение. В начале 50-х годов Д. Нейман занялся изучением самовоспроизведения и заложил основы математической теории "самовоспроизводящихся автоматов". Также он доказал возможность их создания. Существуют также различные нестрогие обоснования возможности самовоспроизведения. К их числу можно отнести все, что связано с функционированием компьютерных вирусов.

- Пятое соображение имеет отношение к теории алгоритмов. Принципиальная возможность автоматизации решения интеллектуальных задач с помощью ЭВМ обеспечивается свойством алгоритмической универсальности. Алгоритмическая универсальность ЭВМ означает, что на них можно программно реализовывать (т. е. представить в виде машинной программы) любые алгоритмы преобразования информации, — будь то вычислительные алгоритмы, алгоритмы управления, поиска доказательства теорем или композиции мелодий. При этом мы имеем в виду, что процессы, порождаемые этими алгоритмами, являются потенциально осуществимыми, т. е. что они осуществимы в результате конечного числа элементарных операций. Практическая осуществимость алгоритмов зависит от имеющихся в нашем распоряжении средств, которые могут меняться с развитием техники. Так, в связи с появлением быстродействующих ЭВМ стали практически осуществимыми и такие алгоритмы, которые ранее были только потенциально осуществимыми.

**Замечание.** Свойство алгоритмической универсальности заключается не только в том, что для всех известных алгоритмов оказывается возможной их программная реализация на ЭВМ. Содержание этого свойства имеет отношение и к будущим возможным реализациям. Всякий раз, когда в будущем какое-либо предписание будет признано алгоритмом, то независимо от того, в какой форме и какими средствами это предписание будет первоначально выражено, его можно будет задать также в виде машинной программы.

Существуют соображения, которые могут быть положены в основу и отрицательного ответа. Этих соображений не меньше, чем лежащих в основе положительного ответа. Мы ограничимся только одним, достаточно очевидным и связанным с рассмотренным выше свойством алгоритмической универсальности. Из этого свойства нельзя сделать вывод, что вычислительные машины и роботы могут в принципе решать любые задачи. Анализ оснований математики привел к доказательству существования таких задач, для которых невозможен алгоритм, решающий все задачи данного типа. Этот факт способствует лучшему пониманию того, что могут делать машины и чего они не могут сделать. Действительно, утверждение об алгоритмической неразрешимости некоторого класса задач является не просто признанием того, что такой алгоритм нам не известен и никем еще не найден. Такое утверждение представляет собой одновременно и прогноз на будущее. Смысл прогноза в том, что подобного рода алгоритмами нами в принципе не может быть построен.

Как же действует человек при решении таких задач? Чаще всего он просто игнорирует их, что не мешает ему жить дальше. Другим вариантом является сужение условий универсальности задачи, когда она решается только для определенного подмножества исходных условий. Еще один вариант заключается в том, что человек методом "проб и ошибок" расширяет множество доступных для себя элементарных операций (например, создает новые материалы, открывает новые месторождения или типы ядерных реакций). Доступны ли такие варианты ЭВМ и соответствующим программам?

## История проблемы

Идея создания искусственного подобия человеческого разума для решения сложных задач и моделирования мыслительной деятельности известна с древнейших времен. Впервые ее выразил Р.Луллий, который еще в XIV в. пытался создать машину для решения различных задач на основе всеобщей классификации понятий. В XVIII в. Г. Лейбниц и Р. Декарт независимо друг от друга развили эту идею, предложив универсальные языки классификации всех наук. Эти идеи легли в основу теоретических разработок в области создания искусственного интеллекта.

Развитие искусственного интеллекта как научного направления стало возможным только после создания ЭВМ. Это произошло в 40-х гг. ХХ в. В это же время Н. Винер опубликовал свои основополагающие работы по новой науке — **кибернетике**.

Термин *искусственный интеллект* (*artificial intelligence*) предложен в 1956 г. на семинаре с аналогичным названием в Стэнфордском университете (США). Семинар был посвящен разработке методов решения логических, а не вычислительных задач. Вскоре после признания искусственного интеллекта самостоятельной отраслью науки произошло разделение на два основных направления: **нейрокибернетику и кибернетику "черного ящика"**.

Основную идею **нейрокибернетики** можно сформулировать следующим образом. Единственный объект, способный мыслить, — это человеческий мозг. Поэтому любое "мыслящее" устройство должно каким-то образом воспроизводить его структуру. Таким образом, нейрокибернетика ориентирована на аппаратное моделирование структур, подобных структуре мозга. Физиологами давно установлено, что основой человеческого мозга является большое количество (до  $10^{21}$ ) связанных между собой и взаимодействующих нервных клеток — нейронов. Поэтому усилия нейрокибернетики были сосредоточены на создании элементов, аналогичных нейронам, и их объединении в функционирующие системы. Эти системы принято называть **нейронными сетями**, или **нейросетями**.

Первые нейросети были созданы в конце 50-х гг. американскими учеными Г. Розенблаттом и П. Мак-Каллоком. Это были попытки создать системы, моделирующие человеческий глаз и его взаимодействие с мозгом. Устройство, созданное ими, получило название **персептрона**. Оно умело различать буквы алфавита, но было чувствительно к их написанию, например, буквы **A**, **A** и **А** для этого устройства были тремя разными знаками. Постепенно в 70-80 гг. количество работ по этому направлению искусственного интеллекта стало снижаться. Слишком неутешительны оказались первые результаты. Авторы объясняли неудачи малой памятью и низким быстродействием существующих в то время компьютеров.

Однако в середине 80-х гг. в Японии в рамках проекта разработки компьютера V поколения, основанного на знаниях, был создан нейрокомпьютер. К этому времени ограничения по памяти и быстродействию были практически сняты. Появились **транспьютеры** — параллельные компьютеры с большим количеством процессоров. От транспьютеров был один шаг до **нейрокомпьютеров**, моделирующих структуру мозга человека.

В основу **кибернетики "черного ящика"** положен принцип, противоположный нейрокибернетике. Не имеет значения, как устроено "мыслящее" устройство. Главное, чтобы на заданные входные воздействия оно реагировало так же, как человеческий мозг.

Это направление искусственного интеллекта было ориентировано на поиски алгоритмов решения интеллектуальных задач на существующих моделях компьютеров. В 1956 -1963 гг. велись интенсивные поиски моделей и алгоритма человеческого мышления и разработка первых программ. Оказалось, что ни одна из существующих наук — философия, психология, лингвистика — не может предложить такого алгоритма. Тогда кибернетики предложили создать собственные модели. Были созданы и опробованы различные подходы.

В конце 50-х гг. родилась модель **лабиринтного поиска**. Этот подход представляет задачу как некоторый граф, отражающий пространство состояний, и в этом графе проводится поиск оптимального пути от входных данных к результирующим. Была проделана большая работа по разработке этой модели, но в решении практических задач идея большого распространения не получила.

Начало 60-х гг.— эпоха **эвристического программирования**. Эвристическое программирование — разработка стратегий действий на основе известных, заранее заданных эвристик. В 1963 - 1970 гг. к решению задач стали подключать методы **математической логики**. На основе **метода резолюций**, позволившего автоматически доказывать теоремы при наличии набора исходных аксиом, в 1973 г. создается язык **Пролог**.

В настоящее время наблюдаются тенденции к объединению этих направлений вновь в единое целое.

Существенный прорыв в практических приложениях искусственного интеллекта произошел в середине 70-х гг., когда на смену поискам универсального алгоритма мышления пришла идея моделировать конкретные знания специалистов-экспертов. В США появились первые коммерческие системы, основанные на знаниях, или экспертные системы. Пришел новый подход к решению задач искусственного интеллекта — *представление знаний*. Созданы MYCIN и DENDRAL — ставшие уже классическими экспертные системы для медицины и химии. Объявлено несколько глобальных программ развития интеллектуальных технологий — ESPRIT (Европейский Союз), DARPA (министерство обороны США), японский проект машин V поколения.

Начиная с середины 80-х гг. происходит коммерциализация искусственного интеллекта. Растут ежегодные капиталовложения, создаются промышленные экспертные системы. Растет интерес к самообучающимся системам, роботам.

### **Дополнительные вопросы**

1. ЕИ & ИИ. Соотношение, возможность установления соотношения.
2. Тест Тьюринга. Основные особенности.
3. Другие критерии ИИ.
4. ИИ и другие области научного знания.

1. ОБЛАСТИ ПРИМЕНЕНИЯ И КОНКРЕТНЫЕ ЗАДАЧИ ИИ.
2. ПРЕДСТАВЛЕНИЕ О ПРОБЛЕМЕ ИИ. ПРИМЕРЫ ЗАДАЧ.
3. ОБЩАЯ СХЕМА РЕШЕНИЯ ЗАДАЧ.
4. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - Задачи. Некоторые свойства задач.
  - Алгоритмы. Связь с задачами и свойства.
  - Примеры задач и их связь с ИИ.

### **Области применения и конкретные задачи ИИ.**

Традиционно к области ИИ относят следующие теоретические и прикладные направления:

- доказательство теорем;
- игры;
- распознавание образов;
- принятие решений;
- обработка данных на естественном языке.

Каждое из перечисленных направлений состоит из ряда задач, имеющих в основном прикладное значение. Рассмотрим эти задачи более подробно.

Доказательство теорем. Смысл направления заключается в поиске доказательства (или опровержения) для некоторого математического утверждения. Цель – в автоматизации поиска. Прикладное значение состоит в том, что логика мышления человека и логика доказательства очень похожи. В некоторых случаях, при решении прикладных задач логика может быть дедуктивной.

Игры. Смысл направления заключается в поиске методов манипуляции символьной информацией. Прикладное значение состоит в систематизации указанных методов. Конкретные задачи – шахматы, шашки, карточные игры, всевозможные ситуационные и ролевые игры.

Распознавание образов. Смысл направления заключается в поиске методов принятия решений на основе прецедентности (опыта). Прикладное значение состоит в разработке методологии решения плохо формализованных задач. Задачи - распознавание изображений, визуализация картографической информации, медицинская диагностика и многие другие.

Принятие решений. Смысл направления заключается в поиске методов постановки, анализа и представления конкретных ситуаций, связанных с понятием решения задачи. Прикладное значение состоит в формализации понятия задачи, принципов формирования целей и т.п. Задачи – робототехника, всевозможные устройства.

Обработка данных на естественном языке. Смысл направления заключается в разработке методов анализа и синтеза текстов на ЕЯ, понимания (кодирования и интерпретация) ЕЯ - информации. Прикладное значение состоит в автоматизации обработки текстовой информации, возможности построения интерфейсов и т.п. Задачи – словари, системы перевода и т.п.

Рассмотренные направления и отдельные задачи, в действительности очень тесно связаны.

### **Представление о проблеме ИИ. Примеры задач.**

Общим для ЕИ и ИИ является наличие **задачи** и необходимость ее решения. Рассмотрим понятие задачи более подробно. Существуют различные аспекты этого понятия. Но т.к. речь идет об ИИ, то нас интересуют не содержательные аспекты, а те, которые соответствуют

формальному представлению о задаче в математике, прикладной математике и информатике.

*понятие задачи в математике, прикладной математике и информатике эквивалентно, но вместе с тем и не формализовано.*

**Примеры:**

**1. задача о поездах; задача поиска кратчайшего маршрута; задача построения интерфейса человек – компьютер.**

Общей во всех рассмотренных задачах является схема их решения. Эту схему в обобщенном виде можно представить с помощью следующих конструкций:

Дано: некоторая предметная область (ПО). В ПО определено пространство исходной информации и объекты в нем. Целью существования этих объектов являются некоторые, возможно выделенные, объекты в пространстве решений (ответов).

Требуется: установить связь, соотношение между объектами в пространствах исходной информации и решений.

**2. Задача распознавания свойства четности**

Пусть задано множество натуральных чисел  $\mathbb{N}$ . Известно, что это множество может быть разбито на подмножества четных и нечетных чисел. Обозначим эти подмножества через  $\mathbb{N}_q$  и  $\mathbb{N}_{nq}$  соответственно. По построению введенные подмножества удовлетворяют следующим простым условиям:

$$\mathbb{N}_q \cup \mathbb{N}_{nq} = \mathbb{N},$$

$$\mathbb{N}_q \cap \mathbb{N}_{nq} = \emptyset.$$

Всю известную нам информацию об разбиении  $\mathbb{N}$  на подмножества  $\mathbb{N}_q$  и  $\mathbb{N}_{nq}$ , обозначим через  $I_0(P_q, P_{nq}, \mathbb{N}_q, \mathbb{N}_{nq})$ . Для решения задачи требуется указать алгоритм (правило, метод) следующего вида:

$$\forall n \in \mathbb{N} \quad (A : n \times I_0(P_q, P_{nq}, \mathbb{N}_q, \mathbb{N}_{nq}) \rightarrow \text{результат}) .$$

Единственное условие, которое накладывается на результат), имеет следующий смысл: он должен допускать интерпретацию в терминах принадлежности подмножествам  $\mathbb{N}_q$ ,  $\mathbb{N}_{nq}$ .

**3. Задача распознавания арабских цифр.**

Задача очень проста и каждый, кто читает этот текст, точно понимает, как и когда он научился узнавать (распознавать) цифры. Поэтому, начнем с содержательной трактовки этой задачи. Для этого обратимся к рисунку 1 [<http://archive.ics.uci.edu/ml/>].

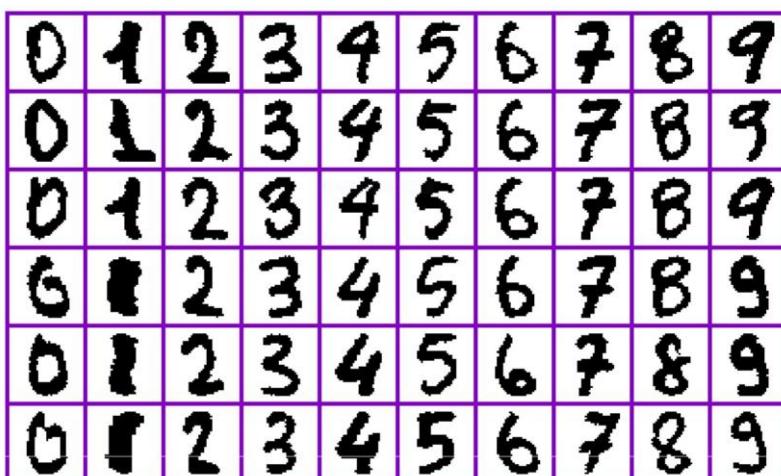


Рисунок 1 – Пример написания арабских цифр

Отметим, что цифры даже на содержательном уровне без труда интерпретируются в качестве объекта исследования в задаче распознавания арабских цифр. Поэтому проблем с формированием пространства  $X$ , которое содержит все возможные варианты записи арабских цифр, нет. Обратим также внимание на то, что таких цифр всего десять. Т.е. можно сказать, что в данном случае речь идет о десяти классах  $X_0, \dots, X_9$ . Цифры, записанные в столбцах приведенной на рисунке 1 матрицы, можно трактовать как обучающие подмножества  $X_0, \dots, X_9$ , а их объединение – в качестве подмножества (обучающей выборки)  $X^0$ .

Понятно, что подмножества  $X_0, \dots, X_9$  не исчерпываются приведенными на рисунке вариантами. Более того, каждый может предложить свои варианты продолжения или формирования подмножеств  $X_0, \dots, X_9$ . И вот здесь возникает очень интересный в методологическом смысле вопрос: как человек приходит в своей практике к формированию данных подмножеств? Или в более расширительной трактовке – почему и в результате каких процессов человек начинает без труда узнавать арабские цифры, допустимы ли ошибки и т.п.?

В некотором идеализированном смысле вариант ответа на сформулированные вопросы может быть получен в результате следующего рассуждения. Цифры мы учимся узнавать в школе. Для этого используются прописи в которых вначале человек долго тренируется записывать одну цифру  $i \in \{0, \dots, 9\}$ . Полученный массив записанных вариантов можно рассматривать в качестве соответствующего подмножества  $X_i^0$ . В результате у него образуется во многом механистическое представление о записи и узнавании этой цифры  $i$ . Затем берется другая цифра  $j \in \{0, \dots, 9\}, j \neq i$  и снова с помощью прописей осуществляется тренировка и формирование подмножества  $X_j^0$ . Этот несложный процесс продолжается до тех пор, пока список цифр не будет исчерпан и/или результат узнавания цифр не будет устраивать учителя.

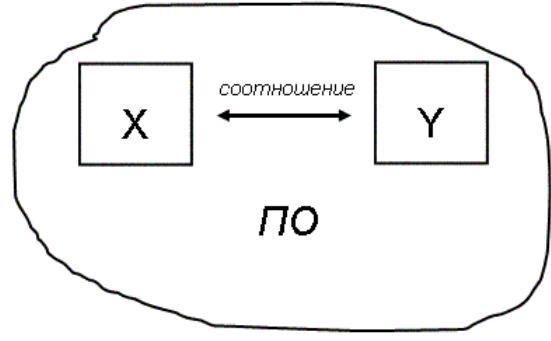
Понятно, что на этом процесс обучения узнаванию цифр может не завершаться, т.к. далее начинается то, что связано с эволюцией представления человека о реальном мире. Элементом эволюции может быть эмпирическое обобщение, интуиция и проч. Результат эволюции в данной задаче заключается в том, что практически никто при узнавании цифр не обращается к школьному опыту и обучающей выборке  $X^0$ , а число ошибок как минимум не увеличивается. Хотя любой такой процесс в качестве необходимого элемента (стартовой точки) всегда содержит некоторую обучающую выборку  $X^0$ , причем она у каждого может быть своя.

Обозначим, теперь через  $I_0(X^0, X_0, \dots, X_9)$  всю информацию, которую мы можем получить в рассматриваемой задаче с учетом всех оговорок, которые были сделаны выше. Требуется указать алгоритм (правило, метод) следующего вида:

$$\forall x \in X \ (A : x \times I_0(X^0, X_0, \dots, X_9) \rightarrow \text{результат}) .$$

По аналогии с задачей распознавания свойства четности, единственное условие, которое накладывается на результат, заключается в следующем: он должен допускать интерпретацию в терминах принадлежности классам  $X_0, \dots, X_9$ . ■

Обозначим пространство исходной информации через  $X = X_1 \times \dots \times X_n$ , а пространство решений – через  $Y = Y_1 \times \dots \times Y_m$ ,  $n, m \in \mathbb{N}$ . Тогда схематично задачу можно описать следующим образом:



Характер соотношения между объектами в пространствах  $X$  и  $Y$  может быть различным. Это может быть функция

$$f : X \rightarrow Y,$$

или, в общем, случае – обычное отношение в декартовом произведении  $X \times Y$ . Но во всех случаях оно должно быть вычисляемым. Пусть даже эта вычислимость будет основана не на точных алгоритмах, а на эвристических соображениях.

### Общая схема решения задач.

Проведенные выше построения относятся в основном к области ЕИ, т.к. они содержательны. В математике, прикладной математике и информатике принято оперировать формальными конструкциями. Решение задачи при этом осуществляется по следующей схеме.

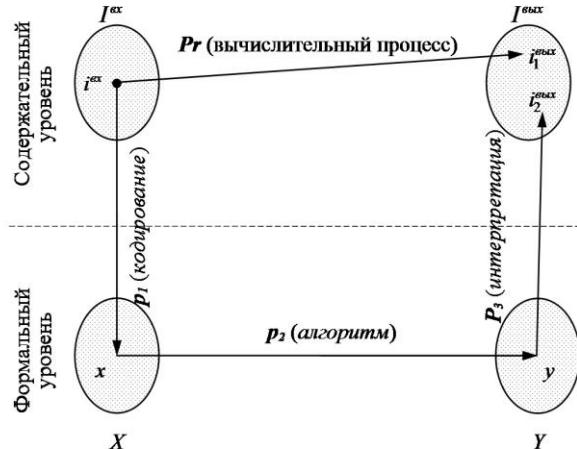


Рисунок 2 – Процесс решения задачи

Приведенная на рисунке 2 диаграмма типична для прикладной математики, т.к. она естественным образом связана с формализацией любой практической задачи. Процессы на рисунке можно представить

$$\Pr(i^{\alpha}) = i_1^{\alpha} \in I^{\alpha},$$

$$p_3 \circ p_2 \circ p_1(i^{\alpha}) = i_2^{\alpha} \in I^{\alpha}.$$

Из диаграммы следует, что основная проблема при решении любой задачи – это установление соотношения:

$$\Pr(i^{\alpha}) \leftrightarrow p_3 \circ p_2 \circ p_1(i^{\alpha}).$$

Чтобы подобная схема могла использоваться для решения конкретной прикладной задачи, необходимо ввести некоторые условия. Например, необходимо условиться, что ПО эквивалентна в каком-то смысле информации об объектах из  $X = X_1 \times \dots \times X_n$  и  $Y = Y_1 \times \dots \times Y_m$ . Должны быть определены также ограничения на характер зависимости между этими множествами. Но самое главное – должно быть определено, что является решением задачи. А если говорить о характеризации классов рассматриваемых задач, то наиболее интересным является вопрос о разрешимости задачи.

Существуют различные основания, по которым принято отличать характер решаемой задачи. Например:

По области получения результата. В этом смысле принято выделять **прямую** и **обратную** задачи.

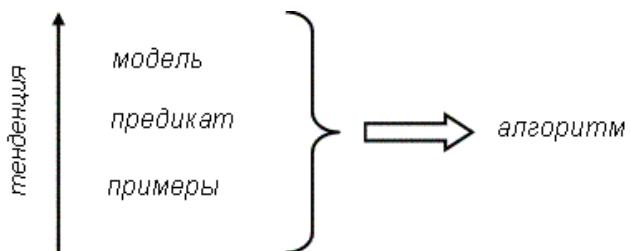
По способу построения модели. Непосредственно через модель отношения (функции) или через соответствующий предикат.

Как известно, каждой функции (отношению)  $f$  можно поставить в однозначное соответствие представляющий ее предикат

$$P_f(x, y) = \begin{cases} I, & \text{если } f(x) = y \\ L, & \text{иначе} \end{cases}$$

Строить можно как непосредственно  $f$ , так и  $P_f(x, y)$ . В общем случае, результаты могут быть различными, т.к. соответствие не взаимно однозначно.

По способу представления информации.



### Некоторые конкретные проблемы ИИ.

Вернемся к приведенной выше схеме решения задачи. Ясно, что эту схему на содержательном уровне можно дополнить человеком, вместо которого или для которого решается задача. На формальном же уровне дополнение будет представлять собой среду, в которой предполагается реализация и функционирование алгоритма. В нашем случае речь может идти о программах, компьютерах и т.д.

Теперь можно говорить о задачах трех типов:

- задачи содержательного уровня;
- задачи формального уровня;
- задачи взаимодействия содержательного и формального уровней.

### Дополнительные вопросы.

1. Задачи. Некоторые свойства задач.
2. Алгоритмы. Связь с задачами и свойства.
3. Примеры задач и их связь с ИИ.

1. ОБЩЕЕ ПРЕДСТАВЛЕНИЕ О ЗАДАЧЕ
2. ЗАДАЧА И ИНФОРМАЦИЯ
3. ДЕДУКТИВНЫЕ И ИНДУКТИВНЫЕ ЗАДАЧИ
4. ХАРАКТЕРИСТИЧЕСКИЕ СВОЙСТВА ЗАДАЧ ИИ

### Общее представление о задаче

Понятие задачи является плохо формализуемым. Между тем это понятие широко используется как в формальных теориях (математика, информатика и проч.), так и в обиходе. Часто это понятие подменяется еще одним очень близким ему по смыслу понятием – проблема. Почему понятие задачи играет такую важную роль, особенно в формальных теориях? Ответ здесь простой – любая теория (наука) однозначно характеризуется каким-либо многообразием однотипных задач и полностью ими исчерпывается. Например, математический анализ (в качестве науки или теории) объединяет задачи анализа функций, алгебра – анализ отношений. Для чего нужно объединять однотипные задачи? Во-первых, никто их не объединяет искусственно. Как правило, такое объединение становится возможным только на определенном этапе развития науки и свидетельствует, как правило, о “хорошем” уровне развития теории. Во-вторых, специализация задач всегда предпочтительнее. А такая специализация как раз и является следствием объединения однотипных задач. Кроме того, однотипность задач приводит к разработке и специальной методологии по следующей схеме:



Рисунок 1 – Структурная схема формирования науки

Методология, в свою очередь, используется для уточнения спектра задач и т.д. по циклу. Какова конечная цель таких исследований? Безусловно, вначале необходимо научиться решать те задачи, которые сформировали теорию. Но не менее важно – выстроить все исследования таким образом, чтобы была возможность отличать одну науку от другой. Конечно, это не значит, что все теории между собой не связаны. В основе упомянутых выше математического анализа и алгебры лежит одно фундаментальное математическое понятие – отношения. Поэтому эти теории связаны и по применяемому математическому аппарату, и по сущности результатов, по требованию к результатам и проч.

Обладает ли ИИ такими признаками, которые позволяют отличать задачи ИИ от остальных задач в области информатики? Вопрос не праздный, т.к. в случае положительного ответа по-

является возможность говорить об общей методологии ИИ, общих ответах на все вопросы в таких областях как нейронные сети, генетические алгоритмы. В конечном итоге, наличие общей методологии позволит говорить о том, что ИИ – наука. А с другой стороны, выведет решение задач на новый уровень.

## Задача и информация

Рассмотрим в каких терминах можно описать процесс формирования задачи. Для этого обратимся к содержательному уровню (см. рис. 2, лекция 2). Описание реальности можно выполнить в терминах рис. 1 (лекция 1). Степень детализации объектов, характер связей и проч. вопросы зависят от задачи. Но во всех случаях общая схема понимания задачи такова. На первом шаге выделяются объекты, на втором – выделяются и классифицируются связи. Объекты, в свою очередь, разделяются на входные (участвующие в постановке задачи) и выходные (определяющие характер ответов или результатов в задаче). При таком разделении участвуют и связи, которые установлены между объектами. Выделенные таким образом объекты и связи формируют информационное поле задачи. Если воспользоваться терминологией рис. 2, то на содержательном уровне всякую задачу можно отождествить с некоторым подмножеством декартова произведения:  $I^{ex} \times I^{vых}$ . Обозначим это подмножество и задачу для краткости через  $Z$ , т.е. по определению будем полагать  $Z \subset I^{ex} \times I^{vых}$ . Отметим, что на формальном уровне в лекции 2, мы задачу отождествляли с подмножеством в декартовом произведении  $X \times Y$ . Необходимо отметить, что существует различие между пониманием задачи (на содержательном уровне) и постановкой задачи (на формальном уровне). Первичным является именно понимание задачи и все результаты оцениваются на содержательном уровне, хотя получаются на формальном.

В каких терминах может быть описано подмножество  $I^{ex} \times I^{vых}$ , а значит и задача? Для ответа на этот вопрос нам необходимо специализировать информацию и определить сущность процессов, которые выполняются при решении задачи. Для этого вернемся к уровням сознания (лекция 1).



Общим элементом, позволяющим объединить все уровни сознания является информация. Одновременно информация выступает в роли всеобщего объединителя всех объектов или поля (по аналогии с энергией), в котором происходит идентификация объектов. Осуществляется такая идентификация с помощью сознания. На каждом уровне сознания используется

свой вид информации. Интеллекту соответствует вид информации, который получил название знаний. Как эволюция характеризует зависимость между уровнями сознания, так в терминах эволюции можно говорить о зависимости между видами информации. Интуиции не сопоставлен никакой вид информации, хотя его можно описать, например, термином предвидение. Из приведенного рисунка можно сделать вывод: *при описании задачи И<sup>2</sup> необходимо использовать вид информации, который соответствует ЕИ (знания), и этот вид информации должен иметь отношение к универсальному процессу эволюции.*

### Дедуктивные и индуктивные задачи

Обратимся снова к схеме решения задачи на рис. 2 (лекция 2). Можно записать

$$\begin{aligned} \Pr(i^{ex}) &= i_1^{ex} \in I^{ex}, \\ p_3 \circ p_2 \circ p_1(i^{ex}) &= i_2^{ex} \in I^{ex}. \end{aligned} \quad (1)$$

Для того, чтобы что-то сказать о решении задачи  $Z$  в терминах (1), требуется установить отношение между  $i_1^{ex}$  и  $i_2^{ex}$ . В идеале, это отношение может иметь вид

$$\forall i^{ex} \in I^{ex} ((\Pr(i^{ex}) = p_3 \circ p_2 \circ p_1(i^{ex})) \Leftrightarrow (i_1^{ex} = i_2^{ex})). \quad (2)$$

Вначале заметим, что условие (2) является следствием одного из видов отношений, которое может быть введено на декартовом произведении  $I^{ex} \times I^{ex}$ . А именно, отношения эквивалентности, порожденного функцией равенства в указанном произведении. В более общем случае, любое отношение  $I^{ex} \times I^{ex}$  может быть описано функцией

$$\psi : I^{ex} \times I^{ex} \rightarrow R^+,$$

где  $R^+$  - подмножество неотрицательных действительных чисел. При этом  $\psi$  реализует какой-либо вариант "похожести" элементов в  $I^{ex}$ . Это может быть близость ( $\psi(i^{ex}, i^{ex}) = 0$ ), подобие ( $\psi(i^{ex}, i^{ex}) = 1$ ) либо другой какой-то вариант.

Введем теперь еще одну функцию вида

$$\varphi : R^+ \rightarrow [0,1],$$

и потребуем, чтобы она была монотонной, а также удовлетворяла следующему условию ( $r, r_1, r_2 \in R^+$ )

$$\varphi(r) = \begin{cases} 1, & r = r_1, \\ 0, & r = r_2. \end{cases} \quad (3)$$

Выбор функции  $\varphi$ , удовлетворяющей условию (3), определяется характером отображения  $\psi$ . В случае близости  $r_1 = 0, r_2 = +\infty$ , а для подобия  $-r_1 = 1, r_2 = 0 \vee +\infty$ . Во всех случаях, при таком выборе, имеет смысл суперпозиция  $\varphi \circ \psi$ , которая является основанием для введения следующей функции

$$\Phi(I^{ex}) = \sum_{i^{in} \in I^{input}} \varphi(\psi(\Pr(i^{ex}), p_3 \circ p_2 \circ p_1(i^{ex}))) \cdot |I^{ex}|^{-1}. \quad (4)$$

Нетрудно видеть, что (2) является частным случаем (4). При подходящем выборе отображений  $\varphi$  и  $\psi$ , условие (2) может быть записано:  $\Phi(I^{ex}) = 1$ .

Задачи, для которых выполняется условие

$$\Phi(I^{ex}) = 1 \quad (5)$$

Условие (5) играет особую роль в формальных теориях (математике, информатике и проч.). Связано это с тем, что такого решения, удовлетворяющее условию (5), называют обоснованным. Известно, что логическим основанием для вывода об обоснованности решения является принцип дедукции. Поэтому, соответствующий класс задач, для которого обоснованность может быть установлена, можно назвать дедуктивно разрешимым (или алгоритмически дедуктивно разрешимым). Для краткости мы будем называть такие задачи дедуктивными. Надо отметить, что задач (или точнее – классов задач), которые являются дедуктивными, значительно меньше, чем задач, для которых условие (5) не может быть получено. Такие задачи, в противоположность дедуктивным, будем называть индуктивно разрешимыми (или кратко – индуктивными). Для этого класса задач не существует аналога принципа дедукции, который можно было бы назвать принципом индукции. Тем не менее, такие задачи решаются, их большинство (в том числе и в математике). Задачи, которые имеют практический смысл, вообще все являются индуктивными по построению.

## Характеристические свойства задач ИИ

Объединяя перечисленные выше особенности задач  $Z$ , можно выделить следующие характеристические свойства задач из области ИИ:

1. **Индуктивность** (индуктивная разрешимость);
2. **Знания** (наличие знаний как вида информации);
3. **Эволюция** (как характеристика процесса получения решения).

Перечисленные свойства – это минимальный набор особенностей задачи ИИ. Он может быть дополнен. Все эти свойства будут обсуждаться в последующих лекциях. Сейчас же кратко обсудим методологические особенности решения таких задач.

Перечисленные свойства должны оказывать влияние на методы решения и даже на способ организации такого решения. Данные вопросы, как известно (см. рис. 1), относятся к области методологии. Рассмотрим их более подробно.

Для этого нам потребуется детализация и конкретное наполнение процесса получения решения, представленного на рис. 2 (лекция 2). При рассмотрении содержательного уровня можно выделить несколько компонент, образующих процесс решения. Любая задача описывается некоторой информацией, и получение решения связано с ее переработкой. Поэтому можно сказать, что задача и соответствующий процесс имеет информационную и вычислительную составляющие. Последовательность шагов, которая преобразует информацию, с помощью соответствующих средств реализуются в некотором окружении (среде). Данная среда естественным образом накладывает отпечаток на информационную и вычислительную составляющие. Поэтому можно выделить еще одну составляющую, которую можно назвать технологической.

Эти же компоненты можно выделить в процессе получения решения и на формальном уровне. А так как методология занимается проблемами организации процессов решения, то указанные компоненты естественным образом переносятся и на уровень формальных построений. Суть организации (управления) в этом случае сводится к решению проблем представления и оценивания в целях достижения решения. С проблемами представления все просто, это типичный круг проблем при математической формализации. Понятие цели связано с двумя сторонами процесса решения. С одной стороны, необходимо обеспечить саму возможность получения решения. С другой – оценить его качество с позиции обоснованности. Эти цели одновременно связаны с результатом и являются характеристикой процесса.

Ну и, наконец, реализация любого эвристического процесса связана с оцениванием результата и возвратом на исходный шаг (одновременно это можно рассматривать как эволю-

цию). На каждом шаге выполняется стандартная универсальная процедура, схема которого представлена на рис. 2. По результатам оценивания задача может корректироваться через ее компоненты. Схематично такая корректировка представлена в виде обратной связи. Этот процесс продолжается до тех пор, пока достижение локальных целей (на уровне отдельной задачи) возможно. Затем среди всех таких задач выбирается та, для которой получены наилучшие результаты. Дальнейшее развитие теории происходит в направлении такой задачи и до тех пор, пока это возможно, либо просто практически целесообразно.

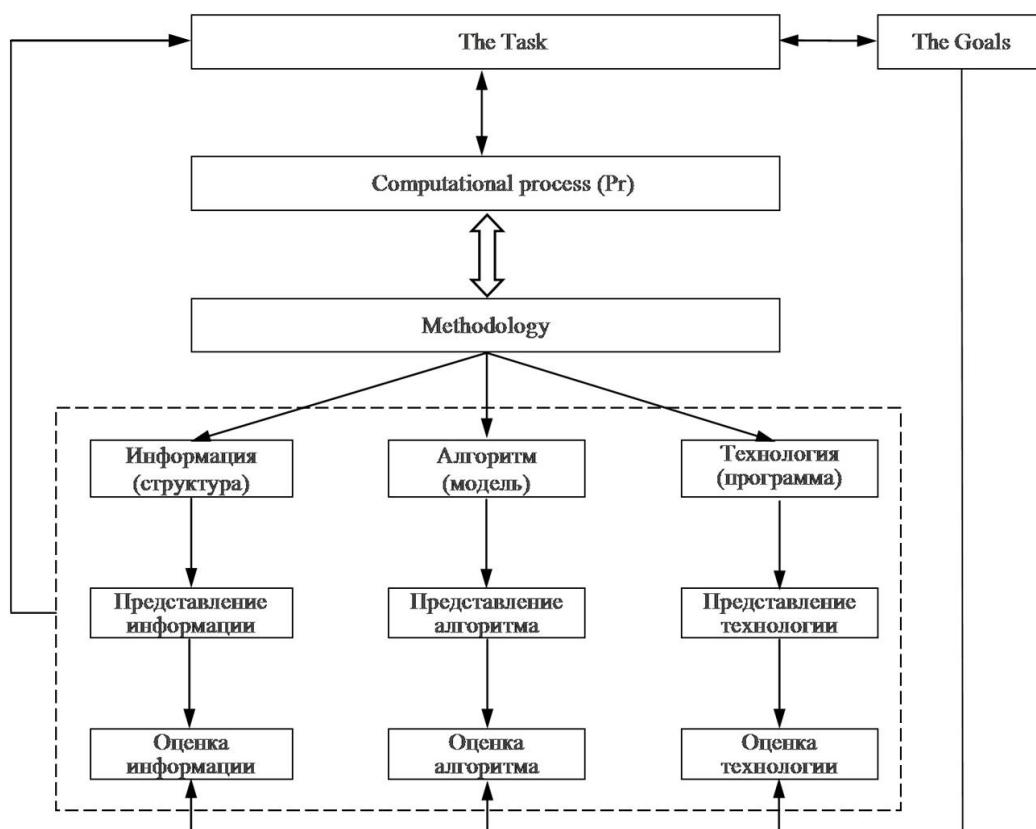


Рисунок 2 – Схема общего шага методологии

## Лекция 4

1. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И МОДЕЛИРОВАНИЕ.
2. ЛОГИКА – ОБЩАЯ ЧАСТЬ ЛЮБОГО МОДЕЛИРОВАНИЯ. КЛАССИФИКАЦИЯ ЛОГИЧЕСКИХ ФОРМАЛИЗМОВ.
3. ПОНЯТИЕ О ЛОГИЧЕСКОЙ СИСТЕМЕ.
4. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - Структура информации, уровни и языки ее описания.
  - Структура и модель.
  - Требования к результату в ЕИ, ИИ и конкретных науках.

### Искусственный интеллект и моделирование.

Схема моделирования:

*Реальность  $\Rightarrow$  представления о реальности  $\Leftrightarrow$  теория, формализация*

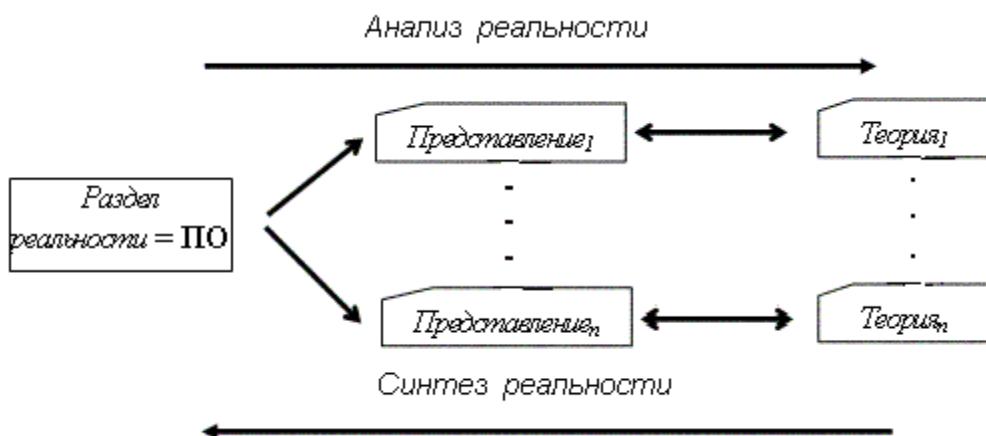
Реальность является средой существования *естественного интеллекта* (ЕИ). Для анализа (синтеза) реальности используется определенный набор средств. Такой набор обязательно должен содержать две компоненты:

- представления элементов реальности;
- манипулирования реальностью с целью получения информации о существующих (синтезируемых) элементах.

В процессе анализа (синтеза) ЕИ формируется некоторое представление о реальности, которое может быть *истинным, истинным в определенной степени либо ложным*. В зависимости от этого получаются *различные результаты*. Прямой зависимости между результатами и средствами, которые используются для их достижения, нет. В идеальном случае такие средства являются частью представления или прямым следствием последнего. Тогда они не “ухудшают” результат в смысле согласования степеней истинности результата и формируемого представления.

Возможность формирования любого представления о реальности предполагает некую формализацию, которая используется для “*моделирования*”. Так возникают теории, которые являются с одной стороны следствием соответствующего представления, а с другой – сами способны влиять не только на представление, но и на реальность. Для этого теория должна иметь методологию, быть обоснованной хотя бы в каком-то смысле и т.п.

Все сказанное выше относится к сфере компетенции ЕИ и схематично может быть представлено следующим образом:



ИИ можно определить как некоторую совокупность средств и методов, которые предназначены для "моделирования" ЕИ в приведенной выше схеме. При этом можно говорить не о "моделировании" вообще (как некоторой абстракции), а лишь о компьютерной его разновидности. Это следует из универсальности математической формализации.

Моделирование в таком контексте можно определить через результаты, которые имеют место в действительности и получены в результате использования модели. Они должны быть похожими в заранее определенном смысле. И поэтому модели могут различаться, можно даже говорить об их сравнении, обосновании, свойствах и т.п.

Любую часть приведенной схемы нельзя связывать с ИИ, т.к. ИИ не сводится к некоторой вычислительной реализации. (Цель моделирования в ИИ – не числа, а понимание).

### **Пример.**

Предположим, что некоторая база данных содержит сведения об отношениях "x — ОТЕЦ у" и "x — МАТЬ у". Чтобы обработать запросы типа:

ИВАНОВ А.И. — ДЕД ПЕТРОВА В.А.?

ПЕТРОВ В.А. — ВНУК ИВАНОВА А.И.?

необходимо либо ввести в базу данных также и сведения об отношениях "x — ДЕД у" и "x — ВНУК у", либо объяснить системе, как из отношений ОТЕЦ, МАТЬ извлечь исключительную информацию. Реализация первой возможности связана с неограниченным ростом избыточности базы данных. Вторая возможность при традиционном алгоритмическом подходе требует написания все новых и новых программ для реализации новых типов запросов.

Логический вывод позволяет расширять возможности "общения" наиболее просто и наглядно. Так, для приведенных типов запросов системе достаточно будет сообщить три правила:

1. x—ДЕД у если x—ОТЕЦ а и а—РОДИТЕЛЬ у
2. x—РОДИТЕЛЬ у если x—ОТЕЦ у или x—МАТЬ у
3. x—ВНУК у если у—ДЕД x

Эти правила содержат естественные и очевидные определения понятий ДЕД, РОДИТЕЛЬ, ВНУК. Поясним в чем состоит логический вывод для запроса "A—ДЕД B?" в предположении, что в базе данных имеются факты: A—ОТЕЦ B и B—МАТЬ В. При этом для упрощения опустим тонкости, связанные с падежными окончаниями. Пользуясь определением 1 система придет к необходимости проверки существования такого индивидуума а, что факты A—ОТЕЦ а и а—РОДИТЕЛЬ В истинны. Если такой а существует, то A—ДЕД B, если не существует такого а, то A не является дедом B.

## **Логика – общая часть любого моделирования. Классификация логических формализмов.**

Определим вначале, с чем имеет дело ЕИ. Начнем с того, что информацию о любой ПО можно представить, точно определив объекты в этой ПО. Тогда семантика (которая собственно и является предметом рассмотрения в ЕИ) очевидным образом связана с подмножествами декартового произведения: {**объект**}<sup>n</sup>, где n может быть произвольным числом (может и бесконечным). При этом отношения, которые определяются на множестве объектов, представляют собой обычные **связи**. Остается как-то назвать эти связи и в итоге получаем **семантику ПО**. Все остальное в ЕИ является производным от семантики – и возможные представления, и связанные с ними теории и т.д.

