Теория принятия решений
Введение

Лекция 1

Определение ИИ

- Одно из определений ИИ: Системы, которые думают подобно людям.
- Новое захватывающее направление работ по созданию компьютеров, способных думать, машин, обладающих разумом, в полном и буквальном смысле этого слова.
- Автоматизация действий, которые мы ассоциируем с человеческим мышлением, т.е. таких действий как принятие решений, решение задач, обучение.

Тест Тьюринга

Тест Тьюринга – проверка того, способен ли компьютер действовать подобно человеку. Компьютер должен обладать следующими возможностями:

1. **Средствами обработки текстов** на естественных языках, позволяющий свободно общаться с компьютером на языке.
2. **Средства представления знаний,** с помощью которых компьютер может писать в память то, что он узнает или прочитает.
3. Средства автоматического **формирования логических выводов**, обеспечивающих возможность использовать хранимую информацию для поиска ответов на вопросы
4. Средствами **машинного обучения**, которые позволяют приспособиться к новым обстоятельствам, обнаруживать стандартные ситуации.
5. **Машинное зрение** для восприятия объектов
6. **Средствами робототехники** для манипулирования объектами и перемещения в пространстве.

Как мыслить по-человечески?

- Когнитивное моделирование.
- Запись результатов проводимых компьютером рассуждений совпадает с регистрацией рассуждений людей.
- Когнитология совместно использует компьютерные модели и экспериментальные методы, взятые из психологии для разработки точных и обоснованных теорий работы человеческого мозга.

Как мыслить рационально?

- Подход, основанный на использовании законов мышления.
- Научное направление – логика определяет законы мышления, которые управляют работой ума. Логика имеет систему утверждений о предметах и об отношениях между ними, формальный язык. Компьютеры могут решать любую разрешимую задачу.

Как мыслить рационально?

Подход, основанный на использовании рациональных агентов. Агентом называется все, что действует.

Компьютерные агенты имеют свойства:

- Функционируют под автономном управлением.
- Воспринимают среду. Адаптируются к изменению среды.
- Способны достигать поставленной цели.
- Рациональный агент способен достичь наилучшего результата в условиях неопределенности. Поэтому он способен логическим путем прийти к правильному решению и действовать в соответствии с принятым решением.

Развитие искусственного интеллекта

- **Философы** (начиная с 400 года до н.э.) заложили основы искусственного интеллекта, сформулировали идеи, что мозг напоминает машину, он оперирует знаниями, закодированными на внутреннем языке, мышление позволяет выбирать наилучшие действия.
- **Математики** предложили средства для манипулирования высказываниями, являющимися логически достоверными или не достоверными вероятностными высказываниями. Показали, что представляют собой вычисления, формирование алгоритмов.
- **Экономисты** формализовали теорию принятия решений, максимизирующий ожидаемый выигрыш от принятия решения.

Развитие искусственного интеллекта

- **Психологи** подтвердили идею, что люди могут рассматриваться как машины обработки информации. Лингвисты показали, что процессы использования естественного языка укладываются в эту модель.
- **Компьютерные инженеры** разработали программы для искусственного интеллекта и предоставили высокопроизводительные средства для их реализации.
- **Теория управления** посвящена проектированию устройств, которые действуют оптимально на основе связи со средой.

Теория принятия решений

- Теория решений объединяет **теорию вероятностей** и **теорию полезности**, предоставляет методы принятия решений в условиях неопределенности. В условиях, когда среда характеризуются вероятностными распределениями.
- **Теория решений** используется для экономических задач.
- **Теория игр** используется для экономических задач, в которых действия одного игрока влияют на полезность действий другого.
- **Исследование операций** изучает последовательные решения которые моделируются марковскими процессами принятия решений.
- Исследование операций возникло во время Второй мировой войны при решении задач оптимизации работы радарных установок.

