

## **Лекция 1. 13.02**

Intellectus (лат.) - ум, рассудок, мыслительные способности человека. Интеллектом можно называть способность мозга решать (интеллектуальные) задачи путем приобретения, запоминания и целенаправленного преобразования знаний в процессе обучения на опыте и адаптации к разнообразным обстоятельствам.

Искусственный интеллект (artificial intelligence, AI, ИИ) - это наука о концепциях, позволяющих ЭВМ делать такие вещи, которые у людей выглядят разумными.

Искусственный интеллект (ГОСТ 33707-2016) - способность функционального блока выполнять функции, обычно ассоциирующиеся с интеллектом человека такие, как, например, рассуждения и обучение.

Искусственный интеллект (основное определение) - это одно из направлений информатики, цель которого - разработка программных и аппаратных средств, способных выполнять функции, традиционно считающиеся интеллектуальными, - понимание языка, логический вывод, использование накопленных знаний, обучение, планирование действий.

ИИ позволяет пользователю-непрограммисту ставить и решать свои интеллектуальные задачи, общаясь с ЭВМ на ограниченном подмножестве естественного языка.

### **Факторы широкого использования ИИ:**

- способен автоматизировать даже те процессы, которые ранее требовали участия человека
- может быстро обрабатывать и анализировать гигантские объемы информации и просчитывать варианты, используя множество переменных
- машина не подвержена человеческому фактору, а ее работоспособность не зависит от эмоций и личных проблем

Виды искусственного интеллекта: ИИ узкого назначения (слабый), ИИ общего назначения (сильный), Суперинтеллект.

### **Слабый ИИ**

Слабый ИИ - система, разработанная для выполнения конкретных задач в ограниченной области подчас лучше, чем человек.

При этом, превышая возможности человека в узкой области, такая система, как правило, не имеет интеллектуальных способностей в других сферах, в отличие от человека, который обучается решению задач в самых разных областях.

Он не обладает самосознанием и не способен к обобщению знаний для решения проблем, которые не предусмотрены его алгоритмами.

Такие системы могут работать в режиме реального времени, способны обрабатывать большие объемы данных и выполнять действия быстрее и точнее, чем человек, но они не могут выйти за пределы запрограммированных функций.

### **Сильный ИИ**

ИИ общего назначения, или сильный ИИ, может успешно выполнять любые умственные задачи, которые под силу людям.

Сейчас компьютеры не способны мыслить абстрактно, продумывать стратегию, а также использовать мысли и воспоминания, чтобы принимать обоснованные решения или выдвигать творческие идеи.

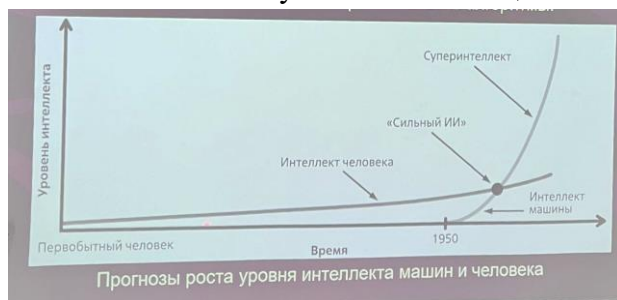
Ожидается, что сильный ИИ сможет рассуждать, справляться с проблемами, выносить суждения в условиях неопределенности, планировать, учиться, интегрировать предыдущие знания в процесс принятия решений, а также предлагать новаторские идеи. Нельзя построить сильный ИИ, постепенно наращивая мощных известных решений на базе слабого ИИ и интегрируя их между собой. Для построения сильного ИИ нужен качественный скачок, новая архитектура.

### **Суперинтеллект**

В отличие от сильного и слабого ИИ, суперинтеллект будет не просто выполнять те же задачи, а также генерировать новые подходы и решения, которые выходят за пределы возможностей человеческого разума. Такая система способна быстро адаптироваться, обучаться и совершенствовать алгоритмы.

Особенности супер-ИИ:

- Превосходит интеллект человека
- Способен к самообучению и эволюции



### **Интеллектуальные задачи**

Всякая задача, для которой неизвестен алгоритм решения, априорно относится к задачам ИИ.

Решение должно получаться за приемлемое время.

Особенности интеллектуальных задач:

- используется информация в символьной форме
- предполагается наличие выбора

### **Подходы к построению систем ИИ**

**Структурный подход** - построение ИИ путем моделирования структуры человеческого мозга.

$10^{11}$  -  $10^{12}$  нейронов мозга

-понять функционирование мозга

-создать системы, выполняющие функции, сходные с функциями мозга

Данные, поступающие на вход нейросети, проходят обработку на каждом слое сети.

При этом каждый нейрон имеет определенные параметры, которые могут изменяться в зависимости от получения результатов - в этом и заключается обучение сети.

1943г., Уоррен Маккалок и Уолтер Питтс - формальный логический нейрон

1951 г., Марвин Минский - первый нейрокомпьютер SNARC (Stochastic Neural Analog Reinforcement Calculator), содержащий 40 нейронов

1958 г., Фрэнк Розенблатт - первая нейронная сеть

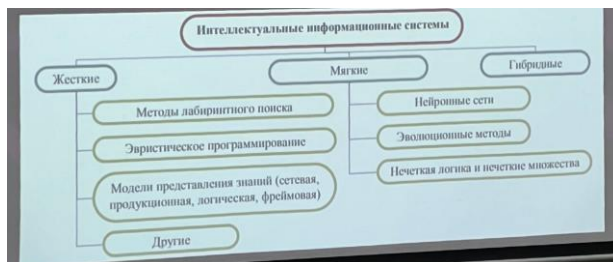
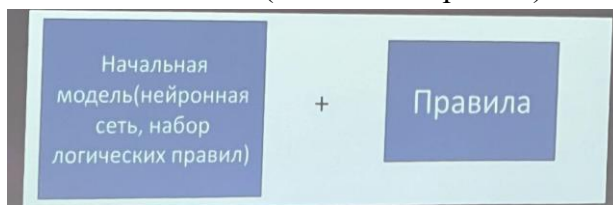
2010 г. - база данных ImageNet, содержащая 15 миллионов изображений в 22 тысячах категорий

**Имитационный (эвристический) подход** - построение ИИ путем имитационного моделирования интеллектуальной деятельности человека по результату.

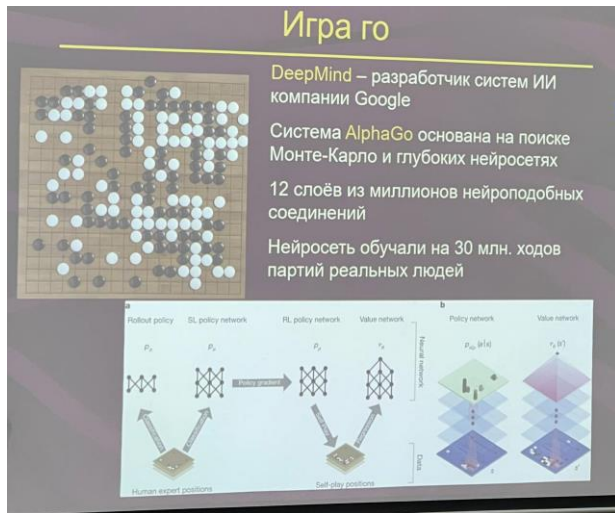
**Эвристики** - правила, которые позволяют сделать выбор при отсутствии точных теоретических оснований.



**Эволюционный подход** - построение начальной модели и правил, по которым она может изменяться (эволюционировать).



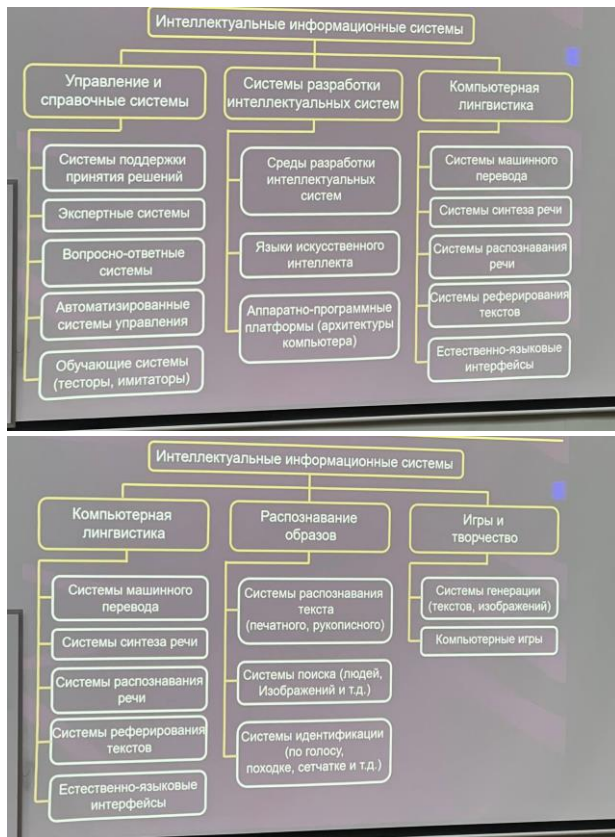
1957 г. - программа “Логик-Теоретик” (доказательство теорем в исчислении высказываний).



## Классификации ИИ по решаемым задачам

Интеллектуальные информационные системы:

- Управление и справочные системы
  1. Системы поддержки принятия решений
  2. Экспертные системы
  3. Вопросно-ответные системы
  4. Автоматизированные системы управления
  5. Обучающие системы (тесторы, имитаторы)
- Системы разработки интеллектуальных систем
  1. Среды разработки интеллектуальных систем
  2. Язык искусственного интеллекта
  3. Аппаратно-программные платформы (архитектуры компьютера)
- Компьютерная лингвистика
  1. Системы машинного перевод
  2. Системы синтеза речи
  3. Системы распознавания речи
  4. Системы реферирования текстов
  5. Естественнo-языковые интерфейсы)
- Распознавание образов
  1. Системы распознавания текста (печатного, рукописного)
  2. Системы поиска (людей, изображений и т.д.)
  3. Системы идентификации (по голосу, по сетчатке, по походке)
- Игры и творчество
  1. Системы генерации (текстов, изображений)
  2. Компьютерные игры



## Лекция 2. 20.02.25

### Знания и данные

Обычная программа = Алгоритм + Данные

Интеллектуальная программа = Знания + Стратегия обработки знаний

В стратегию обработки знаний входят методы, модели и алгоритмы, ориентированные на автоматическое накопление знаний на основе анализа и обобщения.



уровня

Будем рассматривать только 3 нижних

**Данные** - это совокупность фактов, значений, наблюдений или записей, представленных в форме, пригодной для обработки и анализа (представляют собой неструктурированную информацию для ее дальнейшего использования)

**Основные характеристики данных:**

- Сырой материал: представляют собой необработанную информацию, которая требует анализа и интерпретации
- Низкая степень осмысленности: сами по себе не несут значимого контекста, поэтому они требуют обработки для превращения в информацию
- Разнообразие форм: числа, текст, изображения, звук и т.п.

Примеры данных: показатели температуры, цены товаров, результаты тестов и т.д.

**Информация** - данные, обработанные таким образом, что они приобретают смысл и становятся полезными для пользователя.

**Основные характеристики информации:**

- Контекст: несет определенный смысл, так как она привязана к конкретному контексту или ситуации
- Применимость: предназначена для использования в конкретных задачах или для ответов на вопросы
- Ценность: имеет ценность для пользователя, так как позволяет принимать обоснованные решения

Примеры информации: отчет о продажах, график изменения цен, демографический анализ и т.п.

**Знания** - обобщенная и интерпретированная информация, которая была понятна, усвоена и может быть использована для решения задач, принятия решений и прогнозирования.

**Знания** - связи и закономерности предметной области (принципы, модели, законы), полученные в результате практической деятельности и профессионального опыта, позволяющего ставить и решать задачи в данной области. (Быстриковой это определение нравится больше)

**Основные характеристики знаний:**

- Осмысленность: представляет собой структурированную и осознанную информацию, которая может быть передана другим
- Понимание закономерностей: основываются на понимании связей, причин и следствий, позволяет предсказывать результаты и выбирать наилучшие действия
- Применение в практике: не только воспринимаются, но и используются для решений конкретных задач

Примеры знаний: понимание рынка на основе опыта работы, научные теории, экспертные выводы в медицине или инженерии.