### **Вывод 1. Можно отождествить ПО с парой [объекты, связи].**

При построении теории (а уж тем более – при переходе к моделированию ЕИ средствами ИИ) необходимо определить **синтаксис представления** указанной пары и **средства манипулирования** объектами и связями.

### **Вывод 2. Базой ИИ является какое-либо решение проблем представления и манипулирования некоторыми синтаксическими объектами, которые подчиняются определенным семантическим ограничениям.**

Целью ЕИ является определение качественных характеристик изучаемых объектов или связей в терминах "истинно" – "ложно" (может быть в какой-то степени). Если при этом используется количественная характеристика, то и она через представляющие предикаты

сводится к качественной. Кроме того, прежде чем использовать модель нужно что-то сказать о ее поведении (качестве). Любое качественное суждение имеет соответствующую логическую реализацию (не обязательно в терминах булевой логики).

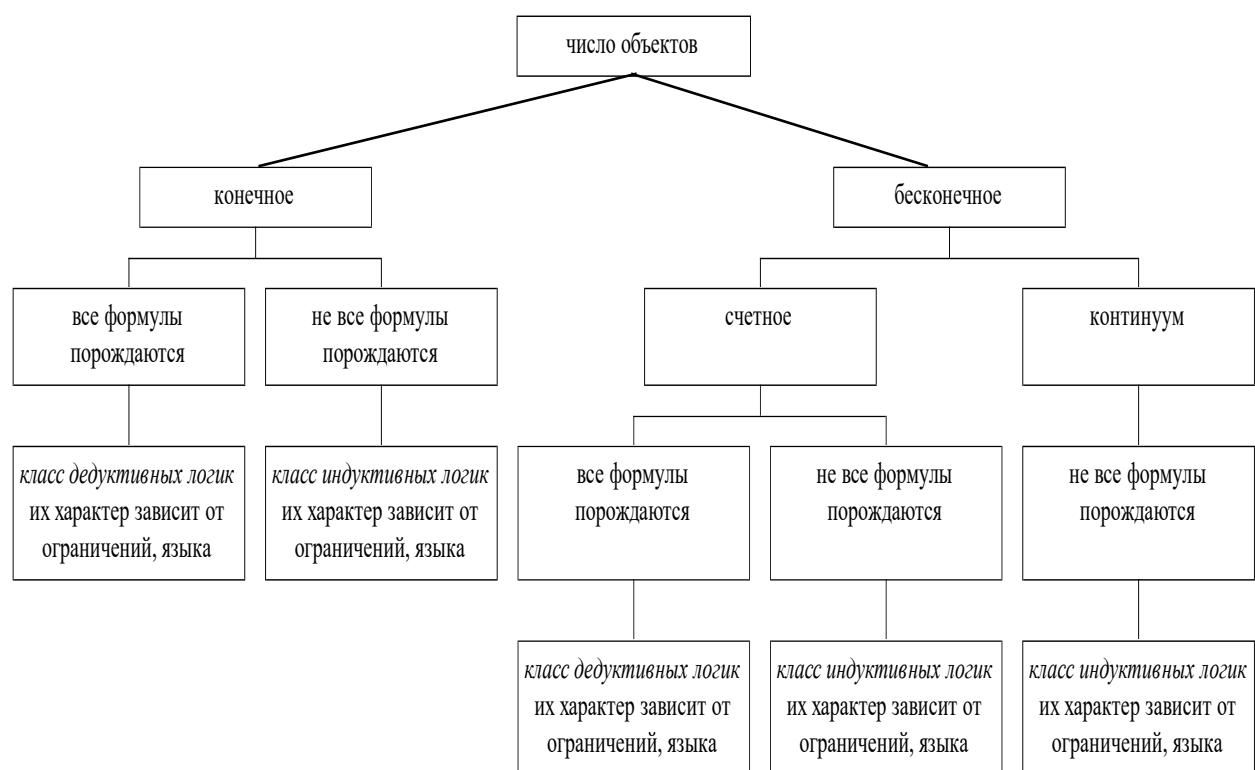
### **Вывод 3. Логика является центральной частью любого интеллекта (ЕИ, ИИ).**

Исходя из этого логические конструкции (различные по аппарату, синтаксису и семантике) являются базой методов ИИ. Независимо от возможных вариантов представления объектов и связей, можно говорить в основном о логическом манипулировании ими. Правда, лучше не забывать и о количественных методах.

Логические теории могут быть классифицированы по следующим основаниям:

- **числу формул, которые описывают объекты и связи – конечное и бесконечное (счетное либо континуум);**
- способу порождения формул теории – **все могут быть построены либо часть; требованиям к такому способу – достаточно существования или необходимо конструктивное построение объекта;**
- **языку, который используется для описания теории.**

Возможны и другие основания, позволяющие получить более детальную классификацию. По перечисленным же основаниям, классификация может быть представлена схематически следующим образом:



**Дедуктивные логики** в зависимости от ограничений и языка подразделяются на: **булевы, модальные и интуиционистские**.

**Индуктивные логики** все можно ассоциировать с **нечеткими**. Более подробная их классификация будет приведена в дальнейшем.

Ясно, что при построении систем ИИ встречаются все ситуации, описанные в приведенной выше схеме и, фактически, этой схемой они исчерпываются. Поэтому говорить о возможности конкретной логики для моделирования ЕИ можно в контексте ее особенностей: **выразительной силы языка, полноты, непротиворечивости, принципов обоснования** и т.д. Посмотрим с этой позиции на дедуктивные логики (на примере логики исчисления высказываний и частично - исчисления предикатов).

## Понятие о логической системе

Логическая модель представляет собой совокупность абстрактных объектов, не связанных с внешним миром, в которой имеются правила оперирования объектами, позволяющие получать новые объекты.

Логическая модель считается определенной, если:

1. задан конечный алфавит (множество символов);
2. определена процедура построения множества формул или слов формальной системы;
3. выделено множество формул, называемых **аксиомами**;
4. задано конечное множество правил вывода, которые позволяют получать из выделенного множества аксиом другое множество формул. Эти правила могут быть представлены в виде

$$X_1 \text{ и } X_2 \text{ и } \dots \text{ и } X_n \Rightarrow Y_1 \text{ и } Y_2 \text{ и } \dots \text{ и } Y_m$$

где  $X_i, Y_j$  – формулы формальной системы,  $\Rightarrow$  – символ следования.

Логические модели иногда называют аксиоматической системой. В п.1 определяется конечное множество символов, которое используется для построения формул. Символы бывают трех типов: константы, переменные и операторы. Иногда такое множество называется словарем. Аксиомы – это общезначимые формулы, т.е. истинные всегда при подстановке любых символов из словаря. В этом смысле правила вывода позволяют получать из аксиом новые общезначимые формулы, которые называются **теоремами**. Процесс их получения называется обычно **доказательством**. Тот факт, что формула  $A$  выводима, обозначается обычно  $E \vdash A$ . То, что  $A$  выводима из множества формул  $E$  обозначают  $E \vdash A$ . При этом, в  $E$  могут содержаться либо аксиомы, либо некоторые другие формулы. Если последние не являются выводимыми, то они называются **гипотезами**. Множество  $E$  называется невыполнимым, если  $E \vdash \perp$  (ложь).

Различают два типа правил вывода:

1. правила, применяемые к формулам, рассматриваемы как единое целое. В этом случае правила называют **продукциями**.
2. правила, которые применяют к любой отдельной части формулы, при условии, что эти части сами являются формулами. Такие правила называют правилами **переписывания**.

### Примеры.

- a). продукция:  $x < y$  и  $y < z \rightarrow x < z$ ;
- b). переписывание:  $x - x \rightarrow 0$ .

Обычно к логической модели предъявляются определенные требования. К числу таких требований можно отнести:

1. невозможность вывода отрицания уже доказанного утверждения, т.е.

$$E \vdash A \Rightarrow E \text{ и } \neg A \vdash A \quad (\neg - \text{знак отрицания})$$

2. логическая модель должна быть минимальной, т.е. не содержащей бесполезных аксиом и правил вывода. Это приводит к следующему определению. Некоторая формула называется независимой от аксиом  $E$ , если  $A$  нельзя вывести из  $E$  с помощью заданных правил вывода. В формальной системе, которая является минимальной, все аксиомы должны быть независимы.
3. логическая модель должна быть адекватной, т.е. любая теорема должна быть общезначимой (доказуемой, выводимой). Этот факт записывается

$$\vdash A \Rightarrow \models A.$$

логическая модель должна быть также полной, т.е. всякая общезначимая формула должна быть теоремой. Этот факт записывается

$$\models A \Rightarrow \vdash A.$$

В зависимости от принятия тех или иных свойств типа перечисленных выше, получаются разные логические модели.

### Примеры.

#### a. логическая модель **JP**

1. словарь: {a, b, =}.
2. формулы: любая комбинация символов из множества {a, b,  $\square$ }.
3. аксиомы:  $a \square a$ .
4. правила вывода:  $c_1 = c_2 \rightarrow bc_1 = bc_2$

Очевидно, что в данной модели могут быть получены следующие теоремы:

$$\begin{aligned} a &= a \\ ba &= ba \\ bba &= bba \\ \dots &\dots \end{aligned}$$

Между тем,  $bab = a$  – является формулой, но не выводимой.

#### b. логическая модель **DH**

1. словарь: {M, I, U}.
2. формулы: любая последовательность символов из множества {M, I, U}.
3. аксиомы: MI.
4. правила вывода:

$$\begin{aligned} mI &\rightarrow mIU \\ Mm &\rightarrow Mmm \\ II\rightarrow U \\ UU &\rightarrow \end{aligned}$$

здесь m – любая последовательность символов.

В данной модели может быть следующий вывод:

$$MI \Rightarrow (2) MII \Rightarrow (2) MIII \Rightarrow (1) MIIIIU \Rightarrow (3) MIUU \Rightarrow (4) MI$$

При задании логической модели возникает естественный вопрос: возможно ли, рассматривая какую-либо формулу, определить, является она доказуемой или нет? Иначе говоря, необходимо определить является ли рассматриваемая формула теоремой или нет и как это доказать. Если существует способ (процедура, алгоритм) такого доказательства, то соответствующая логическая модель называется **разрешимой**. Не всякая логическая модель является разрешимой. Дело в том, что если даже применить правила вывода последовательно ко всем объектам логической модели и получить все теоремы (м.б. при бесконечном их числе), то не существует все же подходящего способа, чтобы перечислить все не теоремы.

Логическая модель всегда представляет собой модель некоторой реальности (м.б. и математической). Распространение исходных положений какой-либо логической модели на реальность называется **интерпретацией**. Интерпретация придает смысл каждому символу логической модели и позволяет установить соответствие между символами и реальными объектами. Теоремы логической системы, будучи однажды интерпретированными, становятся утверждениями в обычном смысле. Относительно этих утверждений можно делать выводы об их истинности или ложности. Одна и та же логическая модель может быть интерпретирована в терминах различных реальностей. В этом смысле логическая модель тем интереснее, чем больше она допускает различных интерпретаций (т.е. чем для большего числа реальностей она является моделью).

Из введенных определений видно, что существует различие между концепциями доказательства и истинности, т.к. эти два понятия относятся к различным областям. Всего имеется 4 варианта взаимоотношения между доказательством и значением истинности. Обозначим теоремы через **T**, не теоремы – **NT**. Их интерпретация может быть истинной (**I**), либо ложной (**L**). Соотношение между этими вариантами можно представить следующим образом

Т, И	Т, Л
NT, И	NT, Л

Два выделенных варианта используются для корректировки логической системы.

### **Примеры.**

Рассмотрим две интерпретации системы JP.

1. Условимся, что символ а означает 0, а b – последующее целое число за символом, стоящим справа от него. Т.е.  $bba \rightarrow 2$ ,  $bbba \rightarrow 3$  и т.д. Символом же  $\square$  обозначим знак равенства  $=$ . Тогда аксиома интерпретируется как  $0 = 0$ . Теоремы же –  $1 = 1$ ,  $2 = 2$  и т.д. Очевидно, что и аксиома и теоремы являются истинными при такой интерпретации. В противоположность этому формула  $1 = 2$ , которая не интерпретирует никакую из теорем, является ложной. Заметим также, что в этой интерпретации множества (Т, Л) и (NT, И) пусты.
2. Пусть теперь а означает высказывание “Сократ смертен”, b – символ отрицания, а  $=$  – идентичность двух высказываний. Тогда аксиома формулируется: “Сократ смертен  $=$  Сократ смертен”. Формула “Сократ бессмертен  $=$  Сократ бессмертен”, которая является теоремой – также истинна. При такой интерпретации формула “ $\neg$ Сократ смертен  $=$  Сократ смертен”, которая является не теоремой, также истинна. В этой интерпретации непустыми будут классы (NT, И) и (NT, Л).

### **Дополнительные вопросы.**

1. Структура информации, уровни и языки ее описания.
2. Структура и модель.
3. Требования к результату в ЕИ, ИИ и конкретных науках.

## Лекция 5

1. ИСЧИСЛЕНИЕ ВЫСКАЗЫВАНИЙ. ОБЩИЕ ПОЛОЖЕНИЯ.
2. ВЫПОЛНИМЫЕ И ОБЩЕЗНАЧИМЫЕ ФОРМУЛЫ.
3. ПРОБЛЕМА ДОКАЗУЕМОСТИ ФОРМУЛ.
4. АЛГОРИТМЫ ДОКАЗАТЕЛЬСТВА РАЗРЕШИМОСТИ.
5. ПРИНЦИП РЕЗОЛЮЦИЙ.
6. АЛГОРИТМ РЕЗОЛЮЦИИ ДЛЯ ФОРМУЛ, НЕ ПРИВЕДЕННЫХ К КНФ.
7. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - ИИ и исчисление высказываний.

### Исчисление высказываний. Общие положения.

Исчисление высказываний изучает предложения, которые могут быть либо **истинными**, либо **ложными**.

#### Примеры.

1. За четверть часа до своей смерти он был еще жив – *истина языка*.
2. Земля вертится – *фактическая истинна*.
3. Если верно, что когда идет дождь, то дорога мокрая, то справедливо также и следующее утверждение: если дождя нет, то дорога сухая. Это пример *логической истины*, справедливой при подстановке и других высказываний.

Эта формальная система определяется следующим образом:

1. Алфавит
  - a. буквы  $p, q, r, s, t, \dots$
  - b. логические операторы  $\neg, \supset$
  - c. скобки  $(\ )$
2. Построение формул
  - a. всякое высказывание (буква) есть формула
  - b. если  $m$  – формула, то  $(m)$  – также формула
  - c. если  $m_1, m_2, m_3$  – формулы, то  $\neg m_1, m_2 \supset m_3$  – также формулы.
  - d. Формулы однозначно получаются с помощью правил а-в.
3. Аксиомы ( $m_1, m_2, m_3$  – формулы или буквы)
  - (A1)  $(m_1 \supset (m_2 \supset m_1))$
  - (A2)  $((m_1 \supset (m_2 \supset m_3)) \supset ((m_1 \supset m_2) \supset (m_1 \supset m_3)))$
  - (A3)  $((\neg m_1 \supset \neg m_2) \supset ((\neg m_1 \supset m_2) \supset m_1))$

#### 4. Правила вывода

если  $m_1$  и  $(m_1 \supset m_2)$  – теоремы, то  $m_2$  также теорема.

Это правило обычно называют *modus ponens* или правилом отделения.

Заметим, что существуют и другие способы аксиоматизации исчисления высказываний. Для этого могут использоваться дополнительные логические связки:  $\vee$  (дизъюнкция),  $\wedge$  (конъюнкция),  $\equiv$  (эквивалентность). Однако, эти аксиоматические системы могут быть приведены к указанной выше, т.к.

$$\begin{aligned} (m_1 \equiv m_2) \text{ равносильно } & (m_1 \supset m_2) \wedge (m_2 \supset m_1) \\ (\neg m_1 \vee m_2) \text{ равносильно } & (m_1 \supset m_2) \\ \neg(m_1 \wedge m_2) \text{ равносильно } & (\neg m_1 \vee \neg m_2) \end{aligned}$$

В исчислении высказываний ни правило вывода, ни аксиомы не используют специальных букв. Поэтому эта система инвариантна относительно используемых символов.

$$K(p \supset \neg q) \supset (r \supset s) \text{ равносильно } K(t \supset \neg s) \supset (q \supset r)$$

В исчислении высказываний разрешается вводить новые буквы, называемые свободными переменными. В этой системе отсутствуют константы.

Каждая формула исчисления высказываний может быть интерпретирована, т.е. ей может быть приписано одно из двух значений – И или Л. При этом значение формулы – есть функция значений ее составляющих. Обычно для интерпретации используются следующие таблицы истинности.

Для бинарных операций

$p$	$q$	$p \wedge q$	$p \vee q$	$p \supset q$	$p = q$
И	И	И	И	И	И
И	Л	Л	И	Л	Л
Л	И	Л	И	И	Л
Л	Л	Л	Л	И	И

Для унарных

$p$	$\neg p$
И	Л
Л	И

Для определения значения конкретной формулы могут использоваться приведенные выше таблицы. Так, используя таблицы, можно показать, что формула

$$(p \supset (q \supset r)) \equiv ((p \wedge q) \supset r) \quad (\text{упражнение})$$

истинна при любых интерпретациях.

Заметим, что между логическими операциями и связками естественного языка существует взаимосвязь. В ЕЯ гораздо больше связок, но ИВ не станет более выразительным, если ввести дополнительные связки. Т.е. то, что можно выразить в ИВ с помощью логических операторов  $\neg$ ,  $\supset$  эквивалентно тому, что выражается с помощью более широкого набора операторов  $\neg$ ,  $\supset$ ,  $\wedge$ ,  $\vee$ ,  $\equiv$ .

### Выполнимые и общезначимые формулы.

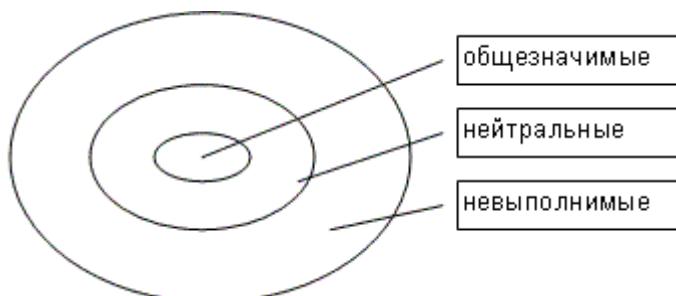
Формула называется **выполнимой**, если при некоторых значениях входящих в нее символов (высказываний) она допускает интерпретацию со значением И.

**Пример.**  $p \wedge q$ ,  $p \vee q$  – выполнимые формулы,  $p \neg p$  – не выполнимая.

Формула называется **общезначимой**, если она истинна, независимо от значений входящих в нее символов (высказываний).

**Пример.**  $p \vee \neg p$ ,  $\neg \neg p \equiv p$  – общезначимые формулы.

Из этих определений следует, что все множество формул в ИВ можно разбить на три подмножества: общезначимые, невыполнимые и нейтральные.



Общезначимые формулы в ИВ называют тавтологиями и обозначают  $\models A$ . Запись  $E \models A$  означает, что при всех интерпретациях, при которых истинны формулы из  $E$ , истинна также

и А. Такая формула А называется логическим следствием из Е. В этом смысле, тавтология – логическое следствие из пустого множества.

### Проблема доказуемости формул.

Главная проблема в ИВ – это проблема доказуемости формул, т.е. возможность вывода формул. В 1921г. Пост доказал **теорему**:

*Формула А доказуема в ИВ тогда и только тогда, когда она является тавтологией.*

Из этой теоремы следует, что ИВ характеризуется:

1. *непротиворечивостью*, т.е. А и  $\neg A$  не могут быть одновременно выводимыми.
2. *полнотой*, т.е. теоремы в точности соответствуют тавтологиям.

3. *разрешимостью*, т.е. существует процедура решения проблемы доказуемости.

В силу теоремы Поста все сказанное выше справедливо для тавтологий. По отношению к другим формулам правомерен прием редуцирования проблем, основанный на следующем **утверждении**:

А является логическим следствием из В тогда и только тогда, когда импликация ( $B \supset A$ ) есть тавтология, т.е.

$$B \models A \Leftrightarrow \models (B \supset A)$$

Это утверждение имеет и следующую, более общую, формулировку:

$$(H_1, \dots, H_n) \models A \Leftrightarrow \models ((H_1, \dots, H_n) \supset A)$$

В этом утверждении  $H_i$  называют гипотезами, а формулы А – заключениями. В последнем виде способ доказательства называют **принципом дедукции**.

Следствием принципа дедукции является универсальный механизм вывода, который применим во всех логических системах. Этот механизм – *алгоритм (метод) резолюций*. Данный метод эффективно используется для доказательства доказуемости формул. Введем следующее определение: множество формул Е называется **выполнимым**, если все они могут быть истинными одновременно. В семантическом смысле это определение позволяет рассматривать множество Е как конъюнкцию формул, входящих в Е. Далее остается воспользоваться принципом дедукции в следующей форме: формула А является логическим следствием конечного множества формул Е тогда и только тогда, когда формула ( $E \wedge \neg A$ ) невыполнима. Заметим также, что невыполнимость Е означает ( $E \models \perp$ ). Тогда принцип дедукции можно переписать:

$$(H_1 \wedge \dots \wedge H_n) \models A \Leftrightarrow (H_1 \wedge \dots \wedge H_n \wedge \neg A) \models \perp$$

В дальнейшем, при рассмотрении метода резолюции, мы ограничимся случаем конечного  $n$ . Еще одним следствием принципа дедукции являются два принципа **математической индукции**:

- *принцип полной математической индукции*

$$(P(1) \wedge P(n) \Rightarrow P(n+1)) \Rightarrow \forall n P(n)$$

- *принцип трансфинитной индукции*

$$(P(1) \wedge \forall m < n P(m) \Rightarrow P(n)) \Rightarrow \forall n P(n)$$

где  $P$  – некоторое свойство (предикат),  $m, n$  – натуральные числа.

Оба принципа математической индукции (обобщения, от частного к общему) являются дедуктивными по характеру доказательства. В основном, они используются для сокращения доказательства выводимости в логических системах. Но, вообще говоря, их можно рассматривать как определенный шаг в направлении “не дедуктивного” вывода.

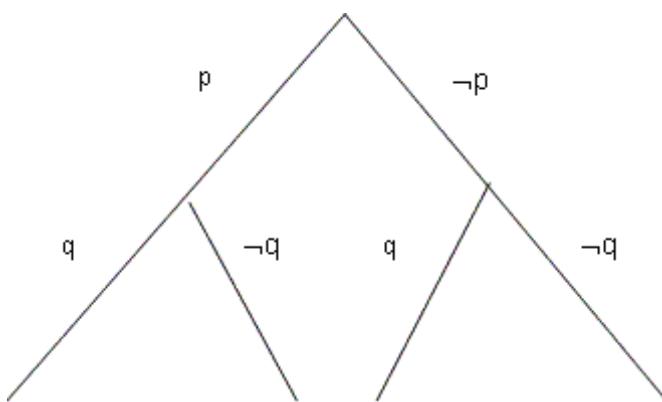
## Алгоритмы доказательства разрешимости.

Явно или неявно различные алгоритмы используют понятие **семантического дерева**. Если дано конечное или счетное множество высказываний  $P = \{P_1, \dots, P_n, \dots\}$ , то семантическое дерево – это бинарное корневое дерево, удовлетворяющее следующим условиям:

1. каждая дуга помечена позитивной или негативной литерой, взятой из  $P$ ;
2. литерами, которыми помечены две дуги, выходящие из одного узла, противоположны;
3. никакая ветвь не содержит более одного вхождения каждой литеры;
4. никакая ветвь не содержит пару противоположных литер.

**Пример.**  $P = \{p, q\}$ .

Соответствующее семантическое дерево имеет вид



Каждому узлу дерева соответствует частичная интерпретация, т.е. функция  $I_N$ , которая ставит в соответствие истинностное значение по выбранному элементу из  $P$ . Частичная интерпретация  $I_N$  сопоставляет значение И (соответственно – Л), если некоторая дуга пути, соединяющего выбранную вершину с корнем, помечена выбранной литературой из  $P$  (соответственно - отрицанием). Частичная интерпретация не определена для  $p$ , если ни одна из литер  $p$  и  $\neg p$  не встречается на соответствующем пути. Конечное семантическое дерево называется полным, если каждый его лист соответствует некоторой всюду определенной интерпретации.

### Тривиальный алгоритм.

Требует просмотра полного семантического дерева, соответствующего множеству высказываний, встречающихся в  $P$ . Для каждого листа этого дерева, формула оценивается согласно соответствующей интерпретации. Формула является выполнимой, если, по крайней мере, для одного из листьев получается значение И.

Этот алгоритм неэффективен. Если  $P$  содержит  $n$  различных высказываний, то нужно рассмотреть  $2^n$  интерпретаций. Т.е. алгоритм является экспоненциальным.

### Алгоритм Куайна.

Представляет собой усовершенствование тривиального алгоритма. Основан на следующей идее:

если при всех возможных расширениях некоторой частичной интерпретации, исследуемая формула принимает одно и то же истинностное значение из {И, Л}, то бесполезно строить поддерево, исходящее из узла, соответствующего этой частичной интерпретации.

**Пример.**  $((p \wedge q) \supset r) \wedge (r \supset p)$

Начинаем с произвольного упорядочения множества  $\{p, q, r\}$ . Например, пусть это будет порядок  $(p, q, r)$ . Сначала рассмотрим те интерпретации, при которых  $p$  принимает значение И. В этом случае, формула сводится к виду

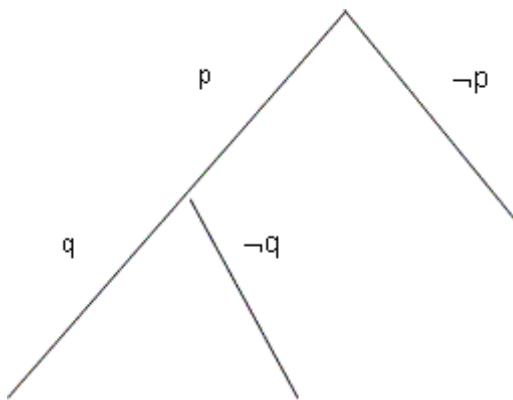
$$((q \supset r) \wedge q) \supset r$$

Если в ней  $q$  принимает значение И, то получим  $(r \supset r)$ , что общезначимо. Если же  $q$  принимает значение Л, то имеем  $(\perp \supset r)$ , что также общезначимо.

Далее, рассмотрим интерпретации, при которых  $r$  принимает значение Л. Имеем:

$$(\perp \supset r) \supset I,$$

что всегда истинно. Следовательно, исходная формула всегда истинна, т.е. общезначима. Соответствующее семантическое дерево имеет вид:



### **Алгоритм редукции.**

Позволяет доказывать общезначимость формул путем приведения к абсурду. Алгоритм особенно удобен, когда формула содержит много импликаций.

**Пример.**  $((p \wedge q) \supset r) \supset (p \supset (q \supset r))$ .

Предположим, что при некоторой интерпретации  $I$  эта формула принимает значение Л. Это возможно только если

$$I((p \wedge q) \supset r) = I, I(p \supset (q \supset r)) = L$$

Из второй формулы следует, что  $I(p)=I$ ,  $I(q)=I$ ,  $I(r)=L$ . Но тогда  $I((p \wedge q) \supset r) = L$ , что противоречит первой формуле. Это противоречие означает, что исходные предположения не верны. Отсюда следует общезначимость формулы.

Все приведенные алгоритмы сравнимы по сложности и относятся к классу NP-сложных алгоритмов. Т.е. алгоритмы в общем случае неэффективны. Поэтому необходимы не переборные алгоритмы. Такие алгоритмы должны организовывать более эффективный просмотр семантического дерева.

### **Принцип резолюций.**

Этот принцип основан на предварительном преобразовании формулы к некоторой канонической форме.

**Дизъюнктом** называется дизъюнкция конечного числа литер, т.е. формула вида:

$$(I_1 \vee \dots \vee I_n)$$

**Конъюнктивной нормальной формой (КНФ)** называется конъюнкция конечного числа дизъюнктов.

Справедливо следующее **утверждение**:

Любая формула имеет логически эквивалентную ей КНФ.

### **Алгоритм приведения к КНФ.**

1. заменяем  $(m_1 \equiv m_2)$  на  $(m_1 \supset m_2) \wedge (m_2 \supset m_1)$ , а затем  $(m_1 \supset m_2)$  на  $(\neg m_1 \vee m_2)$ . Это делается для исключения операций:  $\equiv, \supset$ .
2. добиваемся того, чтобы операция  $\neg$  оставалась только непосредственно перед литерами. Для этого необходимое число раз применяем преобразования:

$$\begin{aligned}\neg(m_1 \wedge m_2) &\rightarrow (\neg m_1 \vee \neg m_2), \\ \neg(m_1 \vee m_2) &\rightarrow (\neg m_1 \wedge \neg m_2), \\ \neg\neg m_1 &\rightarrow m_1\end{aligned}$$

3. необходимое число раз применяются преобразования, основанные на дистрибутивности операций:

$$\begin{aligned}m_1 \vee (m_2 \wedge m_3) &\rightarrow (m_1 \vee m_2) \wedge (m_1 \vee m_3), \\ m_1 \wedge (m_2 \vee m_3) &\rightarrow (m_1 \wedge m_2) \vee (m_1 \wedge m_3).\end{aligned}$$

4. дизъюнкты, содержащие литеру (высказывание) и ее отрицание  $(m \vee \neg m) = I$  могут быть опущены, т.к. они общезначимы. Опускаются также повторения литер (высказываний) в пределах одного дизъюнкта.

Полученная этим алгоритмом КНФ называется приведенной.

**Пример.**

$$\begin{aligned}a. \quad &(p \supset (q \supset r)) \\ &\neg p \vee (q \supset r) \rightarrow \neg p \vee (\neg q \vee r) \rightarrow (\neg p \vee \neg q \vee r) \\ b. \quad &((p \wedge s) \supset r) \\ &\neg(p \wedge s) \vee r \rightarrow (\neg p \vee \neg s) \vee r \rightarrow (\neg p \vee \neg s \vee r)\end{aligned}$$

**Замечание.** Описание задач и алгоритмов в терминах дизъюнктов составляет основу логического программирования. Используется в языках ПРОЛОГ, LISP.

Для КНФ справедливо **утверждение**:

Формула Р в КНФ является общезначимой тогда и только тогда, когда каждый дизъюнкт общезначим.

Для проверки выполнимости КНФ не существует эффективного алгоритма. Тем не менее, есть достаточно удобный метод для выявления невыполнимости множества дизъюнктов. Действительно, множество дизъюнктов невыполнимо в том и только том случае, когда Л является логическим следствием из него. Таким образом, невыполнимость множества формул S можно проверить порождая логические следствия из S до тех пор, пока не получим пустой дизъюнкт (Л).

Для порождения логических следствий может использоваться простая схема рассуждений. Пусть  $m_1, m_2, m_3$  - формулы. Предположим, что две формулы  $(m_1 \vee m_3)$  и  $(m_2 \vee \neg m_3)$  истинны. Если формула  $m_3$  истинна, то можно сделать заключение, что  $m_2$  истинна. Если же формула  $m_3$  ложна, то  $m_1$  истинна. Во всех случаях формула  $(m_1 \vee m_2)$  является истинной. Получается следующее правило:

$$\{m_1 \vee m_3, m_2 \vee \neg m_3\} \models m_1 \vee m_2$$

или в эквивалентной форме

$$\{\neg m_1 \supset m_3, \neg m_2 \supset \neg m_3\} \models m_1 \vee m_2$$

В том частном случае, когда  $m_3$  - высказывание, а  $m_1, m_2$  - дизъюнкты, это правило называется **правилом резолюций**. Общезначимость правила резолюций выражается в следующем **утверждении**:

Пусть  $s_1, s_2$  - дизъюнкты некоторой формулы S, I – литер. Если  $I \in s_1$  и  $\neg I \in s_2$ , то дизъюнкт

$$r = (s_1 \setminus l) \vee (s_2 \setminus \neg l)$$

является логическим следствием формулы  $S$ .

### Замечания.

1. Нетрудно показать, что формулы  $S$  и  $S \wedge r$  эквивалентны.
2. Дизъюнкт  $r$  называется **резольвентой** дизъюнктов  $s_1, s_2$ .

### Алгоритм.

1. при условии, что  $\mathcal{L} \notin S$  переходим к шагу 2. В противном случае – к шагу 5.
2. выбираем  $l, s_1, s_2$  такие, что  $l \in s_1$  и  $\neg l \in s_2$ .
3. вычисляем резольвенту  $r$ .
4. заменяем  $S$  на  $S \wedge r$  и переходим к шагу 1.
5. конец работы алгоритма.

**Пример.** Проверим невыполнимость множества

$$S = \{p \vee q, p \vee r, \neg q \vee \neg r, \neg p\}$$

Занумеруем вначале дизъюнкты формулы  $S$ . Получим следующий список:

$$1. p \vee q \quad 2. p \vee r \quad 3. \neg q \vee \neg r \quad 4. \neg p$$

Теперь вычисляем резольвенты, и результаты добавляем к  $S$ . В списке, который приводится ниже, каждый дизъюнкт – резольвента некоторых предыдущих дизъюнктов. Номера их приводятся в скобках справа.

- |                    |        |
|--------------------|--------|
| 5. $p \vee \neg r$ | (1, 3) |
| 6. $q$             | (1, 4) |
| 7. $\neg r$        | (3, 6) |
| 8. $p$             | (2, 7) |
| 9. Л               | (4, 8) |

Из примера видно, что стратегия построения дизъюнктов может значительно влиять на эффективность алгоритмов. Тем не менее, независимо от стратегии, существуют важные свойства формулы  $S$ . Например, если  $S$  не содержит ни одной пары дизъюнктов, допускающих резольвенту, то  $S$  выполнимо. Это справедливо при условии, что  $S$  не содержит пустой дизъюнкт.

Представляет также интерес свойство гарантированности результата, который получается алгоритмом резолюций. Это свойство обосновывается в следующих утверждениях.

1. Если  $S$  невыполнимо и содержит резольвенты своих дизъюнктов, то оно обязательно содержит и пустой дизъюнкт.

2. Если  $S$  невыполнимо, то этот факт всегда можно установить алгоритмом резолюций.

Алгоритм резолюций применяется в качестве механизма доказательства при реализации принципа дедукции. Для доказательства того, что некая формула  $A$  является логическим следствием гипотез  $H_1, \dots, H_n$ , необходимо применить резолюцию к множеству

$\{H_1, \dots, H_n, \neg A\}$ . Эти гипотезы и формула  $\neg A$  должны иметь вид дизъюнктов.

**Пример.**

$$\{h, \neg h \vee p \vee q, \neg p \vee c, \neg q \vee c\} \supset c$$

- |                           |        |
|---------------------------|--------|
| 1. $h$                    |        |
| 2. $\neg h \vee p \vee q$ |        |
| 3. $\neg p \vee c$        |        |
| 4. $\neg q \vee c$        |        |
| 5. $\neg c$               |        |
| 6. $p \vee q$             | (1, 2) |
| 7. $\neg p$               | (3, 5) |
| 8. $\neg q$               | (4, 5) |
| 9. $q$                    | (6, 7) |
| 10. Л                     | (8, 9) |

Этот вывод часто представляют в форме перевернутого семантического дерева.

### Алгоритм резолюции для формул, не приведенных к КНФ.

Приведение формулы к КНФ может оказаться довольно трудоемким. В этом случае может использоваться т.н. неклаузальная форма алгоритма резолюций. Эта форма предназначена для распространения механизма доказательства на произвольные логические формулы, т.е. когда гипотезы и отрицание заключения не являются дизъюнктами.

Пусть  $m_1, m_2$  - формулы ИВ. Ведем определения

$$T_p(m_1, m_2) \stackrel{\text{def}}{=} m_1^{p=\Lambda} \wedge m_2^{p=I}$$

$$\perp_p(m_1, m_2) \stackrel{\text{def}}{=} m_1^{p=\Lambda} \vee m_2^{p=I}$$

Символом  $m_i^{p=\Lambda(I)}$  обозначена формула, полученная из формулы  $m_i$  заменой всех вхождений переменной  $p$  на  $\Lambda$  ( $I$ ). Оператор  $T$  употребляется для доказательства общезначимости, а  $\perp$  - невыполнимости.

#### Пример.

Доказать невыполнимость

$$\models \{(p \wedge r) \vee (q \wedge \neg r), (\neg p \wedge r) \vee (\neg q \wedge \neg r)\}$$

1.  $(p \wedge r) \vee (q \wedge \neg r)$
2.  $(\neg p \wedge r) \vee (\neg q \wedge \neg r)$
3.  $(q \wedge \neg r) \vee (\neg q \vee \neg r)$  (1, 2)  $\perp_p$
4.  $(p \wedge r) \vee (\neg p \wedge r)$  (1, 2)  $\perp_q$
5.  $\perp$  (4, 3)  $\perp_r$

### Дополнительные вопросы.

1. ИИ и исчисление высказываний.

## Лекция 6

1. ФОРМАЛИЗАЦИЯ ИСЧИСЛЕНИЯ ПРЕДИКАТОВ.
2. ДОКАЗАТЕЛЬСТВО ВЫПОЛНИМОСТИ В ИП.
3. МЕТОД РЕЗОЛЮЦИЙ.
4. АЛГОРИТМ РЕЗОЛЮЦИИ С УНИФИКАЦИЕЙ.
5. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - Исчисление предикатов и высказываний. Соотношение.
  - ИИ и исчисление предикатов.

### Формализация исчисления предикатов.

Исчисление высказываний позволяет формализовать лишь небольшую часть множества рассуждений. Например, рассмотрим следующее рассуждение Аристотеля:

Все люди смертны, Сократ – человек. Следовательно, Сократ смертен.

Очевидно, что это рассуждение выходит за рамки ИВ. Если обозначить высказывания, входящие в это рассуждение через  $p$ ,  $q$  и  $r$  соответственно, то формулу рассуждения можно записать:  $(p \wedge q) \supset r$ . Эта формула является не общезначимой. В тоже время, само рассуждение Аристотеля является истинным при любых условиях, т.е. общезначимо. Происходит это по той причине, что высказывание может иметь внутреннюю структуру, а в ИВ все рассуждения формализуются как неделимый объект.

Если провести анализ внутренней структуры приведенного рассуждения, то его можно переписать следующим образом:

Для всех  $x$ , если  $x$  является человеком, то  $x$  смертен. Сократ является человеком, следовательно, Сократ смертен.

Если теперь ввести одноместные предикатные константы  $H(x)$ ,  $M(x)$ , которые будут означать свойства “являться человеком” и “быть смертным” соответственно, то исходное рассуждение может быть приведено к виду:

$$\forall x(H(x) \supset M(x)) \wedge H(\text{Сократ}) \Rightarrow M(\text{Сократ})$$

Предикатные константы – это формулы, которые связаны с подходящим числом переменных или параметров, но главное – не меняют своего значения истинности при подстановке.

1. Алфавит.
  - a. константы  $a, b, c, \dots$
  - b. индивидуальные переменные  $x, y, z, \dots$
  - c. предикаты  $A, B, C, \dots$
  - d. логические операторы  $\neg, \supset$
  - e. квантор всеобщности  $\forall$

#### 2. Построение формул.

Формулы ИП образуются аналогично формулам ИВ. Кроме того:

- a. каждому предикату приписывается вес  $k$ . Выражение  $A(x_1, \dots, x_k)$  является формулой, если и только если вес  $A$  равен  $k$ .
- b. выражение  $(\forall x_1 A(x_1, \dots, x_k))$  представляет собой формулу, в которой  $x_1$  есть связанныя переменная, а все остальные  $x_i$  ( $i \geq 2$ ) - свободные.

#### 3. Аксиомы.

Имеются все три аксиомы ИВ (A1-A3), а также:

$$(A4) \quad (\forall t)A(t) \supset A(I)$$

$$(A5) \quad (\forall t)(m_1 \supset m_2) \supset (m_1 \supset (\forall t)m_2)$$

где  $m_1, m_2$  - формулы,  $t$  – не является свободной переменной в  $m_1$ .

#### 4. Правила вывода.

$$\begin{array}{ll} (m_1 \wedge (m_1 \supset m_2) \supset m_2 & \text{modus ponens} \\ m_1 \supset (\forall t)m_1 & \text{обобщение} \end{array}$$

где  $t$  – свободная переменная в  $m_1$ .

Говорят, что в  $x$  **связанная** переменная в формуле, если первое вхождение  $x$  описывает связь, которой подчинено второе вхождение  $x$ .

**Пример.**

1.  $f(x) = \sum_{i=1}^n \alpha_i \cdot x_i$ ,  $i$  – связанная переменная.

2.  $\forall x(H(x) \supset M(x))$ ,  $x$  – связанная и квантифицированная переменная.

В остальных случаях переменная называется **свободной**.

Также как в ИВ, в ИП используются операции  $\vee$  (дизъюнкция),  $\wedge$  (конъюнкция),  $\equiv$  (эквивалентность). Кроме того, в ИП используется квантор существования  $\exists$ . Между кванторами и операциями установлена следующая связь:

$$((\exists x)A(x)) \equiv \neg((\forall x)\neg A(x))$$

Такие системы ИП принято называть логическими системами ИП первого порядка. В них действие кванторов распространяется только на предикатные переменные и внутренние символы в формулах.

В ИП формулы также делятся на три подмножества: общезначимые (истинные при всех интерпретациях), невыполнимые (ложные при всех интерпретациях) и нейтральные (истинные хотя бы для одной интерпретации).

#### Доказательство выполнимости в ИП.

Для доказательства выполнимости формул в ИП используется тот же механизм, что и в ИВ. Предварительно каждая формула приводится к КНФ. Алгоритм приведения к КНФ состоит из четырех основных этапов.

##### 1. Приведение к предваренной (префиксной) форме.

Предваренной называется формула, которая имеет вид

$$Q_1 x_1 \dots Q_n x_n M,$$

где  $Q_i$  обозначает один из кванторов  $\forall, \exists$ ,  $M$  – формула, не содержащая кванторов.

Справедливо следующее **утверждение**:

Для любой формулы существует логически эквивалентная ей предваренная формула.