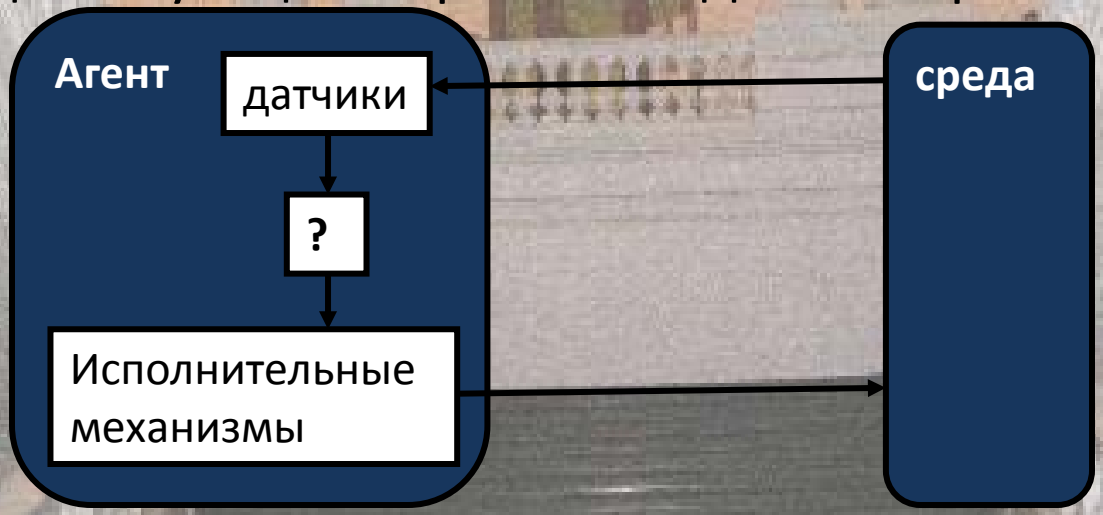
Современное состояние разработок

- **Автономное планирование и составление расписаний.** Программа Remote Agent составила расписание операций для комического аппарата.
- **Ведение игр.** Программа Deep Blue победила чемпиона мира по шахматам.
- **Автономное управление.** Система управления автомобилем NavLab, оснащенная компьютерным зрением, использовалась для проезда более 4500 км по автомобильным дорогам.
- **Диагностика.** Медицинские диагностические программы позволяют устанавливать диагноз и пояснять свои решения. Система дает лучшие решения, чем опытные врачи.
- **Планирование снабжения.** Развернута программа динамического анализа и планирования DART для снабжения военной операции в Персидском заливе.
- **Понимание естественного языка** и решения задач. Программа Porverb решет кроссворды лучше большинства людей.

Интеллектуальные агенты

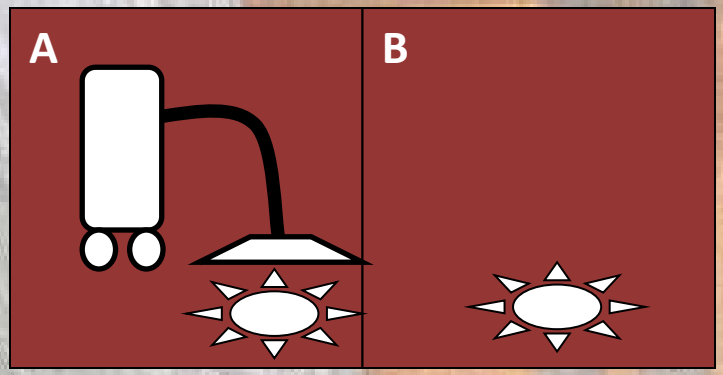
Интеллектуальные агенты

- **Агентом** является объект, воспринимающий окружающую среду с помощью датчиков и действующий на среду с помощью исполнительных механизмов.
- **Функция агента** определяет действие, предпринимаемое агентом в ответ на любую последовательность актов восприятия.
- **Программа агента** – конкретная реализация функции агента, действующего в рамках заданной архитектуры.