Знания и данные			
Параметр	Данные	Информация	Знания
Смысл	Не имеют самостоятельного смысла	Содержат контекст, приобретают смысл	Представляют обобщенную и осознанную информацию
Степень обработки	Сырой материал	Обработаны и структурированы	Интерпретированы и поняты
Применение	Требуют обработки	Полезны для решения задач	Используются для принятия решений и прогнозирования
Примеры	Числа, факты, записи	Отчеты, анализы, графики	Теории, рекомендации, опыт

### Особенности знаний:

1. Внутренняя интерпретируемость: каждая информационная единица знаний должна иметь уникальное имя по которому интеллектуальная система находит ее, а также отвечает на запросы в которых это имя упомянуто.
2. Структурированность: информационные единицы должны обладать гибкой структурой, причем для них может выполняться “принцип Матрешки” (вложенность). Отношения “часть-целое”, “род-вид”, “элемент-класс” (иерархическая структура).
3. Связность: между информационными единицами должна быть предусмотрена возможность установления связей.
4. Семантическая метрика: свойство, которое позволяет создавать отношение, характеризующее ситуационную близость информационных единиц, позволяет выявлять одинаковые ситуации при решении однотипных задач.
5. Активность: выполнение действий в интеллектуальной системе должно инициироваться не командами, а текущим состоянием знаний, представленных в системе, тогда источником активности интеллектуальной системы должно стать появление новых фактов, установление новых связей и т.п.

### Знания:

- Формализованные (знания, которые содержатся в книгах, письмах, отчетах, то есть те знания которые уже задокументированы или которые можно задокументировать)
- Неформальные (знания, которые трудно или невозможно формализовать. Наше мнение, ощущения, впечатления и т.п.)

### Знания:

- Декларативные (знания об объектах и явлениях окружающего мира, которые носят описательный характер. Отвечают на вопросы Кто? Что? Как это был? и т.д.)



- Процедурные (знания, которые хранятся в интеллектуальных системах в виде описания процедур, с помощью которых их можно получить)

### Поиск решения интеллектуальных задач

Процесс решения задачи:

- Представление знаний
- Поиск решения
  1. Использование пространства состояний (поиск решения задачи среди альтернативных вариантов)
  2. Редукция задачи

### Граф пространства состояний

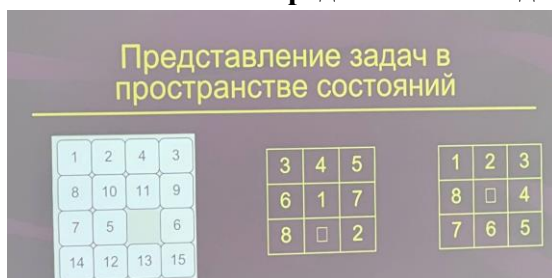


**Узлы** - состояния игрового поля.

**Дуги** - разрешенные ходы, приводящие от одной конфигурации игры к другой.

**Эффективная игровая стратегия** - все пути, ведущие к наибольшему числу побед и наименьшему числу поражений.

### Представление задач в пространстве состояний



**Состояние** характеризует момент решения задачи.

**Начальное** состояние соответствует исходной информации поставленной задачи. (Есть задачи которые всегда имеют одно и то же начальное состояние: крестики-нолики, шахматы).

**Целевое** состояние может быть:

- задано явно (игра в восемь (пятнашки, но поменьше))
- определено неявно (охарактеризовано неким свойством)



**Оператор** - допустимый ход в задаче. Оператор преобразует одно состояние в другое, является функцией определенной на множестве состояний и принимающий значение из этого множества.

**Решение задачи** - определенная последовательность операторов, преобразующих начальное состояние в целевое.

**Пространство состояний** - множество всех состояний, достижимых из начального состояния при помощи заданных операторов. Графически представляется в виде направленного графа, в котором вершины графы - состояния, дуги - операторы. Решение задачи - путь в графе, ведущий от корневой вершины к конечной.

### **Формализация задачи в пространстве состояний**

Задача -  $(S_n, S_c, O)$

$S_n$  - множество начальных состояний

$S_c$  - множество целевых состояний

$O$  - множество операторов (с учетом ограничений)

Дальнейшая формализация решения задачи предполагает выбор конкретной формы описания состояния задачи.

Любой граф может быть определен явно или неявно.

При явном определении граф задается связью вершин и дуг.

При неявном определении задается начальная вершина и набор операторов.

**Решение задачи** подразумевает просмотр неявно заданного графа:

1. В исходной точке к начальному состоянию применяется тот или иной оператор и строится новая вершина-состояние, а также дуги, связывающие ее с корневой вершиной.
2. На каждом следующем шаге поиска к одной из уже полученных вершин-состояний применяется допустимый оператор и строится еще одна вершина графа и связывающие дуги.
3. Процесс поиска продолжается до тех пор, пока не будет построена вершина, соответствующая целевому состоянию.

## Примеры пространств состояний

Задача 1. Игра в восемь

Начальные состояния: 8 пронумерованных фишек, произвольно расположенных в квадрате из 9 клеток

Целевое состояние:

1	2	3
8	□	4
7	6	5

Операторы: перемещение пустой клетки вверх, вниз, влево, вправо

Ограничения: выход за пределы игрового поля

## Примеры пространств состояний

Задача 2. «Крестики-нолики»

Начальное состояние: пустая игровая доска

Целевые состояния: 3 крестика располагаются в одной строке, столбце или по диагонали

Операторы: записывание x или o в пустую клетку

### Лекция 3. 27.02.25

## Поиск решения в пространстве состояний

**Начальная вершина** - вершина графа, соответствующая начальному состоянию.

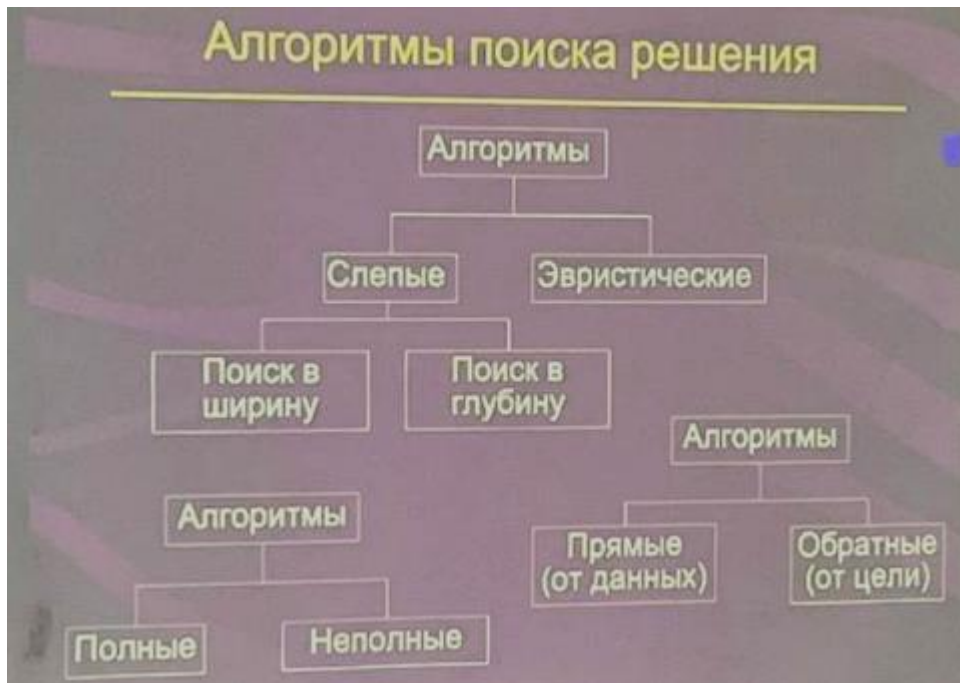
**Целевая вершина** - вершина, соответствующая целевому состоянию.

**Раскрытие вершины** - построение всех ее потомков путем применения к соответствующему состоянию задачи всех допустимых операторов.

**Поиск в пространстве состояний** - процесс постепенного раскрытия вершин и проверки свойств порождаемых вершин.

## Алгоритмы поиска

- использование эвристической информации
- порядок перебора (раскрытия) вершин
- полнота просмотра пространства состояний
- направление поиска



Алгоритмы делятся на:

- В слепых алгоритмах местоположения в пространстве целевой вершины никак не влияет на тот порядок, в котором рассматривается вершина.
- Эвристические алгоритмы используют дополнительную информацию об общем виде пространства состояний и/или о том, где расположена цель, поэтому для раскрытия выбирается наиболее перспективные вершины.

Слепые делятся на:

- поиск в ширину
- поиск в глубину

Алгоритмы:

- Полные - при необходимости осуществляется полный просмотр графа пространства состояний и гарантирует нахождение целевого состояния, если оно существует.
- Неполные - просматривают лишь некоторую часть пространства состояний и если оно не содержит целевую вершину, то оно не будет найдено.

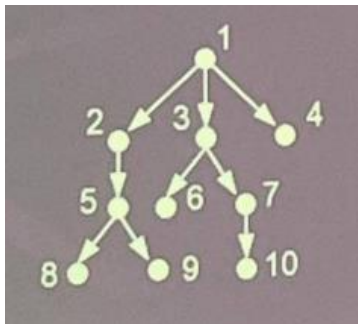
Алгоритмы:

- Прямые (от данных) - начинаются с условия задачи и выполняются путем применения операторов для получения новых состояний
- Обратные (от цели) - начинаются с обращения к цели и продолжается путем определения операторов, которые могут привести к цели и построение цепочки от цели к исходным данным задачи.



## Алгоритмы перебора

В алгоритме перебора в ширину, вершины раскрываются в том порядке, в котором они строятся.



Поиск в ширину исследует порядок вершин по уровню и если вершин на данном уровне нет, то переходим к следующему уровню.

В алгоритме перебора в глубину прежде всего раскрываются те вершины, которые были построены последними.



При поиске в глубину сначала исследуются потомки, если их нет затем исследуются братья.

**Брат** - тот, у кого общий родитель.

Сначала рассматривается базовый алгоритм для графов деревьев (каждая вершина имеет только одного предка).

### Базовый алгоритм поиска в ширину

OPEN - список построенных, но нераскрытых вершин.

Шаг 1. Поместить начальную вершину (от которой нужно найти путь) в список Open .

Шаг 2. Если список Open пуст, то конец алгоритма с сообщением о неудаче поиска, иначе перейти к 3

Шаг 3. Выбрать первую вершину из списка Open (назовем ее X) и удалить ее из Open

Шаг 4. Если X - целевая вершина, то конец алгоритма и выдача решения, иначе перейти к 5

Шаг 5. Образовать всех потомков вершины X и добавить их в конец списка Open.

Построить указатели от этих вершин к X. Перейти к 2.

Указатели хранятся в виде пары, вершины и ее родителя.



Поскольку при поиске в ширину вершины графа рассматриваются по уровням, то сначала проверяются те состояния, путь к которым короче т.е. поиск в ширину гарантирует нахождение самого короткого пути.

### Расширенный алгоритм поиска в ширину

CLOSED - список исследованных (раскрытых) вершин.

Шаг 1. Поместить начальную вершину в список Open. Создать пустой список Closed

Шаг 2. Если список Open пуст, то конец алгоритма с сообщением о неудаче поиска, иначе перейти к 3

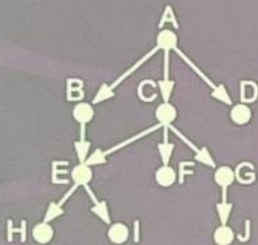
Шаг 3. Выбрать первую вершину X из списка Open и перенести ее в список Closed, удалив из Open

Шаг 4. Если X - целевая вершина, то конец алгоритма и выдача решения, иначе перейти к 5

Шаг 5. Образовать потомков X. Не находящихся в Open и Closed добавить в конец списка Open. Построить указатели от этих вершин к X. Перейти к 2.

## Поиск в ширину

A – начальное состояние, F – целевое состояние



Шаг	X	Open	Closed
		(A)	
1	A	(B,C,D)	(A)
2	B	(C,D,E)	(A,B)
3	C	(D,E,F,G)	(A,B,C)
4	D	(E,F,G)	(A,B,C,D)
5	E	(F,G,H,I)	(A,B,C,D,E)
6	F	цель	

Указатели такие же как в прошлой табличке

## Поиск в глубину

Шаг 1. Поместить начальную вершину в список Open. Создать пустой список Closed

Шаг 2. Если список Open пуст, то конец алгоритма с сообщением о неудаче поиска, иначе перейти к 3

Шаг 3. Выбрать первую вершину X из списка Open и перенести ее в список Closed, удалив из Open.

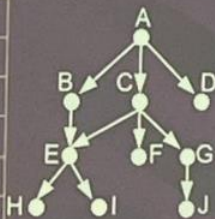
Шаг 4. Если X - целевая вершина, то конец алгоритма и выдача решения, иначе перейти к 5

Шаг 5. Образовать потомков X. Не находящихся в Open и Closed добавить в начало списка Open. Построить указатели от этих вершин к X. Перейти к 2.

## Поиск в глубину

Шаг 5. Образовать потомков X. Не находящихся в Open и Closed добавить в **начало** списка Open. Построить указатели от этих вершин к X. Перейти к 2.

Шаг	X	Open	Closed
		(A)	
1	A	(B,C,D)	(A)
2	B	(E,C,D)	(A,B)
3	E	(H,I,C,D)	(A,B,E)
4	H	(I,C,D)	(A,B,E,H)
5	I	(C,D)	(A,B,E,H,I)
6	C	(F,G,D)	(A,B,E,H,I,C)
7	F	цель	



В общем виде поиск в глубину может привести к бесконечному процессу. Это происходит если поиск в глубину пошел по ветке не содержащей целевую вершину.