Алгоритм приведения к предваренной форме:

- исключить операции эквивалентности и импликации;
- если необходимо, переименовать связанные переменные таким образом, чтобы никакая переменная не имела бы одновременно свободных и связных вхождений;
- удалить те кванторы, которые не распространяются на квантифицированную переменную;

- d. преобразовать формулу в соответствии со следующими правилами (снятие отрицаний или уменьшение радиуса действия операции  $\neg$ ):

$$\begin{aligned}\neg \forall x A \rightarrow \exists x \neg A, \quad \neg \exists x A \rightarrow \forall x \neg A, \\ \neg(A \wedge B) \rightarrow (\neg A \vee \neg B), \quad \neg(A \vee B) \rightarrow (\neg A \wedge \neg B), \quad \neg \neg A \rightarrow A\end{aligned}$$

- e. переместить все кванторы в начало формулы. Для этого используются следующие правила:

$$\begin{aligned}(\forall x A \wedge \forall x B) \rightarrow \forall x(A \wedge B), \quad (\forall x A \vee \forall x B) \rightarrow \forall x(A \vee B) \\ (\forall x A \wedge B) \rightarrow \forall x(A \wedge B), \quad (\forall x A \vee B) \rightarrow \forall x(A \vee B) & \quad (\text{если } B \text{ не содержит } x) \\ (A \wedge \forall x B) \rightarrow \forall x(A \wedge B), \quad (A \vee \forall x B) \rightarrow \forall x(A \vee B) & \quad (\text{если } A \text{ не содержит } x) \\ (\exists x A \wedge B) \rightarrow \exists x(A \wedge B), \quad (\exists x A \vee B) \rightarrow \exists x(A \vee B) & \quad (\text{если } B \text{ не содержит } x) \\ (A \wedge \exists x B) \rightarrow \exists x(A \wedge B), \quad (A \vee \exists x B) \rightarrow \exists x(A \vee B) & \quad (\text{если } A \text{ не содержит } x)\end{aligned}$$

### Примеры.

1.  $\forall x(P(x) \wedge \forall y \exists x(\neg Q(x, y) \supset \forall z R(a, x, y)))$

$$\begin{aligned}\forall x(P(x) \wedge \forall y \exists x(\neg \neg Q(x, y) \vee \forall z R(a, x, y))) & \quad (\text{п. а алгоритма}) \\ \forall x(P(x) \wedge \forall y \exists u(\neg \neg Q(u, y) \vee \forall z R(a, u, y))) & \quad (\text{п. б алгоритма}) \\ \forall x(P(x) \wedge \forall y \exists u(\neg \neg Q(u, y) \vee R(a, u, y))) & \quad (\text{п. с алгоритма}) \\ \forall x(P(x) \wedge \forall y \exists u(Q(u, y) \vee R(a, u, y))) & \quad (\text{п. д алгоритма}) \\ \forall x \forall y \exists u(P(x) \wedge (Q(u, y) \vee R(a, u, y))) & \quad (\text{п. е алгоритма})\end{aligned}$$

2.  $\forall x(P(x) \supset \exists y((P(y) \vee \neg R(a, x, y)) \supset \forall z(\neg S(y, z))))$

$$\begin{aligned}\forall x(\neg P(x) \vee \exists y(\neg(P(y) \vee \neg R(a, x, y)) \vee \forall z(\neg S(y, z)))) \\ \forall x(\neg P(x) \vee \exists y((\neg P(y) \wedge \neg \neg R(a, x, y)) \vee \forall z(\neg S(y, z)))) \\ \forall x(\neg P(x) \vee \exists y((\neg P(y) \wedge R(a, x, y)) \vee \forall z(\neg S(y, z)))) \\ \forall x \exists y \forall z(\neg P(x) \vee ((\neg P(y) \wedge R(a, x, y)) \vee (\neg S(y, z))))\end{aligned}$$

**Замечание.** Одна формула может допускать много эквивалентных предваренных нормальных форм. Вид полученного результата зависит от порядка применения правил описанного выше алгоритма, а также от выбора символов при переименовании.

### 2. Удаление кванторов существования (приведение к сколемовской нормальной форме).

Этот этап позволяет сопоставить каждой формуле  $A$  некоторую формулу  $S_A$  очень простого строения. При этом гарантируется, что обе формулы  $A$  и  $S_A$  выполнимы либо нет одновременно.

Приведение к сколемовской нормальной форме осуществляется по следующим правилам:

- сопоставить каждой  $\exists$  - квантифицированной переменной список  $\forall$  - квантифицированных переменных, предшествующих ей, а также некоторую еще не использованную функцию  $f$ , число переменных у которой равно мощности составленного списка;
- в формуле заменить каждое вхождение  $\exists$  - квантифицированной переменной на терм в виде функции  $f$ , полученной на предыдущем шаге;
- удалить  $\exists$  - квантификацию.

### Примеры.

1.  $\forall x \forall y \exists u(P(x) \wedge (Q(u, y) \vee R(a, u, y)))$

подстановка:  $u \rightarrow f(x, y)$

сколемовская нормальная форма:  $\forall x \forall y (P(x) \wedge (Q(f(x, y), y) \vee R(a, f(x, y), y)))$

2.  $\forall x \exists y \forall z (\neg P(x) \vee ((\neg P(y) \wedge R(a, x, y)) \vee \neg S(y, z)))$

подстановка:  $y \rightarrow f(x)$

сколемовская нормальная форма:  $\forall x \forall z (\neg P(x) \vee ((\neg P(f(x)) \wedge R(a, x, f(x))) \vee \neg S(f(x), z)))$

3.  $A: \exists u \forall v \exists w \forall x \forall y \exists z M(u, v, w, x, y, z).$

подстановки:  $u \rightarrow a, w \rightarrow f(v), z \rightarrow g(v, x, y)$

сколемовская нормальная форма  $S_A: \forall v \forall x \forall y M(a, v, f(v), x, y, g(v, x, y))$

Доказано, что формула  $S_A \supset A$  общезначима. В силу этого второй этап алгоритма приводит не к логически эквивалентной форме некоторой исходной формулы  $A$ , а к эквивалентной в смысле выполнимости формуле  $S_A$ .

### 3. Удаление кванторов всеобщности.

В формуле  $S_A$  все переменные связаны либо квантором  $\forall$ , либо вообще не связаны. Кванторы всеобщности можно опустить. Действительно, пусть задана произвольная формула  $A$ , в которой определены некоторые свободные переменные  $x_1, \dots, x_n$ . Обозначим  $U_A$  -  $\forall$  - замыкание формулы  $A$  (т.е.  $\forall x_1 \dots \forall x_n A$ ), и  $E_A$  -  $\exists$  - замыкание  $A$  (т.е.  $\exists x_1 \dots \exists x_n A$ ). Тогда, можно доказать, что:

$$|=A \Leftrightarrow |=U_A, \quad |=\neg A \Leftrightarrow |=\neg E_A$$

В силу первой эквивалентности, кванторы  $\forall$  можно опускать.

### 4. Приведение к КНФ.

Полученную на предыдущих этапах формулу преобразовать к виду КНФ. Для этого можно использовать те же правила, которые использовались для приведения к КНФ формулы ИВ.

## Метод резолюций.

Для ИП не существует алгоритма, позволяющего распознать общезначимость, нейтральность или невыполнимость произвольной формулы. Это обусловлено существованием бесконечного числа возможных интерпретаций для формул ИП. Частные, но важные результаты в этом направлении были получены Эрбраном. Эти результаты позволяют упростить проверку выполнимости формул.

Основная идея такой проверки состоит в следующем. Пусть  $A$  – произвольная формула, а  $S_A$  – КНФ формулы  $A$ , полученная приведенным в пункте 2 алгоритмом. В общем случае доказано, что:

$$|=A \Leftrightarrow |=S_A, \quad |=\neg A \Leftrightarrow |=\neg S_A$$

В свою очередь, невыполнимость  $S_A$  эквивалентна случаю, когда  $S_A$  интерпретируется со значением Л при всех значениях входящих в нее переменных. В общем случае, число таких интерпретаций бесконечно и несчетно, что приводит к невозможности алгоритмической проверки. Поэтому было бы неплохо попытаться выявить такую область, которая тесно связана с КНФ и также удовлетворяет условию:  $S_A$  является невыполнимой тогда и только тогда, когда она принимает значение Л при всех интерпретациях именно на этой области. Такая область действительно существует и называется Эрбрановой.

Эрбранова область формулы  $S_A$  (приведенной к КНФ) – это минимальное множество  $H_A$ , удовлетворяющее следующим условиям:

- для любой константы  $a$ , имеющей вхождение в  $A$ , область  $H_A$  содержит соответствующую Эрбранову константу;
- если  $A$  не содержит константы, то тем не менее  $H_A$  содержит Эрбранову константу, обозначаемую с;

- c. для каждой  $n$  – местной функции  $f$ , имеющей вхождение в  $A$ , вводится функциональная константа  $F$ . Принадлежность элементов  $h_1, \dots, h_n$  к  $H_A$  влечет принадлежность к  $H_A$  также  $F(h_1, \dots, h_n)$ .

Элементы Эрбрановой области не имеют конкретного значения – они лишь простые семантические объекты.

**Пример.** Задана формула  $G = (P(x) \vee Q(x)) \wedge \neg Q(x)$ . Она не содержит констант и функций.

Поэтому  $H_G = \{c\}$ .

После приведения формулы  $A$  к КНФ каждый константе, переменной и функции сопоставляется значение из области  $H_A$ . К полученной в итоге формуле применяется алгоритм (метод) резолюций.

**Пример.** Доказать, что  $C$  является логическим следствием гипотез  $H_1, H_2$ .

$$\begin{aligned} H_1 &: \forall x(P(x) \supset Q(x)) \\ H_2 &: \forall x(Q(x) \supset R(x)) \\ C &: \forall x(P(x) \supset R(x)) \end{aligned}$$

На основании принципа дедукции это будет доказано, если  $(H_1 \wedge H_2 \wedge \neg C)$  невыполнимо. Приведем вначале последнюю формулу к КНФ.

$$\begin{aligned} &\forall x((P(x) \supset Q(x)) \wedge (Q(x) \supset R(x)) \wedge \exists x \neg(P(x) \supset R(x))) \\ &\forall x((\neg P(x) \vee Q(x)) \wedge (\neg Q(x) \vee R(x)) \wedge \exists x \neg(\neg P(x) \vee R(x))) \\ &\forall x((\neg P(x) \vee Q(x)) \wedge (\neg Q(x) \vee R(x)) \wedge \exists x (\neg \neg P(x) \wedge \neg R(x))) \\ &\forall x((\neg P(x) \vee Q(x)) \wedge (\neg Q(x) \vee R(x)) \wedge \exists y (\neg \neg P(y) \wedge \neg R(y))) \\ &\exists y \forall x((\neg P(x) \vee Q(x)) \wedge (\neg Q(x) \vee R(x)) \wedge (P(y) \wedge \neg R(y))) \\ &((\neg P(x) \vee Q(x)) \wedge (\neg Q(x) \vee R(x)) \wedge P(c) \wedge \neg R(c)) \end{aligned}$$

Эрбранова область содержит единственный элемент  $\{c\}$ . Поэтому последнюю формулу можно записать:

$$((\neg P(c) \vee Q(c)) \wedge (\neg Q(c) \vee R(c)) \wedge P(c) \wedge \neg R(c))$$

Остается теперь применить алгоритм резолюций и показать невыполнимость полученной формулы.

1.  $\neg P(c) \vee Q(c)$
2.  $\neg Q(c) \vee R(c)$
3.  $P(c)$
4.  $\neg R(c)$
5.  $Q(c)$  (1, 3)
6.  $\neg Q(c)$  (2, 4)
7. Л (5, 6)

### Алгоритм резолюции с унификацией.

Т.к. в общем случае дизъюнкты допускают бесконечно много интерпретаций, то метод резолюций в ИП является неэффективным. Более эффективный способ получается, если работать прямо с дизъюнктами без явного рассмотрения интерпретаций. Для этого используется операция **унификации**.

Пусть задана некоторая формула  $S$ , содержащая два дизъюнкта:

$$c_1 = l_1 \vee \dots, c_2 = \neg l_1 \vee \dots \text{ и } c_1, c_2 \in S$$

Предположим, что  $l_1$  и  $l_2$  являются унифицируемыми, т.е. можно указать подстановки

$$l_1 \rightarrow l'_1, l_2 \rightarrow l'_2,$$

такие, что  $l'_1 = l'_2 = l'$ . Тогда из  $c'_1, c'_2$  получается резольвента

$$r' = (c'_1 \setminus l') \vee (c'_2 \setminus \neg l')$$

И в итоге применимым является следующий **алгоритм резолюции с унификацией**:

1. при условии, что  $\mathcal{L} \notin S$  переходим к шагу 2. В противном случае – к шагу 5.
2. выбираем  $l_1, l_2, s_1, s_2$  такие, что  $s_1, s_2$  есть дизъюнкты или резольвенты,  $l_1 \in s_1$ ,  $\neg l_2 \in s_2$ , а  $l_1, l_2$  являются унифицируемыми.
3. вычисляем резольвенту  $r'$ .
4. заменяем  $S$  на  $S \wedge r'$  и переходим к шагу 1.
5. конец работы алгоритма.

#### **Замечание.**

Идея невыполнимости используется при доказательстве теорем. На практике же иногда бывает важно получить решение в явном виде. Для этого используется так называемый **ответный предикат** вида:

$$(\neg \text{заключение}(x) \vee \text{вывод}(x)), \text{ либо } (\text{заключение}(x) \wedge \text{вывод}(x)).$$

Эта идея, в частности, положена в основу известного языка PROLOG.

#### **Пример.**

Предположим, что кто-то из студентов желает договориться с преподавателем о консультации. Преподаватель находится на кафедре и требуется указать № телефона кафедры. Ведем несколько предикатов:

$J(p, n)$  – требуется соединить  $p$  по номеру телефона  $n$ ;

$A(x, l)$  –  $x$  находится в месте  $l$ ;

$T(l, m)$  –  $l$  имеет номер телефона  $m$ ;

$V(y, z)$  – означает, что  $y$  желает получить консультацию у  $z$ .

Для введенных предикатов можно отметить следующие свойства:

- a. если  $y$  желает получить консультацию у  $z$ :  $V(y, z)$ , и  $z$  находится в месте  $e$ :  $A(z, e)$ , то  $y$  может получить консультацию только в месте  $e$ :  $A(y, e)$ . Формально это можно записать:

$$(V(y, z) \wedge A(z, e)) \supset A(y, e), \text{ или в виде ДНФ: } A(y, e) \vee \neg V(y, z) \vee \neg A(z, e).$$

- b. если  $x$  находится в месте  $e$ :  $A(x, e)$  и  $e$  имеет номер телефона  $m$ :  $T(e, m)$ , то с  $x$  можно соединиться по телефону  $m$ :  $J(x, m)$ . Формально это можно записать:

$$(A(x, e) \wedge T(e, m)) \supset J(x, m), \text{ или в виде ДНФ: } J(x, m) \vee \neg A(x, e) \vee \neg T(e, m).$$

Допустим теперь, что фамилии студента и преподавателя конкретизированы:  $V$  (фамилия студента, фамилия преподавателя).

Получается следующая система предложений:

A1.  $J(x, m) \vee \neg A(x, e) \vee \neg T(e, m)$ ;

A2.  $A(y, e) \vee \neg V(y, z) \vee \neg A(z, e)$ ;

A3.  $A$  (фамилия преподавателя, кафедра)

A4.  $V$  (фамилия студента, фамилия преподавателя);

A5.  $T$  (кафедра, номер телефона);

A6.  $\neg J$  (фамилия студента, n)  $\vee$  вывод (n);  
Предложение A6 – это используемая в данном примере форма ответного предиката. Результат, очевидно, будет получен, если определить: вывод (n). Воспользуемся для этого алгоритмом резолюции с унификацией.

- A7.  $\neg A$  (фамилия студента, e)  $\vee \neg T$  (e, n)  $\vee$  вывод (n) (6, 1)  
*подстановки:* (x  $\rightarrow$  фамилия студента, m  $\rightarrow$  n).
- A8.  $\neg V$  (фамилия студента, z)  $\vee \neg A$  (z, e)  $\vee \neg T$  (e, n)  $\vee$  вывод (n) (7, 2)  
*подстановки:* (y  $\rightarrow$  фамилия студента).
- A9.  $\neg A$  (фамилия преподавателя, e)  $\vee \neg T$  (e, n)  $\vee$  вывод (n) (8, 4)  
*подстановки:* (z  $\rightarrow$  фамилия преподавателя).
- A10.  $\neg T$  (кафедра, n)  $\vee$  вывод (n) (9, 3)  
*подстановки:* (e  $\rightarrow$  кафедра).
- A11. вывод (номер телефона) (10, 5)  
*подстановки:* (n  $\rightarrow$  номер телефона).

### Дополнительные вопросы.

1. Исчисление предикатов и высказываний. Соотношение
2. ИИ и исчисление предикатов.

## Лекция 7

1. ХАРАКТЕРИЗАЦИЯ ДЕДУКТИВНОЙ ЛОГИКИ.
2. ДРУГИЕ МОДЕЛИ ДЕДУКТИВНЫХ ЛОГИК..
3. ПРИМЕРЫ ИНДУКТИВНЫХ ЛОГИК.

### Характеризация дедуктивной логики.

Ниже описана стандартная схема определения математической логики (включающей как разновидность логику исчисления высказываний и исчисления предикатов). В последующем это потребуется для определения неклассических логик. Схема, как это принято в математике, имеет следующий вид: *множества*  $\Rightarrow$  *характеристическая функция*  $\Rightarrow$  *абстракция (логическая переменная)*  $\Rightarrow$  (*операции над множествами*  $\Leftrightarrow$  *операции над логическим переменным*)  $\Rightarrow$  *построение логических систем и требования к ним*

#### 1. Множества.

Под множеством в математике принято понимать произвольный (конечный либо бесконечный) набор элементов. Сам этот набор, помеченный каким либо образом – чаще всего символом, и представляет собой множество. Множество можно задавать различными способами – *перечислением* (т.е. явным указанием элементов в него входящих) либо с помощью *принципа свертывания* (т.е. указанием общего свойства элементов множества). Все остальные способы (графический, табличный и проч.) являются производными от указанных.

**Замечание.** Способ задания множества в ИИ имеет первостепенное значение. Достаточно обратить внимание хотя бы на то обстоятельство, что перечислением можно задать только конечные множества.

Каждому множеству  $X$  можно поставить во взаимно-однозначное соответствие *характеристическую функцию*:

$$\mu_X(x) = \begin{cases} 1, & x \in X \\ 0, & \text{otherwise} \end{cases}$$

Над множеством вводятся операции: *объединения*, *пересечения*, *дополнения* и некоторые другие. Результат любой операции можно определить в терминах характеристической функции:

$$\begin{aligned} z \in \bar{X} &\Leftrightarrow \mu_{\bar{X}}(z) = 1 - \mu_X(z) = 1 \Leftrightarrow z \notin X \\ z \in X \cap Y &\Leftrightarrow \mu_{X \cap Y}(z) = \mu_X(x) \wedge \mu_Y(y) = 1 \Leftrightarrow x, y \in X \cap Y \\ z \in X \cup Y &\Leftrightarrow \mu_{X \cup Y}(z) = \mu_X(x) \vee \mu_Y(y) = 1 \Leftrightarrow x, y \in X \cup Y \end{aligned}$$

Исходя из приведенных соотношений, рассмотрение теоретико-множественных конструкций можно проводить в терминах соответствующих характеристических функций. Или, если ввести логическую переменную (элемент пространства  $B^2 = \{0,1\}$ ) вместо характеристической функции, то получим обычную булеву алгебру. Нетрудно установить эквивалентность между теорией множеств и булевой алгеброй.

#### 2. Построение логических систем.

Логика как предмет исследований в математике существует достаточно давно (Аристотель). Первоначально перед ней ставилась задача формализации “материальной” логики – мышления, умозаключений и проч. После введения в математическую практику теоретико-множественных конструкций и установления связи с логическими формализмами, булева логика (и некоторые естественные варианты ее расширения) стала рассматриваться несколько шире – как аппарат, позволяющий обосновать “правильность” математики в целом. Исходя из этого, вытекает и способ построения логических систем – через *аксиоматизацию*.  
**Примеры.**

1. Стандартная аксиоматическая система  $\Rightarrow$  доказательство “в смысле Гильберта”

Заданы аксиомы

$$\begin{array}{ll} (A1) & (X \supset (Y \supset X)) \\ (A2) & ((X \supset (Y \supset Z)) \supset ((X \supset Y) \supset (X \supset Z))) \\ (A3) & ((\neg X \supset \neg Y) \supset ((\neg X \supset Y) \supset X)) \end{array}$$

и правило вывода (*modus ponens*)

*Если  $X$  и  $(X \supset Y)$  – теоремы, то  $Y$  – есть теорема*

**2. Система натурального вывода  $\Rightarrow$  доказательство “в смысле Генциена”**  
Правила записываются в виде

$$\frac{S_1, \dots, S_n}{S}$$

что обозначает возможность вывода  $S$  из  $S_1, \dots, S_n$ .

**Базисные правила** (формализуют принцип монотонности рассуждений, вывода)

$$\frac{}{E, A \Rightarrow A} \quad u \quad \frac{E \Rightarrow A}{E, B \Rightarrow A}$$

**Правила введения связок**

$$\begin{array}{ll} (\wedge) & \frac{E \Rightarrow A \quad E \Rightarrow B}{E \Rightarrow A \wedge B} \\ (\vee) & \frac{\begin{array}{c} E \Rightarrow A \\ E \Rightarrow B \end{array}}{E \Rightarrow A \vee B} \quad \frac{E \Rightarrow A \vee B}{E \Rightarrow A \vee B} \\ (\supset) & \frac{\begin{array}{c} E, A \Rightarrow B \\ E \Rightarrow A \supset B \end{array}}{E, A \supset B} \\ (\neg) & \frac{\begin{array}{c} E, A \Rightarrow \perp \\ E \Rightarrow \neg A \end{array}}{E \Rightarrow \neg A} \\ (\equiv) & \frac{\begin{array}{c} E, A \Rightarrow B \quad E, B \Rightarrow A \\ E \Rightarrow A \equiv B \end{array}}{E \Rightarrow A \equiv B} \end{array}$$

**Правила удаления связок**

$$\begin{array}{ll} (\wedge) & \frac{E \Rightarrow A \wedge B}{\begin{array}{c} E \Rightarrow A \\ E \Rightarrow B \end{array}} \quad \frac{E \Rightarrow A \wedge B}{\begin{array}{c} E \Rightarrow A \vee B \\ E, A \Rightarrow C \quad E, B \Rightarrow C \end{array}} \\ (\vee) & \frac{E \Rightarrow A \vee B}{\begin{array}{c} E \Rightarrow A \\ E \Rightarrow B \end{array}} \\ (\supset) & \frac{E \Rightarrow A \supset B}{\begin{array}{c} E, A \Rightarrow B \\ E \Rightarrow A \end{array}} \\ (\neg) & \frac{\begin{array}{c} E \Rightarrow A \Rightarrow \perp \\ E \Rightarrow \neg A \end{array}}{E \Rightarrow \perp} \quad \frac{E \Rightarrow \neg \neg A}{E \Rightarrow A} \\ (\equiv) & \frac{\begin{array}{c} E \Rightarrow A \equiv B \\ E \Rightarrow A \Rightarrow B \end{array}}{E, A \Rightarrow B} \quad \frac{E \Rightarrow A \equiv B}{E, B \Rightarrow A} \end{array}$$

**3. Стандартная аксиоматическая система для квантификации  $\Rightarrow$  доказательство “в смысле Гильберта”**

Заданы аксиомы

$$\begin{array}{ll} (A1) & (P \supset (Q \supset P)) \\ (A2) & ((P \supset (Q \supset R)) \supset ((P \supset Q) \supset (P \supset R))) \\ (A3) & ((\neg P \supset \neg Q) \supset ((\neg P \supset Q) \supset P)) \\ (A4) & \forall x(P \supset Q) \supset (P \supset \forall x Q) \end{array}$$

правила вывода (*modus ponens*)

*Если  $X$  и  $(X \supset Y)$  – теоремы, то  $Y$  – есть теорема*

правило обобщения

*Если  $x$  не связана в теореме  $P$ , то  $\forall x P$  – теорема*

**4. Система натурального вывода для квантификации  $\Rightarrow$  доказательство “в смысле Генциена”**

К тому, что введено в примере 3 добавляются

#### *Правила введения кванторов*

$$\begin{array}{c} (\forall) \quad \frac{E \Rightarrow A(x)}{E \Rightarrow \forall x A(x)} \\ (\exists) \quad \frac{E \Rightarrow A(t)}{E \Rightarrow \exists x A(x)} \end{array}$$

#### *Правила удаления кванторов*

$$\begin{array}{c} (\forall) \quad \frac{E \Rightarrow \forall x A(x)}{E \Rightarrow A(t)} \\ (\exists) \quad \frac{E \Rightarrow \exists x A(x)}{E \Rightarrow A(c)} \end{array}$$

Примечание.  $x, t, c$  - термы, с ограничением на вхождение в формулы  $E, A$ .

В таком виде логические системы (логики) представляют теоретический интерес сами по себе и как язык формализации теории множеств. Приведенный список примеров не исчерпывает существующих логических систем.

### **3. Дедуктивность логических систем.**

Исследования логических систем (безотносительно к их применимости на практике) развиваются в следующих направлениях:

- выразительная сила языка, используемого для построения логики;
- сравнение логических систем (в этом смысле логики в примерах 1, 2 и 3, 4 примерно одинаковы);
- конструктивность системы (существование и реализуемость, численные характеристики соответствующих алгоритмов вывода).

Общим в построении всех приведенных логик является возможность использования **принципа** (метатеоремы) **дедукции** для построения всех возможных теорем из сформулированных (или доказанных). Этот принцип для примеров 1, 2 может быть сформулирован следующим образом:

*Необходимым и достаточным условием выводимости  $A$  из гипотез (теорем)  $E, B$  является выводимость  $(B \supset A)$  из  $E$ . Или формально:*

$$E, B \vdash A \Leftrightarrow E \vdash (B \supset A)$$

Принцип дедукции:

- является верным в полной системе типа 1, 2 и может оставаться верным в неполной системе;
- не применим к системам с квантификацией (типа 3, 4), если в выводе формулы  $A$  используется правило обобщения по переменной, имеющей свободное вхождение в  $B$ . (например -  $P \vdash \forall x P$  является законным выводом, в отличие от  $P \supset \forall x P$ ).

Следствием принципа дедукции является универсальный механизм вывода, который применим во всех логических системах. Этот механизм – *метод резолюций*. Еще одним следствием являются два принципа математической индукции (**см. лекцию 4**).

Следствием анализа дедуктивных систем является их характеризация в терминах разрешимости (по определению: *теория называется разрешимой, если существует алгоритм, позволяющий за конечное число шагов определить является ли исследуемая формула теоремой, ее отрицанием либо не тем или другим*). Оказывается, что

- *исчисление высказываний* (примеры 1, 2 и некоторые их бескванторные аналоги) является разрешимым;
- *исчисление предикатов* (примеры 3, 4) неразрешимо.

Фактически результаты о разрешимости (впервые подобный результат доказан К.Геделем) определяют ограничения и возможности использования дедуктивных логик. (Более точную формализацию результатов о разрешимости см. курс “Математическая логика”.)

### **4. Дедуктивная логика и реальный мир: возможность обоснования правильности рассуждений и выводов.**

Поставим теперь следующий вопрос: как соотносится реальный мир и формальные теории, которые могут быть построены средствами описанных выше логик? Или иначе: насколько хороши средства дедуктивной логики для моделирования ЕИ?

Для ответа на этот вопрос с самых общих позиций обычно имеется:

*ЕИ задача*

*Модель формальной логики (ИИ)*

1. некоторые объекты и связи между ними. Число объектов и связей может быть произвольным
2. семантика функционирования этих объектов и взаимодействие по установленным связям
3. определенные цели функционирования задачи
4. методология анализа и/или синтеза может быть известна (если задача имеет решение) либо нет (в противном случае)
1. язык логического символизма и правила построения формул (высказываний)
2. некоторый набор тавтологий
3. правила манипулирования формулами, по которым получаются новые формулы и только формулы
4. алгоритм, позволяющий осуществлять классификацию любой исследуемой формулы. В худшем случае этот алгоритм (результат) не сможет закончить работу
5. ограниченная интерпретация формул в пространстве {истина, ложь}

Теперь сформулированный выше вопрос можно сделать более конструктивным, т.к. по форме он переходит к установлению отношения между моделью ЕИ (описанной приведенными выше условиями) и теми моделями, которые порождаются дедуктивной логикой. Формально подобное соотношение вряд ли может быть установлено. Но наиболее правдоподобным является предположение:

**как бы ни была построена дедуктивная логика (по числу аксиом, правил вывода и др. компонентов), она никогда не будет достаточной для моделирования ЕИ.**

Сформулированное предположение может быть обосновано, по крайней мере, следующим образом:

- даже при заранее зафиксированных целях решения задачи, число объектов/связей может быть увеличено без ущерба для семантики задачи (на уровне ЕИ), т.к. никогда не может быть известно какие объекты/связи и как влияют на достижение цели. В противном случае задача вырождалась бы в простую вычислительную процедуру;
- ограничивая число объектов/связей в задаче, приходится считаться с вносимой при этом неопределенностью;
- желательно, чтобы эта неопределенность как-то учитывалась и в конечном результате. А это практически невозможно при использовании аппарата дедуктивных логик, т.к. они категоричны (ответ м.б. интерпретирован только в терминах {истина, ложь}).

Приведенные аргументы не исчерпывают всего, что может быть сказано на эту тему.

## Другие модели дедуктивных логик.

Определим вначале класс дедуктивных логик. К этому классу обычно относят логики, для которых имеет место: *полнота, непротиворечивость и разрешимость*. Так получается класс дедуктивных логик *по построению*. Вообще говоря, данный класс является довольно узким (примеры 1, 2). Кроме того, логики типа построенных в примере 3, 4, тоже не отнесешь к разряду неклассических. Причина заключается в том, что они достаточно хорошо изучены и имеют неплохие свойства. Но главное – при определенных условиях на информацию (знания, если говорить в контексте ИИ) и ограничениях на аппарат, эти и многие другие логики становятся дедуктивными в обычном смысле. Так получается класс дедуктивных логик *по аппарату*. Объединение всех логик дедуктивных как по аппарату, так и по построению можно назвать *классическими*. В противоположность этому, все остальные виды логик будем считать *неклассическими*.

Указанное объединение имеет смысл и по другим причинам чисто технического свойства. Дело в том, что дифференциацию всех логик одного класса (по крайней мере тех, которые выше названы классическими) можно провести стандартным математическим способом – модификацией описания либо свойств.

### 1. Возможные направления модификации дедуктивных логик.

Выше были описаны принципы построения некоторых логических систем. В основе этого построения лежит несколько моментов:

- для описания конкретной логики должны быть заданы пространства – исходное (где определены переменные) и результирующее (где определены значения операций над переменными). В нашем случае это пространство  $B^2$ ;
- определены наборы исходных аксиом и правил вывода.

Ясно, что конструкции, которые получаются в результате, должны допускать характеристизацию в терминах свойств. В нашем случае это такие свойства:

- **рефлексивность** -  $\{p_1, \dots, p_n, q\} \mapsto q$
- **монотонность** -  $\{p_1, \dots, p_n\} \mapsto q \Rightarrow \{p_1, \dots, p_n, r\} \mapsto q$
- **транзитивность** -  $\{p_1, \dots, p_n\} \mapsto r \wedge \{p_1, \dots, p_n, r\} \mapsto q \Rightarrow \{p_1, \dots, p_n\} \mapsto q$

Замечание: иногда к перечисленным свойствам добавляют **антисимметричность**

$$\{p_1, \dots, p_n, r\} \mapsto q \Leftrightarrow \{p_1, \dots, p_n, \neg q\} \mapsto \neg r$$

Характеризация логик в терминах свойств очень интересна и плодотворна в теоретическом смысле, т.к. позволяет систематизировать все логики с точки зрения их соотношения (к примеру – как алгебра и подалгебра).

Из сказанного выше можно сделать вывод о направлениях возможной модификации логик:

- **изменением  $B^2$ ;**
- **манипуляцией над множеством аксиом либо правил вывода (их исключением или добавлением новых);**
- **описанием структур (через свойства) и определением конструкции подструктур либо надструктур.**

В последнем случае можно ожидать просто получения более систематического описания новых логик, но не получения чего-либо принципиально нового. Так рассмотренная ниже модальная логика может быть получена во втором либо третьем направлениях. Чаще всего значение имеет историческая традиция.

## 2. $k$ – значная логика.

Направление модификации:  $B^2$  заменяется на  $K = \{0, 1, \dots, k-1\}$  и рассматриваются всевозможные конечные декартовы произведения  $K^l$  ( $l$  – натуральное, может быть и бесконечное число).

Дополнительная модификация аппарата: необходимо определить соответствующие логические операции.

Преимущества с позиции применения в ИИ: расширяются возможности интерпретации, но логика остается по-прежнему дискретной.

Пусть  $x \in K^l$ . Функцию алгебры логики  $f(x)$ , равно как и соответствующие операции, можно определить через таблицу

$x_1$	...	$x_l$	$f(x_1, \dots, x_l)$
0	...	0	$f(0, \dots, 0)$
0	...	1	$f(0, \dots, 1)$
...	...	...	...
1	...	0	$f(1, \dots, 0)$
1	...	1	$f(1, \dots, 1)$

Функции  $f$  могут быть определены через “элементарные” (отрицание, конъюнкцию, дизъюнкцию и др.) различным образом. При этом может учитываться и содержательный смысл семантики задачи.

### Примеры.

1.  $x = x + 1 \pmod{k}$ . Эта операция представляет собой обобщение отрицания в смысле “циклического” сдвига значений.
2.  $Nx = k - 1 - x$ . Здесь  $Nx$  (или  $\sim x$ ) является другим обобщением отрицания в смысле “зеркального” отображения значений. В литературе оно называется *отрицанием Лукасевича*.

$$3. \quad I_i(x) = \begin{cases} k-1, & \text{при } x=i \\ 0, & \text{иначе} \end{cases}$$

Функции  $I_i(x)$  при  $i \neq k-1$  являются обобщением некоторых свойств отрицания.

4.  $\min(x_1, x_2)$  - обобщение конъюнкции.
5.  $x_1 x_2 \ (\text{mod } k)$  - еще одно обобщение конъюнкции.
6.  $\max(x_1, x_2)$  - обобщение дизъюнкции.
7.  $x_1 + x_2 \ (\text{mod } k)$  - еще одно обобщение дизъюнкции.

С использованием каких-либо операций, из перечисленных в примерах, нетрудно построить более сложные функции логики, формулы и т.д.

Что касается построения логической системы  $k$ -значной логики, то здесь также можно использовать аксиоматический подход. При этом аксиомы и правила вывода могут быть аналогичны тем, которые использованы в примерах 1-4 (п.1.2.2 из §1).

По отношению к  $k$ -значной логике можно ставить проблему ее полноты, непротиворечивости и разрешимости. Во многом соответствующие результаты аналогичны соответствующим результатам для двузначных логик. Правда, рост значности приводит все-таки к определенным усложнениям формулировок и доказательств. Более того, с помощью системы специальных операций конструкции  $k$ -значной логики могут быть сведены к системе булевозначных функций. И при этом сохраняется эквивалентность систем.

### **3. Интуиционистская логика.**

Направление модификации: из логических систем типа построенной в примере 1 (п.1.2.2 из §1), исключаются все аксиомы, которые содержат в явном виде либо приводят к “закону исключенного третьего”:  $x \vee \neg x$ .

Дополнительная модификация аппарата: необходимо определить соответствующие логические операции. По правилам интуиционистской логики ни одна из операций ( $\neg, \wedge, \vee, \supset$ ) не выражается через другие.

Преимущества с позиции применения в ИИ: Из логики исключаются формулы, истинность которых может быть оценена “от противного”. Это “конструктивизирует” вывод в следующем смысле: допустимыми считаются только такие формулы, доказательство которых либо они сами могут быть построены в явном виде. В обычной логике достаточным считается соответствующее доказательство существования. Логика остается дискретной, но по сравнению с обычной становится более конструктивной.

Замечание. Для интуиционистской логики всякое утверждение представляет собой запись конструкции. Для “интуициониста” заявление об истинности  $\alpha$  равноценно высказыванию “я выполнил (может быть и умозрительно) построение того, что описывается предложением  $\alpha$ ”. Аналогично предложение  $\neg\alpha$  есть запись конструкции, которая показывает, что  $\alpha$  не имеет места. С этой точки зрения “закон исключенного третьего” формулируется следующим образом:

“либо я конструктивно доказал  $\alpha$ , либо я конструктивно доказал ложность  $\alpha$ ”

Если мы теперь возьмем в качестве  $\alpha$  некоторое утверждение типа теоремы Гольдбаха, то предложение  $\alpha \vee \neg\alpha$  не истинно. Действительно, про данную теорему до сих пор не известно, верна она или нет. Следовательно, мы не можем утверждать, что “ $\alpha$  истинно” или “ $\alpha$  ложно”, пока конструктивно не установлено, который из этих случаев имеет место.

Пример. (аксиоматическое построение интуиционистской логики)

Заданы аксиомы (ниже  $\alpha, \beta, \gamma$  - обозначают произвольные предложения)

1.  $\alpha \supset (\alpha \wedge \alpha)$
2.  $(\alpha \wedge \beta) \supset (\beta \wedge \alpha)$
3.  $(\alpha \supset \beta) \supset ((\alpha \wedge \gamma) \supset (\beta \supset \gamma))$
4.  $((\alpha \supset \beta) \wedge (\beta \supset \gamma)) \supset (\alpha \supset \gamma)$
5.  $\beta \supset (\alpha \supset \beta)$
6.  $(\alpha \wedge (\alpha \supset \beta)) \supset \beta$
7.  $\alpha \supset (\alpha \vee \beta)$
8.  $(\alpha \vee \beta) \supset (\beta \vee \alpha)$
9.  $((\alpha \supset \gamma) \wedge (\beta \supset \gamma)) \supset ((\alpha \wedge \beta) \supset \gamma)$
10.  $\neg \alpha \supset (\alpha \supset \beta)$
11.  $((\alpha \supset \beta) \wedge (\alpha \supset \neg \beta)) \supset \neg \alpha$

и правило вывода (*modus ponens*)

*Из предложений  $\alpha$  и  $\alpha \supset \beta$  можно вывести предложение  $\beta$*

Замечание. По сути изложенная в примере система (если к ней добавить “закон исключенного третьего”) эквивалентна приведенной в примере 1 (п. 1.2.2 из §1).

Что касается правил определения операций ( $\neg, \wedge, \vee, \supset$ ) и выводимости (или доказательства), то это довольно сложные конструкции. С ними лучше ознакомиться в специальной литературе.

По отношению к такой логике снова можно говорить о фундаментальных ее свойствах как полнота, непротиворечивость и разрешимость. Во многом соответствующие результаты аналогичны соответствующим результатам для двузначных логик. Но при этом сами конструкции значительно “беднее”. В настоящее время предпринято ряд успешных попыток построения интуиционистской математики в целом. При этом по сравнению с классическим случаем приходится исключать все теоремы существования и все, что доказывается от противного. Получается математика, в которой остаются только конструктивные (алгоритмические) разделы. Интуитивно понятно, что это лучше соответствует реальности (и, следовательно, ИИ). Хотя с точки зрения интерпретации в терминах {истина, ложь, не знаю} возможности логики не сильно расширяются. Поэтому можно говорить о многозначной интуиционистской логике.

#### 4. Модальная логика.

Направление модификации: Вводятся дополнительные операторы (модальные) и/или расширяется условие монотонности (п. 2.1 § 2). Второй вариант используется при алгебраическом (структурном) описании множества модальных логик.

Дополнительная модификация аппарата: Требуется уточнение понятий общезначимости, выводимости и введение дополнительных аксиом, правил вывода, описывающих использование модальности.

Преимущества с позиции применения в ИИ: Объединяет преимущества логик из пп 2.2, 2.3. Правда конструктивность требует уточнения.

Замечание. В обыденном языке часто говорят о допустимости чего-либо. Большая часть фраз языка может быть истинной либо ложной в зависимости от обстоятельств, текущего момента и т.д. Для выражения этого можно ввести специальные операторы – **модальные**, которые будут выражать, к примеру, “возможность” / “необходимость”. Это должно способствовать расширению интерпретирующей силы языка логики. Ниже речь будет идти только об указанных модальностях, хотя в литературе изучаются и другие их типы. Так логика знания (основанная на модальностях “знаю” / “верю”), временная (тэмпоральная) логика (основанная на парных модальностях “редко” / “часто”) представляют собой также разновидности модальной логики. В принципе, все они строятся по одной схеме.

Модальные операторы вводятся по аналогии с кванторами всеобщности и существования. Для обозначения модальности “необходимо” используется символ  $\Box$ . Формула  $\Box F$  читается “необходимо, чтобы  $F$ ”. Двойственный для  $\Box$  оператор обозначается символом  $\Diamond$ . Формула  $\Diamond F$  читается “возможно, что  $F$ ”. Один из этих операторов принимается за основной, а другой определяется через него и отрицание:  $\Box F = \neg \Diamond \neg F$ .

Использование таких модальных операторов предполагает некоторую интерпретацию. В частности, естественной является такая:

для оператора  $\Box$ :

- $\Box F$  истинно тогда и только тогда, когда
- $F$  необходимо истинно (a)
- $F$  абсолютно истинно (b)

для оператора  $\Diamond$ :

- $\Diamond F$  истинно, если
- $F$  может оказаться истинным (a)
- $F$  условно истинно (b)

Синтаксис модальной логики ничем не отличается от синтаксиса логики исчисления предикатов. А семантика – отличается. В обычной логике рассматривается два класса формул – истинные (1) и ложные (0). В модальной же вводится несколько классов, семантика которых определяется внелогическими соображениями. Обозначим через N класс “необходимых” высказываний, через P – “возможных”, через I – “невозможных” и через C – “нейтральных” (возможно ложных). Допустим теперь, что никакое высказывание не принадлежит одновременно N и C или I и P. Это сразу определяет закон противоречия. Далее, класс N содержится в P, а класс I – в C. Это может быть отражено в виде законов следования таким образом:

любое необходимое высказывание возможно,  
любое невозможное высказывание не является необходимым.

Существуют высказывания, которые являются возможными и нейтральными одновременно. Их называют “проблематичными” (обозначим это множество через U). Теперь можно сформулировать “закон исключенного четвертого”:

любое высказывание принадлежит либо N, либо U, либо I

Теперь можно определить таблицы истинности для логических операций и модальных операторов. Для этого сопоставим всем подмножествам N, U, I числовые значения 2, 1 и 0 соответственно. Тогда таблицы будут иметь вид:

		$F \wedge G$			$F \vee G$			$F \supset G$			$F \equiv G$			
		$\neg F$	G	0	1	2	0	1	2	0	1	2	$\Box F$	$\Diamond F$
F	0	2	0	0	0	0	1	2	2	2	2	2	1	0
	1	1	0	1	1	1	1	1	2	1	2	2	1	0
	2	0	0	1	2	2	2	2	0	1	2	0	1	2

Далее в модальной логике все строится по такой же схеме, как и в обычной. Вводится набор аксиом (только в них могут участвовать и аксиомы с операторами модальности) и фиксируются некоторые правила вывода.

### Пример.

Аксиома может быть такой:  $\vdash \Box(F \Rightarrow G) \Rightarrow (\Box F \Rightarrow \Box G)$ ;

правило: если  $\vdash F$ , то  $\vdash \Box F$ .

Для модальной логики также имеются результаты о полноте, непротиворечивости и разрешимости. Для вывода используется аналог метода резолюций.