Табулированная функция агента

Мир пылесоса, в котором имеется только два местоположения

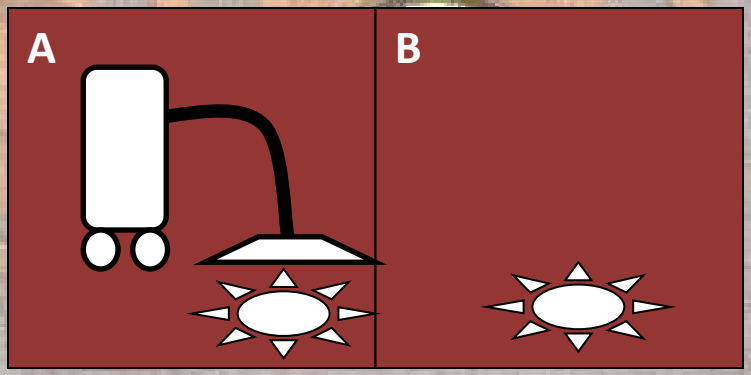


Восприятие	Действие
[A,clean]	Right
[A,dirty]	Suck
[B,clean]	Left
[B,dirty]	Suck
…	
[A,clean] [A,clean]	Right
[A,clean] [B,dirty]	Suck
…	
[A,clean] [A,clean] [A,clean] …	Right
…	
[A,clean] [A,clean] [A,clean] …	Right

Для фрагмента игры в шахматы
Число строк таблицы: 10^{150}
Число атомов во Вселенной: 10^{80}

Показатели производительности пылесоса

- 1. Объем собранного мусора
- 2. Количество чистых квадратов в среднем за все время действия агента
- 3. Количество чистых квадратов и штраф за потребленную электроэнергию



Рациональный агент

- Поведение агента в среде оценивают показатели производительности.
- **Показатели производительности** лучше всего разработать в соответствии с тем, что действительно необходимо добиться в данной среде, а не в соответствии с тем, что думает проектировщик о поведении агента.
- **Рациональный агент** действует так, чтобы максимизировать ожидаемые значения показателей производительности, с учетом полученной к данному моменту времени информацией в результате выполнения последовательности актов восприятия.

Рациональность

- **Рациональный агент** для каждой последовательности измерений должен выбрать действие, которое, как ожидается, максимизирует его показатели производительности, с учетом фактов, полученных с помощью измерений и всех встроенных знаний, которыми он обладает.
- **Факторы :**

1. Последовательность измерений агента, которые получены до настоящего времени
2. Знания агента о среде, известные априори
3. Показатели производительности
4. Действия, которые могут быть выполнены агентом

Всезнание, обучение и автономность

- **Всезнающий агент** априори знает окружающую среду и знает результат всех своих действий. Всезнание невозможно.
- Агент, который основан на априорных знаниях своего разработчика имеет недостаточную автономность.
- **Рациональный агент** должен собирать всю информацию, исследовать ситуацию, выполнять **обучение.** Рациональный агент должен быть автономным.
- **Полностью автономный** агент не имеет априорных знаний, заложенных разработчиком, поэтому он поначалу действует случайным образом.

Определение характера среды

- **Проблемная среда** характеризуется показателями производительности, типом среды, возможностью использования датчиков, возможностью использования исполнительных механизмов.

Проблемная среда автоматического вождения автомобиля:

- **Производительность:** безопасная быстрая езда в рамках правил, максимизация прибыли.
- **Среда:** дороги, другие автомобили, пешеходы.
- **Исполнительные механизмы:** рулевое управление, акселератор, тормоз, …
- **Датчики:** видеокамеры, ультразвуковой дальномер, глобальная система навигации, …

Варианты проблемной среды

- **Полностью наблюдаемая** среда обеспечивает доступ датчиков к полной информации о среде. **Частично наблюдаемая** среда характеризуется неточностью датчиков, отсутствием информации от датчиков.
- **Детерминированная** полностью наблюдаемая среда (мир пылесоса) наиболее благоприятна для агента. **Стохастическая** среда имеет неопределенность (вождение автомобиля).
- **Эпизодическая** среда состоит из отдельных ситуаций, решение для каждой ситуации принимается независимо (контроль деталей). **Последовательная** среда состоит связанных кадров, для которых возможно предсказание (игра в шахматы, вождение такси).