### Глубина просмотра дерева:

- глубина корневой вершины равна нулю
- глубина каждой некорневой вершины на единицу больше глубины ее родительской вершины

**Ограниченный перебор в глубину** - алгоритм поиска, в ходе которого можно строить только те вершины, глубина которых не превышает некоторую заданную граничную глубину.

## Сравнение алгоритмов

### Поиск в ширину

- +: самый короткий путь к целевой вершине
  - : комбинаторный взрыв - чрезмерное количество элементов в списке Open
- Если каждое состояние порождает в среднем  $B$  потомков, то общее число состояний на  $n$ -ом уровне -  $B^n$

### Поиск в глубину

- : пропускает более короткие пути к целевой вершине
- +: если каждое состояние порождает в среднем  $B$  потомков, то общее число состояний на уровне -  $B \cdot n$

## Бэктрекинг

Бэктрекинг (режим возвратов) предлагает определенную организацию перебора всех возможных вариантов решения задачи.



В каждой точке процесса решения, где существует несколько равноправных альтернативных путей дальнейшего продолжения (развилок), выбрать один из них и следовать ему.

Предварительно следует запомнить другие альтернативные пути, чтобы в случае неудачи выбранного пути вернуться в указанную точку и выбрать следующий альтернативный вариант.

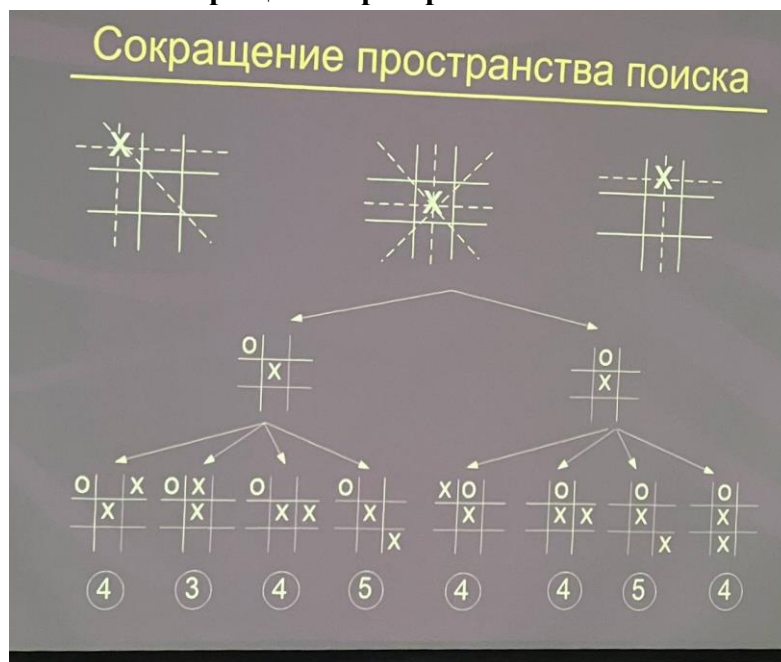
#### Лекция 4 06.03.25

### Эвристический поиск

**Эвристика** - это закон правил для выбора тех ветвей из пространства состояний, которые с наибольшей вероятностью приведут к приемлемому решению проблемы.

1. Проблема может не иметь точного решения из-за неопределенности в постановке задачи и/или в исходных данных.
2. Проблема может иметь точное решение, но стоимость его поиска может быть слишком высокой.

### Сокращение пространства поиска



В 1й конфигурации - 3 выигрышных линии

Во 2й - 4 выигрышных линии

В 3й - 2 выигрышных линии

Эвристика для игры в крестики-нолики

Делать такие ходы, в которых крестики/нолики имеют наибольшее количество потенциально выигрышных линий.

В основе эвристических алгоритмов лежит идея оценивания не раскрытых вершин с помощью эвристической функции и выбор для продолжения поиска наиболее перспективной вершины.

$Est(V)$  - функция, которая определяет перспективность раскрытия вершины. Она определяется на множестве вершин (то есть для каждой вершины) пространства состояний и принимает числовое значение.

$Est(V1) < Est(V2) \Rightarrow V1$  более перспективна, чем  $V2$  (не всегда, зависит от конкретной задачи)

### Алгоритм эвристического перебора

Шаг 1. Поместить начальную вершину в список Open. Создать пустой список Closed.

Шаг 2. Если список Open пуст, то конец алгоритма с сообщением о неудаче поиска, иначе перейти к 3.

Шаг 3. Выбрать первую вершину X из списка Open и перенести ее в список Closed, удалив из Open.

Шаг 4. Если X - целевая вершина, то конец алгоритма и выдача решения, иначе перейти к 5.

Шаг 5. Образовать потомков X. Для не находящихся в Open и Closed вычислить оценку (значение оценочной функции) и добавить в список Open.

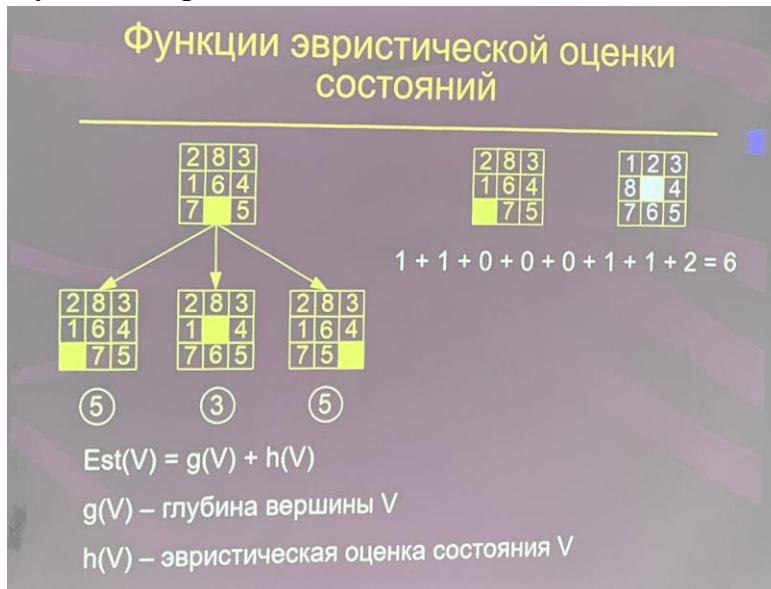
Шаг 6. Переупорядочить вершины в списке Open в соответствии с оценками (минимальные слева). Перейти к шагу 2.

**Эвристический поиск**

А – начальное состояние,  
О – целевое

X	Open	Closed
	(A-7)	()
A-7	(B-5,D-5,C-6)	(A)
B-5	(D-5,C-6,F-6,E-7)	(A,B)
D-5	(H-4,I-5,C-6,F-6,E-7)	(A,B,D)
H-4	(I-5,C-6,F-6,E-7)	(A,B,D,H)
I-5	(O-4,C-6,F-6,E-7)	(A,B,D,H,I)
O-4	цель	

## Функции эвристической оценки состояний



Эвристика состоит в том, чтобы подсчитать количество фишек, стоящих не на своем месте.

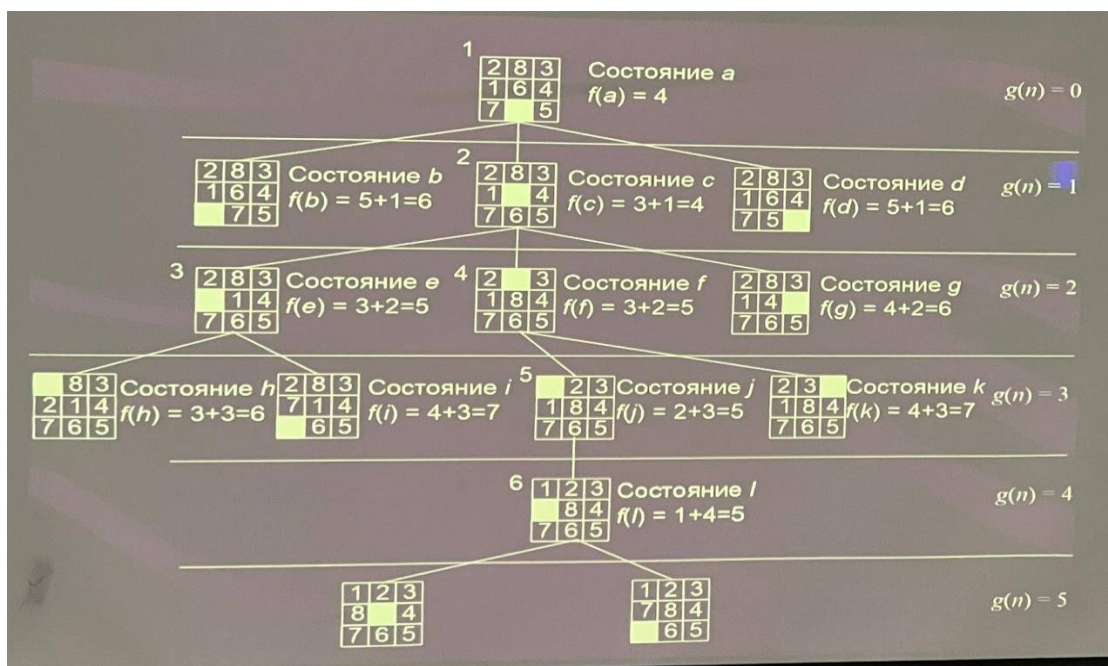
Суммируем все расстояния между фишками, находящимися на своем месте и полем, куда ее надо переместить для достижения цели.

Если два состояния имеют одинаковые или почти одинаковые эвристические значения, то предпочтительнее из них выбрать то, которое ближе лежит к начальному состоянию.

$$Est(V) = g(V) + h(V)$$

$g(V)$  - глубина вершины  $V$

$h(V)$  - эвристическая оценка состояния  $V$



## Редукция задач

**Редукция задачи** - анализ исходной задачи с целью выделения такого набора подзадач, решение которых означает решение исходной задачи.


Каждая из выделенных подзадач:

- в общем случае является более простой, чем исходная
- может быть решена каким-либо методом, в том числе и с использованием пространства состояний

Продолжение процесса последовательным выделением подзадач из возникающих задач до получения элементарных задач, решение которых уже известно.

### Задача о пирамиде (ханойской башне)

**Задача о пирамиде (ханойской башне)**



**Формализация в пространстве состояний:**

Состояние задачи – (место первого диска, место второго диска, место третьего диска)

Начальное состояние – (A, A, A)

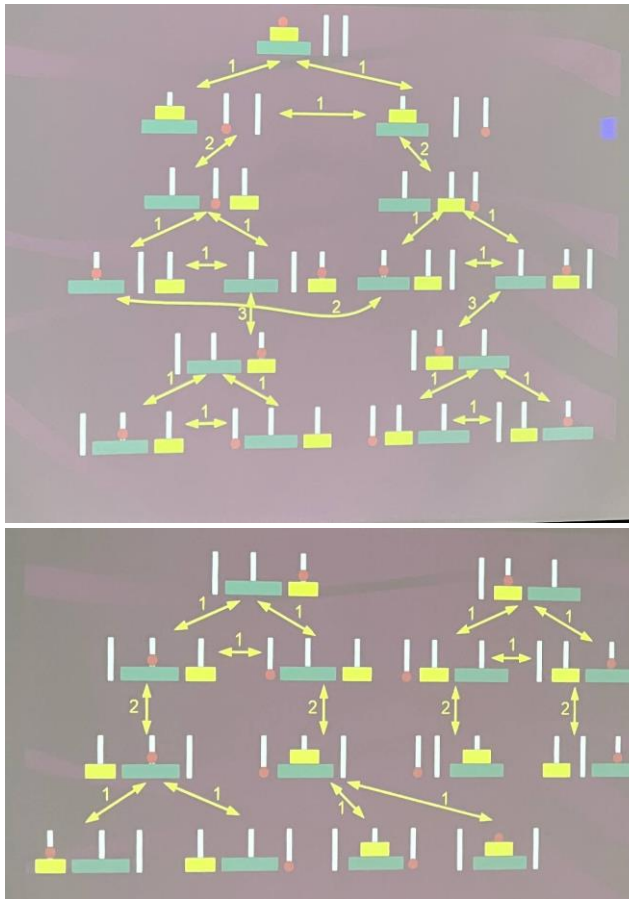
Целевое – (C, C, C)

Формализация задач в пространстве состояний:

Состояние задачи - (место первого диска, место второго диска, место третьего диска)

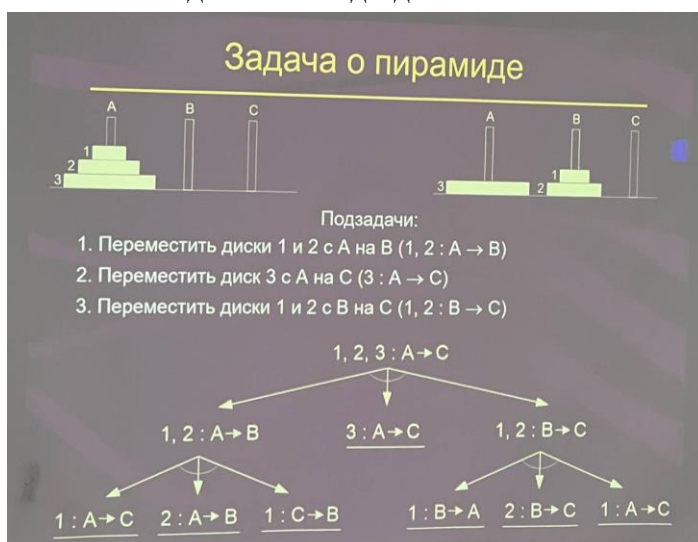
Начальное состояние - (A, A, A)

Целевое - (C, C, C)



Для перемещения всей пирамиды необходимо переложить нижний диск, а это возможно, если располагающаяся над ним часть пирамиды перенесена на другой стержень.