### Замечания.

1. В зависимости от семантики приведенные выше таблицы истинности могут иметь и другой вид.
2. Модальная логика является примером немонотонной логики. В ней условие монотонности редуцируется до ограниченной монотонности:

*Если  $\{p_1, \dots, p_n\} \vdash q$  и  $\{p_1, \dots, p_n, r\} \vdash r$ , то  $\{p_1, \dots, p_n, q\} \vdash r$*

Такая монотонность не позволяет моделировать взаимно несовместимые альтернативы рассуждений. А нужно это по той причине, что модальная и другие подобные ей логики предназначены для моделирования модифицируемых рассуждений (зависящих, к примеру, от времени, условий и т.п.). Такие рассуждения не могут быть общезначимыми в классическом смысле. Поэтому для немонотонных логик требуется модификация понятий общезначимости, выводимости и некоторых других.

## **5. Дедуктивные логики и ИИ.**

Все рассмотренные выше логики являются дедуктивными потому, что при некоторых естественных ограничениях на аксиомы и правила вывод новой формулы имеет “доказательную силу”. В этом смысле доказываемое (выводимое) утверждение является, по сути, теоремой. Многочисленные приложения таких логик к ИИ можно трактовать в контексте доказательства теорем.

В чисто техническом смысле такое доказательство может быть построено как в прямом так и в обратном направлениях. Соответствующая дедуктивная схема (вывод) называется прямой и обратной. В системах *прямой дедукции* выводы применяют к фактам и правилам, чтобы построить новые факты, правила либо цели. В системах же *обратной дедукции* выводы применяют к цели и правилам, чтобы получить новые частичные цели. Исходя из содержательного смысла можно сказать, что обратная дедукция лучше приспособлена для ответа на вопросы типа: “как достичь поставленной цели?”, “что нужно сделать, чтобы достичь определенной цели?” и т.д. Поэтому в системах ИИ обратная дедукция чаще всего используется для *объяснения*.

Заметим, наконец, что в техническом смысле дедуктивность не имеет никакой специфики. Именно доказательность вывода играет решающую роль в определении отличительной особенности дедуктивных логик. Она же определяет и области применения таких логик в ИИ.

## **Примеры индуктивных логик.**

Индуктивная логика – это логика подтверждений. При этом “сила” подтверждения может быть определена как в качественном, так и в количественном смысле. Но во всех случаях речь идет об определении порядка на множестве утверждений, по отношению к которым не может быть применена дедуктивная логика. Причины этого могут быть различными, но во всех случаях причины чисто технического характера – вторичны. Т.е. по построению индуктивная логика может ничем не отличаться от дедуктивной. Главная же причина заключается в том, что в индуктивной логике используются фактические истины (т.е. такие факты, знания истинность или ложность которых обосновывается только опытным, экспериментальным путем). Из-за этого истинность вывода зависит от истинности указанных фактов. И более того, из конечного вывода такие факты просто не могут быть исключены (элиминированы) в процессе доказательства.

Если же посмотреть на причины с теоретико-множественной точки зрения, то картина будет приблизительно следующей. Индуктивная логика, построенная каким-либо образом, описывает некоторое множество утверждений. Точного соответствия между этим множеством и реальностью не может быть установлено. Поэтому с позиции логических построений реальность является не полностью или частично определенной. Тем не менее в реальности существуют некие знания по поводу которых необходимо высказать суждение (например, в терминах истинности). Как это может быть реализовано? В частности определением степеней истинности, т.е. числом не из множества {0,1}, а из интервала [0,1]. Остается определиться с порядком на множестве степеней истинности, может быть ввести этот порядок в логику и то, что получается в результате как раз и будет индуктивной логикой. Иногда такие логики называют еще *эвристическими, эмпирическими, опытными* и т.д.

Проделать все сказанное можно и с дедуктивными логиками. Из-за этого возникает очень интересный и содержательный вопрос о соотношении дедуктивных и индуктивных логик.

Рассмотрим простые примеры, которые демонстрируют возможную модификацию вывода с тем, чтобы получить индуктивный вывод. Обсудим также применимость такого вывода в ИИ.

### **1. Логика подтверждений Д.Пойа.**

В этой логике какой-то общей схемы нет. Она по сути ориентирована на установление связи между доказательными выводами (которыми оперирует дедуктивная логика) и выводами подтверждения (индуктивная логика). В итоге получается логика “правдоподобных рассуждений”.

Для установления такой связи необходимо каждой известной доказательной схеме вывода поставить в соответствие эвристическую схему. Предполагается, что последняя должна рассматриваться как конкретное правило или критерий подтверждения, которыми можно пользоваться в подходящих случаях.

Соответствие доказательных и эвристических схем в логике Д.Пойа приводится ниже в таблицах 1-3.

Таблица 1

	<b>доказательная схема</b>	<b>эвристические схемы</b>		
<b>название</b>	modus ponens	затушеванная доказательная	затушеванная индуктивная	индуктивная
<b>посылки</b>	$A \Rightarrow B$ A истинно	$A \Rightarrow B$ A более правдоподобно	$A \Rightarrow B$ A менее правдоподобно	$A \Rightarrow B$ A ложно
<b>заключение</b>	$B$ истинно	$B$ более правдоподобно	$B$ несколько менее правдоподобно	$B$ менее правдоподобно

Таблица 2

	<b>доказательная схема</b>	<b>эвристические схемы</b>		
<b>название</b>	modus tollens	затушеванная доказательная	затушеванная индуктивная	индуктивная
<b>посылки</b>	$A \Rightarrow B$ $B$ ложно	$A \Rightarrow B$ $B$ менее правдоподобно	$A \Rightarrow B$ $B$ более правдоподобно	$A \Rightarrow B$ $B$ истинно
<b>заключение</b>	$A$ ложно	$A$ менее правдоподобно	$A$ несколько более правдоподобно	$A$ более правдоподобно

Таблица 3

	<b>доказательная схема</b>	<b>эвристические схемы</b>		
<b>название</b>	modus несомненности	затушеванная доказательная	затушеванная индуктивная	индуктивная
<b>посылки</b>	$A   B$ $B$ истинно	$A   B$ $B$ более правдоподобно	$A   B$ $B$ менее правдоподобно	$A   B$ $B$ ложно
<b>заключение</b>	$A$ ложно	$A$ несколько менее правдоподобно	$A$ несколько более правдоподобно	$A$ более правдоподобно

(| - штрих Шефера, исключающее или). Каждой доказательной схеме соответствует по три эвристических, причем в каждой из них эвристичность получается за счет ослабления одной из посылок. В т. 1, 2  $A, B$  – неравноценные посылки и поэтому переход к индуктивной выводимости осуществляется за счет ослабления меньших. В т. 3  $A, B$  – равноценные посылки и поэтому выбор ослабляемой несущественен.

Как видно из доказательной схемы, процесс непосредственного доказательства может иметь направленность от посылок к следствиям либо наоборот. Нет противопоказаний, чтобы не предположить непрерывность подобных процессов и построить (как это сделано в эвристических схемах) упорядочение следствий / посылок в зависимости от тенденции направления истинности. Кроме того, в доказательных схемах истинность заключения безотносительна, отделена и зависит только от состояния логических переменных в посылках. Иначе обстоит дело в эвристических схемах. Для выведения заключения в таких схемах сверх информации, заключенной в посылках, всегда требуется некоторая внешняя по отношению к этим посылкам информация. От наличия или изменения последней существенно зависит ценность самого заключения.

В принципе, логические схемы могут строиться на базе уже введенных типов правил. По крайней мере, правдоподобные рассуждения на этом построить возможно. Но если необхо-

димо каким-то образом измерять численные величины “силы” заключений, то такого языка недостаточно.

## 2. Вероятностная логика.

Построение отношений следования между событиями, которым могут быть приписаны вероятностные характеристики, существенно зависит от того, в какой концепции (классической либо частотной) определены вероятности событий. Разница имеет семантический характер. Так в частотной вероятность может быть приписана только последовательности событий, но нециальному событию. Ниже изложен подход к построению систем вероятностной логики, в котором указанное различие не учитывается. Оно как бы переносится в плоскость измерений вероятности.

### 2.1. Вероятностная логика подтверждений.

Строится по аналогии с логикой правдоподобных рассуждений Д.Пойа. В ее основе лежит следующая идея. Вероятность в любой из концепций имеет субъективный характер и поэтому численное значение вероятности нельзя приписать какому-либо событию. Смысл имеет только отношение событий и тенденция роста / уменьшения вероятности некого одного события по сравнению с другим. Это снова приводит к необходимости установления связи между доказательными и эвристическими схемами

Соответствие доказательных и эвристических схем в вероятностной логике отношений приводится ниже в таблицах 4-6.

Таблица 4

	<b>доказательная схема</b>	<b>эвристические схемы</b>		
<b>название</b>	modus ponens	затушеванная до-казательная	затушеванная ин-дуктивная	индуктивная
<b>посылки</b>	$A \Rightarrow B$ $P(A) = 1$	$A \Rightarrow B$ $\uparrow P(A)$	$A \Rightarrow B$ $\downarrow P(A)$	$A \Rightarrow B$ $P(A) = 0$
<b>заключение</b>	$P(B) = 1$	$\uparrow P(B)$	$\downarrow P(B)$	$\downarrow P(B)$

Таблица 5

	<b>доказательная схема</b>	<b>эвристические схемы</b>		
<b>название</b>	modus tollens	затушеванная до-казательная	затушеванная ин-дуктивная	индуктивная
<b>посылки</b>	$A \Rightarrow B$ $P(B) = 0$	$A \Rightarrow B$ $\downarrow P(B)$	$A \Rightarrow B$ $\uparrow P(B)$	$A \Rightarrow B$ $P(B) = 1$
<b>заключение</b>	$P(A) = 0$	$\downarrow P(A)$	$\uparrow P(A)$	$\uparrow P(A)$

Таблица 6

	<b>доказательная схема</b>	<b>эвристические схемы</b>		
<b>название</b>	modus несовместимости	затушеванная до-казательная	затушеванная ин-дуктивная	индуктивная
<b>посылки</b>	$A \mid B$ $P(B) = 1$	$A \mid B$ $\uparrow P(B)$	$A \mid B$ $\downarrow P(B)$	$A \mid B$ $P(B) = 0$
<b>заключение</b>	$P(A) = 0$	$\downarrow P(A)$	$\uparrow P(A)$	$\uparrow P(A)$

( $\uparrow P(B), \downarrow P(A)$  – означает увеличение и уменьшение численного значения вероятности соответственно).

По сути такого типа вероятностная логика представляет собой логику подтверждений, построенную на основе вероятностной терминологии.

## 2.2. Аксиоматическое построение вероятностной логики.

### а). Логика подтверждений Р.Карнапа.

Представляет собой обобщение изложенной выше вероятностной логики подтверждений, но строится путем аксиоматического определения функции подтверждения. Последняя вводится следующим образом

$$c: h \times e \rightarrow [0,1]$$

где  $h, e$  - гипотеза и основание соответственно. Степень подтверждения определяется как вероятность гипотезы  $h$  в отношении к ее эмпирическим данным  $e$ . Степень подтверждения при таком определении является числом, которое исходя из семантики его определения можно трактовать как вероятность.

Такая функция должна удовлетворять некоторым условиям, чтобы с ее помощью можно было построить систему индуктивной логики. В логике подтверждений Карнапа эти условия имеют вид:

1.  $c$  – имеет значение для любых двух высказываний  $h$  и  $e$ , если  $e$  не является логически ложным.
2. значение  $c$  – является точным количественным измерением вероятности.
3.  $c$  – удовлетворяет четырем требованиям адекватности
  - из логической эквивалентности оснований вытекает равенство степеней подтверждения гипотезы

$$e \equiv e' \Rightarrow c(h, e) = c(h, e')$$

- то же самое имеет место для логически эквивалентных гипотез

$$h \equiv h' \Rightarrow c(h, e) = c(h', e)$$

- степень подтверждения конъюнкции двух гипотез равна

$$c(h \wedge h', e) = c(h, e) \cdot c(h', e \wedge h)$$

- степень подтверждения дизъюнкции двух гипотез (при условии, что все три высказывания образуют ложную конъюнкцию) равна

$$c(h \vee h', e) = c(h, e) + c(h', e)$$

4. степень подтверждения тавтологического высказывания равна 1:  $c(t, e) = 1$  ( $t$  – тавтология).

Задача логики, в которой функция  $c$  используется для измерения подтверждения высказывания  $h$  при условии  $e$ , заключается в построении такой  $c$ , которая:

- a). удовлетворяет указанным выше условиям (и поэтому является функцией подтверждения);
- b). позволяет поставить в соответствие любым высказываниям некоторые сравнительные численные характеристики (и потому является логикой подтверждений).

### б). Логика подтверждений Х.Джефриса.

Идея, на которой строится данная логика, состоит в том, что степени подтверждения различных высказываний по отношению к одному основанию (посылке) могут быть упорядочены. Для определения такого порядка необходимо, чтобы имела место следующая система аксиом (ниже через  $P$  обозначена степень подтверждения).

**Аксиома 1.** При данном  $p$ , если  $q$  и  $r$  – высказывания, то либо  $q$  более вероятно, чем  $r$ , либо  $q$  менее вероятно, чем  $r$ , либо они равновероятны. При этом не более одного из этих утверждений истинно.

**Аксиома 2.** Отношение “ $q$  более вероятно, чем  $r$ ” транзитивно. Т.е. при данном  $p$ ,  $q$  более вероятно, чем  $r$  и  $r$  более вероятно, чем  $s$ , влечет  $q$  более вероятно, чем  $s$ .

**Аксиома 3.** При данном  $p$ , если  $q, r'$  и  $q', r'$  исключают друг друга, и, кроме того,  $q, q'$  и  $r, r'$  – одинаково подтверждаются  $p$ , то  $q \vee r$  и  $q' \vee r'$  также одинаково подтверждаются  $p$ . Эта аксиома символически может быть записана

$$P(q \vee r, p) = P(q, p) + P(r, p)$$

**Аксиома 4.** При данном  $p$  степень подтверждения конъюнкции высказываний равна произведению степени подтверждения первого высказывания при данном  $p$  на степень подтверждения второго высказывания при данных  $p, q$ . Символически

$$P(q \wedge r, p) = P(q, p) \cdot P(r, p \wedge q)$$

Предполагается также, что данные, к которым относятся высказывания, не должны быть

взаимно противоречивыми (т.к. в противном случае из их конъюнкции можно вывести любое высказывание).

### c). Логика релевантности.

Недостатком всех описанных выше логических систем является отсутствие аппарата для измерения численного значения вероятности, подтверждающей высказывание. Для того, чтобы устранить указанный недостаток, требуется определение значений вероятности логических операций над высказываниями ввести в систему аксиом. Таким путем строится релевантная логика. Рассмотрим пример системы аксиом для такой логики.

**Пример.** (A, B, C, D – некоторые события, а / - символ условной вероятности)

1.  $A \Rightarrow A \wedge A$
2.  $A \wedge B \Rightarrow A$
3.  $((A \Rightarrow B) \Rightarrow (\neg B \wedge C)) \Rightarrow \neg(C \wedge A)$
4.  $A, A \Rightarrow B \mapsto B$
5.  $0 \leq P(A/B) \leq 1$
6.  $P(A/A) = 1$
7.  $P(A \wedge B/C) = P(A/C) \cdot P(B/A \wedge C)$
8. Если  $\mapsto \neg B$ , то  $P(\neg A/B) = 1 - P(A/B)$
9. Если  $\mapsto (A \equiv C) \wedge (B \equiv D)$ , то  $P(A/B) = P(C/D)$

Имея, к примеру, такую систему аксиом остается определить значение логических операций над высказываниями. Это может быть сделано традиционным для логики способом – через определение таблиц истинности.

$P(A)$	$P(B)$	$P(A/B)$	$P(A \vee B)$	$P(A \wedge B)$	$P(A \Rightarrow B)$	$P(\neg A)$
$p$	$q$	$u$	$p + q - pu$	$pu$	$1 - p + pu$	$1 - p$

Надо отметить, что такой вариант определения таблицы истинности является не единственным. Но если он выбран, то все последующие выводы должны определяться на его базе. Ниже при определении нечеткой логики будут показаны другие варианты определения операций.

## Лекция 8

1. ОБЩЕЕ ПРЕДСТАВЛЕНИЕ О ЗАДАЧАХ РАСПОЗНАВАНИЯ ОБРАЗОВ.
2. СОДЕРЖАТЕЛЬНАЯ ПОСТАНОВКА ЗАДАЧИ РАСПОЗНАВАНИЯ ОБРАЗОВ.
3. АЛГОРИТМЫ РАСПОЗНАВАНИЯ.
4. АЛГОРИТМЫ КЛАСТЕРИЗАЦИИ (ТАКСОНОМИИ).
5. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - Примеры задач РО.
  - ИИ и задачи РО.

### Общее представление о задачах распознавания образов.

В настоящее время все более широкое распространение получают системы, основанные на т.н. новых информационных технологиях. Неотъемлемой характеристикой таких систем является их “интеллектуальность” в следующем смысле:

1. ориентированы на пользователей-непрофессионалов в области программирования, математики, т.е. обладают достаточно развитым интерфейсом;
2. предназначены, как правило, для автоматизации решения широкого спектра задач, попадающих в сферу интересов конкретного пользователя.

Второй из перечисленных признаков “интеллектуальности” информационных систем требует более детального рассмотрения. Начнем с примера.

В теоретической физике уже довольно давно решается проблема построения единой модели мира. Дело в том, что сегодня существует две фундаментальные теории – квантовая теория поля (КТП) и общая теория относительности (ОТО), описывающие один и тот же реальный мир. Ни одна из этих теорий не сводится к другой и поэтому более полная математическая модель должна включать их обе. Начала такого объединения были заложены в теории струн.

Всякий раз, когда сталкиваешься с проблемой такого рода, возникает естественный вопрос: а будет ли такая единая модель (в частности, теория струн) универсальной, т.е. справедливой для каждого объекта реального мира? Какие-либо варианты ответов на этот вопрос могут быть найдены разными путями (экспериментами и т.п.). Правомерен также и следующий подход. Введем предикаты:

$$P_{OTO}(x) = \begin{cases} 1, & \text{если объект } x \text{ подчиняется законам ОТО,} \\ 0, & \text{в противном случае.} \end{cases}$$

$$P_{KTP}(x) = \begin{cases} 1, & \text{если объект } x \text{ подчиняется законам КТП,} \\ 0, & \text{в противном случае.} \end{cases}$$

Теперь заметим, что для всякого объекта  $x$  реального мира справедливо:

$$P_{OTO}(x) \& P_{KTP}(x) = 0,$$

и, следовательно, множества

$$M_{OTO} = \{x \mid P_{OTO}(x) = 1\}, \quad M_{KTP} = \{x \mid P_{KTP}(x) = 1\}$$

не пересекаются. В такой ситуации вопрос, сформулированный выше, в каком-то условном смысле сводится к решению следующих проблем:

- описания множеств  $M_{OTO}$  и  $M_{KTP}$  либо их границ;
- отнесения произвольного объекта  $x$  к одному из множеств.

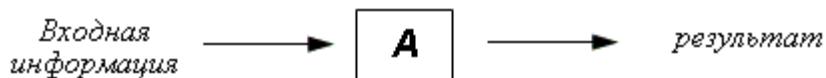
При решении перечисленных проблем строится некий алгоритм, позволяющий вычислять предикаты  $P_{ОТО}, P_{КТП}$ . Иначе говоря, вначале решается качественная задача определения свойств объекта. После этого используется соответствующая физическая модель (ОТО или КТП), с помощью которой вычисляются количественные характеристики объектов.

Качественные задачи, подобные рассмотренной в примере, возникают в интеллектуальных информационных системах в различном контексте – при выборе моделей, проверке допустимости объектов и т.п. Кроме того, такие задачи представляют интерес независимо от решения т.н. количественных задач (как в примере). Поэтому возникает необходимость разработки соответствующего математического аппарата.

По постановке такие качественные задачи примыкают к известной **проблеме выбора** или **проблеме принятия решений**. В наиболее широкой трактовке они могут быть сформулированы в одном из следующих вариантов.

1. На множестве объектов  $X$  произвольной природы заданы подмножества  $X_1, \dots, X_l$  ( $l \in N$ ) в некотором смысле однородных (однотипных) объектов. Требуется указать алгоритм **A**, определенный на всем множестве  $X$ , результат работы которого для каждого  $x \in X$  можно интерпретировать в терминах принадлежности подмножествам  $X_i$  ( $i = 1, \dots, l$ ).
2. Задано множество  $X$  объектов произвольной природы. Требуется указать алгоритм **A**, определенный на всем множестве  $X$ , результат работы которого можно интерпретировать в терминах разбиения  $X$  на подмножества  $X_1, \dots, X_l$  ( $l \in N$ ) в каком-то смысле однородных (однотипных) объектов.

Рассмотрим теперь возможные подходы к решению сформулированных задач. Начнем с того, что всякая задача, в которой требуется в конечном итоге построить алгоритм, решается по следующей классической схеме:



Из этой схемы непосредственно можно сделать вывод, что алгоритмы (подходы к решению задачи) могут отличаться настолько, насколько различаются способы задания входной информации и представления результатов. В этом смысле можно выделить три принципиально различных подхода к решению сформулированных выше задач.

1. математическое моделирование. Когда известны и обоснованы в каком-то смысле закономерности образования подмножеств  $X_1, \dots, X_l$ . Для этого подхода (в отличие от последующих) характерна универсальность алгоритма **A** и (возможно, как следствие) доказательство правильности его работы на всем множестве  $X$ ;
2. моделирование свойств объектов. Когда известны, но не обоснованы закономерности образования подмножеств  $X_1, \dots, X_l$  в множестве  $X$ . Способ задания подмножеств может быть различным – логический (как, например, в исчислении высказываний), структурный (фреймы, сети) и некоторые другие;
3. моделирование по прецедентности. Когда заданы примеры объектов, которые являются носителями закономерностей, присущих подмножествам  $X_1, \dots, X_l$ . В данном случае алгоритм **A** реализует принцип “похожести” объектов на основе сравнения с заданными примерами (т.е. по прецедентности).

Последний из подходов отличается наличием минимального количества информации о предметной области и, в этом смысле, он является отправной точкой, в процессе построения математических моделей для каждой из перечисленных выше задач. В настоящее время этот подход развивается в рамках теории распознавания образов.

## **Содержательная постановка задачи распознавания образов.**

Термином “распознавание образов” обозначается обширное поле деятельности, связанной как с реальными жизненными потребностями – распознавание людьми “образов” в окружающей их среде, так и с решениями научных и технических задач – считывание символов, автоматизированная диагностика и т.п. Появление и широкое распространение такого рода деятельности обусловлено целым рядом причин:

- стремлением расширить область применения компьютеров с целью придать им новые, ранее недоступные функции;
- возрастанием роли автоматизированных систем обработки информации;
- появлением новых научных и технических направлений, призванных облегчить интеллектуальную деятельность человека.

Рассмотрим содержательную интерпретацию задач, рассматриваемых в теории распознавания образов. Пусть задано конечное множество объектов (явлений, процессов) некоторой произвольной природы. Предположим, что:

1. Априори известно, что в заданном множестве существуют подмножества (классы, категории) объектов, качественно отличающиеся между собой;
2. Имеется некоторое устройство (алгоритм), которому доступно описание заданного множества объектов, а информация о принадлежности к подмножествам может быть известна (т.е. устройство может быть обучено), или неизвестна.

От такого устройства естественно потребовать решения следующих задач:

- a) задача распознавания с обучением (учителем).

В ситуации, когда известна информация о принадлежности заданных объектов к классам (в этом случае говорят, что задана обучающая выборка), для каждого вновь поступившего объекта необходимо указать его принадлежность к одному из классов;

- b) задача распознавания без обучения (самообучение, кластеризация, таксономия).

Информация о принадлежности заданных объектов к классам неизвестна. Требуется построить их разбиение на классы.

Следует отметить, что вторая задача имеет в некотором смысле вспомогательный характер, т.к. является, по сути, задачей сортировки, анализа данных. Но в математическом смысле задачи совершенно равноправны.

Сфера применения методов, разработанных для решения таких задач, довольно широка. Краткий и далеко не исчерпывающий перечень таких сфер при представлении об алгоритме, как о “черном ящике” приводится в таблице.

задача	вход	результат
Прогнозирование погоды	Метеорологические измерения	Прогноз погоды
Распознавание знаков рукописного текста	Оптический образ	Наименование буквы
Медицинская диагностика	Симптомы	Наименование болезни
Распознавание диктора	Звуковые колебания	Имя диктора
Диагностика технических устройств	Технические характеристики	Прогноз работоспособности устройств

Несмотря на внешнее различие перечисленных задач, все они могут быть описаны с единых позиций, с использованием единого математического языка и для их решения используется один и тот же математический аппарат теории распознавания образов.

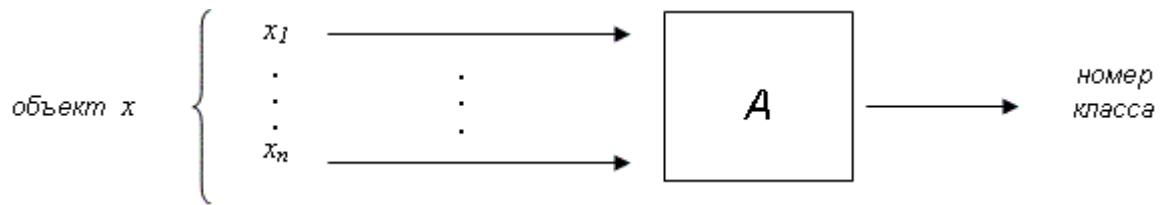
Детализируем теперь содержательную постановку задачи распознавания образов. Далее в этом пункте речь будет идти в основном о задаче распознавания с обучением.

Предположим, что задано конечное множество объектов (векторов) пространства  $R^n$ , т.е.  $x = (x_1, \dots, x_n) \in R^n$ . Каждый объект  $x \in R^n$  является **образом** некоторого реального явления, процесса и т.п. Компоненты  $x_j$  ( $j=1, \dots, n$ ) в описании вектора  $x$  принято называть **признаками**.

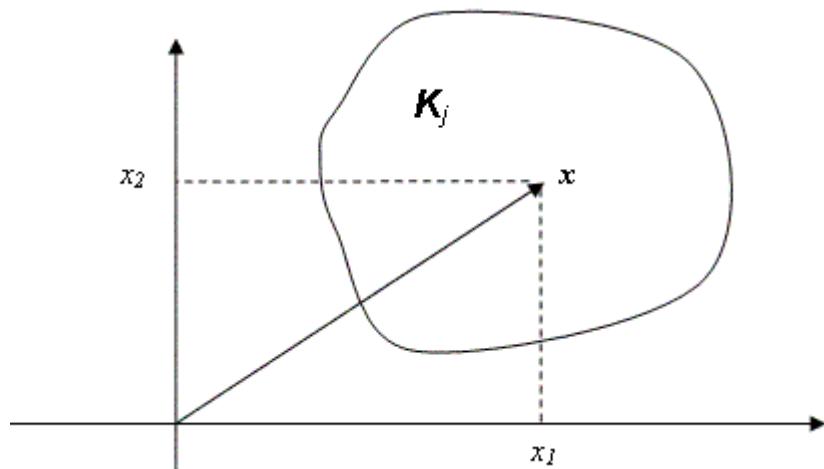
**ками.** Пусть, кроме того, известно, что все множество  $X = \{x\}$  разбито на  $l \in N$  классов  $K_1, \dots, K_l$ .

Каждое устройство, которое позволяет отнести вектор  $x \in R^n$  к одному из классов  $K_1, \dots, K_l$ , называется **классификатором** или **алгоритмом распознавания**.

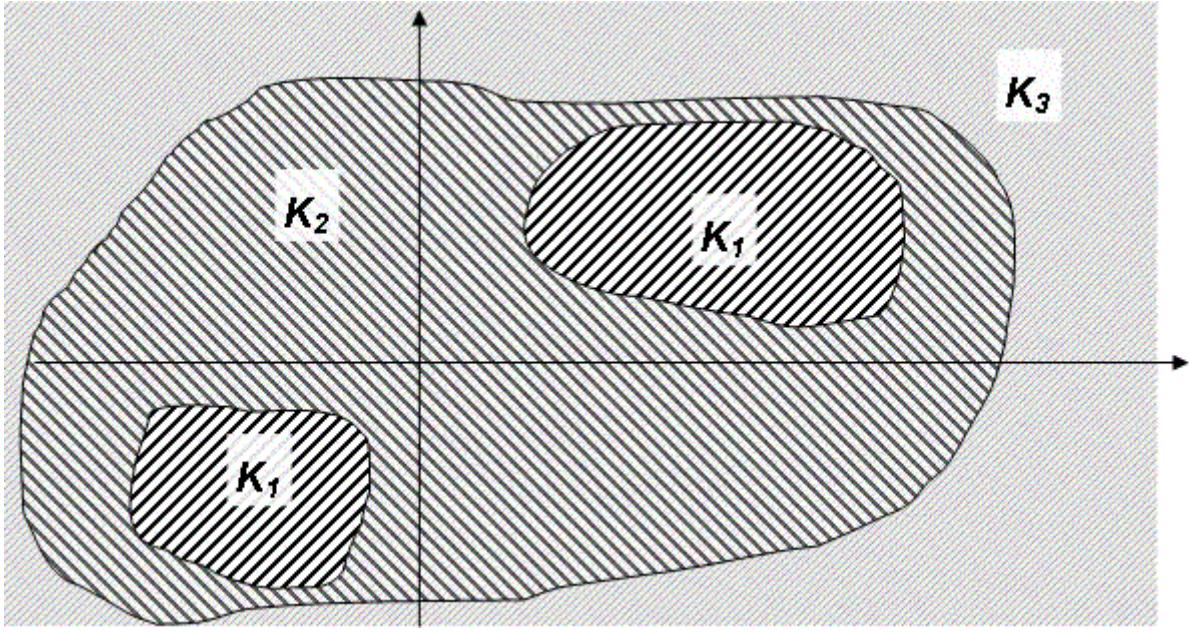
Рассмотрим вопрос о том, что представляет собой алгоритм распознавания и что необходимо определить, чтобы его задать. В качестве простой модели первого приближения такого алгоритма может рассматриваться устройство, имеющее  $n$  входов и один выход и работающее по следующей схеме:



Задачи распознавания образов имеют удобную геометрическую интерпретацию. Рассмотрим пространство  $R^n$  ( $n = 2$ ). Тогда каждый объект  $x = (x_1, x_2)$  представляет собой точку или вектор пространства  $R^2$ .



Классу  $K_j$  может соответствовать некоторая область пространства  $R^2$ . Взаимное расположение классов может быть самым произвольным. Например, для случая  $l = 3$  это может быть следующая конфигурация:



Множество точек, принадлежащих классу  $K_j$ , называют иногда **собственной областью** класса  $K_j$  ( $j=1, \dots, l$ ). Поверхность (для  $n > 2$  – гиперповерхность), которая разделяет собственные области классов  $K_j$ , называют **разделяющей**.

Следует отметить, что по разным причинам алгоритмы  $A$  могут отказываться от распознавания (например, иногда объект  $x$  принадлежит разделяющей поверхности). С учетом этого, алгоритм  $A$  представляет собой следующее отображение:

$$(*) \quad \forall x \in R^n \quad (A : x \rightarrow \{0, \dots, l\}),$$

при условии, что 0 соответствует отказу от классификации. Алгоритм (\*) строит некоторые разделяющие поверхности или области (называются иногда **областями решений** алгоритма  $A$ ). При этом, т.к. алгоритм  $A$  лишь моделирует работу классификатора, т.к. построенные ими разделяющие поверхности могут не совпадать с истинными. **Основная проблема** при выборе и построении алгоритма  $A$  сделать указанное несовпадение минимальным. Конкретизируем теперь представление алгоритма  $A$ . Для этого, очевидно, необходимо рассмотреть вопрос о том, как строить области решения или разделяющие поверхности. В основу такой конкретизации может быть положен следующий тезис (или гипотеза):

*всякий алгоритм  $A$  вида (\*) может быть представлен с помощью  $l$  функций  $g_1, \dots, g_l$ , таких, что*

$$\forall x \in R^n \quad (A(x) = i \Leftrightarrow g_i(x) = \max_{j \in \{1, \dots, l\}} \{g_j(x)\}).$$

Приведенный тезис может иметь и другую форму, не меняющую, однако, сути дела. Функции  $g_i$  ( $i=1, \dots, l$ ) называют **решающими** или **дискриминантными**.

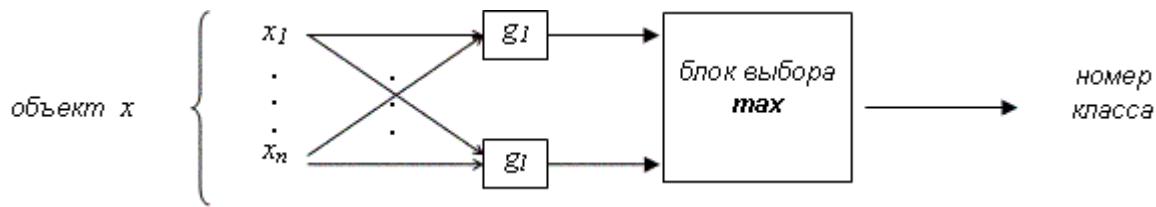
Нетрудно заметить, что в этом случае область решения алгоритма  $A$ , соответствующая классу  $K_i$ , представляет собой множество:

$$\{x \mid x \in R^n, g_i(x) > g_j(x), j \neq i\},$$

а разделяющая поверхность для классов  $K_i, K_j$  образует множество (или описывается уравнением):

$$\{x \mid x \in R^n, g_i(x) = g_j(x)\}.$$

В случае принятия такой гипотезы алгоритм распознавания может быть детализирован следующим образом:



Рассмотрим теперь вопрос о выборе дискриминантных функций. Этот вопрос имеет две стороны:

1. выбор класса функций (линейные, нелинейные и т.п.);
2. выбор конкретных функций в заданном классе.

Выбор класса функций для решения задачи в некоторой предметной области – это одна из наиболее сложных проблем в теории распознавания образов. Второй вопрос имеет в большей степени методологический характер. Как правило, он решается следующим образом. Выбираются вначале некоторые функции (в заданном классе) и определяется степень несовпадения разделяющих поверхностей алгоритма  $A$  и истинных разделяющих плоскостей. Затем эта степень минимизируется путем корректировки функций.

Процесс корректировки функций на реальных объектах называется **обучением алгоритма**. Совокупность объектов, предназначенная для реализации процесса обучения, называется **обучающей выборкой**.

Несколько замечаний по вопросу о выборе классов дискриминантных функций. Понятно, что характеристики класса определяются в первую очередь типом (свойствами) рассматриваемых объектов. С этой точки зрения принято выделять следующие основные типы функций:

- **статистические** (тип объекта – вероятностный),
- **детерминистские** (тип объекта – векторы континуальных пространств),
- **логические** (тип объекта – дискретный).

Кроме того, внутри класса алгоритмы могут различаться с позиции принятия некоторого другого аналога рассмотренной выше гипотезы. Так, вместо максимума, может использоваться минимум и т.п. Некоторые, наиболее широко распространенные классы алгоритмов распознавания описаны ниже.

## Алгоритмы распознавания.

Построение таких алгоритмов заключается в определении решающих функций, которые в пространстве признаков порождают границы, отделяющие объекты одного класса от другого. Для этого выдвигаются некоторые предположения, в соответствии с которыми накладываются на вид разделяющих функций. Эти функции содержат неизвестные параметры, и для того, чтобы осуществить требуемое разбиение (построить границы классов), необходимо установить их значения. Данная задача решается на основе объектов обучающей выборки.

### Линейные алгоритмы распознавания.

Пусть задано  $l \in N$  классов  $K_1, \dots, K_l$ . Общая идея таких алгоритмов распознавания базируется на предположении о том, что существует  $l$  функций  $g_1, \dots, g_l$  вида

$$g_j = \sum_{i=1}^n a_i x_i + a_{n+1}, \quad j = 1, \dots, l$$

а алгоритм  $A$  заносит произвольный допустимый объект  $x$  в класс с номером  $j \in \{1, \dots, l\}$  в том и только том случае, когда

$$g_j = \max_t \{g_t(x)\}$$

Построение конкретных значений коэффициентов линейных функций  $g_1, \dots, g_l$ , отвечающих оптимальному алгоритму  $A$ , может быть осуществлено различными способами. Наиболее распространенный способ основан на следующих очевидных соображениях. Исходя из вида функций  $g_1, \dots, g_l$ , уравнение границы между классами  $K_j$  и  $K_t$  можно записать следующим образом

$$g_{jt}(x) = g_j(x) - g_t(x) = \sum_{i=1}^n b_i x_i + b_{n+1} = 0$$

И тогда задача построения алгоритма  $A$  без труда сводится к решению совокупности систем линейных неравенств. Например, для случая  $l=2$  такая система будет иметь вид:

$$(**) \quad \left\{ \begin{array}{l} b_1 x_1^1 + \dots + b_n x_n^1 + b_{n+1} > 0 \\ \dots \\ b_1 x_1^m + \dots + b_n x_n^m + b_{n+1} > 0 \\ b_1 (-x_1^{m+1}) + \dots + b_n (-x_n^{m+1}) + b_{n+1} > 0 \\ \dots \\ b_1 (-x_1^k) + \dots + b_n (-x_n^k) + b_{n+1} > 0 \end{array} \right.$$

где  $\{x^1, \dots, x^m\}, \{x^{m+1}, \dots, x^k\}$  - объекты обучающей выборки классов  $K_1, K_2$  соответственно. Эти объекты в задаче распознавания образов с обучением считаются заданными.

Заметим, что обобщение такого подхода на случай  $l > 2$  может быть выполнено различным образом. Наибольшее распространение получили методы, основанные на дихотомии (по парной группировке) классов. Нетрудно видеть, что при этом задача построения алгоритма  $A$  сводится к решению конечного числа систем вида (\*\*).

Для решения систем (\*\*) чаще всего используются итерационные методы, суть которых заключается в следующем. Выбирается начальное значение вектора  $(b_1, \dots, b_n, b_{n+1})$ . Данный выбор может быть либо случайным, либо основанным на некоторых содержательных соображениях. Затем последовательно выполняется рассмотрение неравенств

$$b_1 x_1^i + \dots + b_n x_n^i + b_{n+1} > 0$$

для всех  $i = 1, \dots, m, m+1, \dots, k$ . Если для какого-то объекта  $x^i$  неравенство не выполняется, то происходит корректировка вектора  $(b_1, \dots, b_n, b_{n+1})$ . Корректировка может выполняться различными способами, но во всех случаях суть ее одинакова – необходимо добиться выполнения соответствующего неравенства. Такая процедура повторяется до тех пор, пока на некоторой итерации (очередном шаге просмотра объектов  $x^1, \dots, x^m, x^{m+1}, \dots, x^k$ ) значение вектора  $(b_1, \dots, b_n, b_{n+1})$  не будет изменяться.

Один из первых результатов теории распознавания образов (теорема Новикова) доказывает сходимость рассмотренной процедуры в случае, если классы отделены линейными функциями.

## 2. Алгоритм вычисления оценок.

Данный класс алгоритмов реализует принцип прецедентности (принятия решений по примерам). Алгоритмы вычисляют некие оценки похожести, которые характеризуют “близость” распознаваемого объекта с объектами-эталонами (представителями классов). При этом “близость” вычисляется по системе ансамблей признаков, представляющих собой систему подмножеств исходного множества признаков.

Перейдем непосредственно к описанию таких алгоритмов. Предположим, что эталонные объекты  $x_1, \dots, x_m$  ( $x_i \in R^n$ ) сведены в таблицу  $T_{mn}$  (обучающая выборка классов  $K_1, \dots, K_l$ ). Рассмотрим набор признаков  $N = \{1, \dots, n\}$  и выделим всевозможные подмножества  $\{I\}$  множества  $N$  (т.н. система опорных множеств). Очевидно, что каждому опорному множеству  $I = \{i_1, \dots, i_k\}$  можно поставить во взаимно однозначное соответствие булевый вектор  $\omega$  с единичными координатами  $i_1, \dots, i_k$ . Удалим теперь из таблицы  $T_{mn}$  все столбцы, за исключением тех, которые имеют номера  $i_1, \dots, i_k$ . В результате получим соответствующую  $\omega$ -часть таблицы. Объекты этой таблицы будем обозначать через  $\omega x_1, \dots, \omega x_m$  ( $\omega$ -части объектов).

Введем функцию  $B_\omega$ , определенную на множестве  $\{\omega x_i \times \omega x_j\}$  и описывающую степень похожести соответствующих  $\omega$ -частей. Если, например,

$$\omega x_j = \{x_{i_1}^j, \dots, x_{i_k}^j\},$$

то в качестве  $B_\omega$  можно выбрать функцию, значение которой равно числу выполненных неравенств

$$|x_i^j - x_i^t| < \varepsilon_i,$$

где  $\varepsilon_i \in R$ . Функция  $B_\omega$  может быть еще проще

$$B_\omega(\omega x^j, \omega x^t) = \begin{cases} 1, & \text{если } \nu \leq \varepsilon \\ 0, & \text{в противном случае,} \end{cases}$$

где  $\varepsilon \in N$ ,  $\nu$  - число выполненных неравенств  $|x_i^j - x_i^t| < \varepsilon_i$ .

На основе функции  $B_\omega$  строится оценка похожести произвольного объекта (точнее его  $\omega$ -части) на группу объектов таблицы  $T_{mn}$  (чаще всего в эту группу входят объекты одного класса). При этом могут использоваться различные параметры. Например:  $\gamma(x^j)$  - степень важности объекта-эталона,  $p_{i_1}, \dots, p_{i_k}$  - веса признаков. Вычисление такой оценки дает возможность определить ее для классов  $K_1, \dots, K_l$  по заданному опорному множеству  $I$ .

Обозначим  $\Gamma_\omega$  - каким-либо образом параметризованную оценку близости пары объектов, полученную на основе  $B_\omega$ . Пусть теперь для класса  $K_j$  с эталонными объектами  $x^1, \dots, x^{m_j}$  вычислены оценки  $\Gamma_\omega(x^1, x), \dots, \Gamma_\omega(x^{m_j}, x)$  (здесь  $x$  – произвольный объект из  $R^n$ ). Тогда оценка для класса  $K_j$  может быть определена следующим образом:

$$\Gamma_\omega^j(x) = \varphi(\Gamma_\omega(x^1, x), \dots, \Gamma_\omega(x^{m_j}, x)).$$

Наконец, вычисляются оценки для класса  $K_j$  по системе опорных множеств  $\{I\} = \{I_1, \dots, I_k\}$

$$\Gamma_j(x) = \psi(\Gamma_{\omega_1}^j(x), \dots, \Gamma_{\omega_k}^j(x)).$$

В результате для произвольного допустимого объекта  $x$  получаем  $l$  оценок  $\Gamma_1(x), \dots, \Gamma_l(x)$ . Теперь остается выполнить классификацию. Для этого используется решающее правило

$$r: \Gamma_1(x) \times \dots \times \Gamma_l(x) \rightarrow \{0, 1, \dots, l\}.$$

Например, в качестве такого решающего правила можно использовать функцию:

$$r(\Gamma_1(x), \dots, \Gamma_l(x)) = \begin{cases} j, & \text{если } \Gamma_j(x) = \max_{t \in \{1, \dots, l\}} \{\Gamma_t(x)\} \\ 0, & \text{в противном случае.} \end{cases}$$

Необходимо заметить, что эффективность применения алгоритмов вычисления оценок во многом зависит от того, насколько просто решается задача вычисления оценок  $\Gamma_j$  ( $j=1, \dots, l$ ). Например, в приведенной выше модели с параметрами  $k$  (длина опорного множества),  $\varepsilon, \varepsilon_1, \dots, \varepsilon_n$  (пороги функции близости),  $\gamma(x^1), \dots, \gamma(x^m)$  и  $p_1, \dots, p_n$  (веса важности объектов-эталонов и признаков) число слагаемых в выражении  $\Gamma_j(x)$  равно  $m_j C_n^k$ , что существенно затрудняет вычисление оценок. Во многих практических случаях удается получить более эффективные формулы для оценки  $\Gamma_j(x)$ .