Варианты проблемной среды

- **Статическая** среда не изменяется при выполнении последовательных действий (решение кроссворда). **Динамическая** среда изменяется в процессе работы агента (вождение автомобиля) .
- **Дискретная** среда имеет конечное количество различных состояний (игра в шахматы). **Непрерывная** среда имеет изменяющиеся во времени состояния (непрерывное рулевое управление).
- **Одноагентная** среда содержит одного агента (кроссворды). **Мультиагентная** среда содержит множество конкурирующих агентов (шахматы – **конкурентная** среда) или множество взаимодействующих агентов (футбол – **кооперативная** игра).

Среда для вождения такси

- Стохастическая,
- Частично наблюдаемая,
- Последовательная,
- Динамическая,
- Непрерывная
- Мультиагентная

Структура агентов

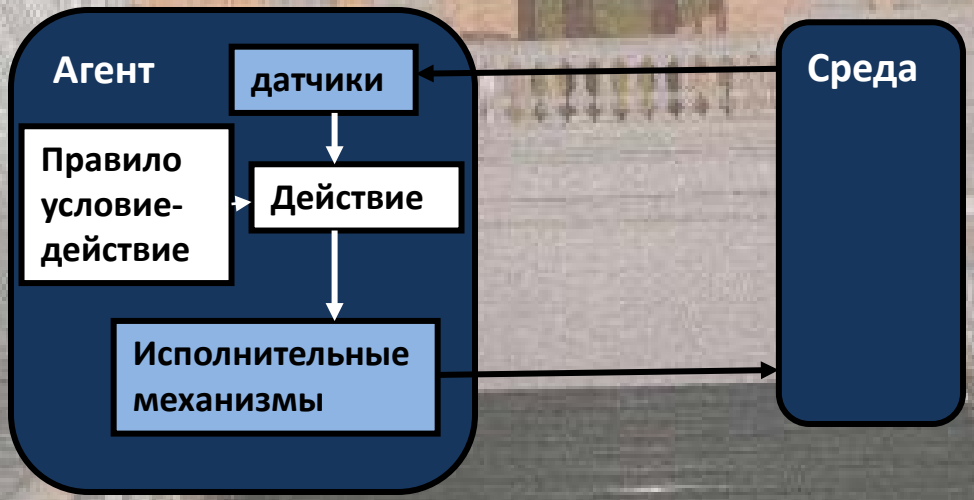
- Таблицы для управления агентами имеют огромное число строк.
- **Агент = Архитектура + Программа**
- **Программа агента** реализует функцию агента.
- Существует множество программ агентов, соответствующих характеру явно воспринимаемой информации, которая используется в процессе принятия решений. Выбор подходящего проекта зависит от характера среды.

Среда для вождения такси

- Стохастическая,
- Частично наблюдаемая,
- Последовательная,
- Динамическая,
- Непрерывная
- Мультиагентная