Процесс редукции схематически представляется в виде дерева. Вершины дерева соответствуют подзадачам, листья дерева (потомки) - элементарным задачам, дуги связывают задачи с ее подзадачами.



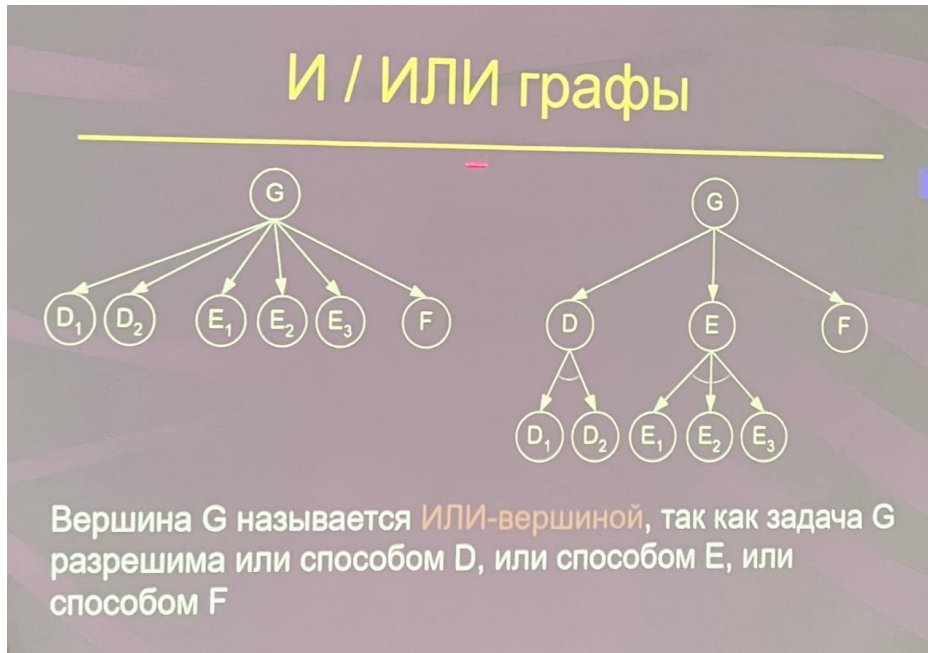
Получаем пространство подзадач

**Формализация задач в рамках подхода, основанного на редукции задач**



1. Нужно выделить форму описания подзадач.
2. Определить множество элементарных задач.
3. Необходимо определить множество операторов.

## И/ИЛИ графы



Вершина G называется ИЛИ-вершиной, так как задача G разрешима или способом D, или способом E, или способом F.

Вершина D называется И-вершиной, поскольку требует решения всех подчиненных задач.

### Разрешимость вершин

Цель поиска на И/ИЛИ-графе - показать, что разрешима исходная задача, то есть начальная вершина.

**Разрешающий граф** - это подграф И/ИЛИ-графа, состоящий только из разрешимых вершин и доказывающий разрешимость начальной вершины.

- заключительные вершины разрешимы, так как они соответствуют элементарным задачам
- ИЛИ-вершина разрешима тогда и только тогда, когда разрешима по крайней мере одна из ее дочерних вершин
- И-вершина разрешима тогда и только тогда, когда разрешима каждая из ее дочерних вершин

## Класс игр двух лиц с полной информацией

В играх участвуют два игрока, которые поочередно делают свои ходы.

В каждый момент игры, каждому игроку известно все что произошло в игре к этому моменту и все что может быть сделано в данный момент.

Игра заканчивается либо выигрышем одного игрока, что означает проигрыш другого (игры с нулевой суммой) или ничьей.

Для изучения игровых стратегий в данном классе игр используется метод редукции (см. прошлую лекцию).

Задача поиска выигрышной стратегии одного из игроков из некоторой конфигурации игры.

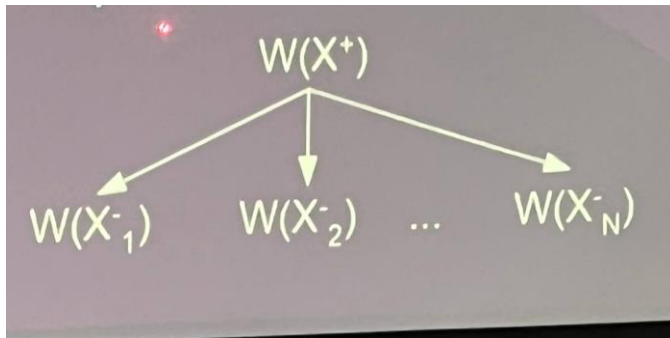
$X^S$  - некоторая конфигурация игры ( $X^+$ ,  $X^-$ )

$W(X^S)$  - задача доказать то, что ПЛЮС может выиграть из конфигурации  $X^S$

$V(X^S)$  - задача доказать то, что МИНУС может выиграть из конфигурации  $X^S$

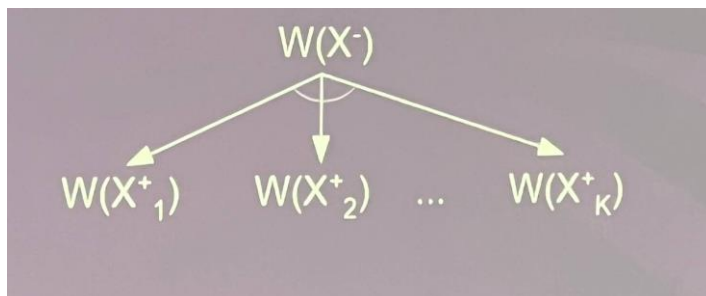
### Операторы редукции:

- Если в конфигурации  $X^S$  имеется  $N$  допустимых ходов, приводящих к конфигурациям  $X_1^-, X_2^-, \dots, X_N^-$ , то для решения задачи  $W(X^+)$  необходимо решить по крайней мере одну из подзадач  $W(X_i^-)$



- Если в конфигурации  $X^-$  имеется  $K$  допустимых ходов, приводящих к конфигурациям  $X_1^+, X_2^+, \dots, X_K^+$ , то для решения задачи  $W(X^-)$  требуется решить каждую из подзадач  $W(X_i^+)$ .

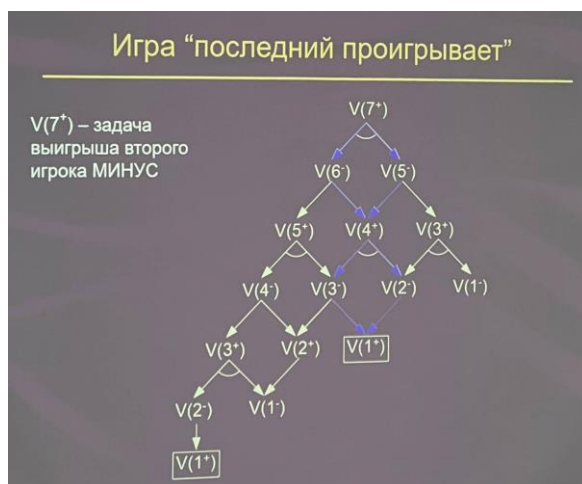
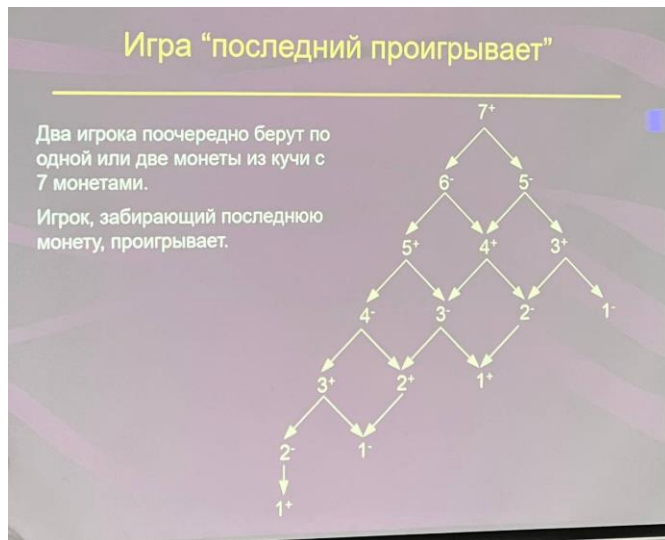
Последовательное применение данной схемы редукции к исходной конфигурации порождает И/ИЛИ дерево (дерево игры). Для конфигурации где ход принадлежит  $X^+$  появляется ИЛИ-вершина, для позиции  $X^-$  появляется ИЛИ-вершина. Построение дерева заканчивается на конфигурациях, выигрышных для одного из игроков.





**Цель построения** игрового дерева - получение решающего поддеревья для задачи  $W(X^S)$ , показывающего, как игрок ПЛЮС может выиграть игру из позиции  $X^S$  независимо от ответов противника.

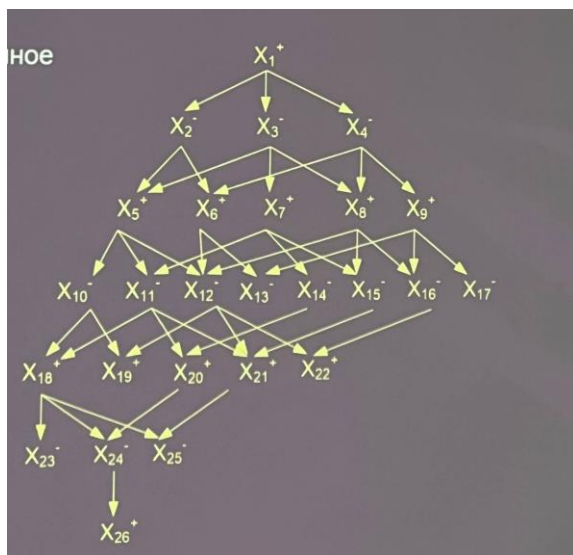
Решающее дерево заканчивается на позициях, выигрышных для ПЛЮСА, и содержит полную стратегию достижения им выигрыша: для каждого возможного продолжения игры, выбранного противником, в дереве есть ответный ход, приводящий к победе.





### Выделение решающего поддерева

1) Построить полное дерево игры

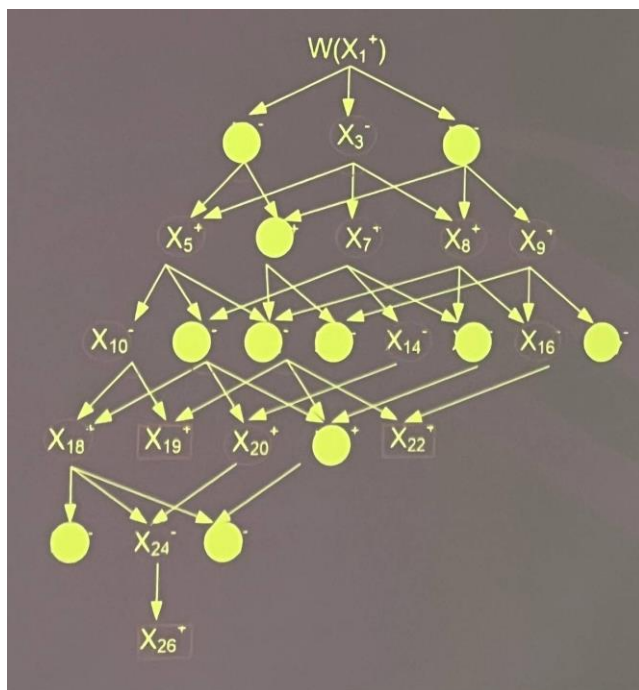


2) Выделить на дереве концевые вершины, которые соответствуют выигрышу ПЛЮСА.