### **3. Статистические алгоритмы распознавания.**

Статистические алгоритмы чаще всего используются на том основании, что природные процессы носят случайный характер. Такая точка зрения на природные процессы вполне правомерна, т.к. почти всегда ничего нельзя сказать о механизме порождения объектов и, поэтому, его можно рассматривать с вероятностных позиций. Имеют место различные неопределенности ("шумы", неточность классификации и т.п.). В такой ситуации естественно рассматривать объект  $x = (x_1, \dots, x_n)$  как случайную  $n$ -мерную величину, а принадлежность  $x$  к какому-либо классу определять с некоторой долей вероятности.

Статистический анализ дает аппарат для построения алгоритма распознавания оптимального в том смысле, что его применение обеспечивает в среднем наименьшую вероятность совершения ошибок. Покажем, как можно использовать теорию статистических решений для построения алгоритмов распознавания.

Пусть задано  $l \in N$  классов  $K_1, \dots, K_l$ . Предположим теперь, что:

1. известна или может быть оценена априорная вероятность класса  $K_j$ :  $p(K_j), j=1, \dots, l$  (т.е. вероятность наблюдения объекта из класса  $K_j$ );
2. функция плотности вероятности  $p(x|K_j)$  известна или может быть оценена;
3. апостериорная вероятность  $p(K_j|x)$  неизвестна.

Если алгоритм распознавания принимает решение, что объект  $x \in K_j$  и в действительности ошибается, то в этом случае алгоритм должен быть "ощтрафован" на некоторую величину  $a_{ij}$ . Так как объект  $x$  может принадлежать любому из  $l$  классов, то математическое ожидание потерь, связанных с решением  $x \in K_j$ , можно определить следующим выражением

$$r_j(x) = \sum_{i=1}^l a_{ij} p(K_i | x)$$

Эта величина называется **условным средним риском**.

Работа алгоритма заключается в определении решения, которое минимизировало бы условный риск в смысле следующей процедуры:

шаг 1. для объекта  $x$  вычисляем значения условного риска  $r_1(x), \dots, r_l(x)$ .

шаг 2. объект  $x$  заносим в класс  $K_j$ , если выполняется следующее условие

$$r_j(x) = \min_i \{r_i(x)\}$$

Данный алгоритм минимизирует условные потери. Следовательно, и математическое ожидание полных потерь на множестве всех решений также будет минимизировано. Со статистической точки зрения такой алгоритм является оптимальным по качеству распознавания. Называется этот алгоритм обычно **байесовским классификатором**.

Для реализации байесовского классификатора необходимо каким-либо образом вычислять величины  $p(K_j | x)$ . Согласно известной формуле Байеса

$$p(K_j | x) = (p(K_j) \cdot p(x | K_j)) \cdot (p(x))^{-1}$$

Тогда

$$r_j(x) = (p(x))^{-1} \cdot \sum_{i=1}^l a_{ij} \cdot (p(K_i) \cdot p(x | K_i))$$

Заметим, что множитель  $(p(x))^{-1}$  является общим для всех  $r_j(x)$ ,  $j = 1, \dots, l$  и его можно не учитывать.

Решение оптимизационной задачи значительно упрощается, если функцию потерь (величины  $a_{ij}$ ) выбрать в виде

$$a_{ij} = 1 - \delta_{ij},$$

где  $\delta_{ij}$  - символ Кронекера. Величина  $a_{ij}$  называется в этом случае **нуль-единичной** функцией потерь. Нетрудно видеть, что в этом случае величина условного среднего риска вычисляется по формуле

$$r_j(x) = p(x) - p(K_j) \cdot p(x | K_j),$$

и ее минимизация эквивалентна максимизации выражения  $p(K_j) \cdot p(x | K_j)$ . Такой байесовский классификатор называется **классификатором по максимуму правдоподобия**. Ранее отмечалось, что функции распределения, как правило, неизвестны, но могут быть оценены с помощью объектов обучающей выборки. В некоторых случаях известным считается вид функции распределения, и по обучающей выборке оцениваются только параметры этих функций (математическое ожидание, дисперсия и др.).

### **Алгоритмы кластеризации (таксономии).**

Рассмотренные выше алгоритмы предназначены для решения задачи распознавания образов с обучением. Основным условием в этом случае является наличие необходимой априорной информации о предметной области, которая задается в виде обучающей выборки. Однако, не всегда такая информация имеется. В этом случае используются алгоритмы кластеризации. Кластеризация (или таксономия) – это классификация объектов без обучения, когда не задано никакой априорной информации о прототипах классификации.

Пусть в пространстве  $R^n$  задано  $m \in N$  объектов  $x_1, \dots, x_m$ . Задачу кластеризации можно сформулировать следующим образом:

*необходимо указать алгоритм, позволяющий выделить в  $R^n$  области  $K_1, \dots, K_l$  такие, что каждый объект  $x_i$  входит в одну и только одну из областей  $K_j$  ( $i \in \{1, \dots, m\}$ ,  $j \in \{1, \dots, l\}$ ) и, кроме того,*

$$\bigcup_{i=1}^l K_i \subseteq R^n, K_i \cap K_j = \emptyset \text{ при } i \neq j.$$

Алгоритмы кластеризации включают объекты в классы (кластеры) в соответствие с мерой сходства, которая отражает естественные связи между объектами. Предполагается, что “сила” этих связей намного выше среди объектов внутри кластера и ниже среди объектов различных кластеров. Мера сходства (или различия) задается в числовой форме и указывает “силу” естественных связей между объектами, между объектом и группой объектов, а также между группами объектов.

В качестве мер сходства используются различные функции. Опишем некоторые из них. Евклидово расстояние. В случае, когда все признаки равнозначны, эта мера имеет вид

$$d(x, x') = \left( \sum_{i=1}^n (x_i - x'_i)^2 \right)^{\frac{1}{2}}$$

В случае же, когда признаки неравнозначны, мера может иметь следующий вид

$$d(x, x') = \left( \sum_{i=1}^n \alpha_i \cdot (x_i - x'_i)^2 \right)^{\frac{1}{2}}$$

Расстояние Махalanобиса. Используется в вероятностных пространствах и имеет вид

$$d(x, x') = \left( (x - x')^T \cdot c^{-1} \cdot (x - x') \right)^{\frac{1}{2}}$$

где  $c^{-1}$  - обратная ковариационная матрица.

Заметим, что определения меры сходства недостаточно для полного решения задачи. Необходимо введение критериев для разбиения на кластеры. В качестве такого критерия наиболее часто используется

$$F = \sum_{i=1}^l \sum_{x \in K_i} (x - M_i)^2$$

где  $l$  – число кластеров,  $K_i$  - множество объектов кластера с номером  $i$ ,  $M_i$  - вектор его выборочных средних значений.

Данный критерий определяет совокупное расстояние между объектом и математическим ожиданием кластера, в который включается объект  $x$ . Очевидно, что если потребовать минимизации  $F$ , то это будет означать, что объект включается в кластер так, чтобы последний имел наибольшую возможную “плотность”.

К настоящему времени разработано большое число алгоритмов кластеризации. В зависимости от того, используется ли в процессе построения кластеров критерии, подобные рассмотренному выше, принято выделять следующие типы алгоритмов:

- прямые (конструктивные) или “собирательные”;
- обратные (оптимизационные) или “разделительные”.

В первом случае процесс кластеризации стартует с изолированного объекта и объединяет ближайшие в некотором смысле объекты в группы. Во втором – от исходного множества объектов в зависимости от значения некоторого критерия объекты распределяются в кластеры. Могут использоваться алгоритмы комбинированного типа.

В основу классификации алгоритмов кластеризации может быть положена также следующая существенная информация: известно априори или нет число классов, на которое необходимо разбить исходное множество объектов. В каждом из этих случаев используется своя схема построения алгоритма.

Рассмотрим случай, когда число классов априори неизвестно. Пусть  $x_1, \dots, x_m$  - исходное множество объектов. Первый кластер (объект) может быть выбран произвольно. Далее в него пытаются включить некоторые другие объекты. Такое включение может быть выполнено, например, когда расстояние от объекта до кластера меньше некоторого порога. Если объекты не включаются в текущий кластер, то образуется новый кластер. Процесс повторяется до тех пор, пока все объекты  $x_1, \dots, x_m$  не будут распределены в кластеры. Следует отметить, что после включения объектов в кластеры параметры последнего могут уточняться. Такого рода уточнения являются основой итерационных процессов построения алгоритмов кластеризации.

Хотя приведенная схема обладает рядом очевидных недостатков (зависимость результатов кластеризации от выбора первого кластера, от порядка рассмотрения объектов, от значения порогов), она позволяет быстро получить приблизительные оценки основных характеристик заданного набора объектов. Кроме того, она не сложна с вычислительной точки зрения. Практически же, для того, чтобы хорошо понять геометрию распределения объектов, необходимо проводить многочисленные эксперименты с различными значениями порога и различными исходными точками кластеризации.

В том случае, когда число кластеров известно, может быть использована итерационная схема, основанная на минимизации следующего функционала (критерия):

$$F = \sum_{x \in K_i(t)} (x - z_i)^2,$$

где  $K_i(t)$  - область  $i$ -го кластера с центром  $z_i$ , полученная на  $t$ -м шаге итерации.

Практическая реализация алгоритма, построенного по данной схеме, также требует проведения численных экспериментов.

## Лекция 9

1. НЕОБХОДИМОСТЬ НЕЧЕТКОЙ МАТЕМАТИКИ.
2. ОПРЕДЕЛЕНИЕ НЕЧЕТКОГО МНОЖЕСТВА.
3. ОПЕРАЦИИ НАД НЕЧЕТКАМИ МНОЖЕСТВАМИ.
4. НЕЧЕТКИЕ ЛОГИЧЕСКИЕ ПЕРЕМЕННЫЕ И ОПЕРАЦИИ НАД НИМИ.
5. ПРИЛОЖЕНИЯ НЕЧЕТКОЙ МАТЕМАТИКИ.
6. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - ИИ и нечеткая математика.

### Необходимость нечеткой математики.

Основные сложности в процессе решения задачи возникают тогда, когда условия, которые необходимо учитывать, оказываются неопределенными и когда они, в то же время, сильно влияют на результаты решения. При построении формальных моделей чаще всего используют детерминированные методы и тем самым вносят определенность в те ситуации, где ее в действительности не существует. Неточность задания тех или иных условий при расчетах практически не принимается во внимание. Иногда с учетом определенных предположений и допущений, неточные параметры заменяются экспертными оценками или средними значениями. Возникающие при этом нарушения равенств, балансовых соотношений и т.д. приводят к необходимости модифицировать параметры метода для точного удовлетворения заданных условий и получения приемлемого результата. Такого рода ситуации чаще всего возникают вследствие недостаточной изученности объектов. Особенность соответствующих систем состоит в том, что значительная часть информации, необходимой для их математического описания, существует в форме представлений или пожеланий экспертов. Но в языке традиционной математики нет объектов, с помощью которых можно было бы достаточно точно отразить нечеткость представлений экспертов.

Обычные методы моделирования по своей сути мало пригодны и не эффективны для такого рода систем. Это выражается так называемым принципом несовместимости: *чем сложнее система, тем менее мы способны дать точные и в то же время имеющие практическое значение суждения об ее поведении*. Для систем, сложность которых превосходит некоторый пороговый уровень, точность и практический смысл становятся почти несовместными. Именно в этом смысле точный количественный анализ в реальных задачах в области ИИ не имеет требуемого практического значения.

Иной подход опирается на предположение о том, что элементами мышления человека являются не числа, а элементы некоторых нечетких множеств или классов объектов, для которых переход от "принадлежности к классу" к "не принадлежности" не скачкообразен, а непрерывен. Теория нечетких (размытых) множеств была предложена американским математиком Л. Заде в 1965 г. и предназначалась для преодоления трудностей представления неточных понятий, анализа и моделирования систем, в которых участвует человек. Следует отметить, что для манипуляции с неточно определенными величинами традиционно используется аппарат теории вероятностей. Однако случайность связана с неопределенностью, касающейся принадлежности некоторого объекта к обычному множеству. Это различие между нечеткостью и случайностью приводит к тому, что математические методы нечетких множеств совершенно не похожи на методы теории вероятностей. Они во многих отношениях проще вследствие того, что понятию вероятностной меры в теории вероятностей соответствует более простое понятие функции принадлежности в теории нечетких множеств. По этой причине даже в тех случаях, когда неопределенность может быть представлена вероятностной моделью, обычно удобнее оперировать с ней методами теории нечетких множеств без привлечения аппарата теории вероятностей.

Подход на основе теории нечетких множеств является, по сути дела, альтернативой общепринятым количественным методам, используемым при построении формальных моделей. Он имеет три отличительные черты:

1. в дополнение к числовым переменным используются нечеткие величины или, так называемые, "лингвистические" переменные;

2. простые отношения между переменными описываются с помощью нечетких высказываний;
3. сложные отношения описываются нечеткими алгоритмами.

Такой подход дает приближенные, но в то же время эффективные способы описания поведения систем, настолько сложных и плохо определенных, что они не поддаются точному анализу с помощью методов традиционной математики. Теоретические же основания данного подхода вполне точны и строги в математическом смысле и не являются сами по себе источником неопределенности. В каждом конкретном случае степень точности решения может быть согласована с условиями задачи и точностью имеющихся данных. Подобная гибкость составляет одну из важных черт нечеткой математики.

Основные приложения данного подхода находятся в таких областях, как искусственный интеллект, лингвистика, поиск информации, процессы принятия решений, распознавание образов, медицинская диагностика, психология, право, экономика и других областях человеческой деятельности.

### **Определение нечеткого множества**

В основе нечетких множеств лежит представление о том, что составляющие данное множество  $X$  объекты, обладающие неким общим свойством, могут обладать последним в различной степени. И, следовательно, принадлежать  $X$  в разной степени. Для записи этого, вместо характеристической функции

$$\mu_X(x) = \begin{cases} 1, & \text{если } x \in X, \\ 0, & \text{если } x \notin X. \end{cases}$$

вводится *функция принадлежности*

$$\varphi_X : X \rightarrow [0, 1].$$

Значение  $\varphi_X(x) = \alpha$  (где  $0 \leq \alpha \leq 1$ ) в этом случае интерпретируется следующим образом: объект  $x$  принадлежит множеству  $X$  со степенью  $\alpha$ .

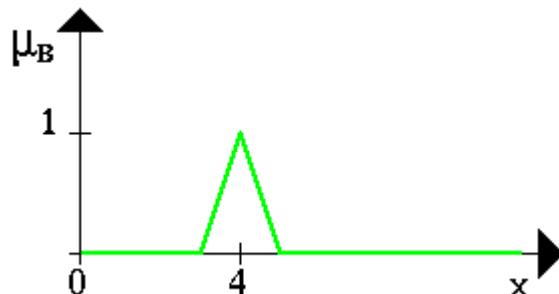
Пусть  $X$  – произвольное множество и  $A \subset X$ . **Нечетким (под) множеством**  $A$  в  $X$  называется совокупность пар вида

$$(x, \varphi_A(x)),$$

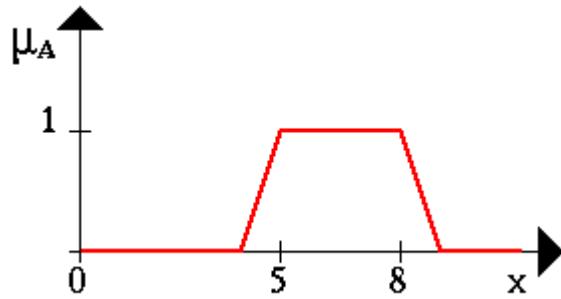
где  $x \in X$  и  $\varphi_A(x)$  - функция принадлежности, зависящая от  $A$ .

**Пример.**

1.  $A = \{x: \text{"значение } x \text{ близко к } 4\}\};$



2.  $A = \{\text{нечеткий интервал от } 5 \text{ до } 8\}$



Чтобы выделять объекты множества А вводится следующее понятие. **Носителем** нечеткого множества А называется множество

$$supp A := \{x : \varphi_A(x) > 0, x \in X\}$$

Уже на уровне введенных понятий очевидной является редукция:  $\varphi_x(x) \rightarrow \mu_x(x)$ , которая существенным образом используется для введения ограничений на конструкцию нечеткой математики.

### Операции над нечеткими множествами.

Операции над нечеткими множествами можно определить разными способами. Ниже для некоторых операций даны различные варианты. Выбор конкретного варианта зависит от смысла, вкладываемого в соответствующие манипуляции над множествами объектов в рамках рассматриваемой задачи ИИ.

Пусть заданы два нечетких множества А, В с функциями принадлежности  $\varphi_A, \varphi_B$  соответственно.

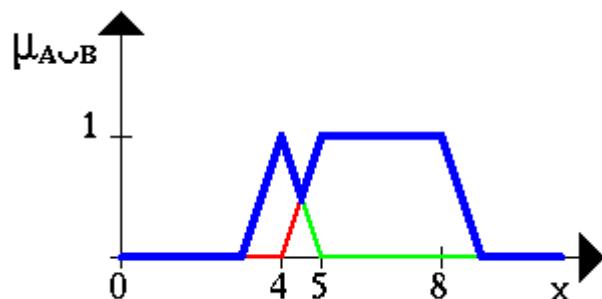
**Объединением** нечетких множеств А, В называется нечеткое множество С, содержащее все объекты исходных множеств и функция принадлежности которого определяется одним из следующих способов:

$$(a) \quad \varphi_C(x) = \max \{\varphi_A(x), \varphi_B(x)\},$$

$$(b) \quad \varphi_C(x) = \begin{cases} 1, & \text{если } \varphi_A(x) + \varphi_B(x) \geq 1 \\ \varphi_A(x) + \varphi_B(x), & \text{иначе} \end{cases}$$

$$(c) \quad \varphi_C(x) = \varphi_A(x) + \varphi_B(x) - \varphi_A(x) \cdot \varphi_B(x)$$

**Пример.** Для введенных выше нечетких множеств, объединение по варианту (a)

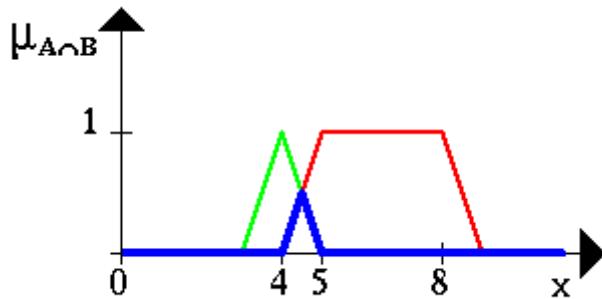


Пересечением нечетких множеств A, B называется нечеткое множество C, содержащее объекты исходных множеств, если только они принадлежат обоим множествам одновременно, и функция принадлежности которого определяется одним из следующих способов:

$$(a) \quad \varphi_C(x) = \min \{ \varphi_A(x), \varphi_B(x) \},$$

$$(b) \quad \varphi_C(x) = \varphi_A(x) \cdot \varphi_B(x)$$

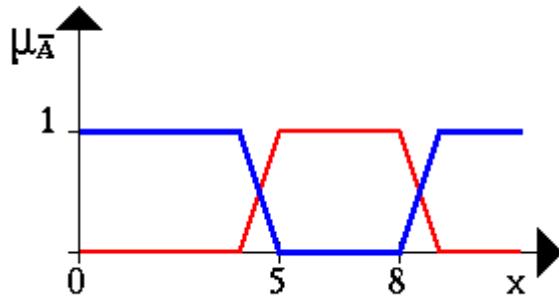
**Пример.** Для введенных выше нечетких множеств, пересечение по варианту (а)



Дополнением нечеткого множества A называется нечеткое множество C, содержащее объекты, не входящие в исходное множество, и функция принадлежности которого определяется следующим способом:

$$\varphi_C(x) = 1 - \varphi_A(x)$$

**Пример.** Дополнение множества A = {нечеткий интервал от 5 до 8}



Все остальные операции над множествами (разность, включение и т.д.) определяются через введенные. И снова, также как и при определении нечеткого множества, главное требование к операциям – возможность их редукции в соответствующие операции для обычных множеств:  $\varphi_X(x) \rightarrow \mu_X(x)$ .

### Нечеткие логические переменные и операции над ними.

Как и в случае обычного множества, нечеткая логическая переменная является результатом абстракции, но не характеристической функции, а функции принадлежности. В общем случае можно считать, что нечеткая логическая переменная определена в  $[0, 1]$  (или  $[0, 1]^n$ ) – подмножестве пространства действительных чисел R (соответственно  $-R^n$ ). При этом, значение  $x \in [0, 1]$  можно трактовать как степень принадлежности нечеткой логической переменной, описываемой данным значением x, к множеству истинных высказываний. Или коро-

че – степенью истинности  $x$ .

В том случае, если речь идет о предикатах, т.е. функциях вида

$$P^F : X^n \rightarrow [0, 1],$$

то мы снова легко можем перейти к нечеткой логической переменной, отождествив предикат (как элемент некоторого множества) с его значением (фактически функцией принадлежности). Правда в этом случае остается проблема вычисления значений логических операций над предикатами. Но и эта проблема решается, после определения конструкции отображения нечетких множеств.

Ниже мы ограничимся рассмотрением логики, которая может быть построена путем введения операций над нечеткими логическими переменными. Пусть  $x_1, x_2 \in [0, 1]$  - нечеткие логические переменные, полученные в результате описанной выше процедуры абстракции. В таких условиях, отталкиваясь от операций над нечеткими множествами, легко можно ввести стандартный набор логических операций.

*Конъюнкцией* называется операция над нечеткими логическими переменными  $x_1, x_2$ , результатом которой является снова логическая переменная, которая определяется одним из следующих способов (по аналогии с пересечением):

$$(a) \quad x_1 \wedge x_2 = \min \{x_1, x_2\},$$

$$(b) \quad x_1 \wedge x_2 = x_1 \cdot x_2$$

*Дизъюнкция* (по аналогии с объединением) определяется следующим образом:

$$(a) \quad x_1 \vee x_2 = \max \{x_1, x_2\},$$

$$(b) \quad x_1 \vee x_2 = \begin{cases} 1, & \text{если } x_1 + x_2 \geq 1 \\ x_1 + x_2, & \text{иначе} \end{cases}$$

$$(c) \quad x_1 \vee x_2 = x_1 + x_2 - x_1 \cdot x_2$$

И, наконец, *отрицание* (по аналогии с дополнением):

$$\neg x = 1 - x$$

Все остальные операции (следования и проч.) определяются через введенные. Кроме того, для нечетких логических переменных могут быть введены операции, не имеющие аналогов в обычной логике (например – выпуклая комбинация).

### Замечания.

1. Нетрудно убедиться, что для введенных операций имеет место редукция в случае перехода к обычным булевым, шкалированным и т.п. переменным;
2. Можно было бы записать характеристацию результатов операций над нечеткими множествами через логические операции над подходящим образом выбранные нечеткие логические переменные. В зависимости от выбранной системы операций получилось бы несколько различных вариантов характеристизации;
3. Для нечетких логических переменных очень важной и нетривиально решаемой в общем случае является проблема вычисления результатов операций.

Что касается вопроса построения нечетких логических систем, то это может быть сделано по аналогии с примерами аксиоматических систем. Но возможности языка нечеткой логики несколько шире и поэтому системы могут быть построены каким-либо и другим образом. Главное требование при этом остается – возможность редукции в обычную логическую систему при переходе к булевым, шкалированным и т.д. переменным.

Очень интересно, что все виды вероятностных логик получаются из подходящим образом построенных нечетких логических систем при минимальном ограничении на нормировку всей совокупности логических переменных.

## Приложения нечеткой математики.

Ниже приведены примеры приложений, в которых реально применяется нечеткая логика:

- Автоматическое управление воротами плотины на **гидроэлектростанциях** (*Tokyo Electric Pow.*)
- Упрощенное управление **роботами** (*Hirota, Fuji Electric, Toshiba, Omron*)
- **Наведение телекамер** при трансляции спортивных событий (*Omron*)
- Замена экспертов при анализе работы **биржи** (*Yamaichi, Hitachi*)
- Предотвращение нежелательных температурных флюктуаций в **системах кондиционирования воздуха** (*Mitsubishi, Sharp*)
- Эффективное и стабильное управление **автомобильными двигателями** (*Nissan*)
- Управление экономичной скоростью **автомобилей** (*Nissan, Subaru*)
- Улучшение эффективности и оптимизация **промышленных систем управления** (*Aptronix, Omron, Meiden, Sha, Micom, Mitsubishi, Nisshin-Denki, Oku-Electronics*)
- Позиционирование приводов в **производстве полупроводников**. wafer-steppers (*Canon*)
- Оптимизированное планирование **автобусных расписаний** (*Toshiba, Nippon-System, Keihan-Express*)
- Системы архивации **документов** (*Mitsubishi Elec.*)
- **Системы прогнозирования** землетрясений (*Inst. of Seismology Bureau of Metrology, Japan*)
- **Медицина**: диагностика рака (*Kawasaki Medical School*)
- Сочетание методов нечеткой логики и **нейронных сетей** (*Matsushita*)
- Распознавание рукописных символов в **карманных компьютерах** (**записных книжках**) (*Sony*)
- Распознавание движения изображения в **видеокамерах** (*Canon, Minolta*)
- Автоматическое управление двигателем **пылесосов** с автоматическим определением типа поверхности и степени засоренности (*Matsushita*)
- Управление освещенностью в **камкордерах** (*Sanyo*)
- Компенсация вибраций в **камкордерах** (*Matsushita*)
- Однокнопочное управление **стиральными машинами** (*Matsushita, Hitachi*)
- **Распознавание** рукописных текстов, объектов, голоса (*CSK, Hitachi, Hosai Univ., Ricoh*)
- Вспомогательные средства полета **вертолетов** (*Sugeno*)
- Моделирование **судебных процессов** (*Meiji Gakuin Univ, Nagoy Univ.*)
- **САПР** производственных процессов (*Aptronix, Harima, Ishikawajima-OC Engineering*)
- Управление скоростью линий и температурой при **производстве стали** (*Kawasaki Steel, New-Nippon Steel, NKK*)
- Управление **метрополитенами** для повышения удобства вождения, точности остановки и экономии энергии (*Hitachi*)
- Оптимизация потребления бензина в **автомобилях** (*NOK, Nippon Denki Tools*)
- Повышение чувствительности и эффективности **управления лифтами** (*Fujitec, Hitachi, Toshiba*)
- Повышение безопасности **ядерных реакторов** (*Hitachi, Bernard, Nuclear Fuel div.*)

**Пример.** Моделирование работы светофора с нечеткой логикой.

**Цель:** Исследование возможностей светофора с нечеткой логикой, установленного на перекрестке, при различных интенсивностях потоков автомашин и сравнение его работы с обычным светофором.

**Постановка:** В обычном светофоре время работы зеленого и красного света, а также время цикла фиксированы. Это создает некоторые трудности в движении машин, особенно, при изменении их потоков в часы пик, что довольно часто приводит к появлению автомобильных пробок.

В предлагаемом нечетком светофоре время цикла остается постоянным, однако, время его работы в режиме зеленого света должно меняться в зависимости от количества подъезжающих к перекрестку машин.

Пусть время цикла традиционного и нечеткого светофоров будет одинаковым и равным 1мин.=60сек. Длительность зеленого света обычного светофора зададим 30сек., тогда красный свет будет гореть тоже 30сек.

Для работы нечеткого светофора на перекрестке улиц Север-Юг (СЮ) и Запад-Восток (ЗВ) необходимо установить 8 датчиков (рис.1), которые считают проехавшие мимо них машины.

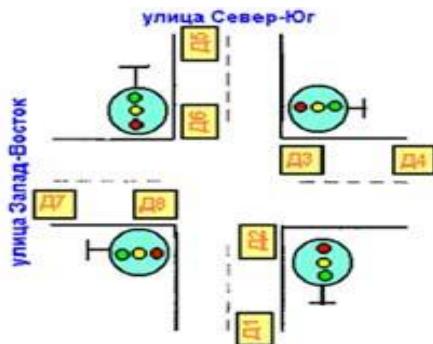


Рис.1. Расположение датчиков на перекрестке.

Светофор использует разности показаний четырех пар датчиков: **(Д1-Д2), (Д3-Д4), (Д5-Д6)** и **(Д7-Д8)**. Таким образом, если для улицы СЮ горит зеленый свет, машины проезжают перекресток и показания двух пар датчиков равны: **Д1=Д2, Д5=Д6**, а, следовательно, их разность равна нулю. В это же время на улице ЗВ перед светофором останавливаются машины, которые успели проехать только **Д4** и **Д7**. В результате можно рассчитать суммарное количество автомобилей на этой улице следующим образом:

$$(Д4-Д3)+(Д7-Д8)=(Д4-0)+(Д7-0)=Д4+Д7.$$

Для сравнения работы обоих светофоров введем показатель эффективности, в качестве которого будем рассматривать число машин, не проехавших перекресток за один цикл светофора.

Данную задачу можно сравнить с системой массового обслуживания (СМО), по двум каналам которой поступают заявки на обслуживание в виде автомашин. Показатель эффективности в этом случае число заявок, получивших отказ.

#### Решение:

Для решения поставленной задачи используется пакет **Mathlab**, т.к. он имеет в своем составе fuzzy-приложение, необходимое для моделирования работы нечеткого светофора. Поскольку работа светофора зависит от числа машин на обеих улицах и текущего времени зеленого света, для нашей подпрограммы предлагается использовать 3 входа: число машин на улице СЮ по окончанию очередного цикла, число машин на улице ЗВ по окончанию цикла и время зеленого света нечеткого светофора.

Теперь для каждой переменной надо задать лингвистические термы, соответствующие некоторым диапазонам четких значений. Так, для переменной время зеленого света предлагаются использовать три терма (рис.2):

- **малое (10-25сек.);**
- **среднее(20-40сек.);**
- **большое(35-50сек.).**

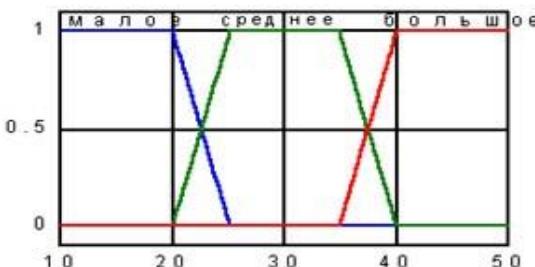


Рис.2. Функция принадлежности первой входной переменной.

Степень принадлежности четких значений термам задается с помощью функций принадлежности (в нашем случае эти функции имеют форму трапеции).

Аналогично, термы для двух оставшихся переменных будут (рис.3):

- **очень малое (0-18);**
- **малое (16-36);**
- **среднее (34-56);**
- **большое (54-76);**
- **очень большое (72-90).**

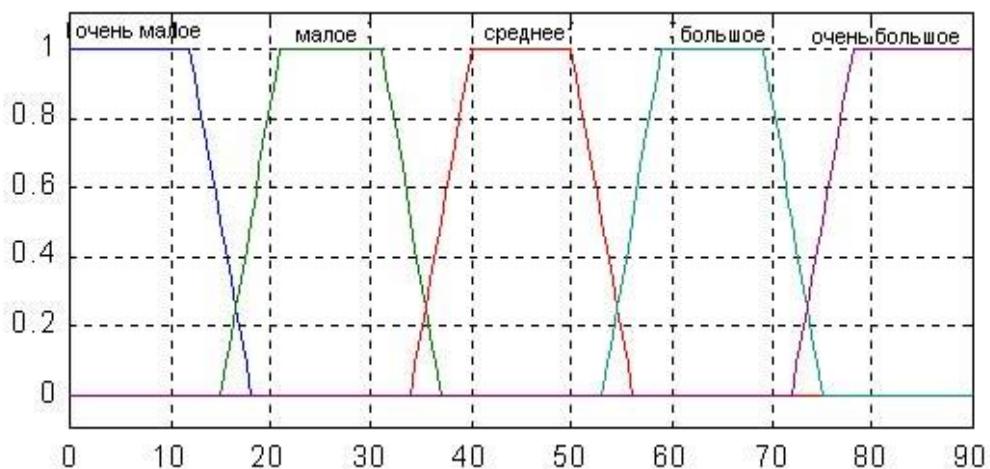


Рис.3. Функция принадлежности второй и третьей входных переменных.

Функции принадлежности здесь также имеют форму трапеции.

Так как суть работы светофора состоит в изменении времени зеленого света, в качестве выходного параметра предлагается использовать величину этого изменения. Термы в этом случае будут следующие (рис.4):

- **уменьшить (-20-0сек.);**
- **не изменять (-15-15сек.);**
- **увеличить (0-20сек.).**



Рис.4. Функция принадлежности выходной переменной.

Функции принадлежности имеют форму Гаусса. Кроме того, в программу записывается таблица правил на основе условных высказываний, которая формирует выходное значение исходя из величин входных параметров, например:

*если (число машин на улице СЮ=малое)&(число машин на улице ЗВ=большое)&(время зеленого света на улице СЮ=большое), то (время зеленого света=уменьшить).*

## Лекция 10

1. СМЫСЛ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ.
2. ОПЕРАЦИИ В ГЕНЕТИЧЕСКИХ АЛГОРИТМАХ.
3. ПРИНЦИП РАБОТЫ ГА.
4. ГЕНЕТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ.
5. ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ.
6. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - ИИ и генетические алгоритмы.

### Смысл генетических алгоритмов.

Генетические алгоритмы (ГА) - это стохастические, эвристические оптимизационные методы. Впервые предложенные Холландом (1975). Основываются на идеи эволюции с помощью естественного отбора, выдвинутой Дарвином.

ГА работают с совокупностью "особей" - популяцией, каждая из которых представляет возможное решение данной проблемы. Каждая особь оценивается мерой ее "приспособленности" согласно тому, насколько "хорошим" является соответствующее ей решение задачи. В природе это эквивалентно оценке того, насколько эффективен организм при конкуренции за ресурсы. Наиболее приспособленные особи получают возможность "воспроизводить" потомство с помощью "перекрестного скрещивания" с другими особями популяции. Это приводит к появлению новых особей, которые сочетают в себе некоторые характеристики, наследуемые ими от родителей. Наименее приспособленные особи с меньшей вероятностью смогут воспроизвести потомков, так что те свойства, которыми они обладали, будут постепенно исчезать из популяции в процессе эволюции. Иногда происходят мутации, или спонтанные изменения в генах.

Таким образом, из поколения в поколение, хорошие характеристики распространяются по всей популяции. Скрещивание наиболее приспособленных особей приводит к тому, что исследуются наиболее перспективные участки пространства поиска. В конечном итоге, популяция будет сходиться к оптимальному решению задачи. Преимущество ГА состоит в том, что он находит приблизительные оптимальные решения за относительно короткое время.

ГА состоит из следующих компонент:

**Хромосома.** Решение рассматриваемой проблемы. Состоит из генов.

**Начальная популяция** хромосом.

**Набор операторов** для генерации новых решений из предыдущей популяции.

**Целевая функция** для оценки приспособленности решений.

Чтобы применять ГА к задаче, сначала выбирается метод кодирования решений в виде строки. Фиксированная длина ( $l$ -бит) двоичной кодировки означает, что любая из  $2^l$  возможных бинарных строк представляет возможное решение задачи. По существу, такая кодировка соответствует разбиению пространства параметров на гиперкубы, которым соответствуют уникальные комбинации битов в строке – хромосоме.

### Операции в генетических алгоритмах.

Стандартные операторы для всех типов генетических алгоритмов это: селекция, скрещивание и мутация.

#### 1 Селекция или отбор

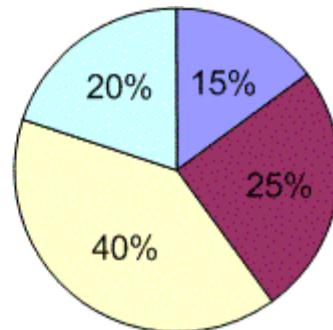
Оператор селекции (reproduction, selection) осуществляет отбор хромосом в соответствии со значениями их функции приспособленности. Существуют как минимум два популярных типа оператора селекции: рулетка и турнир.

**Метод рулетки** (roulette-wheel selection) - отбирает особей с помощью п "запусков" рулетки. Колесо рулетки содержит по одному сектору для каждого члена популяции. Размер i-ого сектора пропорционален соответствующей величине  $P_{sel}(i)$  вычисляемой по формуле:

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

При таком отборе члены популяции с более высокой приспособленностью будут чаще выбираться, чем особи с низкой приспособленностью.

1	000110110	15%
2	111001101	25%
3	000110110	40%
4	111101111	20%

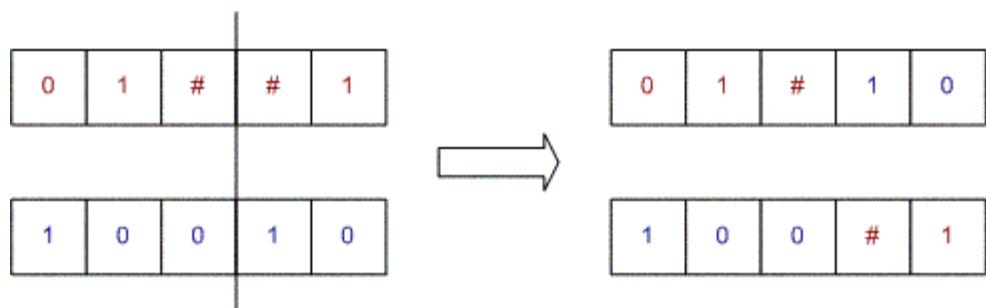


Оператор селекции типа колеса рулетки с пропорциональными функциями приспособленности (в виде секторов)

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

## 2. Скрещивание

Оператор скрещивания (crossover) осуществляет обмен частями хромосом между двумя (может быть и больше) хромосомами в популяции. Может быть одноточечным или многоточечным. Одноточечный crossover работает следующим образом. Сначала, случайным образом выбирается одна из  $I-1$  точек разрыва. Точка разрыва - участок между соседними битами в строке. Обе родительские структуры разрываются на два сегмента по этой точке. Затем, соответствующие сегменты различных родителей склеиваются и получаются два генотипа потомков.



Одноточечный оператор скрещивания (точка разрыва равна трем)

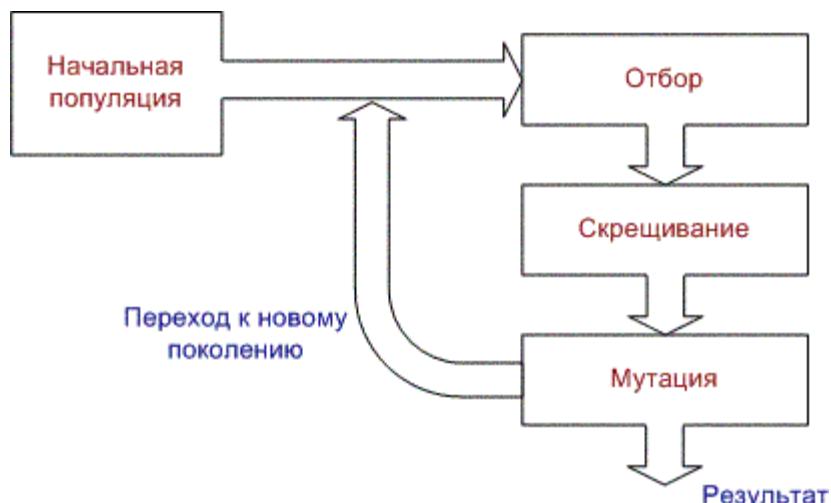
## 3. Мутация

Мутация (mutation) - стохастическое изменение части хромосом. Каждый ген строки, которая подвергается мутации, с вероятностью  $P_{mut}$  (обычно очень маленькой) меняется на другой ген.



## Принцип работы ГА.

Работа ГА представляет собой итерационный процесс, который продолжается до тех пор, пока не выполняются заданное число поколений или какой-либо иной критерий останова. На каждом поколении ГА реализуется отбор пропорционально приспособленности, crossover и мутация. Алгоритм работы простого ГА выглядит следующим образом:



Алгоритм работы классического ГА

1. НАЧАЛО /\* генетический алгоритм \*/
2. Создать начальную популяцию
3. Оценить приспособленность каждой особи
4. ПОКА НЕ останов ВЫПОЛНЯТЬ
5. НАЧАЛО /\* создать популяцию нового поколения \*/
6. ПОВТОРИТЬ (размер\_популяции/2) РАЗ
7. НАЧАЛО /\* цикл воспроизведения \*/

  - a. Выбрать две особи с высокой приспособленностью из предыдущего поколения
  - b. Скрестить выбранные особи и получить двух потомков
  - c. Оценить приспособленности потомков
  - d. Поместить потомков в новое поколение

8. КОНЕЦ
9. ЕСЛИ популяция сошлась, ТО останов = TRUE
10. КОНЕЦ
11. КОНЕЦ

Отбор в генетическом алгоритме тесно связан с принципами естественного отбора в природе следующим образом:

Приспособленность индивидуума	Значение целевой функции на этом индивидууме.
Выживание наиболее приспособленных	Популяция следующего поколения формируется в соответствии с целевой функцией. Чем более приспособлен индивидуум, тем больше вероятность его участия в crossover, т.е. размножении.

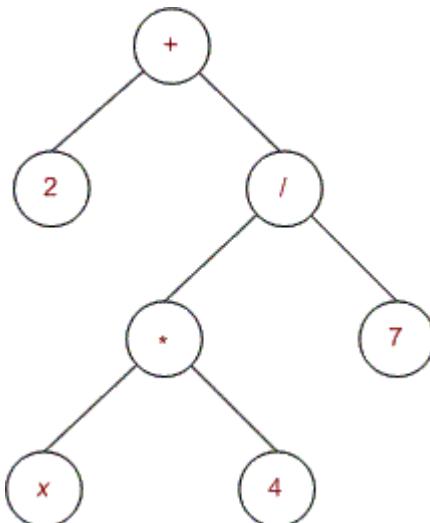
Таким образом, модель отбора определяет, каким образом следует строить популяцию следующего поколения. Как правило, вероятность участия индивидуума в скрещивании берется пропорциональной его приспособленности. Часто используется так называемая *стратегия элитизма*, при которой несколько лучших индивидуумов переходят в следующее поколение без изменений, не участвуя в crossover и отборе. В любом случае, каждое следующее поколение будет в среднем лучше предыдущего. Когда приспособленность индивидуумов перестает заметно увеличиваться, процесс останавливают и в качестве решения задачи оптимизации берут наилучшего из найденных индивидуумов.