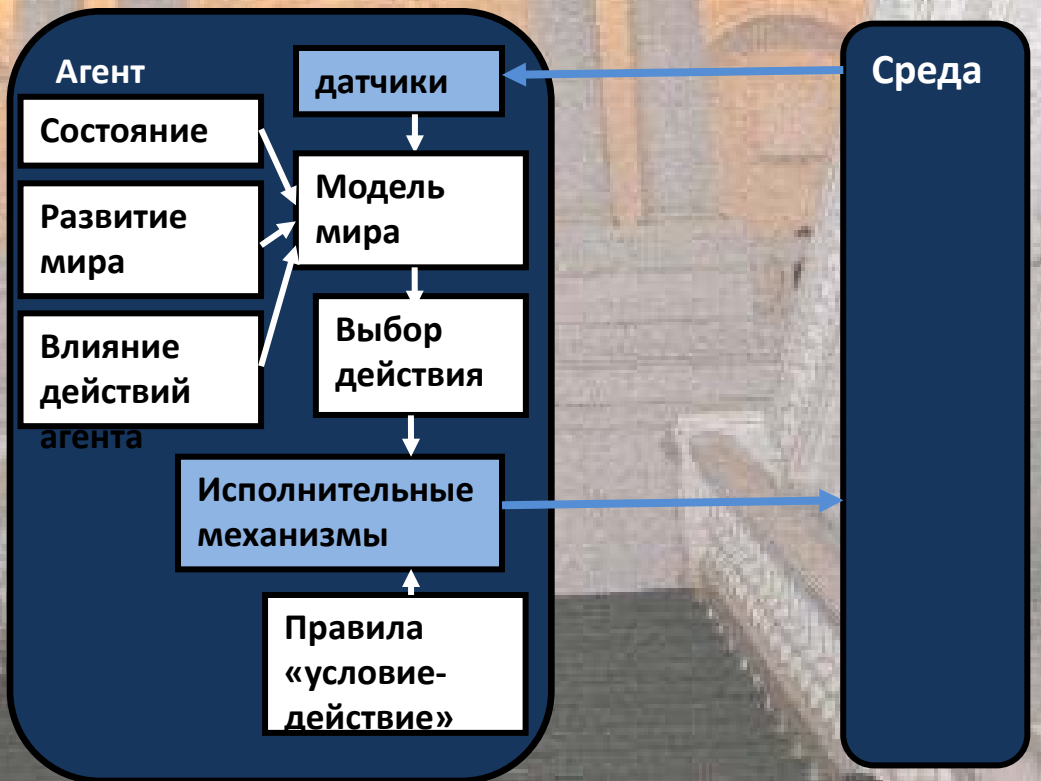
Структура агентов

- **Простые рефлексные агенты** используют принцип «правило-условие-действие».
- `If status=Dirty then return Suck`
- `else If location=A then return Right`
- `else If location=B then return Left`
- Агент работает, если среда является полностью наблюдаемой



Рефлексные агенты, основанные на модели

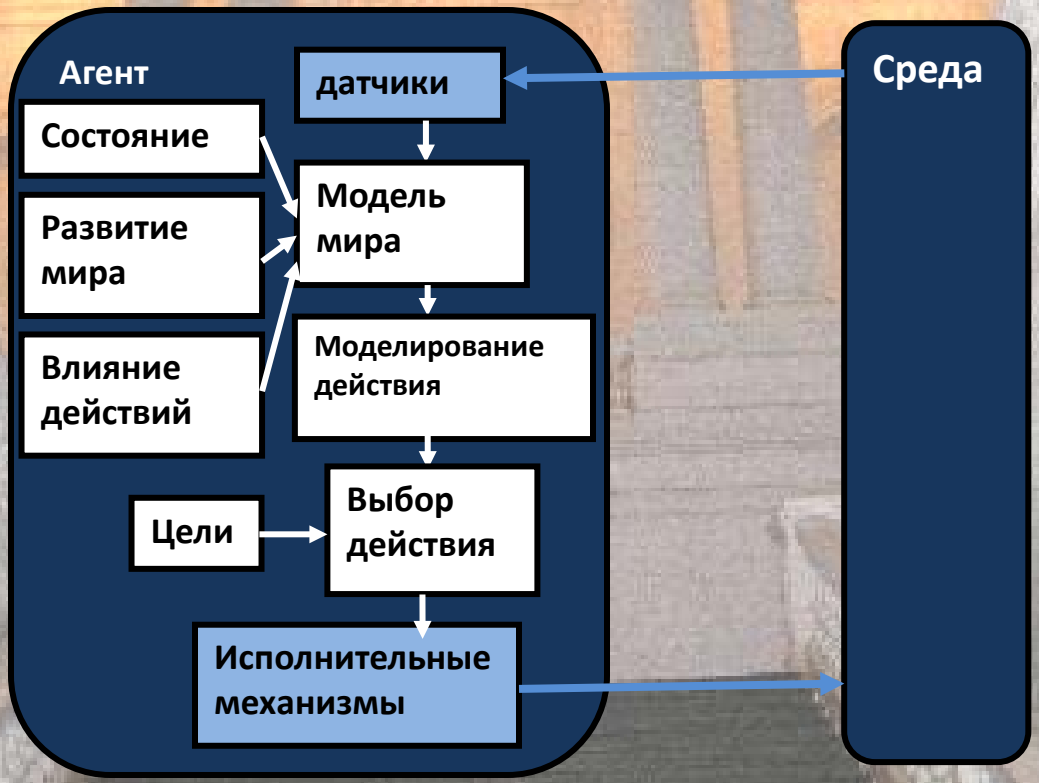
- Основанные на модели рефлексные агенты применимы в **условиях частичной наблюдаемости** среды. Агент должен строить модель для ненаблюдаемых аспектов текущего состояния.
- **При вождении такси:**
- Учет предыдущих кадров видеокамеры для определения необходимости торможения такси.