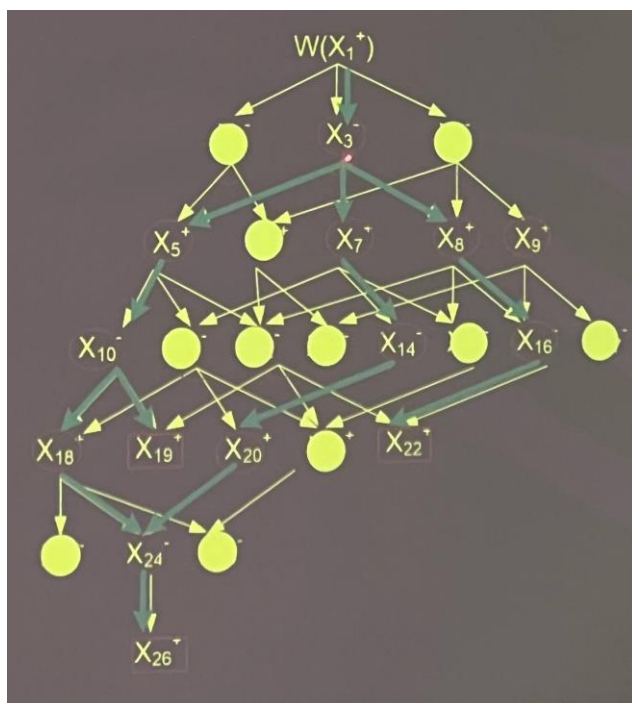
Среди промежуточных вершин дугой указать И-вершины ходов противника.

3) Определить, какие вершины являются выигрышными конфигурациями для игрока ПЛЮС.

ИЛИ-вершина выигрышна тогда и только и тогда, когда выигрышна хотя б одна из ее дочерних вершин. И-вершина выигрышна тогда и только тогда, когда выигрышны все ее дочерние вершины.



4) Выделить на дереве решающее дерево.



### Минимаксная процедура

В большинстве игр построить полное решающее дерево не представляется возможным, поэтому вместо задачи поиска выигрышной стратегии решается более простая задача.

Поиск для заданной позиции игры достаточно хорошего первого хода.

Для этого, просматривается лишь часть игрового дерева, построенного от исходной конфигурации.

В построенном дереве вершины оцениваются и по этим оценкам выделяется наилучший первый ход.

Для оценивания концевых вершин (там где заканчивается дерево по условиям ограничения построения) используется оценочная функция, а для оценивания остальных вершин используется минимаксная процедура.

Оценочная функция дает числовое значение, показывающее достоинство данной позиции.

Задачи выигрыша игрока ПЛЮС:

- статическая оценочная функция положительна в конфигурациях, где имеет преимущества игрок ПЛЮС
- отрицательна в конфигурациях, где имеет преимущества МИНУС
- близка к нулю в позициях, не дающих преимущества ни одному из игроков

$$E(P) = \begin{cases} +\infty, & \text{если } P - \text{позиция выигрыша игрока ПЛЮС} \\ -\infty, & \text{если } P - \text{позиция выигрыша игрока МИНУС} \\ M(P) - O(P), & \text{иначе} \end{cases}$$

o		
x	x	

$E(P) = 4 - 1 = 3$

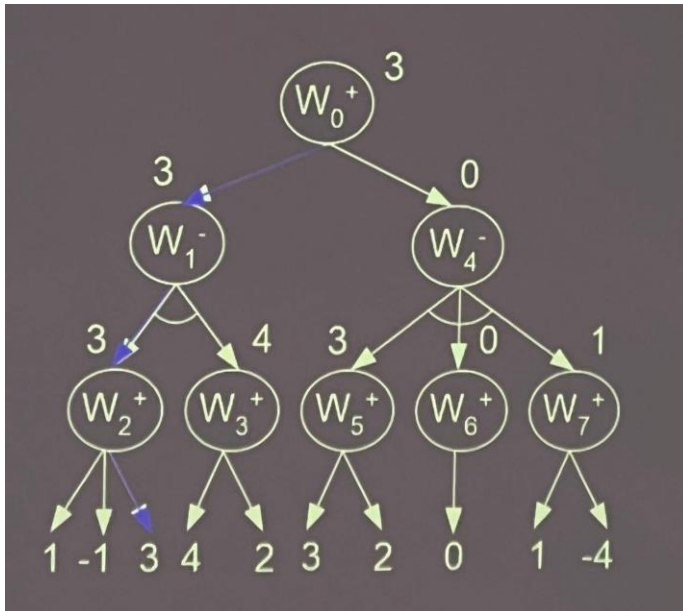
	o	
	x	
	x	
o	x	

$E(P) = 4 - 2 = 2$

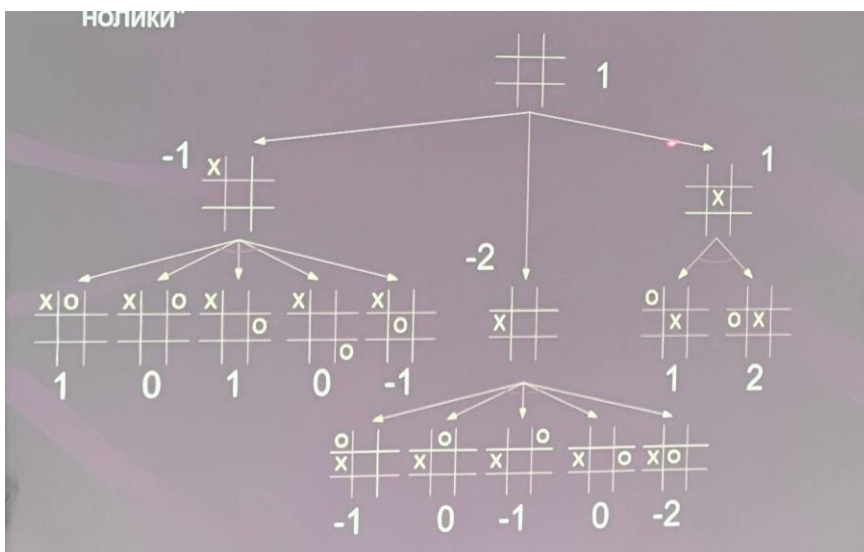
Минимаксный принцип для задачи выигрыша игрока ПЛЮС:

- если ход делает игрок ПЛЮС, то ИЛИ-вершине дерева игры приписывается оценка, равная максимуму оценок ее дочерних вершин
- если ход делает игрок МИНУС, то И-вершине игрового дерева приписывается оценка, равная минимуму оценок ее дочерних вершин

1. Дерево игры строится одним из известных алгоритмов перебора от исходной позиции S до глубины N
2. Для всех концевых вершин дерева вычисляется оценочная функция
3. В соответствии с минимаксным принципом вычисляются оценки всех остальных вершин: сначала оценки вершин, родительских для концевых, затем родительские для этих родительских вершин и т.д. снизу вверх по дереву поиска до начальной вершины S
4. Среди вершин, дочерних к начальной, выбирается вершины с наибольшей оценкой. Ход, который к ней ведет, и есть искомый наилучший ход в игровой конфигурации S



Минимаксно-оптимальная игра, при которой каждый из игроков всегда выбирает наилучший для себя ход



Двухуровневый минимакс для первого хода игры в “крестики-нолики”

**Лекция 6. 20.03.25 (Была в виде семинара)**

**Лекция 7. 27.03.25**

**Модели представления знаний:**

- логические модели
- продукционные модели
- сетевые модели
- фреймовые модели

Математическая логика является формальным языком, т.е. для любой последовательности символов можно сказать, удовлетворяет ли она правилам конструирования выражений в этом языке

В формальной логике разрабатываются методы правильных рассуждений. Правильные рассуждения - это цепочка умозаключений логически последовательных форм.

Формальная система - совокупность абстрактных объектов, в которой представлены правила оперирования множеством символов без учета их смыслового содержания.

Любая формальная система определяется 4 составляющими.

Формальная система  $\langle T, P, A, B \rangle$  :

Алфавит  $T$  - конечное множество символов: константы, переменные и операторы

$P$  - процедура построения формул (ППФ) из элементов алфавита

$A$  - множество формул, называемых аксиомами

$B$  - множество правил вывода, которые позволяют получать из некоторого конечного множества ППФ другое множество формул

Каждая выведенная формула называется теоремой, а используемая при этом последовательность применения правил вывода называется доказательство теоремы.

**Интерпретация** - это придание смысла каждому символу формальной системы, т.е. соответствие между символами и объектами реального мира.

## Исчисление высказываний

1. Алфавит:

пропозициональные символы  $p, q, r$  ( иногда  $T$  и  $F$ )

логические операторы

скобки

2. Построение формул:

любой пропозициональный символ и символ истинности - ППФ

если  $A$  и  $B$  - ППФ, то  $\neg A$ ,  $A \& B$ ,  $A \vee B$ ,  $A \rightarrow B$  и  $(A)$  - ППФ

3. Аксиомы:

$A \rightarrow (B \rightarrow A)$

$(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$  и ряд других законов

4. Правило вывода (правило модус поненс):

Если  $A$  и  $A \rightarrow B$  - ППФ, то  $B$  - также ППФ

Интерпретация символов

# Исчисление высказываний

Символ	Интерпретация
$p$	Сегодня четверг
$q$	Время – 18 часов
$t$	У групп ИДБ-22 занятия по ИиЭС

Формула	Интерпретация
$\neg p$	Сегодня не четверг
$p \& q$	Сейчас четверг и 18 часов
$p \vee q$	Сейчас четверг или время – 18 часов
$p \rightarrow t$	Если сегодня четверг, то у группы ИДБ-2 занятия по ИиЭС

## Исчисление предикатов

Предикаты - функция, результатом которой является истина или ложь.

1. Алфавит:

- константы  $a, b, c, \dots$
- переменные символы  $t, u, w, \dots$
- предикатные символы
- функциональные символы
- логические операторы  $\neg, \&, \vee, \rightarrow$
- кванторы существования  $\exists$  и всеобщности  $\forall$
- скобки

2. Построение формул:

- формулы исчисления предикатов образуются аналогично формулам исчисления высказываний
- если  $A$  - ППФ, то  $\exists x A, \forall x A$  - ППФ

3. Аксиомы:

- аксиомы исчисления высказываний
- $\forall x A(x) \rightarrow A(t)$
- $A(t) \rightarrow \exists x A(x)$

4. Правила вывода:

- модус поненс
- если  $B \rightarrow A(x)$  - ППФ, то  $B \rightarrow \forall x A(x)$  - ППФ (правило обобщения)
- если  $A(x) \rightarrow B$  - ППФ, то  $\exists x A(x) \rightarrow B$  - ППФ (правило конкретизации)

$f(x)$  - унарный предикат “студент (Петров)”



$f(x, y)$  - бинарный предикат “зарплата (Петров, 45000)”

$P$  - "по четвергам у групп ИДБ-22 занятия по ИиЭС" занятие (четверг, ИДБ-22, ИиЭС)

$\forall x$  занятие ( $x$ , ИДБ-22, ИиЭС), где  $x$  - день недели

$\exists x$  занятие ( $x$ , ИДБ-22, ИиЭС)

Переменные позволяют составлять обобщенные утверждения относительно классов объектов.

Унификация - поиск подстановок на место переменных

### **Использование исчисления предикатов в ИИ**

Ситуации и цели в задачах разного вида могут быть описаны правильно построенными формулами исчисления предикатов.

Моделирование мира блоков, конструирование алгоритма управления для руки робота.

Любой вопрос, который задается системе, формулируется как теорема (не ебу что дальше).

### **Применение исчисления предикатов в ИИ**

- доказательство теорем в математике и логике
- в интеллектуальных системах извлечения информации из БД

Одним из наиболее частных и очевидных применений исчисления высказываний является извлечение информации из баз данных, то есть ответить на такой запрос, для которого необходимо провести цепочку размышлений (метод редукции)

1. Иванова работает в бухгалтерии
2. Петров руководит бухгалтерией

РАБОТАЕТ (Иванова, Бухгалтерия)

ГЛАВА (Петров, Бухгалтерия)

$(\text{РАБОТАЕТ}(x, y) \ \& \ \text{ГЛАВА}(z, y)) \rightarrow \text{НАЧАЛЬНИК}(z, x)$

Кто является начальником Ивановой?

$(\exists x) \text{НАЧАЛЬНИК}(x, \text{Иванова})$

### **Пространство состояний логической системы**

В качестве формального языка для отображения узлов графа в пространстве состояний можно использовать как исчисление высказываний, так и исчисление предикатов.

Тогда вершины графа соответствуют символам алфавита, а для создания дуг используются правила вывода модус поненс.

## Пример. Исчисление высказываний

Пример. Исчисление высказываний

$q \rightarrow p$	$s \rightarrow r$	$s$
$r \rightarrow p$	$t \rightarrow r$	$t$
$v \rightarrow q$	$s \rightarrow u$	

```
graph BT; v --> q; q --> p; t --> r; s --> r; s --> u; style t fill:#fff,stroke:#000; style s fill:#fff,stroke:#000;
```

После построения пространства состояний методами (в глубину, в ширину) поиска можно решить ряд задач, например, является ли конкретное выражение логическим следствием из исходного набора утверждений.