### Генетическое программирование.

Идею генетического программирования (ГП) - впервые предложил Коза в 1992 году, опираясь на концепцию генетических алгоритмов (ГА). Эта идея заключается в том, что в отличие от ГА в ГП все операции производятся не над строками, а над деревьями. При этом используются такие же операторы, как и в ГА: селекция, скрещивание и мутация.

В ГП хромосомами являются программы. Программы представлены деревьями с функциональными (промежуточными) и терминальными (конечными) элементами. Терминальными элементами являются константы, действия и функции без аргументов, функциональными - функции, использующие аргументы.

Для примера рассмотрим функцию  $2+x^4/7$ , представленную на рисунке ниже. Терминальные элементы  $T = \{2, x, 4, 7\}$ , функциональные  $F = \{+, *, /\}$ .



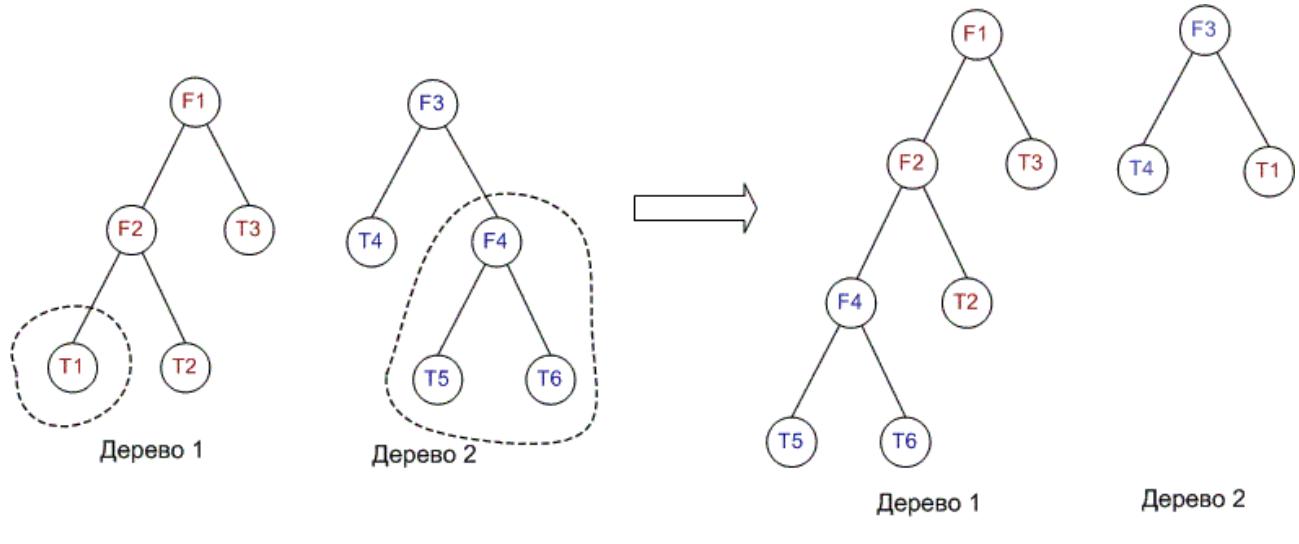
Древовидное представление функции  $2+x^4/7$

Для того чтобы применить ГП к какой-либо проблеме, необходимо определить:

- множество терминальных элементов.
- множество функциональных элементов.
- меру приспособленности (fitness).
- параметры, контролирующие эволюцию.
- критерий останова эволюции.

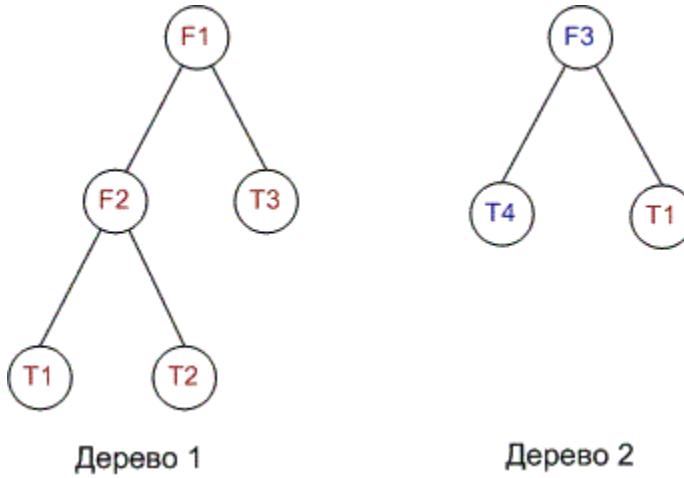
Алгоритм работы ГП такой же как и ГА: селекция, скрещивание и мутация. Однако поскольку ГП оперирует над деревьями, а не над строками, то операторы скрещивания и мутации имеют отличия.

Оператор скрещивания работает следующим образом: выбираются случайные части родительских деревьев и эти части меняются местами.



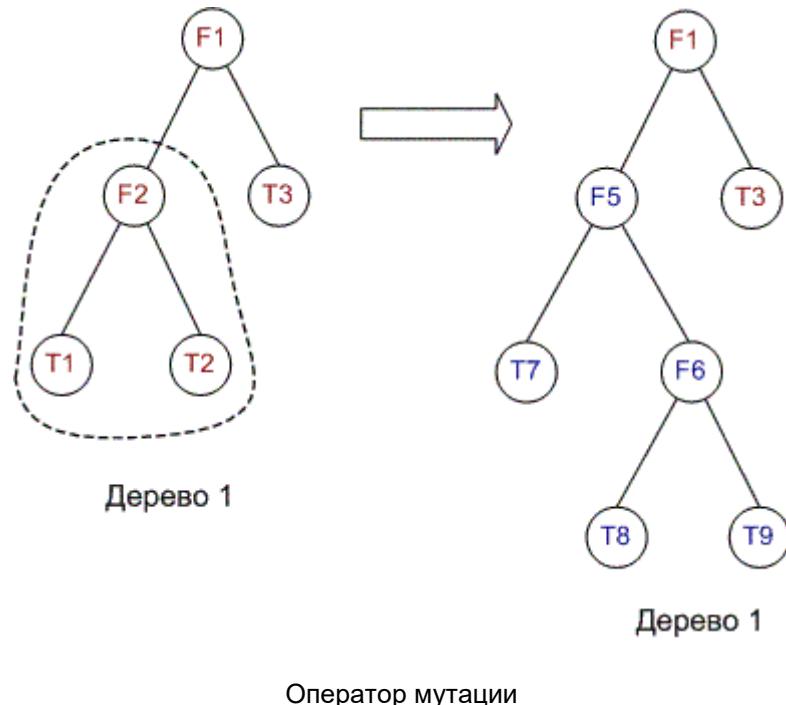
Скрещивание двух деревьев

В качестве особенности необходимо отметить, что в ГП размер хромосомы меняется. Чтобы предотвратить чрезмерное разрастание дерева, вводят максимальное количество функциональных элементов в дереве или максимальную глубину дерева. Однако при операции скрещивания возможна ситуация, когда при скрещивании двух деревьев получится одно из деревьев, превосходящее заданный лимит. В этом случае вместо конфликтного дерева копируется родительское дерево.



Разрешение конфликтной ситуации предыдущего оператора скрещивания при максимальной глубине дерева равной трем

Оператор мутации случайно удаляет часть дерева и заменяет ее новым деревом.



### **Применение генетических алгоритмов.**

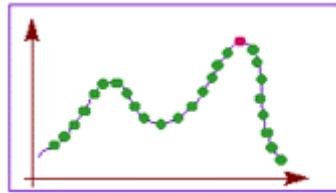
С давних пор известны два основных пути решения задач оптимизации - переборный и локально-градиентный. У этих методов свои достоинства и недостатки, и в каждом конкретном случае следует подумать, какой из них выбрать.

Рассмотрим достоинства и недостатки стандартных и генетических методов на примере классической задачи коммивояжера (TSP - traveling salesman problem). Суть задачи состоит в том, чтобы найти кратчайший замкнутый путь обхода нескольких городов, заданных своими координатами. Уже для 30 городов поиск оптимального пути представляет собой сложную задачу, побудившую развитие различных методов (в том числе, нейронных сетей и генетических алгоритмов).

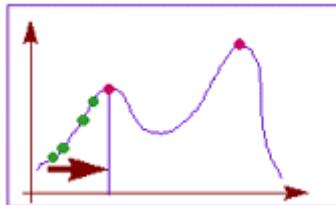


Каждый вариант решения (для 30 городов) - это числовая строка, где на j-ом месте стоит номер j-ого по порядку обхода города. Таким образом, в этой задаче 30 параметров, причем не все комбинации значений допустимы. Естественно, первой идеей является полный перебор всех вариантов обхода.

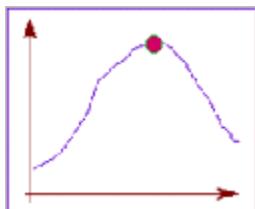
Переборный метод наиболее прост по своей сути и тривиален в программировании. Для поиска оптимального решения (точки максимума целевой функции) требуется последовательно вычислить значения целевой функции во всех возможных точках, запоминая максимальное из них. Недостатком этого метода является большая вычислительная стоимость. Однако, если перебор всех вариантов за разумное время возможен, то можно быть абсолютно уверенным в том, что найденное решение действительно оптимально.



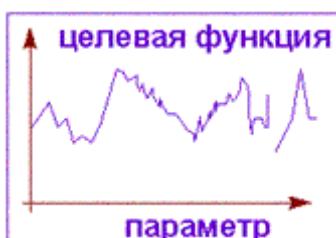
Второй популярный способ основан на методе градиентного спуска. При этом вначале выбираются некоторые случайные значения параметров, а затем эти значения постепенно изменяют, добиваясь наибольшей скорости роста целевой функции. Достигнув локального максимума, такой алгоритм останавливается, поэтому для поиска глобального оптимума потребуются дополнительные усилия.



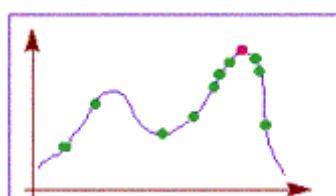
Градиентные методы работают очень быстро, но не гарантируют оптимальности найденного решения. Они идеальны для применения в так называемых **унимодальных** задачах, где целевая функция имеет единственный локальный максимум (он же — глобальный). Отметим, что задача коммивояжера унимодальной не является.



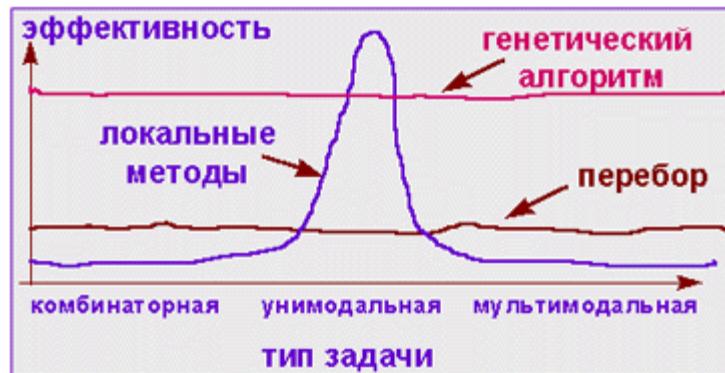
Типичная практическая задача, как правило, мультимодальна и многомерна, то есть содержит много параметров. Для таких задач не существует единого универсального метода, который позволял бы достаточно быстро найти точное решение.



Однако, комбинируя переборный и градиентный методы, можно надеяться получить хотя бы приближенное решение, точность которого будет возрастать при увеличении времени расчета.



Генетический алгоритм представляет собой именно такой комбинированный метод. Механизмы скрещивания и мутации в каком-то смысле реализуют переборную часть метода, а отбор лучших решений - градиентный спуск. На рисунке показано, что такая комбинация позволяет обеспечить хорошую эффективность генетического поиска для любых типов задач.



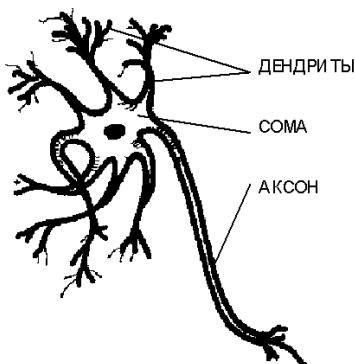
Итак, если на некотором множестве задана сложная функция от нескольких переменных, то генетический алгоритм - это программа, которая за разумное время находит точку, где значение функции как угодно близко к максимальному. Выбирая приемлемое время расчета, мы получим одно из лучших решений, которые вообще возможно получить за это время.

## Лекция 11

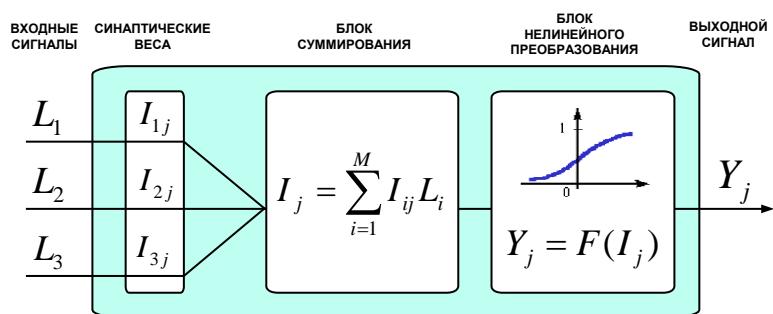
1. БИОЛОГИЧЕСКИЙ НЕЙРОН И ФОРМАЛЬНАЯ МОДЕЛЬ НЕЙРОНА.
2. ПЕРСЕПТРОНЫ.
3. НЕЙРОННЫЕ СЕТИ.
  - Построение нейронной сети.
  - Нейронные сети: обучение без учителя.
  - Нейронные сети Хопфилда и Хэмминга.
4. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - ИИ и нейронные сети.

### Биологический нейрон и формальная модель нейрона.

Биологический нейрон имеет следующий вид:



В 1943 году Дж. Маккалоки и У. Питт предложили формальную модель биологического нейрона как устройства, имеющего несколько входов (входные синапсы – дендриты), и один выход (выходной синапс – аксон):

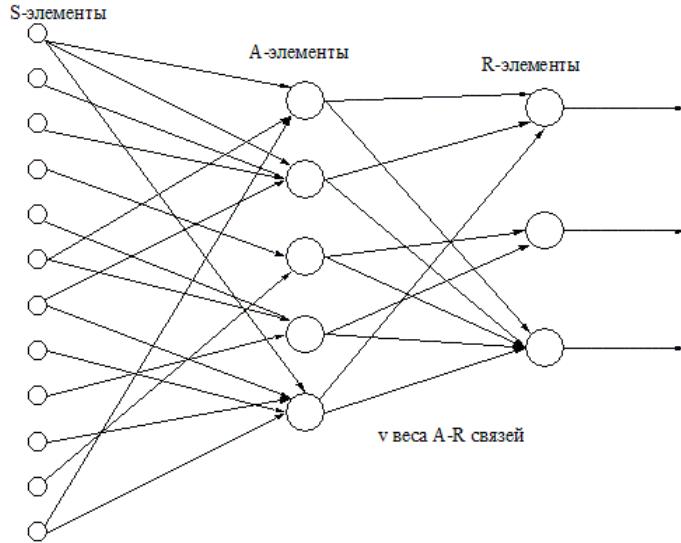


Дендриты получают информацию от источников информации (рецепторов)  $L_i$ , в качестве которых могут выступать и нейроны. Набор входных сигналов  $\{L_i\}$  характеризует объект, его состояние или ситуацию, обрабатываемую нейроном.

Каждому  $i$ -му входу  $j$ -го нейрона ставится в соответствие некоторый весовой коэффициент  $I_{ij}$ , характеризующий степень влияния сигнала с этого входа на аргумент передаточной (активационной) функции, определяющей сигнал  $Y_j$  на выходе нейрона. В нейроне происходит взвешенное суммирование входных сигналов, и далее это значение используется как аргумент активационной (передаточной) функции нейрона.

### Персептроны.

Персептроны рассматривались как эвристические модели механизма мозга. Впоследствии они стали основополагающей схемой в построении кусочно-линейных моделей. В наиболее простом виде персептрон может быть представлен в виде следующей сети:



Персептрон состоит из совокупности сенсорных элементов (S-элементов), на которые поступают входные сигналы. Сенсорные элементы каким-либо образом (возможно и случайно) связаны с совокупностью ассоциативных элементов (A-элементов). Выходной сигнал A-элементов отличается от нуля только тогда, когда некоторое число S-элементов, воздействующих на один A-элемент, имеет определенную величину или форму сигнала. Ассоциативные элементы соединены с реагирующими элементами (R-элементами) связями, коэффициенты усиления ( $v$ ) которых переменны и могут изменяться (например, в процессе обучения). Взвешенные комбинации выходов R-элементов составляют реакцию системы, которая указывает на определенное решение. Число R-элементов обычно связано с числом возможных решений.

Впервые идея персепtronов была применена для решения задач распознавания образов. В этом случае решение связано с определением принадлежности распознаваемого объекта определенному классу. Если распознаются только два класса, то в персептроне устанавливается только один R-элемент, который обладает двумя реакциями — положительной и отрицательной. Если классов больше двух, то для каждого из них устанавливают свой R-элемент, а выход каждого такого элемента представляет линейную комбинацию выходов A-элементов:

$$R_j = \Theta_j + \sum_{i=1}^n v_{ij} \cdot x_i$$

где  $R_j$  — реакция  $j$ -го R-элемента;  $x_i$  — реакция  $i$ -го A-элемента;  $v_{ij}$  — вес связи от  $i$ -го A-элемента к  $j$ -му R элементу;  $\Theta_j$  — некоторая заранее фиксированная величина (порог)  $j$ -го R-элемента.

Аналогично определяется значение  $i$ -го A-элемента:

$$x_i = \Delta_i + \sum_{k=1}^s y_k$$

Здесь сигнал  $y_k$  может быть непрерывным, но чаще всего он принимает только два значения: 0 или 1. Сигналы от S-элементов подаются на входы A-элементов с постоянными весами равными единице, но каждый A-элемент связан только с группой S-элементов. Предположим, что требуется обучить персептрон различать два класса  $V_1$  и  $V_2$ . Будем считать, что в персептроне существует два R-элемента, один из которых соответствует классу  $V_1$ , а другой — классу  $V_2$ . Персептрон будет обучен правильно, если выход  $R_1$  превышает  $R_2$ , ко-

гда распознаваемый объект принадлежит классу  $V_1$ , и наоборот. Разделение объектов на два класса можно провести и с помощью только одного R-элемента. Тогда объекту класса  $V_1$  должна соответствовать положительная реакция R-элемента, а объектам класса  $V_2$  — отрицательная.

Персептрон обучается путем предъявления обучающей последовательности изображений объектов, принадлежащих классам  $V_1$  и  $V_2$ . В процессе обучения изменяются веса  $v_i$  A-элементов. В частности, если применяется система с коррекцией ошибок, прежде всего, учитывается правильность решения, принимаемого персептроном. Если решение правильное, то веса связей всех A-элементов, ведущих к R-элементу, выдавшему правильное решение, увеличиваются, а веса всех остальных A-элементов остаются неизменными. Можно также оставлять неизменными веса первых A-элементов, но уменьшать веса последних. В некоторых случаях веса первых связей увеличивают, а последних — уменьшают. После процесса обучения персептрон сам, уже без коррекции, начинает классифицировать новые объекты.

Если персептрон действует по описанной схеме и в нем допускаются лишь связи, идущие от бинарных S-элементов к A-элементам и от A-элементов к единственному R-элементу, то такой персептрон принято называть элементарным  $\alpha$ -персептроном. Обычно классификация  $C(W)$  задается заранее (например, учителем). Персептрон должен выработать в процессе обучения классификацию, заданную заранее.

Для персептронов было получено несколько фундаментальных результатов, которые позволяют определить основные свойства персептрана.

**Теорема 1.** Класс элементарных  $\alpha$ -персептронов, для которых существует решение для любой классификации, не является пустым.

Эта теорема утверждает, что для любой классификации обучающей последовательности можно подобрать такой набор (из бесконечного набора) A-элементов, в котором будет осуществлено любое заданное разделение обучающей последовательности при помощи линейного решающего правила

$$R_j = \Theta_j + \sum_{i=1}^n v_{ij} \cdot x_i$$

**Теорема 2.** Если для некоторой классификации  $C(W)$  решение существует, то в процессе обучения  $\alpha$ -персептрана с коррекцией ошибок, начинаящегося с произвольного исходного состояния, это решение будет достигнуто в течение конечного промежутка времени.

Смысл этой теоремы состоит в том, что если относительно любой заданной классификации можно найти набор A-элементов, в котором существует решение, то в рамках этого набора оно будет достигнуто в конечный промежуток времени.

Обычно обсуждают свойства бесконечного персептрана, т. е. персептрана с бесконечным числом A-элементов со всевозможными связями с S-элементами (полный набор A-элементов). В таких персептранах решение всегда существует, а раз оно существует, то оно и достижимо в  $\alpha$ -персептранах с коррекцией ошибок.

## Нейронные сети.

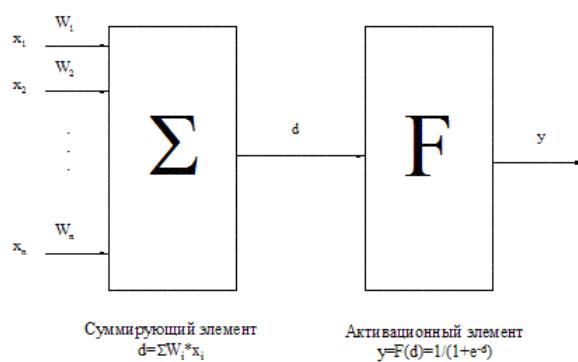
### 1 Построение нейронной сети.

Способность нейронной сети к обучению впервые исследована Дж. Маккалоком и У. Питтом. В 1943 году вышла их работа "Логическое исчисление идей, относящихся к нервной деятельности", в которой была построена модель нейрона, и сформулированы принципы построения искусственных нейронных сетей. Крупный толчок развитию нейрокибернетики дал американский нейрофизиолог Френк Розенблatt, предложивший в 50-х годах свою модель нейронной сети — персептрон. В 70-е годы была предложена модель когнитрона, способного распознавать достаточно сложные объекты независимо от поворота и изменения масштаба изображения.

В 1982 году американский биофизик Дж. Хопфилд предложил оригинальную модель нейронной сети, названную его именем. В последующем было найдено множество эффективных алгоритмов: сеть встречного потока, двунаправленная ассоциативная память и др. В

1986 году Дж. Хинтон опубликовал статью с описанием модели нейронной сети и алгоритмом ее обучения, что дало новый толчок исследованиям в области искусственных нейронных сетей.

Нейронная сеть состоит из множества одинаковых элементов — нейронов, поэтому начнем с них рассмотрение работы искусственной нейронной сети. Биологический нейрон моделируется как устройство, имеющее несколько входов (дendриты), и один выход (аксон). Каждому входу ставится в соответствие некоторый весовой коэффициент ( $w$ ), характеризующий пропускную способность канала и оценивающий степень влияния сигнала с этого входа на сигнал на выходе. В зависимости от конкретной реализации, обрабатываемые нейроном сигналы, могут быть аналоговыми или цифровыми. В теле нейрона происходит взвешенное суммирование входных сигналов, и далее это значение является аргументом активационной функции нейрона, один из возможных вариантов которой представлен на следующем рисунке:



Будучи соединенными определенным образом, нейроны образуют нейронную сеть. В работе нейронной сети может присутствовать этап обучения. Под обучением понимается процесс адаптации сети к предъявляемым эталонным объектам путем модификации (в соответствии с тем или иным алгоритмом) весовых коэффициентов связей между нейронами. Заметим, что этот процесс является результатом работы алгоритма функционирования сети, а не предварительно заложенных в нее знаний человека, как это часто бывает в системах искусственного интеллекта.

Среди различных структур нейронных сетей (НС) одной из наиболее известных является многослойная структура, в которой каждый нейрон произвольного слоя связан со всеми аксонами нейронов предыдущего слоя или, в случае первого слоя, со всеми входами НС. Такие НС называются полносвязанными. Когда в сети только один слой, алгоритм ее обучения с учителем довольно очевиден, так как правильные выходные состояния нейронов единственного слоя заведомо известны, и подстройка синаптических связей идет в направлении, минимизирующем ошибку на выходе сети. По этому принципу строится, например, алгоритм обучения однослойного персептрона. В многослойных же сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, как правило, не известны, и двух или более слойный персептрон уже невозможно обучить, руководствуясь только величинами ошибок на выходах НС. Один из вариантов решения этой проблемы – разработка наборов выходных сигналов, соответствующих входным, для каждого слоя НС, что, конечно, является очень трудоемкой операцией и не всегда осуществимо. Второй вариант – динамическая подстройка весовых коэффициентов синапсов, в ходе которой выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые повлекли уменьшение ошибки на выходе всей сети. Очевидно, что данный метод, несмотря на свою кажущуюся простоту, требует громоздких рутинных вычислений. И, наконец, третий, наиболее приемлемый вариант – распространение сигналов ошибки от выходов НС к ее входам, в направлении обратному распространению сигналов в обычном режиме работы. Этот алгоритм обучения НС получил название процедуры обратного распространения. Рассмотрим его более подробно.

В качестве целевой функции ошибки НС, которую необходимо минимизировать, может быть выбрана функция:

$$E(w) = \frac{1}{2} \sum_{j,p} (y_{j,p}^{(N)} - d_{j,p})^2 \quad (1)$$

где  $y_{j,p}^{(N)}$  – реальное выходное состояние нейрона  $j$  выходного слоя  $N$  нейронной сети при подаче на ее входы  $p$ -го объекта;  $d_{j,p}$  – идеальное (желаемое) выходное состояние этого нейрона. Суммирование ведется по всем нейронам выходного слоя и по всем обрабатываемым сетью образом. Минимизация ведется методом градиентного спуска, что означает подстройку весовых коэффициентов следующим образом:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \frac{\partial E}{\partial w_{ij}} \quad (2)$$

Здесь  $w_{ij}$  – весовой коэффициент синаптической связи, соединяющей  $i$ -ый нейрон слоя  $n-1$  с  $j$ -ым нейроном слоя  $n$ ,  $\eta$  – коэффициент скорости обучения,  $0 < \eta < 1$ .

Показано, что:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \cdot \frac{\partial s_j}{\partial w_{ij}} \quad (3)$$

Здесь под  $y_j$ , как и раньше, подразумевается выход нейрона  $j$ , а под  $s_j$  – взвешенная сумма его входных сигналов, то есть аргумент активационной функции. Так как множитель  $dy/ds$  является производной этой функции по ее аргументу, из этого следует, что производная активационной функции должна быть определена на всей оси абсцисс. В связи с этим, функция единичного скачка и прочие активационные функции с неоднородностями не подходят для рассматриваемых нейронных сетей. В них применяются такие гладкие функции, как гиперболический тангенс. В этом случае

$$\frac{dy}{ds} = 1 - s^2 \quad (4)$$

Третий множитель  $\partial s_j / \partial w_{ij}$  равен выходу нейрона предыдущего слоя  $y_i^{(n-1)}$ . Что касается первого множителя в (3), он раскладывается следующим образом:

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot \frac{\partial s_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot w_{jk}^{(n+1)} \quad (5)$$

Здесь суммирование по  $k$  выполняется среди нейронов слоя  $n+1$ . Введя новую переменную

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \quad (6)$$

получим рекурсивную формулу для расчетов величин  $\delta_j^{(n)}$  слоя  $n$  из величин  $\delta_k^{(n+1)}$  более старшего слоя  $n+1$ .

$$\delta_j^{(n)} = \left[ \sum_k \delta_k^{(n+1)} \cdot w_{jk}^{(n+1)} \right] \cdot \frac{dy_j}{ds_j} \quad (7)$$

Для выходного же слоя

$$\delta_l^{(N)} = (y_l^{(N)} - d_l) \cdot \frac{dy_l}{ds_l} \quad (8)$$

Теперь мы можем переписать (2) в следующем виде:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot y_i^{(n-1)} \quad (9)$$

Иногда для придания процессу коррекции весов некоторой инерционности, сглаживающей резкие скачки при перемещении по поверхности целевой функции, (9) дополняется значением изменения веса на предыдущей итерации

$$\Delta w_{ij}^{(n)}(t) = -\eta \cdot (\mu \cdot \Delta w_{ij}^{(n)}(t-1) + (1-\mu) \cdot \delta_j^{(n)} \cdot y_i^{(n-1)}) \quad (10)$$

где  $\mu$  – коэффициент инерционности,  $t$  – номер текущей итерации.

Таким образом, полный алгоритм обучения НС с помощью процедуры обратного распространения строится так:

- Подать на входы сети один из возможных объектов и в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних

$$s_j^{(n)} = \sum_{i=0}^M y_i^{(n-1)} \cdot w_{ij}^{(n)} \quad (11)$$

где  $M$  – число нейронов в слое  $n-1$  с учетом нейрона с постоянным выходным состоянием +1, задающего смещение;  $y_i^{(n-1)} = x_{ij}^{(n)}$  –  $i$ -ый вход нейрона  $j$  слоя  $n$ .

$$y_j^{(n)} = f(s_j^{(n)}), \text{ где } f() \text{ – сигмоид} \quad (12)$$

$$y_q^{(0)} = l_q, \quad (13)$$

где  $l_q$  –  $q$ -ая компонента вектора входного объекта.

- Рассчитать  $\delta^{(n)}$  для выходного слоя по формуле (8). Рассчитать по формуле (9) или (10) изменения весов  $\Delta w^{(n)}$  слоя  $N$ .
- Рассчитать по формулам (7) и (9) (или (7) и (10)) соответственно  $\delta^{(n)}$  и  $\Delta w^{(n)}$  для всех остальных слоев,  $n=N-1, \dots, 1$ .
- Скорректировать все веса в НС

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t) \quad (14)$$

- Если ошибка сети существенна, перейти на шаг 1. В противном случае – конец.

Сети на шаге 1 попеременно в случайном порядке предъявляются все объекты обучающей выборки, чтобы сеть, образно говоря, не забывала одни по мере запоминания других. Из выражения (9) следует, что когда выходное значение  $y_i^{(n-1)}$  стремится к нулю, эффективность обучения заметно снижается. При двоичных входных векторах в среднем половина весовых коэффициентов не будет корректироваться, поэтому область возможных значений выходов нейронов  $[0, 1]$  желательно сдвинуть в пределы  $[-0.5, +0.5]$ , что достигается простыми модификациями логических функций. Например, сигмоид с экспонентой преобразуется к виду

$$f(x) = -0.5 + \frac{1}{1 + e^{-\alpha \cdot x}} \quad (15)$$

Для нейронных сетей очень важным является вопрос о емкости, то есть числе объектов, предъявляемых на ее входы, которые она способна научиться распознавать. Для сетей с

числом слоев больше двух, он остается открытым. Показано, что для НС с двумя слоями, то есть выходным и одним скрытым слоем, детерминистская емкость сети  $C_d$  оценивается так:

$$N_w/N_y < C_d < N_w/N_y \cdot \log(N_w/N_y) \quad (16)$$

где  $N_w$  – число подстраиваемых весов,  $N_y$  – число нейронов в выходном слое. Следует отметить, что данное выражение получено с учетом некоторых ограничений. Во-первых, число входов  $N_x$  и нейронов в скрытом слое  $N_h$  должно удовлетворять неравенству  $N_x + N_h > N_y$ . Во-вторых,  $N_w/N_y > 1000$ . Однако вышеприведенная оценка выполнялась для сетей с активационными функциями нейронов в виде порога, а емкость сетей с гладкими активационными функциями, например – (15), обычно больше. Кроме того, фигурирующее в названии емкости прилагательное "детерминистский" означает, что полученная оценка емкости подходит абсолютно для всех возможных входных образов, которые могут быть представлены  $N_x$  входами. В действительности распределение входных объектов, как правило, обладает некоторой регулярностью, что позволяет НС проводить обобщение и, таким образом, увеличивать реальную емкость. Так как распределение объектов, в общем случае, заранее не известно, мы можем говорить о такой емкости только предположительно, но обычно она раза в два превышает емкость детерминистскую.

В продолжение разговора о емкости НС логично затронуть вопрос о требуемой мощности выходного слоя сети, выполняющего окончательную классификацию объектов. Дело в том, что для разделения множества входных образов, например, по двум классам достаточно всего одного выхода. При этом каждый логический уровень – "1" и "0" – будет обозначать отдельный класс. На двух выходах можно закодировать уже 4 класса и так далее. Однако результаты работы сети, организованной таким образом, можно сказать – "под завязку", – не очень надежны. Для повышения достоверности классификации желательно ввести избыточность путем выделения каждому классу одного нейрона в выходном слое или, что еще лучше, нескольких, каждый из которых обучается определять принадлежность объекта к классу со своей степенью достоверности. Например: высокой, средней и низкой. Такие НС позволяют проводить классификацию входных объектов, объединенных в нечеткие (размытые или пересекающиеся) множества. Это свойство приближает подобные НС к условиям реальной жизни.

Рассматриваемая НС имеет несколько "узких мест". Во-первых, в процессе обучения может возникнуть ситуация, когда большие положительные или отрицательные значения весовых коэффициентов смещают рабочую точку на сигмоидах многих нейронов в область насыщения. Малые величины производной от логической функции приведут в соответствие с (7) и (8) к остановке обучения. Во-вторых, применение метода градиентного спуска не гарантирует, что будет найден глобальный, а не локальный минимум целевой функции. Эта проблема связана еще с одной, а именно – с выбором величины скорости обучения. Доказательство сходимости обучения в процессе обратного распространения основано на производных, то есть приращении весов и, следовательно, скорость обучения должны быть бесконечно малыми, однако в этом случае обучение будет происходить неприемлемо медленно. С другой стороны, слишком большие коррекции весов могут привести к постоянной неустойчивости процесса обучения. Поэтому в качестве  $\eta$  обычно выбирается число меньше 1, но не очень маленькое, например, 0.1, и оно, вообще говоря, может постепенно уменьшаться в процессе обучения. Кроме того, для исключения случайных попаданий в локальные минимумы иногда, после того как значения весовых коэффициентов стабилизируются,  $\eta$  кратковременно сильно увеличивают, чтобы начать градиентный спуск из новой точки. Если повторение этой процедуры несколько раз приведет алгоритм в одно и то же состояние НС, можно более или менее уверенно сказать, что найден глобальный максимум, а не какой-то другой. Существуют и иные методы исключения локальных минимумов, а заодно и остановки обучения.

## 2. Нейронные сети: обучение без учителя.

Рассмотренный в предыдущем пункте алгоритм подразумевает наличие некоего внешнего звена, предоставляющего сети обучающую информацию. Алгоритмы, использующие подобную концепцию, называются алгоритмами обучения с учителем. Для их успешного функционирования необходимо наличие экспертов, создающих на предварительном этапе для каждого входного объекта эталонный выходной. Так как создание искусственного интеллекта

движется по пути копирования природных прообразов, ученые не прекращают спор на тему, можно ли считать алгоритмы обучения с учителем натуральными или же они полностью искусственны. Например, обучение человеческого мозга, на первый взгляд, происходит без учителя: на зрительные, слуховые, тактильные и прочие рецепторы поступает информация извне, и внутри нервной системы происходит некая самоорганизация. Однако, нельзя отрицать и того, что в жизни человека не мало учителей – и в буквальном, и в переносном смысле, – которые координируют внешние воздействия. Вместе с тем, чем бы ни закончился спор приверженцев этих двух концепций обучения, они обе имеют право на существование.

Главная черта, делающая обучение без учителя привлекательным, – это его "самостоятельность". Процесс обучения, как и в случае обучения с учителем, заключается в подстраивании весов синапсов. Некоторые алгоритмы, правда, изменяют и структуру сети, то есть количество нейронов и их взаимосвязи, но такие преобразования правильнее назвать более широким термином – самоорганизацией, и в рамках данной главы они рассматриваться не будут. Очевидно, что подстройка синапсов может проводиться только на основании информации, доступной в нейроне, то есть его состояния и уже имеющихся весовых коэффициентов. Исходя из этого соображения и, что более важно, по аналогии с известными принципами самоорганизации нервных клеток, построены алгоритмы обучения Хебба.

Сигнальный метод обучения Хебба заключается в изменении весов по следующему правилу:

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \cdot y_i^{(n-1)} \cdot y_j^{(n)} \quad (16)$$

где  $y_i^{(n-1)}$  – выходное значение нейрона i слоя (n-1),  $y_j^{(n)}$  – выходное значение нейрона j слоя n;  $w_{ij}(t)$  и  $w_{ij}(t-1)$  – весовой коэффициент синапса, соединяющего эти нейроны, на итерациях t и t-1 соответственно;  $\alpha$  – коэффициент скорости обучения. Здесь и далее, для общности, под n подразумевается произвольный слой сети. При обучении по данному методу усиливаются связи между возбужденными нейронами.

Существует также и дифференциальный метод обучения Хебба.

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \cdot [y_i^{(n-1)}(t) - y_i^{(n-1)}(t-1)] \cdot [y_j^{(n)}(t) - y_j^{(n)}(t-1)] \quad (17)$$

Здесь  $y_i^{(n-1)}(t)$  и  $y_i^{(n-1)}(t-1)$  – выходное значение нейрона i слоя n-1 соответственно на итерациях t и t-1;  $y_j^{(n)}(t)$  и  $y_j^{(n)}(t-1)$  – то же самое для нейрона j слоя n. Как видно из формулы (17), сильнее всего обучаются синапсы, соединяющие те нейроны, выходы которых наиболее динамично изменились в сторону увеличения.

Полный алгоритм обучения с применением вышеприведенных формул будет выглядеть так:

1. На стадии инициализации всем весовым коэффициентам присваиваются небольшие случайные значения.
2. На входы сети подается входной объект, и сигналы возбуждения распространяются по всем слоям согласно принципам классических прямопоточных (feedforward) сетей, то есть для каждого нейрона рассчитывается взвешенная сумма его входов, к которой затем применяется активационная (передаточная) функция нейрона, в результате чего получается его выходное значение  $y_i^{(n)}$ ,  $i=0\dots M_i-1$ , где  $M_i$  – число нейронов в слое i;  $n=0\dots N-1$ , а N – число слоев в сети.
3. На основании полученных выходных значений нейронов по формуле (16) или (17) производится изменение весовых коэффициентов.
4. Цикл с шага 2, пока выходные значения сети не стабилизируются с заданной точностью.

Применение этого нового способа определения завершения обучения, отличного от использовавшегося для сети обратного распространения, обусловлено тем, что подстраиваемые значения синапсов фактически не ограничены.

На втором шаге цикла попеременно предъявляются все объекты из входного набора. Следует отметить, что вид откликов на каждый класс входных объектов не известен заранее и будет представлять собой произвольное сочетание состояний нейронов выходного слоя, обусловленное случайнм распределением весов на стадии инициализации. Вместе с тем, сеть способна обобщать схожие объекты, относя их к одному классу. Тестирование обучен-

ной сети позволяет определить топологию классов в выходном слое. Для приведения откликов обученной сети к удобному представлению можно дополнить сеть одним слоем, который, например, по алгоритму обучения однослойного персептрона необходимо заставить отображать выходные реакции сети в требуемые классы.

Другой алгоритм обучения без учителя – алгоритм Кохонена – предусматривает подстройку синапсов на основании их значений от предыдущей итерации.

$$w_{ij}(t) = w_{ij}(t-1) + \alpha \cdot [y_i^{(n-1)} - w_{ij}(t-1)] \quad (18)$$

Обучение в этом случае сводится к минимизации разницы между входными сигналами нейрона, поступающими с выходов нейронов предыдущего слоя  $y_i^{(n-1)}$ , и весовыми коэффициентами его синапсов.

Полный алгоритм обучения имеет примерно такую же структуру, как в методах Хебба, но на шаге 3 из всего слоя выбирается нейрон, значения синапсов которого максимально походят на входной объект, и подстройка весов по формуле (18) проводится только для него. Эта, так называемая, аккредитация (термин НС) может сопровождаться затормаживанием всех остальных нейронов слоя и введением выбранного нейрона в насыщение. Выбор такого нейрона может осуществляться, например, расчетом скалярного произведения вектора весовых коэффициентов с вектором входных значений. Максимальное произведение дает выигравший нейрон.

Другой вариант – расчет расстояния между этими векторами в р-мерном пространстве, где р – размер векторов.

$$D_j = \sqrt{\sum_{i=0}^{p-1} (y_i^{(n-1)} - w_{ij})^2}, \quad (19)$$

где j – индекс нейрона в слое n, i – индекс суммирования по нейронам слоя (n-1),  $w_{ij}$  – вес синапса, соединяющего нейроны; выходы нейронов слоя (n-1) являются входными значениями для слоя n. Корень в формуле (19) брать не обязательно, так как важна лишь относительная оценка различных  $D_j$ .

В данном случае, "побеждает" нейрон с наименьшим расстоянием. Иногда слишком часто получающие аккредитацию нейроны принудительно исключаются из рассмотрения, чтобы "уравнять права" всех нейронов слоя. Простейший вариант такого алгоритма заключается в торможении только что выигравшего нейрона.

При обучении по алгоритму Кохонена существует практика нормализации входных объектов, а также – на стадии инициализации – и нормализации начальных значений весовых коэффициентов.

$$x_i = x_i / \sqrt{\sum_{j=0}^{n-1} x_j^2}, \quad (20)$$

где  $x_i$  – i-ая компонента вектора входного образа или вектора весовых коэффициентов, а n – его размерность. Это позволяет сократить длительность процесса обучения.

Инициализация весовых коэффициентов случайными значениями может привести к тому, что различные классы, которым соответствуют плотно распределенные входные объекты, сольются или, наоборот, раздробятся на дополнительные подклассы в случае близких объектов одного и того же класса. Для избежания такой ситуации используется метод выпуклой комбинации. Суть его сводится к тому, что входные нормализованные объекты подвергаются преобразованию:

$$x_i = \alpha(t) \cdot x_i + (1 - \alpha(t)) \cdot \frac{1}{\sqrt{n}} \quad (21)$$

где  $x_i$  – i-ая компонента входного образа,  $n$  – общее число его компонент,  $\alpha(t)$  – коэффициент, изменяющийся в процессе обучения от нуля до единицы, в результате чего вначале на входы сети подаются практически одинаковые объекты, а с течением времени они все больше сходятся к исходным. Весовые коэффициенты устанавливаются на шаге инициализации равными величине

$$w_o = \frac{1}{\sqrt{n}}, \quad (22)$$

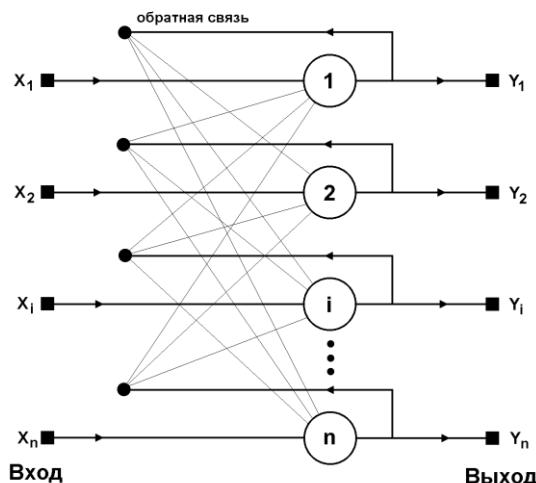
где  $n$  – размерность вектора весов для нейронов инициализируемого слоя.