Агенты, действующие на основе цели

Агенты организуют свои действия так, чтобы достигнуть своих целей.

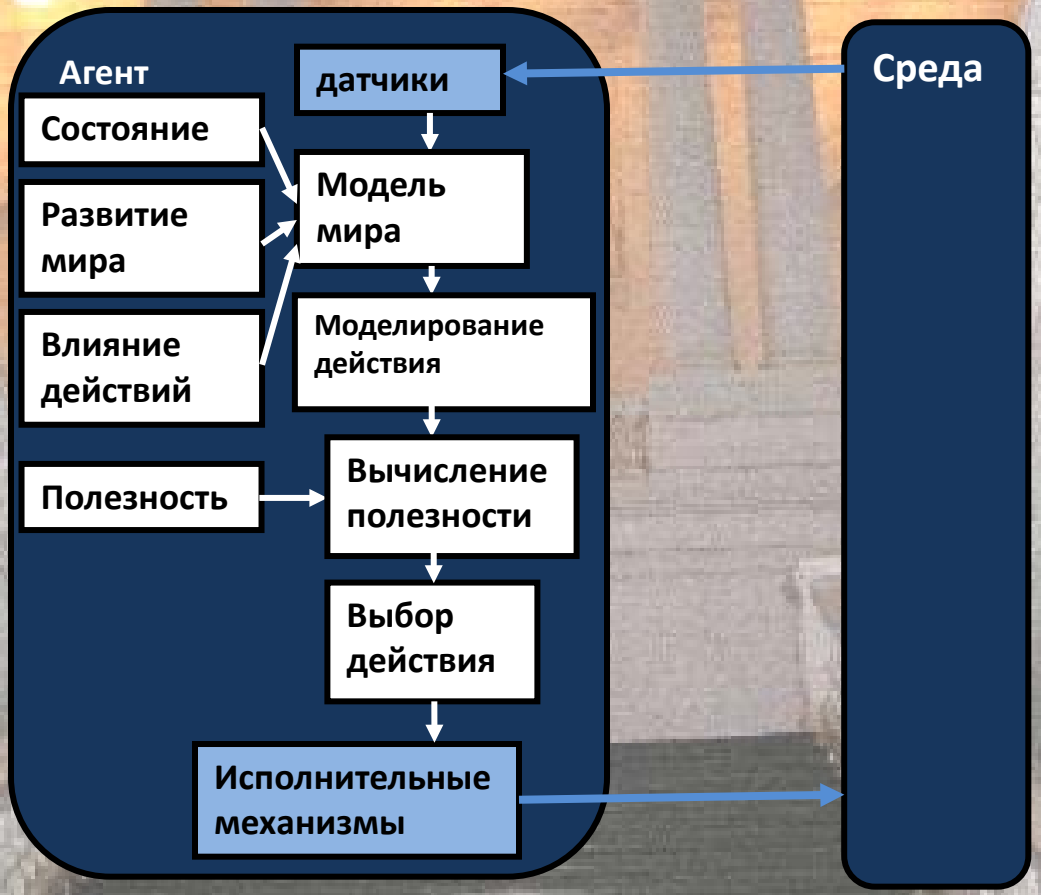
- **При вождении такси:**
- Учет цели – доставка пассажира. Необходимо рассмотреть разные пути движения, выполнить **поиск и планирование.**