## И/ИЛИ графы

И / ИЛИ графы

$(q \ \& \ r) \rightarrow p$

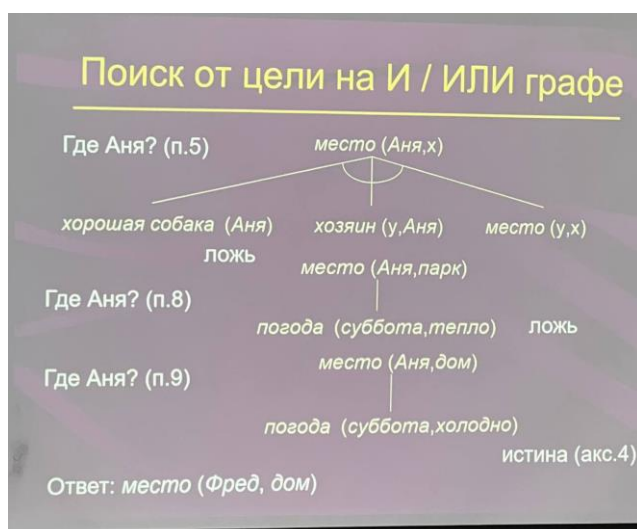
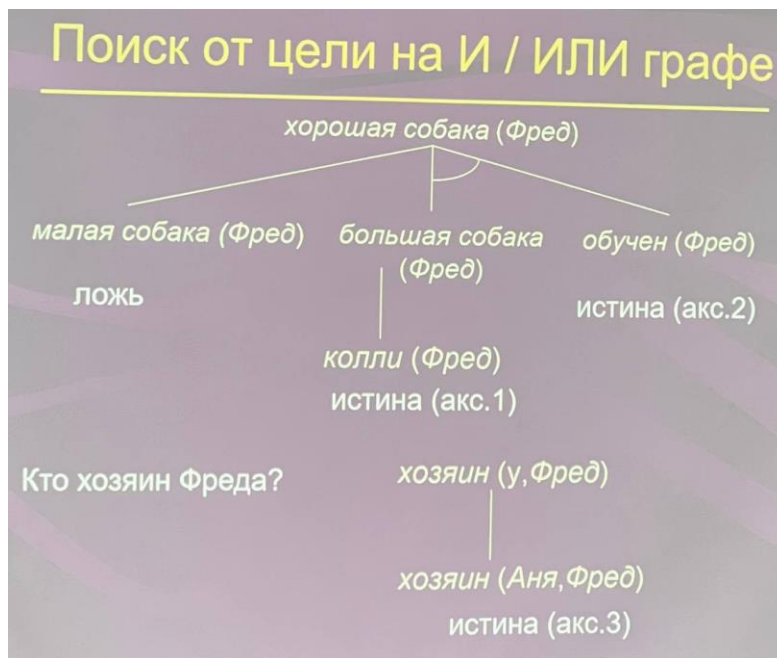
$(q \ \vee \ r) \rightarrow p$

И / ИЛИ графы

$(a \ \vee \ b) \rightarrow d$	$b$
$(b \ \& \ c) \rightarrow e$	$c$
$(d \ \& \ e) \rightarrow h$	
$h \rightarrow g$	
$(a \ \vee \ h) \rightarrow f$	

**Задача.** У девочки Ани есть обученная собака, породы Колли, по кличке Фред. Хорошие собаки находятся рядом с хозяевами. К хорошим собакам относят маленьких собак или обученных собак больших пород. В теплые субботние дни Аня гуляет в парке, а в холодные сидит дома. Где сейчас Фред, если известно, что сегодня холодная суббота?

1. колли (Фред)
2. обучен (Фред)
3. хозяин (Аня, Фред)
4. погода (суббота, холодно)
5.  $\forall (x, y, z) [ \text{хорошая собака} (x) \ \& \ \text{хозяин} (y, x) \ \& \ \text{место} (y, z) ] \rightarrow \text{место} (x, z)$
6.  $\forall x [ \text{малая собака} (x) \vee \text{большая собака} (x) \ \& \ \text{обучен} (x) \rightarrow \text{хорошая собака} (x) ]$
7.  $\forall x \text{ колли} (x) \rightarrow \text{большая собака} (x)$
8. погода (суббота, тепло)  $\rightarrow$  место (Аня, парк)
9. погода (суббота, холодно)  $\rightarrow$  место (Аня, дом)



**Лекция 8. 03.04.25 (Была кр на лекции)**

**Лекция 9. 10.04.25**

## Продукционные модели

### 1. Набор продукционных правил (продукций), образующих базу знаний.

$A \Rightarrow B$  - часть знаний, необходимых для решения задачи. (A - условная часть продукции, B - часть действия).

1. знак логического следования B из истинного A.
2. A описывает некоторое условие, необходимое для совершения действия B.
3. часть B описывает следующий шаг решения задачи.

Набор из семи продукционных правил:

(R1)	$A \Rightarrow E$	H, K	
(R2)	$B \Rightarrow D$		$H \Rightarrow A$ (R3)
(R3)	$H \Rightarrow A$		$A \Rightarrow E$ (R1)
(R4)	$E \& G \Rightarrow C$		$E \& K \Rightarrow B$ (R5)
(R5)	$E \& K \Rightarrow B$		$B \Rightarrow D$ (R2)
(R6)	$D \& E \& K \Rightarrow C$		$D \& E \& K \Rightarrow C$ (R6)
(R7)	$G \& K \& F \Rightarrow A$		

Продукции:

- Детерминированные - при выполнении условной части A, часть действия B выполняется обязательно.
  - ❖ Однозначные
  - ❖ Альтернативные - в правой части указываются альтернативные возможности выбора, которые оцениваются специальными весами. (Если A, то чаще всего надо делать B1, реже B2, в самых крайних случаях B3)
- Недетерминированные - при выполнении условной части A, часть B может как выполняться, так и не выполняться.

Интерпретация: если A, то возможно B.

Если A, то с вероятностью P реализовать B.

Термин продукции был введен начиная с 1974г. Амер. логик Эмиль Леон Пост

Метод продукции используется:

- для определения языков и формальных грамматик
- в экспертных системах
- в языках логического программирования

**2. Рабочее пространство** (рабочая память, база фактов) содержит описание текущего состояния задачи.

**3. Интерпретатор** (система управления):

- 1) Рабочая память инициализируется начальным описанием задачи.

- 2) Текущее состояние решения задачи представляется набором фактов в рабочей памяти.
- 3) Факты сопоставляются с условиями правил, что порождает конфликтное подмножество правил вывода.
- 4) Разрешение конфликта - выбор и выполнение одной из продукций конфликтного множества.
- 5) Цикл управления повторяется для измененной рабочей памяти.
- 6) Процесс заканчивается, если содержимое рабочей памяти не соответствует ни одному условию.

Чистая или классическая продукционная модель не имеет механизма выхода из тупиков, а современные продукционные системы содержат механизмы возврата в предыдущее рабочее состояние.

### Стратегии разрешения конфликтов

В процессе разрешения конфликта, из конфликтного множества выбирается одно правило, но, если интеллектуальная система реализована на ЭВМ с параллельной архитектурой, то из конфликтного множества можно выбрать несколько правил, то есть сколько параллельных ветвей поддерживает ЭВМ.

**Стратегии разрешения конфликтов**

Набор продукций: 1.  $ba \Rightarrow ab$     2.  $ca \Rightarrow ac$     3.  $cb \Rightarrow bc$

Работа продукционной системы:

Шаг	Рабочая память	Конфликтное множество	Применено правило
0	<i>cbaca</i>	1, 2, 3	1
1	<i>cabca</i>	2	2
2	<i>acbca</i>	2, 3	2
3	<i>acbac</i>	1, 3	1
4	<i>acabc</i>	2	2
5	<i>aacbc</i>	3	3
6	<i>aabcc</i>	-	стоп

1. Выбор первого подходящего правила
2. Принцип “стопки книг”

Идея: наиболее часто используемая продукция является наиболее полезной

Из конфликтного множества для исполнения выбирается продукция с максимальной частотой использования.

Этот принцип управления лучше использовать в тех случаях, когда частота выполнения учитывается с учетом предыдущей ситуации, в которой это выполнение имело положительный результат.

3. Принцип наиболее длинного условия

$E \& K \Rightarrow B$      $D \& E \& K \Rightarrow C$

Идея: правила, относящиеся к узкому классу ситуаций, важнее правил, относящихся к широкому классу ситуаций.

#### 4. Принцип приоритетного выбора

Идея: введение статических или динамических приоритетов на продукции.

2 типа выполнения систем продукции:

- прямой - поиск идет от левых частей продукции
- обратный - поиск идет от изначально заданных В, по которым определяются необходимые для В значения А, которые, в свою очередь, отождествляются с правыми частями продукции в системе

(в тесте будет вопрос, чаще выбирают 2 тип)

### Преимущества и недостатки продукционных систем

Преимущества:

1. Знания могут быть записаны в виде продукции
2. Системы продукции являются модульными (добавление новых правил и удаление не приводит к изменению оставшихся)
3. Способность к самообъяснению (система легко прослеживает цепочку правил которые привели к нахождению решения)
4. Естественный параллелизм в системе продукции

Недостатки:

1. Трудность составления продукционного правила, соответствующего элементу знания (предметная область должна быть изучена и определены термины).
2. Трудность записи правила (связано с тем, что у правила есть один единственный формат) (слишком громоздкая левая часть).
3. Сложность проверки непротиворечивости системы продукции.
4. Трудности при проверке корректности работы системы (проверка на всех возможных данных не представляется возможным) (если число продукции достигает 1000, то маловероятно что она будет функционировать во всех случаях).

**Пример продукционной системы**

Игра в 8

Продукции для перемещения пустой клетки (ПК):

ПК не возле левой границы	⇒ переместить ее влево
ПК не возле правой границы	⇒ переместить ее вправо
ПК не возле верхней границы	⇒ переместить ее вверх
ПК не возле нижней границы	⇒ переместить ее вниз

Целевое состояние ⇒ останов

### Сетевые модели

$H = \langle I, C1, C2, \dots, Cn, O \rangle$

I - множество информационных единиц



$C_1, C_2, \dots, C_n$  - множество типов связей

Отображение  $O$  задает между информационными единицами входящими в  $I$ , связи из заданного набора связей.

В зависимости от типов связей различают:

- классифицирующие сети (используют  $C_1$  (отношение структуризации/отношение вычисления)).
- функциональные сети и т.д. (одни информационные единицы вычисляются через другие информационные единицы).

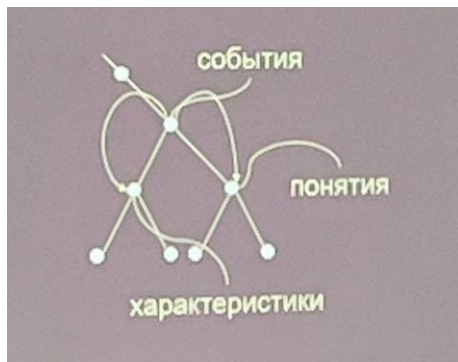
Если в сетевой модели допускаются связи различного типа, то ее обычно называют семантической сетью.

Семантическая сеть - структура для представления знаний в виде узлов, соединенных дугами.

В системах ИИ семантические сети используются:

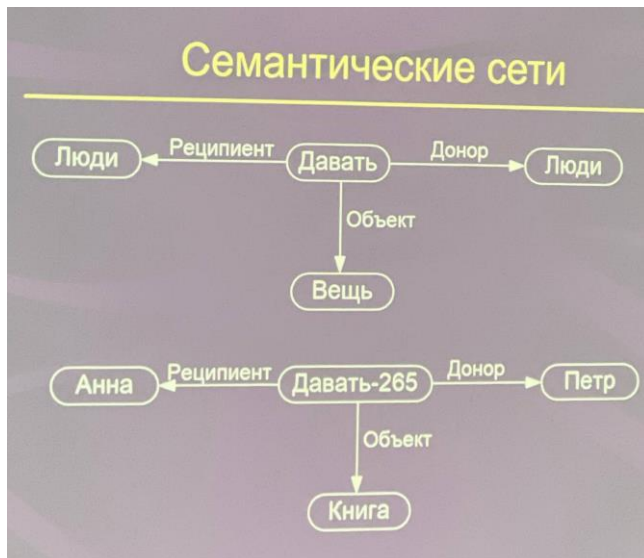
- для представления смысла выражений естественного языка человека.
- в качестве языка-посредника для систем машинного перевода
- для изучения процессов обучения, запоминания и рассуждений

Начиная с конца 50х годов были созданы и применены на практике десятки вариантов семантических сетей



1. Узлы семантических сетей представляют собой концепты (понятия) предметов, событий, состояний.
2. Различные узлы одного концепта относятся к различным значениям, если оно не помечено, что они относятся к одному концепту.
3. Дуги семантических сетей создают отношения между узлами-концептами (метки над дугами указывают на тип отношения).
4. Некоторые отношения между концептами представляют собой лингвистические падежи, другие означают временные, пространственные, логические отношения и отношения между отдельными предложениями.
5. Концепты организованы по уровням в соответствии со степенью обобщенности.