На основе рассмотренного выше метода строятся нейронные сети особого типа – так называемые самоорганизующиеся структуры – self-organizing feature maps. Для них после выбора из слоя  $n$  нейрона  $j$  с минимальным расстоянием  $D_j$  (19) обучается по формуле (18) не только этот нейрон, но и его соседи, расположенные в окрестности  $R$ . Величина  $R$  на первых итерациях очень большая, так что обучаются все нейроны, но с течением времени она уменьшается до нуля. Таким образом, чем ближе конец обучения, тем точнее определяется группа нейронов, отвечающих каждому классу объекты.

### 3. Нейронные сети Хопфилда и Хэмминга.

Среди различных конфигураций искусственных нейронных сетей (НС) встречаются такие, при классификации которых по принципу обучения, строго говоря, не подходят ни обучение с учителем, ни обучение без учителя. В таких сетях весовые коэффициенты синапсов рассчитываются только однажды перед началом функционирования сети на основе информации об обрабатываемых данных, и все обучение сети сводится именно к этому расчету. С одной стороны, предъявление априорной информации можно расценивать, как помочь учителя, но с другой – сеть фактически просто запоминает объекты до того, как на ее вход поступают реальные данные, и не может изменять свое поведение, поэтому говорить о звене обратной связи с "миром" (учителем) не приходится. Из сетей с подобной логикой работы наиболее известны сеть Хопфилда и сеть Хэмминга, которые обычно используются для организации ассоциативной памяти. Далее речь пойдет именно о них.

Структурная схема сети Хопфилда приведена ниже на рисунке. Она состоит из единственного слоя нейронов, число которых является одновременно числом входов и выходов сети. Каждый нейрон связан синапсами со всеми остальными нейронами, а также имеет один входной синапс, через который осуществляется ввод сигнала. Выходные сигналы, как обычно, образуются на аксонах.



Структурная схема сети Хопфилда.

Задача, решаемая данной сетью в качестве ассоциативной памяти, как правило, формулируется следующим образом. Известен некоторый набор двоичных сигналов (изображений, звуковых оцифровок, прочих данных, описывающих некие объекты или характеристики процессов). Сеть должна уметь из произвольного неидеального сигнала, поданного на ее вход, выделить ("вспомнить" по частичной информации) соответствующий класс (если такой есть) или "дать заключение" о том, что входные данные не соответствуют ни одному из классов. В общем случае, любой сигнал может быть описан вектором  $X = \{x_i: i=0...n-1\}$ ,  $n$  – число нейронов в сети и размерность входных и выходных векторов. Каждый элемент  $x_i$  равен либо +1, либо -1. Обозначим вектор, описывающий  $k$ -ый класс, через  $X^k$ , а его компоненты, соответственно,  $x_i^k$ ,  $k=0...m-1$ ,  $m$  – число классов. Когда сеть распознает (или "вспомнит") какой-либо класс на основе предъявленных ей данных, ее выходы будут содержать именно его, то есть  $Y = X^k$ , где  $Y$  – вектор выходных значений сети:  $Y = \{y_i: i=0...n-1\}$ . В противном случае, выходной вектор не совпадет ни с одним образцовым.

Если, например, сигналы представляют собой некие изображения, то, отобразив в графическом виде данные с выхода сети, можно будет увидеть картинку, полностью совпадающую с одной из образцовых (в случае успеха) или же "вольную импровизацию" сети (в случае неудачи).

На стадии инициализации сети весовые коэффициенты синапсов устанавливаются следующим образом:

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, & i \neq j \\ 0, & i = j \end{cases} \quad (23)$$

Здесь  $i$  и  $j$  – индексы, соответственно, предсинаптического и постсинаптического нейронов;  $x_i^k$ ,  $x_j^k$  –  $i$ -ый и  $j$ -ый элементы вектора  $k$ -ого образца.

Алгоритм функционирования сети следующий ( $p$  – номер итерации):

- На входы сети подается неизвестный сигнал. Фактически его ввод осуществляется непосредственной установкой значений аксонов:

$$y_i(0) = x_i, \quad i = 0...n-1, \quad (24)$$

поэтому обозначение на схеме сети входных синапсов в явном виде носит чисто условный характер. Нуль в скобке справа от  $y_i$  означает нулевую итерацию в цикле работы сети.

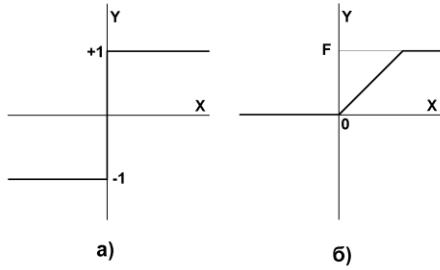
- Рассчитывается новое состояние нейронов

$$s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p), \quad j=0...n-1 \quad (25)$$

и новые значения аксонов

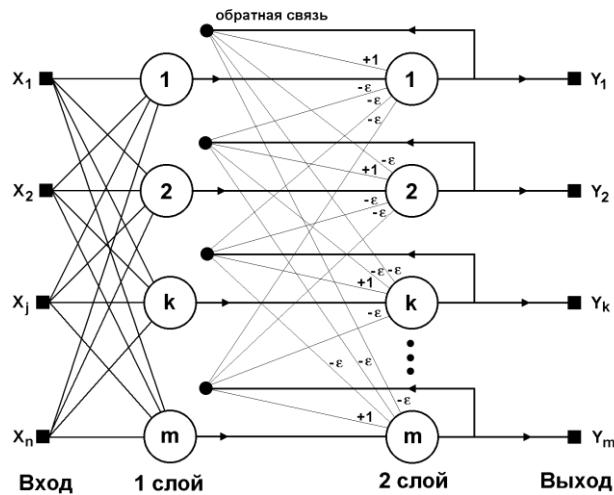
$$y_j(p+1) = f[s_j(p+1)] \quad (26)$$

где  $f$  – активационная функция в виде скачка, приведенная ниже (п.а)



3. Проверка, изменились ли выходные значения аксонов за последнюю итерацию. Если да – переход к пункту 2, иначе (если выходы стабилизировались) – конец. При этом выходной вектор представляет собой класс, наилучшим образом сочетающийся с входными данными.

Как говорилось выше, иногда сеть не может провести распознавание и выдает на выходе несуществующий класс. Это связано с проблемой ограниченности возможностей сети. Для сети Хопфилда число запоминаемых классов  $m$  не должно превышать величины, примерно равной  $0.15 \cdot n$ . Кроме того, если два класса А и Б сильно похожи, они, возможно, будут вызывать у сети перекрестные ассоциации, то есть предъявление на входы сети вектора А приведет к появлению на ее выходах вектора Б и наоборот.



Структурная схема сети Хэмминга.

Когда нет необходимости, чтобы сеть в явном виде выдавала класс, то есть достаточно, скажем, получать номер класса, ассоциативную память успешно реализует сеть Хэмминга. Данная сеть характеризуется, по сравнению с сетью Хопфилда, меньшими затратами на память и объемом вычислений, что становится очевидным из ее структуры. Сеть состоит из двух слоев. Первый и второй слои имеют по  $m$  нейронов, где  $m$  – число образцов. Нейроны первого слоя имеют по  $n$  синапсов, соединенных со входами сети (образующими фиктивный нулевой слой). Нейроны второго слоя связаны между собой ингибиторными (отрицательными обратными) синаптическими связями. Единственный синапс с положительной обратной связью для каждого нейрона соединен с его же аксоном.

Идея работы сети состоит в нахождении расстояния Хэмминга от тестируемого образа до всех образцов. Расстоянием Хэмминга называется число отличающихся битов в двух бинарных векторах. Сеть должна выбрать класс с минимальным расстоянием Хэмминга до неизвестного входного сигнала, в результате чего будет активирован только один выход сети, соответствующий этому образцу.

На стадии инициализации весовым коэффициентам первого слоя и порогу активационной функции присваиваются следующие значения:

$$w_{ik} = \frac{x_i^k}{2}, i=0...n-1, k=0...m-1 \quad (27)$$

$$Tk = n / 2, k = 0...m-1 \quad (28)$$

Здесь  $x_i^k$  – i-ый элемент k-ого образца. Весовые коэффициенты тормозящих синапсов во втором слое берут равными некоторой величине  $0 < \varepsilon < 1/m$ . Синапс нейрона, связанный с его же аксоном имеет вес +1.

Алгоритм функционирования сети Хэмминга следующий:

1. На входы сети подается неизвестный вектор  $\mathbf{X} = \{x_i; i=0...n-1\}$ , исходя из которого рассчитываются состояния нейронов первого слоя (верхний индекс в скобках указывает номер слоя):

$$a. \quad y_j^{(1)} = s_j^{(1)} = \sum_{i=0}^{n-1} w_{ij} x_i + T_j, j=0...m-1 \quad (29)$$

b. После этого полученными значениями инициализируются значения аксонов второго слоя:

$$c. \quad y_j^{(2)} = y_j^{(1)}, j = 0...m-1 \quad (30)$$

2. Вычислить новые состояния нейронов второго слоя:

$$a. \quad s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum_{k=0}^{m-1} y_k^{(2)}(p), k \neq j, j = 0...m-1 \quad (31)$$

b. и значения их аксонов:

$$c. \quad y_j^{(2)}(p+1) = f[s_j^{(2)}(p+1)], j = 0...m-1 \quad (32)$$

d. Активационная функция f имеет вид порога, причем величина F должна быть достаточно большой, чтобы любые возможные значения аргумента не приводили к насыщению.

3. Проверить, изменились ли выходы нейронов второго слоя за последнюю итерацию. Если да – перейди к шагу 2. Иначе – конец.

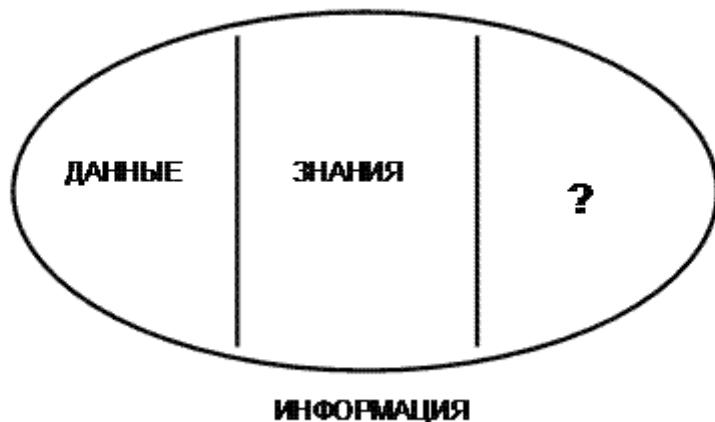
Из оценки алгоритма видно, что роль первого слоя весьма условна: воспользовавшись один раз на шаге 1 значениями его весовых коэффициентов, сеть больше не обращается к нему, поэтому первый слой может быть вообще исключен из сети (заменен матрицей весовых коэффициентов).

## Лекция 12

1. ИНФОРМАЦИЯ, ДАННЫЕ, ЗНАНИЯ.
2. ПРЕДСТАВЛЕНИЕ ОБ ИНФОРМАЦИИ.
  - Источник информации.
  - Модель информации.
  - Определения информационных понятий.
  - Измерения информации.
3. ФОРМАЛИЗАЦИЯ ПОНЯТИЯ ЗНАНИЙ.
  - Что такое знания.
  - Свойства знаний.
4. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - Знания и ИИ.

### Информация, данные, знания.

При решении любых задач используется информация. Собственно решение представляет собой процесс отображения информации. При отображении входная информация преобразуется в выходную. Интуитивно понятно, что информация представляет собой обобщение таких понятий как данные и знания. Схематично соотношение этих понятий можно представить в следующем виде:



Ясно, что между всеми этими понятиями существует зависимость. Характер этой зависимости вытекает из соотношения понятий. С одной стороны:

$$\text{Информация} \supseteq \text{Данные} \cup \text{Знания} \cup ?$$

А с другой:

$$\text{Свойства (Информации)} \subseteq \text{Свойства (Данные} \cup \text{Знания} \cup ?)$$

Рассмотрим эти понятия более подробно.

### Представление об информации.

- a). *Источник информации.* Находится во внешнем мире. Дело в том, что мир – неоднороден (однороден, например, чистый лист бумаги, если не смотреть на его края). Эта неоднородность порождает понятия предметов. Чтобы различать предметы, используется понятие

свойства. Свойства, в свою очередь, представляют собой характеристизацию предметов в другом пространстве. Реальные предметы разнообразны. Для их понимания необходимо их упрощение. Это упрощение заменяет реальные тела их моделями, имеющими ограниченное число свойств. Знания и представляют собой соответствующие модели реального мира.

б). *Модель информации*. Простейшей моделью является объект. Единственным свойством объекта является его существование, а для описания объекта требуется указание его свойств.

**Примеры объектов:** точки, линии, поверхности, геометрические тела, пространства. Главным свойством объекта-точки является отсутствие у неё протяжённости. Свойством неточечных объектов является число размерностей, которыми эти объекты обладают. Моделями границ являются линии (одна размерность) и поверхности (две размерности).

Свойства, касающиеся сразу нескольких объектов, выражаются через понятия связи (отношения). Наличие связей между объектами превращает эти объекты в части некоторой системы, а отсутствие связей – в совокупность объектов.

#### *Комментарий по поводу отсутствия связи.*

Существуют взаимные воздействия предметов, и через эти воздействия проявляется суть предметов. При воздействии на людей возникают понятия, необходимые для описания предметов через свойства соответствующих объектов. Для этого же могут использоваться различные технические устройства. Взаимодействие тел с передачей сведений о структуре одного тела другому является процессом отражения.

в). *Определения информационных понятий*.

**Информация – это:**

следствие процесса отражения.

это след от воздействия одного тела на другое.

это не сам предмет (тело), а то, что его обозначает.

Информация не может существовать без своего носителя (некоторого предмета). Информационный объект – это объект, с внедрённой в него информацией (носитель информации). Информация имеет размерность. Эта размерность представляет собой название источника информации или его элементов. Сообщение – динамическая форма информации, реализуемая в процессе взаимодействия предметов. Данные – статическая форма информации, реализуемая в информационном объекте.

**Информация может быть:**

явной или неявной.

полной, неполной или избыточной.

текстовой или графической.

достоверной или ложной (или находится в промежуточном состоянии).

Процесс порождения информации имеет своей противоположностью процесс интерпретации (понимания информации) данных. Этот процесс заключается в отождествлении поступившей информации с имеющимися у предмета знаниями или трактовке поступившей информации как новые знания. *Знания - это информация, прошедшая процесс интерпретации и рассматриваемая как структура источника информации*. Предмет (например - человек) может иметь априорные (до получения информации) знания о структуре источника информации. Если поступившая информация изменяет априорные знания, то такая информация называется новой информацией. Если же поступившая информация не изменяет априорных знаний, то она всё равно является информацией, но известной. Информация может, как преобразовываться, так и сама преобразовывать другую информацию. Примером могут служить команды компьютера (преобразующая информация) и обрабатываемые данные (преобразуемая информация).

г). *Измерения информации*. Известны две базовых единицы количества информации, называемых одинаково: **бит**. Одна единица введена в теории связи, другая относится к компьютерной технике. Использование одной из указанных единиц зависит от того, как описывается структура источника информации. Если структура источника информации описывается статистически, то и бит вероятностный (теория связи). Если же структура источника информации описывается детерминировано, то и бит детерминированный (компьютерное представление).

Простейший способ измерения информации заключается в подсчете числа бит. Более сложный способ является динамическим. Смысл его заключается в оценке изменения информации.

**Замечание.** Информация - это лишь такие сведения, которые уменьшают существовавшую до их получения неопределенность.

Пусть исходная неопределенность состоит в выборе из некоторого числа  $N$  равновероятных гипотез, и сообщение устраниет эту неопределенность, указывая на одну из них. Количество информации  $I$  будет в этом случае функцией от  $N$ . При некоторых дополнительных предположениях относительно структуры сообщения, показано, что

$$I(N) = \log_2 N \text{ (формула Хартли)}$$

При равновероятных гипотезах вероятность  $p$  каждой из них равна  $1/N$ . Тогда Количество информации через вероятности можно выразить так

$$I(p) = -\log_2 p$$

В более сложном случае, когда имеется  $N$  сигналов  $k$  различных типов, оценивается общий объем информации. Для этого используется формула Шеннона

$$I = -\sum_{i=1}^k p_i \log_2 p_i$$

Эта формула характеризует удельную информацию, не зависящую от числа  $N \rightarrow \infty$ .

**Анкета:** наибольшее количество информации извлекается в случае, когда в анкете ответы да и нет встречаются одинаково часто ( $p_1 = p_2 = 1/2$ ).

Существуют и другие способы измерения информации. Например, в информатике используется способ, предложенный Колмогоровым. Суть его в том, что для получения информации используется алгоритм. Минимальная длина программы, необходимой для реализации алгоритма, также может быть характеристикой информации.

## **Формализация понятия знаний.**

В ЭВМ знания так же, как и **данные**, отображаются в знаковой форме - в виде формул, текста, файлов, информационных массивов и т.п. Поэтому можно сказать, что знания - это особым образом организованные данные. Но это слишком узкое понимание. А между тем, в системах ИИ знания являются основным объектом формирования, обработки и исследования. **База знаний**, наряду с базой данных, - необходимая составляющая программ ИИ. Раздел теории ИИ, связанный с построением KBS – называется *инженерией знаний*.

### a). **что такое знания.**

В настоящее время наиболее распространенными и логически завершенными можно считать следующие две концепции понятия знаний:

- первая имеет отношение к содержательным аспектам понятия и основывается на постулате: *знания есть интерпретированные данные*;
- вторая базируется на выборе определенного способа представления исходной информации. Наиболее распространеными среди них являются фреймы, семантические сети и "аксиоматический" способ (с использованием языков исчисления высказываний, предикатов, модальной логики и т.п.). Основной постулат данной концепции можно сформулировать следующим образом: *знанием является все то, что можно получить на основе перечисленных представлений с помощью подходящих механизмов вывода*.

Очевидны недостатки этих концепций. Первая является неконструктивной, а в рамках второй невозможно поставить вопрос о выборе способа представления при решении практических задач. Вывод: *при построении любой другой концепции знаний необходимо постараться избавиться от указанных недостатков. При этом новая концепция должна содержать упомянутые выше*.

### **Замечание:**

Рассмотрим соотношение понятия знаний с уже известными и семантически с ними связанными понятиями: **информация** и **данные**.

Частью такого формального объекта является множество элементов, которые обладают общностью структуры и способа построения в следующем смысле:

$$\text{объект}: \Pi_1 \times \dots \times \Pi_n \rightarrow D_1 \times \dots \times D_n$$

где  $\Pi_i$  – признаки,  $D_i$  – множество возможных значений признака ( $i=1, \dots, n$ ). Другой частью является некоторая характеристика связи или отношения между объектами. На формальном уровне отношение представляет собой подмножество декартовых степеней множества  $\{\text{объект}\}^m$ . Таким образом, любой формальный объект можно отождествить с парой  $\langle\text{объект, отношение}\rangle$ , или в общем случае – с декартовым произведением  $\{\langle\text{объект, отношение}\rangle\}^m$ . Формальный смысл также имеет представление в терминах  $\{\langle\text{объект, связь}\rangle\}^m$ . В первом случае для описания объектов используются язык алгебры (точнее - алгебраических систем [6]), а во втором – теории графов.

Теперь можно конкретизировать представление функций  $F_i$  ( $i=1,2,3$ ). Очевидно, что они должны реализовывать отображение:

$$F_i: \{\langle\text{объект, связь}\rangle\}^m \rightarrow \{\langle\text{объект, связь}\rangle\}^k \quad (m, k \in N)$$

Таким образом, свойства класса функций  $F_i$  полностью определяются характером связей.

Так при описании данных используются связи **порядка**, которые являются следствием упорядочения и нормализации множества объектов. Это сразу определяет основное свойство класса  $F_1$ : допустимыми являются только те функции, которые сохраняют связи порядка или являются их непосредственным следствием. При этом не так существенно, какие конкретно объекты используются при описании данных.

Чтобы разобраться с характером связей на уровне знаний, сделаем небольшое отступление. Формальный объект является элементом множества, поэтому для него справедливы манипуляции характерные для обычных множеств. Таковыми являются:

- нумерация (индексирование) и упорядочение;
- образование подмножества по какому-либо свойству;
- определение свойств конкретных элементов множества в смысле порядка или включения в подмножество.

Все остальные манипуляции либо являются производными, либо связаны со специализацией языка. В частности, при описании данных и класса функций  $F_1$  речь фактически идет о манипуляциях, связанных с нумерацией и упорядочением. Поэтому естественно ожидать, что при описании знаний будут использованы и другие манипуляции.

В точном соответствии с данной логикой, для характеризации класса  $F_3$  в области значений определяются связи, которые можно назвать **следованием**, **наследованием** или **объединением по свойству**. Выбор конкретного термина не принципиален, но, по крайней мере, два первых встречаются в литературе по ИИ. В случае же с классом функций  $F_2$  соответствующие связи можно назвать **выводимостью**. Объекты в обоих случаях уже не произвольны. Их описание должно быть связано каким-то образом со свойствами.

Обсудим соотношение введенных понятий со второй концепцией знаний. Можно ли основываясь на введенных конструкциях получить представление знаний в виде фреймов, семантических сетей и т.д.? Положительный ответ не вызывает сомнений. Конструкция формальных объектов фактически является минимальным обобщением объектов в представлении знаний с помощью фреймов и семантических сетей. Согласуется она и с объектами в "аксиоматическом" представлении. Причина такой общности и согласованности понятна, т.к. во всех представлениях речь идет о множествах, элементы которых могут быть наделены некоторыми свойствами. Это фактически и положено в основу понятия формального объекта. Конечно, остается еще достаточно много деталей, обсудить которые в рамках данной работы просто не представляется возможным. Но они касаются в основном характеризации классов функций  $F_2$  и  $F_3$ .

Информация, с которой имеют дело ЭВМ, разделяется на **процедурную** и **декларативную**. Процедурная информация овеществлена в **программах**, которые выполняются в процессе решения задач, декларативная информация - в **данных**, с которыми эти программы работают. Стандартной формой представления информации в ЭВМ является **машинное слово**, со-

стоящее из определенного для данного типа ЭВМ числа двоичных разрядов - **битов**. Машинное слово для представления данных и машинное слово для представления команд, образующих программу, могут иметь одинаковое или разное число разрядов. В последнее время для представления данных и команд используются одинаковые по числу разрядов машинные слова. Однако в ряде случаев машинные слова разбиваются на группы по восемь двоичных разрядов, которые называются **байтами**.

Однаковое число разрядов в машинных словах для команд и данных позволяет рассматривать их в ЭВМ в качестве одинаковых информационных единиц и выполнять операции над командами, как над данными. Содержимое памяти образует **информационную базу**.

В большинстве существующих ЭВМ возможно извлечение информации из любого подмножества разрядов машинного слова вплоть до одного бита. Во многих ЭВМ можно соединять два или более машинного слова в слово с большей длиной. Однако машинное слово является основной характеристикой информационной базы, т.к. его длина такова, что каждое машинное слово хранится в одной стандартной ячейке памяти, снабженной индивидуальным именем - адресом ячейки. По этому имени происходит извлечение информационных единиц из памяти ЭВМ и записи их в нее.

Параллельно с развитием структуры ЭВМ происходило развитие информационных структур для представления данных. Появились способы описания данных в виде векторов и матриц, возникли списочные структуры, иерархические структуры. В настоящее время в языках программирования высокого уровня используются **абстрактные типы данных**, структура которых задается программистом. Появление **баз данных** (БД) знаменовало собой еще один шаг на пути организации работы с декларативной информацией. В базах данных могут одновременно храниться большие объемы информации, а специальные средства, образующие **систему управления базами данных** (СУБД), позволяют эффективно манипулировать с данными, при необходимости извлекать их из базы данных и записывать их в нужном порядке в базу.

По мере развития исследований в области ИС возникла концепция знаний, которые объединили в себе многие черты процедурной и декларативной информации.

В ЭВМ **знания** так же, как и **данные**, отображаются в знаковой форме - в виде формул, текста, файлов, информационных массивов и т.п. Поэтому можно сказать, что знания - это особым образом организованные данные. Но это было бы слишком узкое понимание. А между тем, в системах ИИ знания являются основным объектом формирования, обработки и исследования. **База знаний**, наравне с базой данных, - необходимая составляющая программного комплекса ИИ. Машины, реализующие алгоритмы ИИ, называются **машинами, основанными на знаниях**, а подраздел теории ИИ, связанный с построением экспертизных систем, - **инженерией знаний**.

#### б) **Свойства знаний.**

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

Таблица 1

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

Если, например, в память ЭВМ нужно было записать сведения о сотрудниках учреждения, представленные в табл. 1, то без внутренней интерпретации в память ЭВМ была бы занесена совокупность из четырех машинных слов, соответствующих строкам этой таблицы. При этом информация о том, какими группами двоичных разрядов в этих машинных словах закодированы сведения о специалистах, у системы отсутствуют. Они известны лишь программисту, который использует данные табл. 1.1 для решения возникающих у него задач. Система не в состоянии ответить на вопросы типа "Что тебе известно о Петрове?" или "Есть ли среди специалистов сантехник?".

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

2. *Структурированность*. Информационные единицы должны обладать гибкой структурой. Для них должен выполняться "принцип матрешки", т.е. рекурсивная вложимость одних информационных единиц в другие. Каждая информационная единица может быть включена в состав любой другой, и из каждой информационной единицы можно выделить некоторые составляющие ее информационные единицы. Другими словами, должна существовать возможность произвольного установления между отдельными информационными единицами отношений типа "часть - целое", "род - вид" или "элемент - класс".
3. *Связность*. В информационной базе между информационными единицами должна быть предусмотрена возможность установления связей различного типа. Прежде всего эти связи могут характеризовать отношения между информационными единицами. Семантика отношений может носить декларативный или процедурный характер. Например, две или более информационные единицы могут быть связаны отношением "одновременно", две информационные единицы - отношением "причина - следствие" или отношением "быть рядом". Приведенные отношения характеризуют декларативные знания. Если между двумя информационными единицами установлено отношение "аргумент - функция", то оно характеризует процедурное знание, связанное с вычислением определенных функций. Далее будем различать *отношения структуризации, функциональные отношения, каузальные отношения и семантические отношения*. С помощью первых задаются иерархии информационных единиц, вторые несут процедурную информацию, позволяющую находить (вычислять) одни информационные единицы через другие, третьи задают причинно - следственные связи, четвертые соответствуют всем остальным отношениям.

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

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

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

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

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

## Лекция 13

1. ЗНАНИЯ И РЕШЕНИЕ ЗАДАЧИ.
2. СМЫСЛ ПРОБЛЕМЫ ПРЕДСТАВЛЕНИЯ.
3. МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ.
4. ОТ ЗНАНИЙ К БАЗЕ ЗНАНИЙ.
5. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - Проблема представления знаний в ИИ, математике, информатике.

### Знания и решение задачи

Решение задачи, которое является центральной проблемой ИИ, разделяется на две части: качественную и количественную. Наличие **качественной** составляющей приводит к тому, что **формальное решение задач становится невозможным**. Следовательно, первостепенное значение приобретают методика решения задачи, опыт математика, программиста и т.п. **Количественная** составляющая – это алгоритмы, способы численного моделирования и т.п. Здесь также **алгоритмы неизвестны или по каким-то причинам не может быть выполнено их обоснование** (не может быть показана разрешимость задачи, хотя бы обусловленная).

В случае, если в задаче присутствуют перечисленные выше признаки качественной или количественной составляющих, задачу принято относить к области интересов ИИ.

**Пример.** 1.Задача коммивояжера. 2.Задача дистанционного зондирования.

**Замечание.** Дедуктивные схемы описания и индуктивные (эвристические) методы решения задачи. Знания содержатся и в схемах, и в методах. Эти знания различны.

### Смысл проблемы представления

Проблема представления реальных или абстрактных объектов является центральной для ИИ. Дать представление чего-либо означает, прежде всего, умение выделить из общей массы объект, выявить его индивидуальность. Отсюда следует, что его свойства должны быть переведены в форму, удобную для манипулирования объектом. Представление эквивалентно пониманию.

Представление – это действие, делающее некоторое понятие воспринимаемым. Это восприятие может осуществляться посредством фигуры, записи, языка или формализма.

Представление знаний – формализация знаний посредством фигур, записей или языков. Интересны представления, воспринимаемые (распознаваемые) компьютерами. Для этого могут использоваться языки или формализмы для представления знаний. Эти средства позволяют перевести содержательное представление в пригодное для ввода и обработки в компьютере. Результатом такой формализации в общем случае может быть набор инструкций, составляющих часть ЯП.

Представлению знаний присущ пассивный аспект: книга, таблица. В ИИ важен активный аспект: **познание** должно стать операцией, позволяющей извлекать представленные знания и строить рассуждения на их основе.

### Модели представления знаний

Для представления знания в математической логике пользуются логическими формализмами - главным образом *исчислением предикатов*, которое имеет ясную формальную семантику и операционную поддержку в том смысле, что для него разработаны механизмы вывода. Поэтому исчисление предикатов было первым логическим языком, который применили для формального описания предметных областей, связанных с решением прикладных задач.

Описания предметных областей, выполненные в логических языках, называются (формальными) логическими моделями.

Кроме логических, существуют и другие модели представления знаний: семантические и нейронные сети, фреймы.

**Замечание.** В общем случае, для представления знаний используется математическая форма записи. Эта форма базируется на следующих элементах:

- символах, представляющих собой объекты;
- символах, представляющих собой операторы.

**Пример.** Выражение  $x+y-1$ .  $x$ ,  $y$  – объекты,  $+$  и  $-$  – операторы. Операторами могут связываться некоторое число объектов. Это число называется арностью оператора.  $1$  – объект (константа), или выделенный элемент языка.

Использование символов может быть недопустимым. В противном случае выражение называется правильно построенным.

## От знаний к Базе знаний

В любой предметной области (ПО) необходимым условием применения компьютеров является переход на формальный уровень. Для этого требуется построить некое концептуальное представление ПО в терминах **<объект, отношение>**. Здесь под объектом понимается множество элементов, обладающих общностью структуры и способа построения в следующем смысле

$$\text{объект}: \Pi_1 \times \dots \times \Pi_n \rightarrow D_1 \times \dots \times D_n$$

где  $\Pi_i$  – признаки,  $D_i$  – множество возможных значений признака ( $i=1, \dots, n$ ). Отношение – подмножество декартовых степеней множества **{объект}**<sup>l</sup>, которое может быть построено для каждого объекта.

Приведенная конструкция является теоретико-множественной и она хороша только как абстракция. На практике же используются ее различные эквиваленты. Чаще всего в виде **<объект, связь>**. В результате получается конструкция, которая по смыслу наиболее близка к **семантической сети**, и поэтому можно говорить, что при формализации основой является семантика предметной области.

Достаточным условием применения компьютерной технологии является возможность постановки задач, т.е. определение цели существования объектов. При этом как сама цель, так и способы ее достижения существенно зависят от смысла семантики и свойств элементов **<объект, связь>**. Во всех случаях цель обязательно связана с некоторым подмножеством **<объект, связь>**<sup>k</sup> и достигается в результате работы алгоритма

$$A: \{\text{объект, связь}\}^m \rightarrow \{\text{объект, связь}\}^k$$

Вход и выход такого алгоритма представляет собой декартовы степени элементов **<объект, связь>**, определенных при формализации ПО.

Все сказанное выше имеет отношение к информации. Компьютерная технология представляет собой набор средств, обеспечивающих функционирование информации определенного вида. Ясно, что технология должна быть в максимальной степени инвариантна относительно содержания информации. Поэтому определить ее можно через свойства введенных выше элементов концептуального представления. Обсудим с этой позиции, что представляют собой технологии **<Сд+ЯП>** и **<БД+СУБД>**. Результаты сравнительного анализа сведем в таблицу.

	<b>&lt;Сд+ЯП&gt;</b>	<b>&lt;БД+СУБД&gt;</b>
Объекты	наборы векторов, матриц и все-возможные структуры над ними (списки, очереди и т.п.)	наборы кортежей

<b>Связи между объектами</b>	определяются только связи <i>порядка</i>	кроме связей порядка определяются также связи <i>типа</i> (они называются - 1:1, 1:N, M:N)
<b>Алгоритмы получения новых экземпляров &lt;объектов, связей&gt;</b>	встроенные алгоритмы упорядочения	кроме алгоритмов упорядочения используются также алгоритмы, которые базируются на связи типа
<b>Средство реализации</b>	ЯП	СУБД

Необходимо обратить внимание на следующие особенности рассмотренных технологий:

- свойства объекта определяются косвенно, через свойства связей. И в этом смысле для технологии <БД+СУБД> понятие объекта является более широким, почти соответствующим понятию класс в ООП [2];
- алгоритмы являются следствием соглашений о свойствах объектов и связей. Эти алгоритмы являются ядром технологии, ее обязательной частью. Ясно, что средствами ЯП и СУБД могут быть реализованы способы получения объектов, которые не имеют прямого отношения к соответствующей связи. Для этого СУБД, помимо стандартной части (например, SQL), содержит обычно встроенный ЯП.

Замечание. Алгоритмы, которые поддерживаются в рассматриваемых технологиях, можно приблизительно охарактеризовать в терминах соотношения между линейной и реляционной алгебрами (как на уровне данных, так и на уровне операций).

**Вывод: между технологиями существует зависимость вида**

$$\langle \text{СД+ЯП} \rangle \subset \langle \text{БД+СУБД} \rangle$$

**Расширение возможностей последней происходит исключительно за счет свойств связи. Все остальное является следствием.**

Исходя из проведенных выше рассуждений, введем понятие **данных**.

Определение. Под **данными** можно понимать часть информации, которая на концептуальном уровне описывается структурой  $\{\langle \text{объект}, \text{связь} \rangle\}^t$  при связи не сложнее, чем связь типа и алгоритмах, обусловленных именно такими связями ( $t$  – некоторое произвольное, может быть и бесконечное число).

Замечание. Алгоритмы в определении данных нельзя исключить, т.к. при этом можно легко прийти к патологии абстракции, которая будет только содержательной. Например, в ИИ существует понимание знаний как интерпретированных данных, которое в методологическом и технологическом смыслах является не конструктивным.

Знания также имеют алгоритмическое происхождение, т.е. они являются основой и результатом преобразования одной части структуры  $\{\langle \text{объект}, \text{связь} \rangle\}^k$  в другую. Схематично это можно представить следующим образом:



где  $A_1$  – алгоритмы, о которых идет речь в определении данных, а  $A_2$  – алгоритмы получения знаний.

Специфика знаний, как на уровне структуры  $\{<\text{объект}, \text{связь}>\}^k$  так и на уровне алгоритмов, может быть определена с учетом сделанного выше вывода. Как минимум для этого необходимо к связям, фигурирующим в определении данных, что-то добавить. Если пойти по пути максимального обобщения, то такое добавление должно заключаться в *преобразовании связи в связь со всеми признаками объектов*. Это согласуется с идеологией ООП. Но главное заключается в том, что в задачах обработки знаний связь сама может являться целью. И, наконец, только на этом пути следует ожидать, что технология работы со знаниями будет содержать в себе элементы предшествующих технологий.

Нетрудно определить и характер такой связи-объекта. Для этого достаточно вспомнить, что на самом высоком уровне абстракции любой теоретико-множественной структуры находится логика. Т.е. такая связь должна специфицировать логические зависимости между частями структуры  $\{<\text{объект}, \text{связь}>\}^k$ . Интересно, что при этом и математический аппарат, предназначенный для анализа структуры и построения алгоритмов, также будет обобщением аппарата реляционной алгебры. Это аппарат алгебраических систем, который содержит язык описания и алгоритмизации логических элементов.

Теперь можно ввести понятие знаний.

**Определение.** Под знаниями можно понимать часть информации, которая на концептуальном уровне описывается структурой  $\{<\text{объект}, \text{связь}>\}^t$  при связи, которая сама является объектом и алгоритмах, обусловленных именно такими связями. Связь должна устанавливать логические зависимости в структуре  $\{<\text{объект}, \text{связь}>\}^t$ , а в качестве объекта – являться целью решения задачи. Достижение цели (решение задач) должно поддерживаться с помощью алгоритмов логического вывода.

Чтобы знания стали объектом технологии, остается определить понятие базы знаний. Под БЗ будем понимать структуру  $\{<\text{объект}, \text{связь}>\}^k$ , в которой связь может быть любого из упомянутых выше видов (порядка, типа и логическая), в любой комбинации и с точно определенными целями решения задач (алгоритмами). СУБЗ, в свою очередь, представляет собой среду поддержки такой структуры с возможностями интерпретации.

## Лекция 14

1. ЯЗЫКИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ.
2. ЛОГИЧЕСКИЕ МОДЕЛИ.
3. СЕТЕВЫЕ МОДЕЛИ.
4. ПРОДУКЦИОННЫЕ МОДЕЛИ.
5. ФРЕЙМОВЫЕ МОДЕЛИ.
6. ФОРМАЛЬНЫЕ МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ.
7. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - Какая модель лучше?

### Языки представления знаний

Постановка и решение любой задачи всегда связаны с ее "погружением" в подходящую предметную область. Все предметы и события, которые составляют основу общего понимания необходимой для решения задачи информации, называются *предметной областью*. Мысленно предметная область представляется состоящей из реальных или абстрактных объектов, называемых *сущностями*.

Сущности предметной области находятся в определенных *отношениях* друг к другу (ассоциациях), которые также можно рассматривать как сущности и включать в предметную область. Между сущностями наблюдаются различные отношения подобия. Совокупность подобных сущностей составляет *класс сущностей*, являющийся новой сущностью предметной области.

Отношения между сущностями выражаются с помощью суждений. Суждение-это мысленно возможная ситуация, которая может иметь место для предъявляемых сущностей или не иметь места. В языке (формальном или естественном) суждениям отвечают *предложения*. Суждения и предложения также можно рассматривать как сущности и включать в предметную область.

Языки, предназначенные для описания предметных областей, называются *языками представления знаний*. Универсальным языком представления знаний является естественный язык. Однако использование ЕЯ в системах машинного представления знаний затруднительно, в силу присущих ЕЯ нерегулярностям и многозначности смысла фраз. Но главное препятствие заключается в отсутствии формальной семантики естественного языка, которая имела бы достаточно эффективную поддержку с точки зрения манипуляций.

Рассмотрим основные языки представления знаний.

### Логические модели

В основе моделей такого типа лежит *формальная система*, задаваемая четверкой вида:

$$M = \langle T, P, A, B \rangle.$$

Множество  $T$  есть *множество базовых элементов* различной природы, например слов из некоторого ограниченного словаря, деталей конструктора, входящих в состав некоторого набора и т.п. Важно, что для множества  $T$  существует некоторый способ определения принадлежности произвольного элемента к этому множеству. Процедура такой проверки может быть любой, но за конечное число шагов она должна давать положительный или отрицательный ответ на вопрос, является ли  $x$  элементом множества  $T$ . Обозначим эту процедуру  $\Pi(T)$ .

Множество  $P$  есть *множество синтаксических правил*. С их помощью из элементов  $T$  образуют *синтаксически правильные совокупности*. Например, из слов ограниченного словаря строятся синтаксически правильные фразы. Подразумевается существование процедуры  $\Pi(P)$ , с помощью которой за конечное число шагов можно получить ответ на вопрос, является ли некоторая совокупность синтаксически правильной.

Во множестве синтаксически правильных совокупностей выделяется некоторое подмножество  $A$ . Элементы  $A$  называются *аксиомами*. Как и для других составляющих формальной

системы, должна существовать процедура  $\Pi(A)$ , с помощью которой для любой синтаксически правильной совокупности можно получить ответ на вопрос о принадлежности ее к множеству  $A$ .

Множество  $B$  есть множество *правил вывода*. Применяя их к элементам  $A$ , можно получать новые синтаксически правильные совокупности, к которым снова можно применять правила из  $B$ . Так формируется *множество выводимых* в данной формальной системе *совокупностей*. Если имеется процедура  $\Pi(B)$ , с помощью которой можно определить для любой синтаксически правильной совокупности, является ли она выводимой, то соответствующая формальная система называется *разрешимой*. Это показывает, что именно правило вывода является наиболее сложной составляющей формальной системы.

Для знаний, входящих в базу знаний, можно считать, что множество  $A$  образуют все информационные единицы, которые введены в базу знаний извне, а с помощью правил вывода из них выводятся новые *производные знания*. Другими словами, формальная система представляет собой генератор порождения новых знаний, образующих множество *выводимых* в данной системе знаний. Это свойство позволяет хранить в базе лишь те знания, которые образуют множество  $A$ , а все остальные знания получать из них по правилам вывода.

## Сетевые модели

В основе моделей этого типа лежит конструкция, которая называется *семантической сетью*. Введем вначале основные понятия *семантических сетей*.

Под сущностью будем понимать объект произвольной природы. Этот объект может существовать в реальном мире. В этом случае он будет называться  $\Pi$  - *сущностью*. В базе знаний ему соответствует некоторое описание, полнота которого определяется той информацией, которую имеет о  $\Pi$  - сущности ИС. Такое представление в базе знаний называется  $M$  - *сущностью*. Отметим, что могут существовать  $M$ -сущности, для которых в окружающем ИС мире нет соответствующих  $\Pi$  - сущностей. Такие  $M$  - сущности представляют собой абстрактные объекты, полученные в результате операций типа обобщения внутри базы знаний.

Разделение на два типа сущностей позволяет использовать в сетевых моделях идеи, впервые сформулированные в теории *семиотических моделей* и основанном на них *ситуационном управлении*. Под *семиотическими моделями проблемных областей* будет пониматься комплекс процедур, позволяющих отображать в базе знаний  $\Pi$  - сущности и их связи, фиксируемые в проблемной области, в совокупность связанных между собой  $M$ -сущностей. Способ интерпретации взаимосвязанных  $\Pi$  – сущностей называется *денотативной семантикой*, а способ интерпретации взаимосвязанных  $M$ -сущностей - *коннотативной семантикой*.

$\Pi$  - сущность по отношению к соответствующей ей в базе знаний  $M$ -сущности называется *денотатом* или *референтом* этой  $M$ -сущности, а  $M$ -сущность по отношению к исходной  $\Pi$  - сущности - ее *десигнатом, именем, меткой, идентификатором* и т. п. Десигнат - это простейший элемент в сетевой модели. Он входит в класс терминальных объектов сетевой модели. *Терминальным объектом* называется  $M$ -сущность, которая не может быть разложена на более простые сущности. Остальные  $M$ -сущности называются *производными объектами* или *производными  $M$ -сущностями*.

Перечень терминальных объектов, которые могут образовывать классы или типы, задается при проектировании ИС. Ими могут быть целые вещественные числа, идентификаторы, строки, списки и т. п. Семантика терминальных объектов определяется набором допустимых процедур, оперирующих с ними. Например: арифметические действия над числами, сравнение между собой строк или идентификаторов, операции ввода-вывода, включающие необходимые трансформации представлений, и т. д.

Сетевые модели формально можно представить в виде

$$H = \langle I, C_1, C_2, \dots, C_n, \Gamma \rangle.$$

Здесь  $I$  есть множество информационных единиц;  $C_1, C_2, \dots, C_n$  - множество типов связей между информационными единицами. Отображение  $\Gamma$  задает между информационными единицами, входящими в  $I$ , связи из заданного набора типов связей.

В зависимости от типов связей, используемых в модели, различают *классифицирующие сети*, *функциональные сети* и *сценарии*. В классифицирующих сетях используются отношения структуризации. Такие сети позволяют в базах знаний вводить разные отношения между информационными единицами. Функциональные сети характеризуются наличием функциональных отношений. Их часто называют *вычислительными моделями*, т.к. они позволяют описывать процедуры "вычислений" одних информационных единиц через другие. В сценариях используются каузальные отношения, а также отношения типов "средство - результат", "орудие - действие" и т.п. Если в сетевой модели допускаются связи различного типа, то ее обычно называют семантической сетью.

## Продукционные модели

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

В общем случае под продукцией понимается выражение следующего вида:

$$(i); Q;P;A \Rightarrow B;N.$$

Здесь  $i$  - имя продукции, с помощью которого данная продукция выделяется из всего множества продукции. В качестве имени может выступать некоторая лексема, отражающая суть данной продукции (например, "покупка книги"), или порядковый номер продукции в их множестве, хранящемся в памяти системы.

Элемент  $Q$  характеризует сферу применения продукции. Такие сферы легко выделяются в знаниях ЕИ. Разделение знаний на отдельные сферы позволяет экономить время на поиск нужных знаний. Такое же разделение на сферы в базе знаний ИИ целесообразно и при использовании для представления знаний продукционных моделей.

Основным элементом продукции является ее ядро:  $A \Rightarrow B$ . Интерпретация ядра продукции может быть различной и зависит от того, что стоит слева и справа от знака импликации  $\Rightarrow$ . Обычное прочтение ядра продукции выглядит так: ЕСЛИ  $A$ , ТО  $B$ , более сложные конструкции ядра допускают в правой части альтернативный выбор. Например, ЕСЛИ  $A$ , ТО  $B_1$ ; ИНАЧЕ  $B_2$ . Импликация может истолковываться в обычном логическом смысле как знак логического следования  $B$  из истинного  $A$  (если  $A$  не является истинным выражением, то о  $B$  ничего сказать нельзя). Возможны и другие интерпретации ядра продукции. Например,  $A$  описывает некоторое условие, необходимое для того, чтобы можно было совершить действие  $B$ .

Элемент  $P$  есть условие применимости ядра продукции. Обычно  $P$  представляет собой логическое выражение (как правило, предикат). Когда  $P$  принимает значение "истина", ядро продукции активизируется. Если  $P$  "ложно", то ядро продукции не может быть использовано. Элемент  $N$  описывает постусловия продукции. Они актуализируются только в том случае, если ядро продукции реализовалось. Постусловия продукции описывают действия и процедуры, которые необходимо выполнить после реализации  $B$ . Выполнение  $N$  может происходить сразу после реализации ядра продукции.

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

продукционном. В этом случае говорят о работе продукционной системы над семантической сетью.

## Фреймовые модели

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

(Имя фрейма:

Имя слота 1(значение слота 1)

Имя слота 2(значение слота 2)

.....

Имя слота K (значение слота K)).

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

При конкретизации фрейма ему и слотам присваиваются конкретные имена, и происходит заполнение слотов. Таким образом, из протофреймов получаются *фреймы - экземпляры*. Переход от исходного протофрейма к фрейму - экземпляру может быть многошаговым, за счет постепенного уточнения значений слотов.

Например, структура табл.

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

записанная в виде протофрейма, имеет вид

(Список работников:

Фамилия (значение слота 1);

Год рождения (значение слота 2);

Специальность (значение слота 3);

Стаж (значение слота 4)).

Если в качестве значений слотов использовать данные табл. 1.1, то получится фрейм - экземпляр:

(Список работников:

Фамилия (Попов - Сидоров - Иванов - Петров);

Год рождения (1965 - 1946 - 1925 - 1937);

Специальность (слесарь - токарь - токарь - сантехник);

Стаж (5 - 20 - 30 - 25)).

Связи между фреймами задаются значениями специального слота с именем "связь". В ИИ считается, что нет необходимости специально выделять фреймовые модели в представлении знаний, т.к. в них объединены все основные особенности моделей остальных типов.

## Формальные модели представления знаний

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

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

Если перечисленные условия выполняются, то говорят, что система удовлетворяет требованиям *формальной теории*. Ее так и называют *формальной системой* (ФС). Система, построенная на основе формальной теории, называется также *аксиоматической системой*.

Формальная теория должна, таким образом, удовлетворять следующему определению:

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

- наличием алфавита (словаря),  $A$ ,
- множеством синтаксических правил,  $V$ ,
- множеством аксиом, лежащих в основе теории,  $W$ ,
- множеством правил вывода,  $R$ .

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

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

## Лекция 15

1. ЗАДАЧИ ИИ И ПРОГРАММЫ.
2. КОНЦЕПЦИЯ СИСТЕМЫ, ОСНОВАННОЙ НА ЗНАНИЯХ.
3. СТРУКТУРА КБС.
4. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - Структура программных средств для решения задач ИИ.
  - Функциональные, методологические и технологические требования к ИИС

### Задачи ИИ и программы

Объективная необходимость расширения сферы автоматизации порождает ряд проблем. Важнейшая из них связана с разработкой технологий, позволяющих приближать конечного пользователя к ЭВМ, как на этапе разработки, так и на этапе эксплуатации и сопровождения прикладного программного обеспечения (ППО). Дело в том, что в рамках традиционной технологии процессы разработки (рис. 1) и сопровождения (рис. 2) ППО, если их рассматривать в виде цепочек последовательных преобразований информации, можно схематично представить следующим образом.

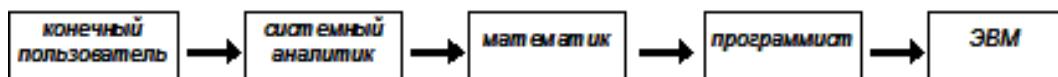


Рис. 1

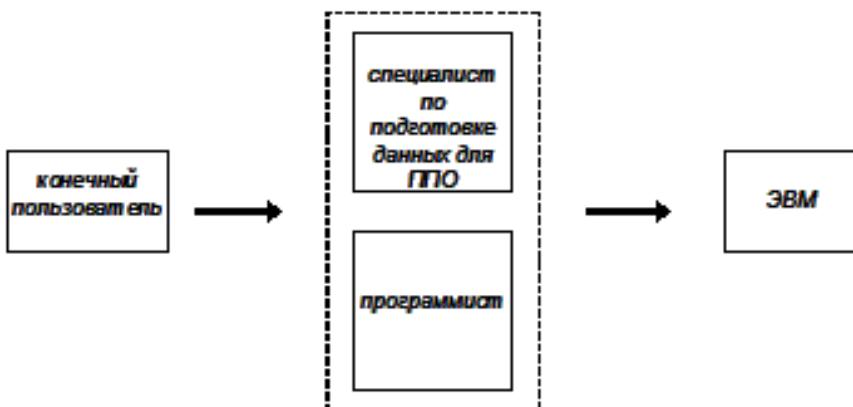


Рис. 2

Из приведенных рисунков нетрудно заключить, что традиционная технология имеет ряд существенных недостатков:

- может быть потеряна суть требований конечного пользователя, что приводит к разработке неадекватного ППО;
- избыточная трудоемкость процессов разработки и сопровождения ППО;
- невозможность эффективной адаптации ППО к динамически изменяющимся требованиям пользователя, т.к. ППО ориентировано на решение фиксированного списка задач заранее определенными способами. И, как следствие, - невозможность интеллектуализации труда пользователя;

Приведенные выше рисунки также показывают, что для устранения перечисленных недостатков (по крайней мере, большинства из них) необходимо исключить промежуточные элементы между конечным пользователем и ЭВМ. Именно в этом смысле говорят о новой ин-

формационной технологии, которая, по сути, представляет собой совокупность программных и технических средств, обеспечивающих реализацию процессов разработки и/или сопровождения ППО непосредственно пользователем (т.е. неспециалистом в области программирования).

## Концепция системы, основанной на знаниях

Исходными посылками для построения концепции KBS являются следующие два обстоятельства:

- с точки зрения пользователя ЭВМ и ППО неотделимы и должны предоставлять пользователю средства общения на языке максимально приближенном к естественному;
- схема решения задачи, которая поддерживается ППО, не зависит от технологии и имеет следующий вид:



Рис.3

С точки зрения этих посылок наивысшим достижением традиционной информационной технологии является отделение данных (входных и выходных) от ППО. Реализована концепция такого отделения в базах данных (БД) и соответствующей технологии – системах управления базами данных (СУБД). В процессе использования этой технологии выяснилось, что средства общения с пользователем могут иметь в лучшем случае процедурно-ориентированный характер. Первая же посылка подразумевает проблемную ориентацию средств общения. Для этого, в точном соответствии с тенденцией развития технологий разработки и сопровождения ППО, был сделан следующий шаг, смысл которого заключается в отделении знаний (о данных, алгоритме и т.п.) от ППО. Ясно, что если при этом под знаниями понимать любые отображения закономерностей, присущих той или иной предметной области, то можно говорить о включении всех предыдущих концепций в новую. Содержательно эволюцию технологий можно представить следующим образом.

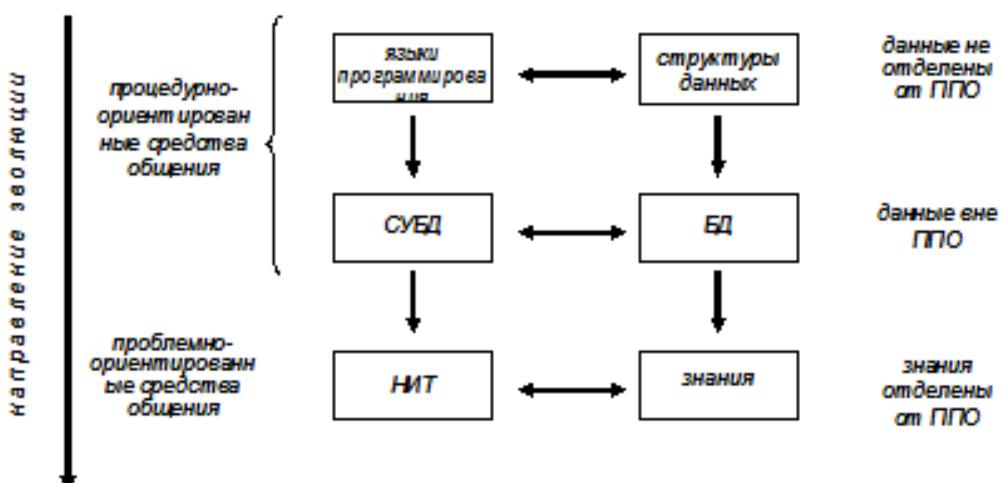


Рис. 4

Таким образом, идея отделения знаний от ППО, составляет ядро системы, основанной на знаниях. Сущность ее заключается во введении базы знаний (БЗ), как элементе, поддерживающем функционирование ППО на этапах разработки и сопровождения. С учетом этого, а

также того обстоятельства, что база знаний с позиции пользователя и ЭВМ обеспечивает выполнение различных функций, схему взаимодействия пользователя и ЭВМ можно представить следующим образом.

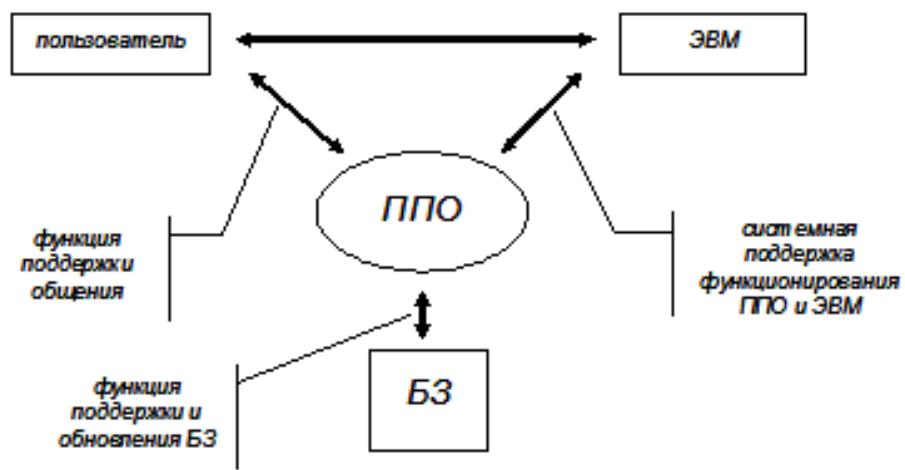


Рис. 5

Такие системы ППО принято называть системами, основанными на знаниях (KBS).

### Структура KBS

Подсистему (совокупность средств) KBS, обеспечивающую функцию поддержки общения, называют естественно-языковым интерфейсом (ЕЯ - интерфейсом). ЕЯ - интерфейс предназначен для удовлетворения информационных потребностей пользователя, как в процессе разработки, так и в процессе сопровождения KBS. Для этого, очевидно, ЕЯ – интерфейс должен выполнять некоторые обязательные функции. К числу которых обычно относят следующие:

- **ведение диалога.** Определение его структуры и той роли, которую пользователь и интерфейс выполняют на текущем шаге диалога;
- **понимание.** Преобразование поступающих от пользователя высказываний во внутренне представление;
- **обработка высказываний.** Формирование и определение заданий на данном шаге диалога;
- **интерпретация.** Формирование выходных высказываний на ЕЯ.

В соответствии с перечисленными функциями обобщенная схема ЕЯ – интерфейса может быть представлена следующим образом.



**Рис. 6**

Ключевым элементом КБС является БЗ. Для ее построения необходимо решить следующие проблемы:

- проблему представления знаний;
- проблему манипуляции знаниями.

При этом, с функциональной (рис. 5) и проблемной (рис. 3) точек зрения БЗ должна содержать:

- компоненту, поддерживающую ЕЯ – интерфейс;
- компоненту, поддерживающую общие знания о предметной области и обеспечивающую хранение и ввод входной информации;
- компоненту, поддерживающую хранение и необходимые манипуляции выходной информацией;
- компоненту, обеспечивающую необходимыми знаниями системную поддержку функционирования КБС и ЭВМ.

Следует отметить, что в наиболее общем смысле проблема построения КБС в настоящее время не решена. Однако существуют и успешно применяются конкретные представители КБС – экспертные системы. Такие системы являются методо-ориентированными представителями КБС, т.к. в ЭС алгоритм (метод) реализован по схеме логического вывода на основе знаний. Однако, указанная специфика не снижает общности структуры ЭС. Это обстоятельство подтверждается, помимо прочего, и тем, что ЭС успешно используются для решения самых разнообразных задач.

## Лекция 16

1. НАЗНАЧЕНИЕ ЭКСПЕРТНЫХ СИСТЕМ.
2. СТРУКТУРА ЭКСПЕРТНЫХ СИСТЕМ.
3. ЭТАПЫ РАЗРАБОТКИ ЭКСПЕРТНЫХ СИСТЕМ.
4. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ.
5. МЕТОДЫ ПОИСКА РЕШЕНИЙ В ЭКСПЕРТНЫХ СИСТЕМАХ.
6. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ.
  - Экспертные системы и KBS.
  - Возможности, примеры.

### Назначение экспертных систем

В начале восьмидесятых годов в исследованиях по искусственному интеллекту сформировалось самостоятельное направление, получившее название "экспертные системы" (ЭС). Цель исследований по ЭС состоит в разработке программ, которые при решении задач, трудных для эксперта-человека, получают результаты, не уступающие по качеству и эффективности решениям, получаемым экспертом. Исследователи в области ЭС для названия своей дисциплины часто используют также термин "инженерия знаний", введенный Е.Фейгенбаумом как "привнесение принципов и инструментария исследований из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов". Важность экспертных систем состоит в следующем:

- технология экспертных систем существенно расширяет круг практически значимых задач, решаемых на компьютерах, решение которых приносит значительный экономический эффект;
- технология ЭС является важнейшим средством в решении глобальных проблем традиционного программирования - длительность и, следовательно, высокая стоимость разработки сложных приложений;
- высокая стоимость сопровождения сложных систем, которая часто в несколько раз пре-восходит стоимость их разработки; низкий уровень повторного использования кода и т.п.;
- объединение технологии ЭС с технологией традиционного программирования добавляет новые качества к программным продуктам. В частности, обеспечиваются: динамичная модификация приложений пользователем, а не программистом; большая "прозрачность"; лучший интерфейс и организация взаимодействия.

По мнению специалистов в области ИИ в недалекой перспективе ЭС найдут следующее применение:

- ЭС будут играть ведущую роль во всех фазах проектирования, разработки, производства, распределения, продажи, поддержки и оказания услуг;
- технология ЭС, получившая коммерческое распространение, обеспечит революционный прорыв в интеграции приложений из готовых интеллектуально - взаимодействующих модулей.
- ЭС предназначены для решения задач, при формализации которых возникают сложности.

Неформализованные задачи обычно обладают следующими особенностями:

- ошибочностью, неоднозначностью, неполнотой и противоречивостью знаний о проблемной области и решаемой задаче;
- большой размерностью пространства решения, т.е. перебор при поиске решения весьма велик;
- динамически изменяющимися данными и знаниями.

Следует подчеркнуть, что неформализованные задачи представляют большой и очень важный класс задач. Считается, что эти задачи являются наиболее массовым классом задач, решаемых компьютерами. Экспертные системы и системы искусственного интеллекта отличаются от систем обработки данных тем, что в них в основном используются символьный (а не числовой) способ представления информации, символьный вывод и эвристический поиск.

По качеству и эффективности решения экспертные системы не уступают решениям эксперта-человека. 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. Необходимо отметить, что в настоящее время технология экспертных систем используется для решения различных типов задач: *интерпретация, предсказание, диагностика, планирование, конструирование, контроль, отладка, инструктаж, управление*.

Подобные задачи возникают в самых разнообразных предметных областях: финансы, нефтяная и газовая промышленность, энергетика, транспорт, фармацевтическое производство, космос, металлургия, горное дело, химия, образование, целлюлозно-бумажная промышленность, телекоммуникации и связь и др.

Коммерческие успехи к фирмам-разработчикам систем искусственного интеллекта (СИИ) пришли не сразу. На протяжении 1960 - 1985 гг. успехи ИИ касались в основном исследовательских разработок, которые демонстрировали пригодность СИИ для практического использования. Начиная примерно с 1985 г. (в массовом масштабе с 1988 - 1990 гг.), в первую очередь ЭС, а в последние годы системы, воспринимающие естественный язык (ЕЯ-системы), и нейронные сети (НС) стали активно использоваться в коммерческих приложениях.

## Структура экспертных систем

Типичная ЭС состоит из следующих основных компонентов (рис. 1):

- решателя (интерпретатора);
- рабочей памяти (РП), называемой также базой данных (БД);
- базы знаний (БЗ);
- компонентов приобретения знаний;
- объяснительного компонента;
- диалогового компонента.

**База данных (рабочая память)** предназначена для хранения исходных и промежуточных данных решаемой в текущий момент задачи. Этот термин совпадает по названию, но не по смыслу с термином, используемым в информационно-поисковых системах (ИПС) и системах управления базами данных (СУБД) для обозначения всех данных (в первую очередь долгосрочных), хранимых в системе.

**База знаний** (БЗ) в ЭС предназначена для хранения долгосрочных данных, описывающих рассматриваемую область (а не текущих данных), и правил, описывающих целесообразные преобразования данных этой области.

**Решатель**, используя исходные данные из рабочей памяти и знания из БЗ, формирует такую последовательность правил, которые, будучи примененными к исходным данным, приводят к решению задачи.

**Компонент приобретения знаний** автоматизирует процесс наполнения ЭС знаниями, осуществляемый пользователем-экспертом.

**Объяснительный компонент** объясняет, как система получила решение задачи (или почему она не получила решение) и какие знания она при этом использовала, что облегчает эксперту тестирование системы и повышает доверие пользователя к полученному результату.

**Диалоговый компонент** ориентирован на организацию общения с пользователем как в ходе решения задач, так и в процессе приобретения знаний и объяснения результатов работы.

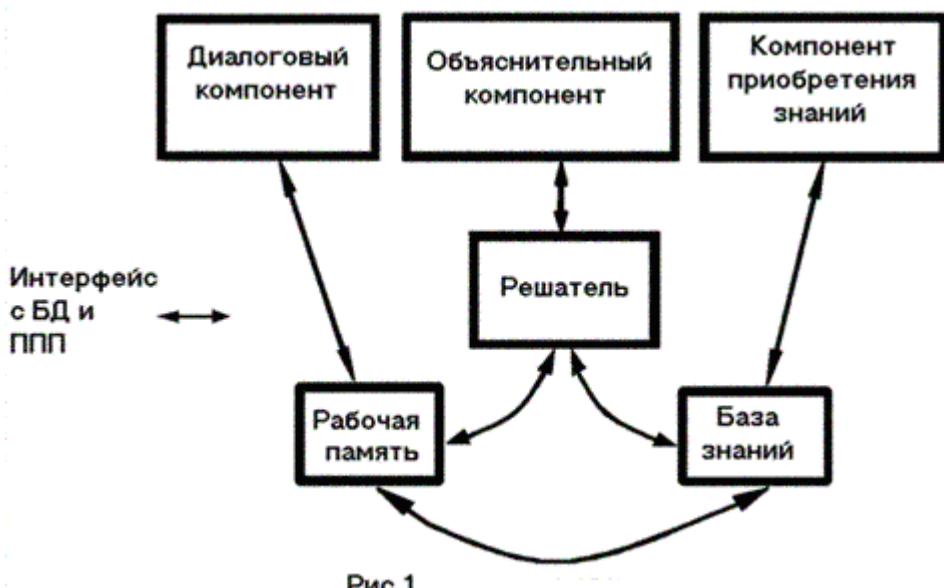


Рис.1

В разработке ЭС участвуют представители следующих специальностей:

- эксперт в проблемной области, задачи которой будет решать ЭС;
- инженер по знаниям - специалист по разработке ЭС (используемые им технологии, методы называют технологией (методами) инженерии знаний);
- программист по разработке инструментальных средств (ИС), предназначенных для ускорения разработки ЭС.

Необходимо отметить, что отсутствие среди участников разработки инженеров по знаниям (т. е. их замена программистами) либо приводит к неудаче процесса создания ЭС, либо значительно удлиняет его.

**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 (традиционными средствами) стандартные функции (типичные для данной проблемной области), которые будут использоваться в правилах, вводимых экспертом.

**Программист** разрабатывает ИС (если ИС разрабатывается заново), содержащее в пределах все основные компоненты ЭС, и осуществляет его сопряжение с той средой, в которой оно будет использовано.

Экспертная система работает в двух режимах: режиме приобретения знаний и в режиме решения задачи (называемом также режимом консультации или режимом использования ЭС).

**В режиме приобретения знаний** общение с ЭС осуществляется (через посредничество инженера по знаниям) экспертом. В этом режиме эксперт, используя компонент приобретения знаний, наполняет систему знаниями, которые позволяют ЭС в режиме решения самостоятельно (без эксперта) решать задачи из проблемной области. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты, их характеристики и значения, существующие в области экспертизы. Правила определяют способы манипулирования с данными, характерные для рассматриваемой области.

Отметим, что режиму приобретения знаний в традиционном подходе к разработке программ соответствуют этапы алгоритмизации, программирования и отладки, выполняемые программистом. Таким образом, в отличие от традиционного подхода в случае ЭС разработку программ осуществляет не программист, а эксперт (с помощью ЭС), не владеющий программированием.

**В режиме консультации** общение с ЭС осуществляется конечным пользователем, которого интересует результат и (или) способ его получения. Необходимо отметить, что в зависимости от назначения ЭС, пользователь может не быть специалистом в данной проблемной области. В этом случае он обращается к ЭС за результатом, не умея получить его сам. Или

быть специалистом. Тогда пользователь может сам получить результат, но он обращается к ЭС с целью либо ускорить процесс получения результата, либо возложить на ЭС рутинную работу. В режиме консультации данные о задаче пользователя после обработки их диалоговым компонентом поступают в рабочую память. Решатель на основе входных данных из рабочей памяти, общих данных о проблемной области и правил из БЗ формирует решение задачи. ЭС при решении задачи не только исполняет предписанную последовательность операций, но и предварительно формирует ее. Если реакция системы не понятна пользователю, то он может потребовать объяснения:

Структуру, приведенную на рис. 1, называют структурой статической ЭС. ЭС данного типа используются в тех приложениях, где можно не учитывать изменения окружающего мира, происходящие за время решения задачи. Первые ЭС, получившие практическое использование, были статическими.

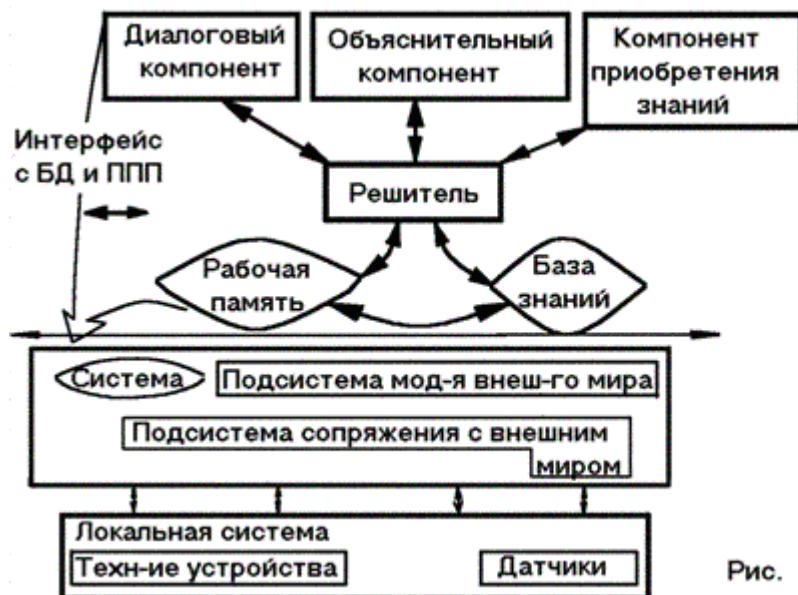


Рис. 2

На рис. 2 показана архитектура динамической ЭС. В нее вводятся два новых компонента: подсистема моделирования внешнего мира и подсистема связи с внешним окружением. Последняя подсистема осуществляет связи с внешним миром через систему датчиков и контроллеров. Кроме того, традиционные компоненты статической ЭС (база знаний и машина вывода) претерпевают существенные изменения, чтобы отразить временную логику происходящих в реальном мире событий.

### Этапы разработки экспертных систем

Разработка ЭС имеет существенные отличия от разработки обычного программного продукта. Опыт создания ЭС показал, что использование при их разработке методологии, принятой в традиционном программировании, либо чрезмерно затягивает процесс создания ЭС, либо вообще приводит к отрицательному результату.

Использовать ЭС следует только тогда, когда разработка ЭС возможна, оправдана и методы инженерии знаний соответствуют решаемой задаче. Чтобы разработка ЭС была возможной для данного приложения, необходимо одновременное выполнение, по крайней мере, следующих требований:

1. существуют эксперты в данной области, которые решают задачу значительно лучше, чем начинающие специалисты;
2. эксперты сходятся в оценке предлагаемого решения, иначе нельзя будет оценить качество разработанной ЭС;
3. эксперты способны вербализовать (выразить на естественном языке) и объяснить используемые ими методы, в противном случае трудно рассчитывать на то, что знания экспертов будут "извлечены" и вложены в ЭС;
4. решение задачи требует только рассуждений, а не действий;

5. задача не должна быть слишком трудной (т.е. ее решение должно занимать у эксперта несколько часов или дней, а не недель);
6. задача хотя и не должна быть выражена в формальном виде, но все же должна относиться к достаточно "понятной" и структурированной области, т.е. должны быть выделены основные понятия, отношения и известные (хотя бы эксперту) способы получения решения задачи;
7. решение задачи не должно в значительной степени использовать "здравый смысл" (т.е. широкий спектр общих сведений о мире и о способе его функционирования, которые знает и умеет использовать любой нормальный человек), так как подобные знания пока не удается (в достаточном количестве) вложить в системы искусственного интеллекта.

Использование ЭС в конкретном приложении может быть возможно, но не оправдано. Применение ЭС может быть оправдано одним из следующих факторов:

- решение задачи принесет значительный эффект, например, экономический;
- использование человека-эксперта невозможно либо из-за недостаточного количества экспертов, либо из-за необходимости выполнять экспертизу одновременно в различных местах;
- использование ЭС целесообразно в тех случаях, когда при передаче информации эксперту происходит недопустимая потеря времени или информации;
- использование ЭС целесообразно при необходимости решать задачу в окружении, враждебном для человека.

Приложение соответствует методам ЭС, если решаемая задача обладает совокупностью следующих характеристик:

1. задача может быть естественным образом решена посредством манипуляции с символами (т.е. с помощью символических рассуждений), а не манипуляций с числами, как принято в математических методах и в традиционном программировании;
2. задача должна иметь эвристическую природу, т.е. ее решение должно требовать применения эвристических правил. Задачи, которые могут быть гарантированно решены (с соблюдением заданных ограничений) с помощью некоторых формальных процедур, не подходят для применения ЭС;
3. 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;
4. задача должна быть достаточно узкой, чтобы решаться методами ЭС, и практически значимой.

При разработке ЭС, как правило, используется концепция "быстрого прототипа". Суть этой концепции состоит в том, что разработчики не пытаются сразу построить конечный продукт. На начальном этапе они создают прототип (прототипы) ЭС. Прототипы должны удовлетворять двум противоречивым требованиям. С одной стороны, они должны решать типичные задачи конкретного приложения, а с другой - время и трудоемкость их разработки должны быть весьма незначительны, чтобы можно было максимально запараллелить процесс накопления и отладки знаний (осуществляемый экспертом) с процессом выбора (разработки) программных средств (осуществляемым инженером по знаниям и программистом). Для удовлетворения указанным требованиям, как правило, при создании прототипа используются разнообразные средства, ускоряющие процесс проектирования.

Прототип должен продемонстрировать пригодность методов инженерии знаний для данного приложения. В случае успеха эксперт с помощью инженера по знаниям расширяет знания прототипа о проблемной области. При неудаче может потребоваться разработка нового прототипа или разработчики могут прийти к выводу о непригодности методов ЭС для данного приложения. По мере увеличения знаний прототип может достигнуть такого состояния, когда он успешно решает все задачи данного приложения. Преобразование прототипа ЭС в конечный продукт обычно приводит к перепрограммированию ЭС на языках низкого уровня, обеспечивающих как увеличение быстродействия ЭС, так и уменьшение требуемой памяти. Трудоемкость и время создания ЭС в значительной степени зависят от типа используемого инструментария.

В ходе работ по созданию ЭС сложилась определенная технология их разработки, включающая шесть следующих этапов (рис. 3): идентификацию, концептуализацию, формализацию, выполнение, тестирование, опытную эксплуатацию.

На этапе *идентификации* определяются задачи, которые подлежат решению, выявляются цели разработки, определяются эксперты и типы пользователей.

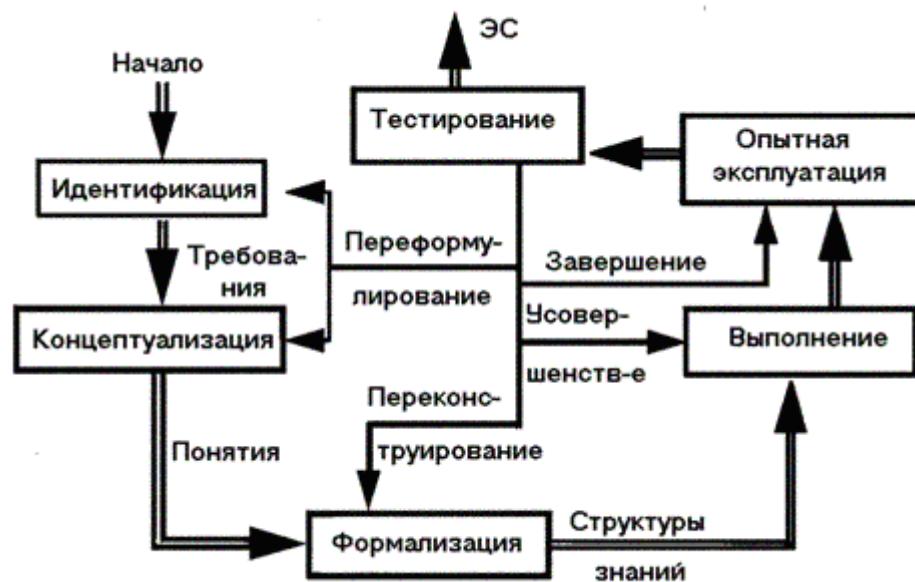


Рис. 3

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

На этапе *формализации* 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.

На этапе *выполнения* осуществляется 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, организацию 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.

### Представление знаний в экспертных системах

Первый и основной вопрос, который надо решить при представлении знаний, - это вопрос определения состава знаний, т.е. определение того, "ЧТО ПРЕДСТАВЛЯТЬ" в экспертной системе. Второй вопрос касается того, "КАК ПРЕДСТАВЛЯТЬ" знания. Необходимо отметить, что эти две проблемы не являются независимыми. Действительно, выбранный способ представления может оказаться непригодным в принципе либо неэффективным для выражения некоторых знаний. Вопрос "КАК ПРЕДСТАВЛЯТЬ" можно разделить на две в значительной степени независимые задачи: как организовать (структурировать) знания и как представить знания в выбранном формализме.

Выделение организации знаний в самостоятельную задачу вызвано тем, что эта задача возникает для любого языка представления, а способы решения этой задачи являются одинаковыми (либо сходными) вне зависимости от используемого формализма.

Итак, в круг вопросов, решаемых при представлении знаний, будем включать следующие:

- определение состава представляемых знаний;
- организацию знаний;
- представление знаний, т.е. определение модели представления.

Состав знаний ЭС определяется следующими факторами:

- проблемной средой;

- архитектурой экспертной системы;
- потребностями и целями пользователей;
- языком общения.

В соответствии с общей схемой статической экспертной системы (см. рис. 1) для ее функционирования требуются следующие знания:

- знания о процессе решения задачи (т.е. управляющие знания), используемые интерпретатором (решателем);
- знания языка общения и способа организации диалога, используемые лингвистическим процессором (диалоговым компонентом);
- знания о способах представления и модификации знаний, используемые компонентом приобретения знаний;
- поддерживающие структурные и управляющие знания, используемые объяснительным компонентом.

Для динамической ЭС, кроме того, необходимы следующие знания:

- знания о методах взаимодействия с внешним окружением;
- знания о модели внешнего мира.

Зависимость состава знаний от требований пользователя проявляется в следующем:

- какие задачи (из общего набора задач) и с какими данными хочет решать пользователь;
- каковы предпочтительные способы и методы решения;
- при каких ограничениях на количество результатов и способы их получения должна быть решена задача;
- каковы требования к языку общения и организации диалога;
- какова степень общности (конкретности) знаний о проблемной области, доступная пользователю;
- каковы цели пользователей.

Состав знаний о языке общения зависит как от языка общения, так и от требуемого уровня понимания.

С учетом архитектуры экспертной системы знания целесообразно делить на *интерпретируемые* и *неинтерпретируемые*. К первому типу относятся те знания, которые способен интерпретировать решатель (интерпретатор). Все остальные знания относятся ко второму типу. Решатель не знает их структуры и содержания. Если эти знания используются каким-либо компонентом системы, то он не "осознает" этих знаний. Неинтерпретируемые знания подразделяются на *вспомогательные* знания, хранящие информацию о лексике и грамматике языка общения, информацию о структуре диалога, и *поддерживающие* знания. Вспомогательные знания обрабатываются естественно-языковой компонентой, но ход этой обработки решатель не осознает, так как этот этап обработки входных сообщений является вспомогательным для проведения экспертизы. Поддерживающие знания используются при создании системы и при выполнении объяснений. Поддерживающие знания выполняют роль описаний (обоснований) как интерпретируемых знаний, так и действий системы. Поддерживающие знания подразделяются на *технологические* и *семантические*. Технологические поддерживающие знания содержат сведения о времени создания описываемых ими знаний, об авторе знаний и т.п. Семантические поддерживающие знания содержат смысловое описание этих знаний. Они содержат информацию о причинах ввода знаний, о назначении знаний, описывают способ использования знаний и получаемый эффект. Поддерживающие знания имеют описательный характер.

Интерпретируемые знания можно разделить на *предметные знания, управляющие знания* и *знания о представлении*. Знания о представлении содержат информацию о том, каким образом (в каких структурах) в системе представлены интерпретируемые знания.

Предметные знания содержат данные о предметной области и способах преобразования этих данных при решении поставленных задач. Отметим, что по отношению к предметным знаниям знания о представлении и знания об управлении являются *метазнаниями*. В предметных знаниях можно выделить описатели и собственно предметные знания. Описатели содержат определенную информацию о предметных знаниях, такую, как коэффициент определенности правил и данных, меры важности и сложности. Собственно предметные знания разбиваются на *факты и исполняемые утверждения*. Факты определяют возможные значения сущностей и характеристик предметной области. Исполняемые утверждения со-

держат информацию о том, как можно изменять описание предметной области в ходе решения задач. Говоря другими словами, исполняемые **утверждения** - это знания, задающие процедуры обработки. Однако мы избегаем использовать термин "процедурные знания", так как хотим подчеркнуть, что эти знания могут быть заданы не только в процедурной, но и в декларативной форме.

Управляющие знания можно разделить на **фокусирующие** и **решающие**. Фокусирующие знания описывают, какие знания следует использовать в той или иной ситуации. Обычно фокусирующие знания содержат сведения о наиболее перспективных объектах или правилах, которые целесообразно использовать при проверке соответствующих гипотез (см. п. 9.2). В первом случае внимание фокусируется на элементах рабочей памяти, во втором - на правилах базы знаний. Решающие знания содержат информацию, используемую для выбора способа интерпретации знаний, подходящего к текущей ситуации. Эти знания применяются для выбора стратегий или эвристик, наиболее эффективных для решения данной задачи.

Качественные и количественные показатели экспертной системы могут быть значительно улучшены за счет использования **метазнаний**, т.е. знаний о знаниях. Метазнания не представляют некоторую единую сущность, они могут применяться для достижения различных целей. Перечислим возможные назначения метазнаний:

1. метазнания в виде стратегических метаправил используются для выбора релевантных правил;
2. метазнания используются для обоснования целесообразности применения правил из области экспертизы;
3. метаправила используются для обнаружения синтаксических и семантических ошибок в предметных правилах;
4. метаправила позволяют системе адаптироваться к окружению путем перестройки предметных правил и функций;
5. метаправила позволяют явно указать возможности и ограничения системы, т.е. определить, что система знает, а что не знает.

Вопросы организации знаний необходимо рассматривать в любом представлении, и их решение в значительной степени не зависит от выбранного способа (модели) представления.

Выделим следующие аспекты проблемы организации знаний:

- организация знаний по уровням представления и по уровням детальности;
- организация знаний в рабочей памяти;
- организация знаний в базе знаний.

Для того чтобы экспертная система могла управлять процессом поиска решения, была способна приобретать новые знания и объяснять свои действия, она должна уметь не только использовать свои знания, но обладать способностью понимать и исследовать их. Иными словами, экспертная система должна иметь знания о том, как представлены ее знания о проблемной среде. Если знания о проблемной среде назвать знаниями нулевого уровня представления, то первый уровень представления содержит метазнания, т.е. знания о том, как представлены во внутреннем мире системы знания нулевого уровня. Первый уровень содержит знания о том, какие средства используются для представления знаний нулевого уровня. Знания первого уровня играют существенную роль при управлении процессом решения, при приобретении и объяснении действий системы. В связи с тем, что знания первого уровня не содержат ссылок на знания нулевого уровня, знания первого уровня независимы от проблемной среды.

Число уровней представления может быть больше двух. Второй уровень представления содержит сведения о знаниях первого уровня, т.е. знания о представлении базовых понятий первого уровня. Разделение знаний по уровням представления обеспечивает расширение области применимости системы.

Выделение уровней детальности позволяет рассматривать знания с различной степенью подробности. Количество уровней детальности во многом определяется спецификой решаемых задач, объемом знаний и способом их представления. Как правило, выделяется не менее трех уровней детальности, отражающих соответственно общую, логическую и физическую организацию знаний. Введение нескольких уровней детальности обеспечивает дополнительную степень гибкости системы, так как позволяет производить изменения на одном уровне, не затрагивая другие. Изменения на одном уровне детальности могут приводить к дополнительным изменениям на этом же уровне, что оказывается необходимым для

обеспечения согласованности структур данных и программ. Однако наличие различных уровней препятствует распространению изменений с одного уровня на другие.

## Методы поиска решений в экспертных системах

Требования пользователя к результату задачи, решаемой с помощью поиска, можно характеризовать количеством решений и свойствами результата и (или) способом его получения. Параметр "количество решений" может принимать следующие основные значения: одно решение, несколько решений, все решения. Параметр "свойства" задает ограничения, которым должен удовлетворять полученный результат или способ его получения. Так, например, для системы, выдающей рекомендации по лечению больных, пользователь может указать требование не использовать некоторое лекарство (в связи с его отсутствием или в связи с тем, что оно противопоказано данному пациенту).

Итак, сложность задачи, определяемая вышеприведенным набором параметров, варьируется от простых задач малой размерности с неизменяемыми определенными данными и отсутствием ограничений на результат и способ его получения до сложных задач большой размерности с изменяемыми, ошибочными и неполными данными и произвольными ограничениями на результат и способ его получения. Из общих соображений ясно, что каким-либо одним методом нельзя решить все задачи. Обычно одни методы превосходят другие только по некоторым из перечисленных параметров.

Рассмотренные ниже методы могут работать в статических и динамических проблемных средах. Для того чтобы они работали в условиях динамики, необходимо учитывать время жизни значений переменных, источник данных для переменных, а также обеспечивать возможность хранения истории значений переменных, моделирования внешнего окружения и оперирования временными категориями в правилах.

Существующие методы решения задач, используемые в экспертных системах, можно классифицировать следующим образом:

- методы поиска в одном пространстве - методы, предназначенные для использования в следующих условиях: области небольшой размерности, полнота модели, точные и полные данные;
- методы поиска в иерархических пространствах - методы, предназначенные для работы в областях большой размерности;
- методы поиска при неточных и неполных данных ;
- методы поиска, использующие несколько моделей, предназначенные для работы с областями, для адекватного описания которых одной модели недостаточно.

Предполагается, что перечисленные методы при необходимости должны объединяться для того, чтобы позволить решать задачи, сложность которых возрастает одновременно по некоторым параметрам.