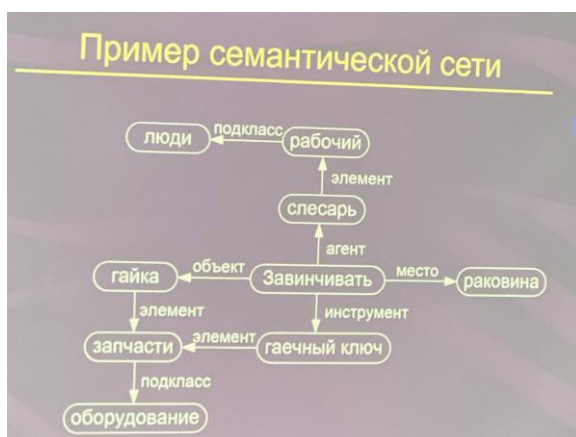
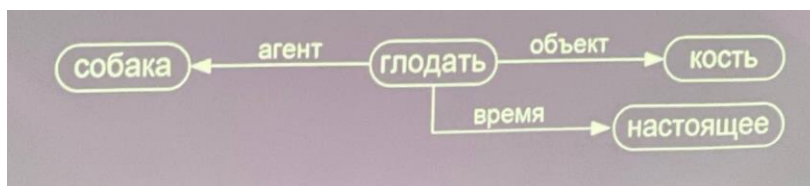
### Семантические сети

Начиная с 1973 года была подчеркнута необходимость стандартизации отношений с учетом семантики отношений.

Связи определяются ролью существительных и групп существительных в действии, описываемом предложением.

- агент
- объект
- инструмент
- место
- время

Падежный фрейм - структура, представляющая предложение набором узлов, один из которых соответствует глаголу, а остальные связанные с ним узлы представляют других участников действия.



**Классификация семантических сетей:**

1. По количеству типов отношений
  - однородные
  - неоднородные
2. По типам отношений:
  - бинарные (2 объекта, во всех примерах были они)
  - n-арные

Наиболее часто в семантических сетях используются следующие отношения:

- связи типа “часть-целое” (“класс-подкласс”, “элемент-множество”)
- функциональные связи (“производит”, “влияет”)
- количественные (больше, меньше, равно)
- пространственные (далеко от, близко от, за, под)
- временные (раньше, позже, в течение)
- атрибутивные связи (иметь свойство, иметь значение)
- логические связи (и, или, не) и др.

Проблема поиска решения в БЗ типа семантической сети сводится к задаче поиска фрагмента сети, соответствующая поставленному вопросу.

## Лекция 10. 17.04.25

### Система фреймов

Фрейм (frame - каркас, рамка), Минский, 70-е гг.

“Фрейм - это единица представления знаний, запомненная в прошлом, детали которой при необходимости могут быть изменены согласно текущей ситуации”

Структуру фрейма автор наполнил необходимой информацией: 1) об объектах и событиях, которые следует ожидать в данной ситуации; 2) как использовать информацию, имеющуюся во фрейме.

Под фреймом понимается абстрактный образ или ситуация.

Комната - помещение со стенами, полом, потолком, окнами и дверью.

Это описание является минимально возможным (при попытке сделать его еще меньше, оно перестанет определять то понятие, для которого предназначалось).

**Фрейм** - это модель знаний, которая активизируется в определенной ситуации и служит для ее объяснения и предсказания.

**Протофрейм** - жесткая структура информационных единиц. В отличие от моделей других типов, во фреймовых моделях фиксируется жесткая структура.

Фреймы с конкретными значениями слотов (**фреймы-примеры, экземпляры**) создаются для отображения реальных ситуаций на основе поступающих данных.

(Имя фрейма:

Имя слота 1 (значение слота 1)

Имя слота 2 (значение слота 2)

...

Имя слота N (значение слота N))

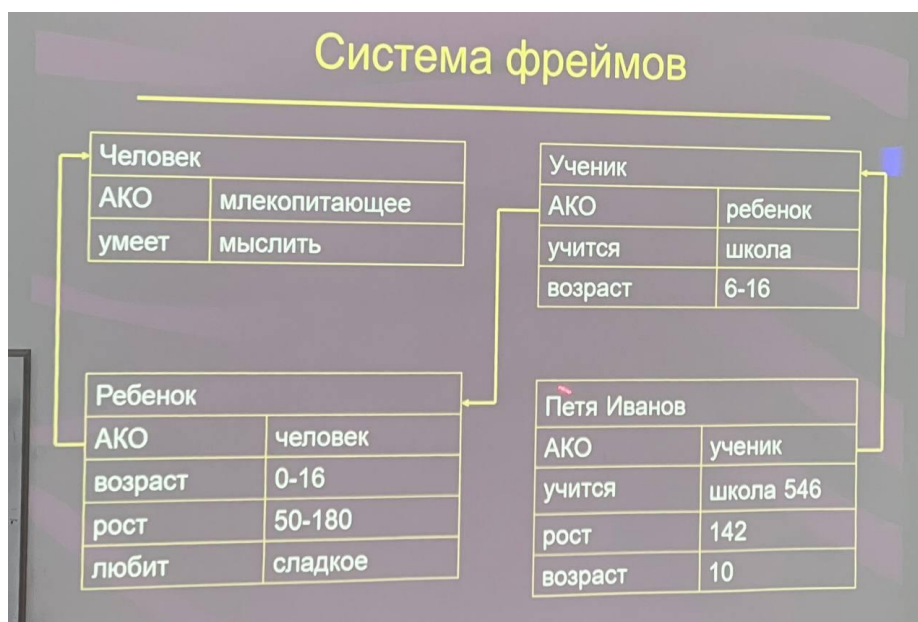
Перемещать		Перемещать	
кто		кто	робот
что		что	блок А
откуда		откуда	стол
куда		куда	блок D
когда		когда	

Значение слота:

- числа или математические соотношения
- тексты на ЕЯ или программы
- правила вывода
- ссылки на слоты этого фрейма или других фреймов
- процедуры (демоны (процедуры, которые активизируются каждый раз, когда происходит доступ к фрейму))

Совокупность фреймов, моделирующая какую-либо предметную область, представляет собой иерархическую структуру, в которой фреймы соединяются с помощью родовидовых связей. На верхнем уровне иерархии находится фрейм, содержащий наиболее общую информацию, истинную для всех остальных фреймов.

Наследование происходит по слоту АКО (A-Kind-Of - это), который указывает на фрейм более высокого уровня иерархии.



Наследование свойств может быть частичным.

Если фреймы и вышестоящие в иерархии фреймы имеют одинаковые названия слотов, то значение слотов фрейма, находящегося ниже в иерархии перекрывают значения слотов, находящиеся выше в иерархии.

Слоты фреймов:

- данные для идентификации фрейма
- взаимосвязь этого фрейма с другими фреймами
- условия на значения (возраст не старше 16 лет)
- значения по умолчанию
- процедурный код

Заполнение ячеек при создании экземпляра фрейма:

- значениями, заданными пользователем
- значениями по умолчанию (могут быть взяты из своего фрейма/значения из вышестоящего в иерархии фрейма-прототипа)
- с помощью процедуры для получения значения

FRL (Frame Representation Language)

KRL (Knowledge Representation Language)

Экспертные системы ANALYST, TRISTAN и МОДИС

## Лекция 11. 24.04.25

Категории источников неопределенности:

- Недостаточно полное знание предметной области
- Недостаточная информация о конкретной ситуации

Теория предметной области может быть неясной или неполной, так как в ней могут использоваться недостаточно четко сформулированные концепции или плохо изученные явления.

Неопределенность предметной области, что правила, даже в простых случаях, дает верный, корректный результат.

Причины недостаточных или ненадежных данных:

- Любой сенсор имеет ограниченную разрешающую способность и не стопроцентную надежность
- При составлении отчетов могут быть допущены ошибки или в них могут попасть недостоверные сведения
- Далеко не всегда можно получить полные ответы на поставленные вопросы
- Не всегда есть возможность быстро получить необходимые данные, когда ситуация требует принятия быстрого решения (пагода времени)

Эксперты пользуются неточными методами по двум причинам:

- Точных методов не существуют
- Если точные методы существуют, но не могут применяться на практике из-за: отсутствия необходимого объема данных или невозможности их получить по соображениям стоимости, рисков или времени.

Утверждение Мак-Карти и Хейеса: «Теория вероятности не является адекватным инструментом для решения задач представления неопределенности знаний и данных»

Аргументы в пользу такого мнения:

- теория вероятности не дает ответа на вопрос, как сочетать вероятности с количественными данными
- назначение вероятности событиям требует информации, которой мы просто не имеем
- как количественно оценивать такие понятия, как "в большинстве случаев", "в редких случаях", "старый" или "высокий"
- обновление вероятностных оценок обходится очень дорого, т.к. требует большого объема вычислений

Байесовские рассуждения используются в распознавание образов и задачах классификации.

Теория Байеса обеспечивает вычисление сложных вероятностей на основе случайной выводки событий.

**Байесовские рассуждения**

Если  $A$  и  $B$  независимы, то вероятность их комбинации:  
вероятность  $(A \& B)$  = вероятность  $(A)$  \* вероятность  $(B)$

**Априорная** (безусловная) вероятность события – вероятность, присвоенная событию при отсутствии знания, поддерживающего его наступление  
 $P(\text{событие})$

**Апостериорная** (условная) вероятность события – вероятность события при некотором заданном основании  
 $P(\text{событие} | \text{основание})$

Условная вероятность события  $d$  при данном  $s$  – это вероятность того, что событие  $d$  наступит при условии, что наступило событие  $s$

$$P(d | s) = \frac{P(d \& s)}{P(s)} \quad P(s | d) = \frac{P(d \& s)}{P(d)}$$

$$P(d \& s) = P(s | d) * P(d) \quad P(d | s) = \frac{P(s | d) * P(d)}{P(s)}$$

в квадрате формула это теорема Байеса

**Байесовские рассуждения**

Симптом  $s$  = боль в груди      Заболевание  $d$  = инфаркт

Условная вероятность того, что пациент страдает заболеванием  $d$ , если у него обнаружен симптом  $s$ , равна:

$$P(d | s) = \frac{P(s | d) * P(d)}{P(s)}$$

$P(\text{боль в груди} | \text{инфаркт})$  проще вычислить, чем  $P(\text{инфаркт} | \text{боль в груди})$

Множество заболеваний  $D$  состоит из  $m$  элементов  
Множество симптомов  $S$  состоит из  $n$  элементов  
 $m \times n$   $P(s | d) + (m+n) P(s) \text{ и } P(d)$

Правило Байеса с комплексными симптомами:

$$P(d | s_1 \& s_2 \& \dots \& s_k) = \frac{P(d) * P(s_1 \& s_2 \& \dots \& s_k | d)}{P(s_1 \& s_2 \& \dots \& s_k)}$$

Нередкие ситуации при которых человек эксперт, оценив вероятности некоторого события значений 0,7, предполагает, что это событие истинно и не учитывает, что это событие может быть ложным.

## Стэндфордская теория фактора уверенности

В теории вероятностей сумма вероятностей отношения и его отрицания должна равняться 1

Оценка достоверности "Порш – быстрая машина" = 0.9

Мера уверенности (доверия) – неформальная оценка, которую человек-эксперт добавляет к заключению

$MB(H)$  – мера уверенности в гипотезе  $H$

$MD(H)$  – мера недостоверности гипотезы  $H$

$0 < MB(H) < 1$ , если  $MD(H) = 0$

$0 < MD(H) < 1$ , если  $MB(H) = 0$

$$CF(H) = MB(H) - MD(H)$$

$CF \rightarrow 1$  – усиление доверия к гипотезе

$CF \rightarrow -1$  – усиление отрицания гипотезы

$CF \rightarrow 0$  – доказательств в пользу гипотезы и против нее сбалансированы

## Лекция 12. 15.05.25

### Экспертные системы

Экспертные системы ориентированы на тиражирование опыта высокопрофессиональных специалистов в тех областях деятельности человека, где качество решений существенно зависит от уровня экспертизы или компетенции экспертов.

Экспертные системы (ЭС) особенно эффективны там, где знания устоялись, являются общечеловеческим достоянием, а решения, которые принимает ЭС мало зависят от того, в какой стране они применяются (медицина, химия, биология).

Особенности, присущие экспертным системам:

- предназначены для пользователей, сфера деятельности которых далека от ИИ (предназначена для помощи обычному пользователю в повседневных вещах)
- возможность получения с помощью ЭС профессиональных неформализованных знаний
- способны давать при необходимости пояснения и разъяснения

Условия определения экспертной системы:

- система должна обладать знаниями
- знания должны быть сконцентрированы на определенную предметную область
- из этих знаний должно непосредственно вытекать решение проблем

ГОСТ 33707-2016:

**Экспертная система** - это система, основанная на знаниях, которая обеспечивает решение задач в конкретной области или в сфере приложений путем логических выводов, извлекаемых из базы знаний, разработанной на основании человеческого опыта.



В 60-х Эдвард Файгенбаум

Первые ЭС		
Название ЭС	Характеристики	
<b>DENDRAL</b> (1965 г.) Эдвард Файгенбаум Джошуа Ледерберг Брюс Бученен	Область применения	химия
	Назначение	определение возможной структуры молекулы на основе химической формулы и масс-спектрограммы
	Способ представления знаний	правила логики ("если - то")
	Инструментарий разработки	язык INTERLISP

Первые ЭС		
Название ЭС	Характеристики	
<b>MYCIN</b> (сер. 1970-х г.) Эдвард Шортлайф	Область применения	медицина
	Назначение	диагностика и лечение инфекционных заболеваний крови
	Способ представления знаний	факты и правила вида «Условие -> Действие» с коэффициентами уверенности, метаправила
	Инструментарий разработки	язык LISP

EMYCIN (Empty MYCIN) ориентирована на решение задач диагностики.

Новые ЭС: NEOMICYN и PUFF.

Первые ЭС		
Название ЭС	Характеристики	
<b>PROSPECTOR</b> 1974–1983 гг.	Область применения	геология
	Назначение	поиск (предсказание) месторождений на основе анализа геологических данных
	Способ представления знаний	нечеткая математика (нечеткие множества, мера оценки истинности данного утверждения)
	Инструментарий разработки	язык INTERLISP

Инструментальная система KAS - оболочка ЭС.

## Классификация ЭС

Классификация ЭС			Название	Предназначение	Типичные задачи
Интерпретирующие системы	Формирование описания ситуаций по результатам наблюдений или данным, получаемым от сенсоров	Распознавание образов, определение химической структуры вещества	Диагностические системы	Обнаружение источников неисправностей по результатам наблюдений за поведением контролируемой системы	Задачи в медицине, механике, электронике и т.д.
Прогнозирующие системы	Логический анализ возможных последствий заданных ситуаций или событий	Предсказание погоды и прогноз ситуаций на финансовых рынках	Системы проектирования	Структурный синтез конфигурации объектов (компонентов проектируемой системы) при заданных ограничениях	Синтез электронных схем, компоновка архитектурных планов

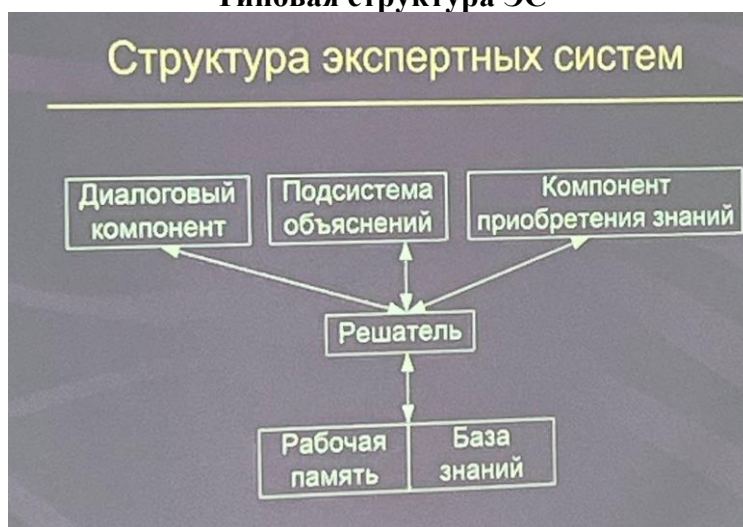
### Отличия ЭС от прикладных программ

Основное отличие ЭС от прикладной программы состоит в том, что ЭС манипулируют знаниями, а обычные - данными.

Экспертная система должна обладать:

- компетентностью (ЭС должна достигать такого же уровня профессионализма, что и человек-эксперт)
  - достигать экспертного уровня суждений
  - быть умелой (применять знания эффективно и быстро)
  - обладать работоспособностью - умением рассуждать в случае некорректных данных или неполных наборов правил
- символьным рассуждением (при решении задач ЭС вместо выполнения математических вычислений, манипулирует символами)
- глубиной (ЭС эффективно решает задачи предметной области, если правил много и эти правила сложные)
- самосознанием (ЭС имеют знания, позволяющие им рассуждать об их собственных действиях) (Пользователи больше доверяют результатам; Ускоряется развитие системы, поскольку ее легче тестировать (отлаживать); Предположения, положенные в основу работы системы, становятся явными)

### Типовая структура ЭС



**Рабочая память** - предназначена для исходных и промежуточных данных решаемой задачи.

**База знаний** - предназначена для хранения долгосрочных данных, описывающих (?) и правила для преобразования этих данных.

**Решатель** (интерпретатор) - используя исходные данные из рабочей памяти и знания из базы знаний, формирует такую последовательность правил, которая будучи примененная к исходным данным приведет к решению задачи.

**Компонент приобретения знаний** - автоматизирует процесс наполнения ЭС знаниями.

**Подсистема объяснений** (объяснительный компонент) - как система получила или не получила результат и какие знания она использовала.

**Диалоговый компонент** - вывод

### **Разработчики ЭС**

1. Эксперт - определяет знания, характеризующие предметную область.
2. Инженер по знаниям (специалист по разработке ЭС)
  - a. помогает эксперту выявить и структурировать знания, необходимые для работы ЭС
  - b. осуществляет выбор инструментальных средств (ИС) и определяет способ представления знаний в этом ИС
  - c. выделяет и программирует традиционными средствами стандартные функции
  - d. все те методы и технологии, которыми пользуется инженер по знаниям называются технологиями или методами инженерии знаний.

Языки программирования : LISP, PROLOG

Экспертные системы-оболочки (ЭСО): KEE, CENTAUR, G2 и GDA, CLIPS, AT\_ТЕХНОЛОГИЯ.

3. Программист - разрабатывает инструментальные средства, которые содержат основные компоненты ЭС и осуществляет сопряжение с той средой.

### **Режимы работы ЭС:**

Приобретение знаний

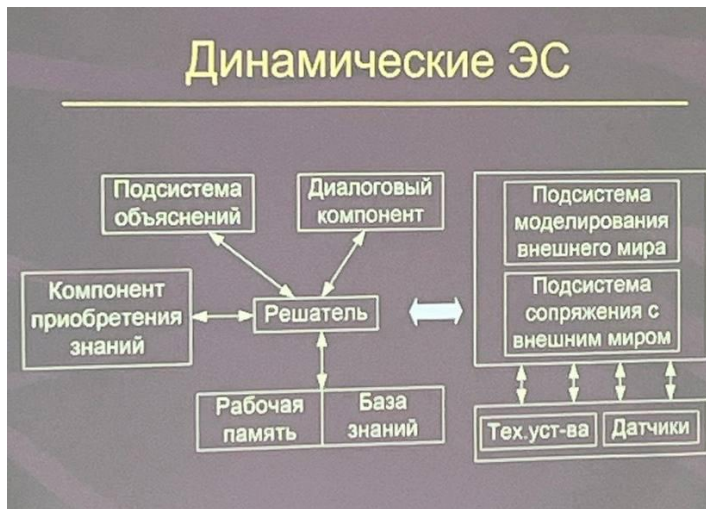
Решение задачи (консультация)

Данные о задаче после обработки их диалоговым компонентом поступают в рабочую память.

Решатель на основе входных данных, общих данных о проблемной области и правил из БЗ формирует последовательность операций для решения задачи, а затем исполняет ее.

**Статические ЭС** используются в тех приложениях, где можно не учитывать изменения окружающего мира, происходящие за время решения задачи.

**Динамические ЭС**



**Лекция 13. 22.05.25**

### Обобщенная структура ЭС



### Трудности при разработке ЭС

- 1) проблема извлечения знаний экспертов
- 2) проблема формализации знаний экспертов
- 3) проблема нехватки времени у эксперта
- 4) правила, формализованные экспертом, не дают необходимой точности
- 5) недостаток ресурсов
- 6) несоответствие инструментальных средств решаемой задачи

Использовать ЭС следует только тогда, когда разработка ЭС оправдана, и методы инженерии знаний соответствуют своим задачам.

### Оправданность разработки ЭС:

- решение задачи принесет значительный эффект
- использование человека-эксперта невозможно
- необходимость выполнять экспертизу одновременно, в разных местах
- если надо решать задачи в окружении, враждебном для человека
- когда при передаче информации эксперту происходит недопустимая потеря информации или потеря времени

## **Разработка экспертных систем**

Характеристики решаемой задачи:

- 1) может быть решена посредством манипуляции с символами, а не с числами
- 2) должна иметь эвристическую, а не алгоритмическую природу
- 3) должна быть достаточно сложна, чтобы оправдать затраты на разработку ЭС

Концепция “быстрого прототипа”:

- решающий типичные задачи конкретного приложения
- время и трудоемкость разработки должны быть незначительны
- в случае неудачи, может потребоваться либо разработка нового прототипа, либо разработчики приходят к выводу о неприменимости методов инженерии знаний для данного приложения
- когда прототип начинает решать все задачи, происходит преобразование прототипа в конечный продукт

## **Этапы разработки ЭС:**

1. Идентификация
2. Концептуализация
3. Формализация
4. Выполнение
5. Тестирование
6. Опытная эксплуатация

## **Этап идентификации**

Связан с осмыслением задач, которые предстоит решить будущей экспертной системе.

Результаты этапа:

- идентификация задачи
- определение участников процесса проектирования и их ролей
- выявление ресурсов и целей

**Идентификация задачи** - составление неформального описания, в котором указываются:

- общие характеристики задачи
- подзадачи, выделяемые внутри данной задачи
- ключевые понятия (объекты), их входные (выходные) данные
- вид решения, а также знания о решаемой задаче

## **Типичные ресурсы при проектировании ЭС:**

- источники знаний
- время разработки
- вычислительные средства
- объем финансирования

#### **Источники знаний:**

- для эксперта - предшествующий опыт, книги, известные примеры решения задач
- для инженера по знаниям - опыт в решении аналогичных задач, методы представления и манипулирования знаниями, программные инструментальные средства
- при недостаточном финансировании предпочтения отдаются не разработке новой ЭС, а доработке существующей

#### **Этап концептуализации**

Содержательный анализ проблемной области, выявление используемых понятий и их взаимосвязи, определение методов решения задачи.

Создание модели предметной области, включающей основные концепции и отношения.

Подходы к процессу построения модели:

- Признаковый (атрибутивный) подход - полученная от экспертов информации представлена в виде троек: объект-атрибут-значение атрибута
- Структурный (когнитивный подход) - выделение элементов предметной области и их взаимосвязи

#### **Этап формализации**

Выражение ключевых понятий и отношений на формальном языке (существующем или новом):

- определяются способы представления декларативных и процедурных знаний (фреймы, сценарии, семантические сети и т.д.)
- осуществление их представления
- определение способов манипулирования знаниями (логический вывод, аналитическая модель, статическая модель и др.)
- формирование описания решения задачи ЭС на предложенном формальном языке

#### **Этап выполнения**

Создание одного или нескольких прототипов требуемой решаемой задачи

Разработка прототипа состоит либо в программировании компонентов или выборе уже известных компонентов и наполнение их знаниями.

#### **Этап тестирования**

На данном этапе производится оценка выбранного способа представления знаний в ЭС.

**Источники неудач в работе системы:**



- тестовые примеры
- ввод-вывод
- правила вывода
- управляющие стратегии

#### **Тестовые примеры могут:**

- оказаться вне предметной области ЭС
- быть слишком однородным и не охватывать предметную область

**Ввод-вывод** - данные, приобретенные от эксперта, и заключения, предъявленные ЭС при объяснении.

#### **Ошибки при вводе:**

- эксперту задавались неправильные вопросы
- собрана не вся необходимая информация
- вопросы системы трудны для понимания, многозначны и не соответствуют знаниям пользователя
- входной язык неудобен для пользователя

#### **Входные данные могут оказаться непонятны, так как:**

- их может быть слишком много или слишком мало
- неудачная организация, упорядоченность заключений
- неподходящий пользователю уровень абстракций с непонятной ему лексикой

#### **В правилах вывода:**

- отсутствует учет взаимозависимости правил
- ошибочность, противоречивость и неполнота правил

#### **Этап опытной эксплуатации**

Проверяется пригодность экспертной системы для конечного пользователя.

Пригодность определяется удобством работы и полезностью.

**Полезность ЭС** - способность в ходе диалога определять потребности пользователя, выявлять и устранять причины неудач в работе, а также решать поставленные задачи.

#### **Удобство работы с ЭС:**

- естественность взаимодействия с ней
- гибкость ЭС
- устойчивость системы к ошибкам

По результатам тестирования и опытной эксплуатации создается конечный продукт, пригодный для промышленного использования.