Агенты, действующие с учетом полезности

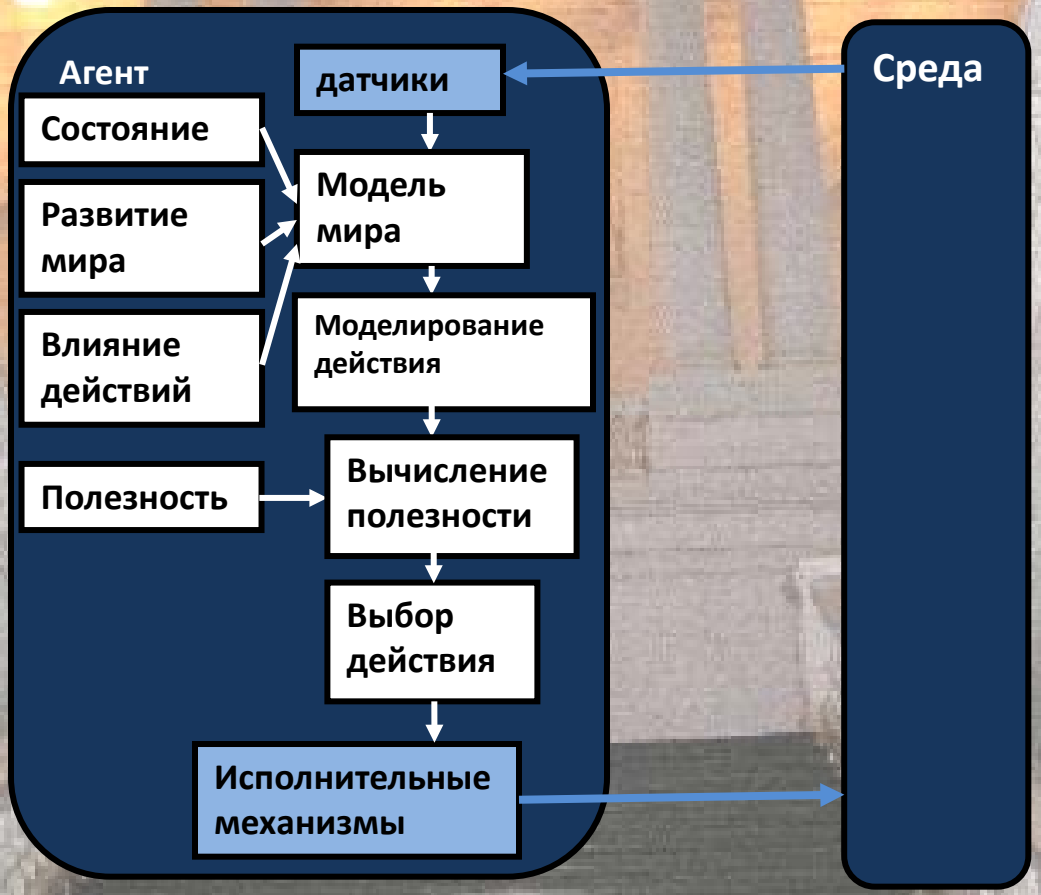
Агенты пытаются максимизировать свою собственную ожидаемую **«полезность».**

Функция полезности отображает последовательность состояний на вещественное число.



Агенты, действующие с учетом полезности

1. Имеются несколько конкурирующих целей (скорость и безопасность). Функция полезности **дает компромисс.**
2. Имеются несколько целей. Функция полезности дает взвешенную оценку **вероятности успеха.**



Обучающиеся агенты

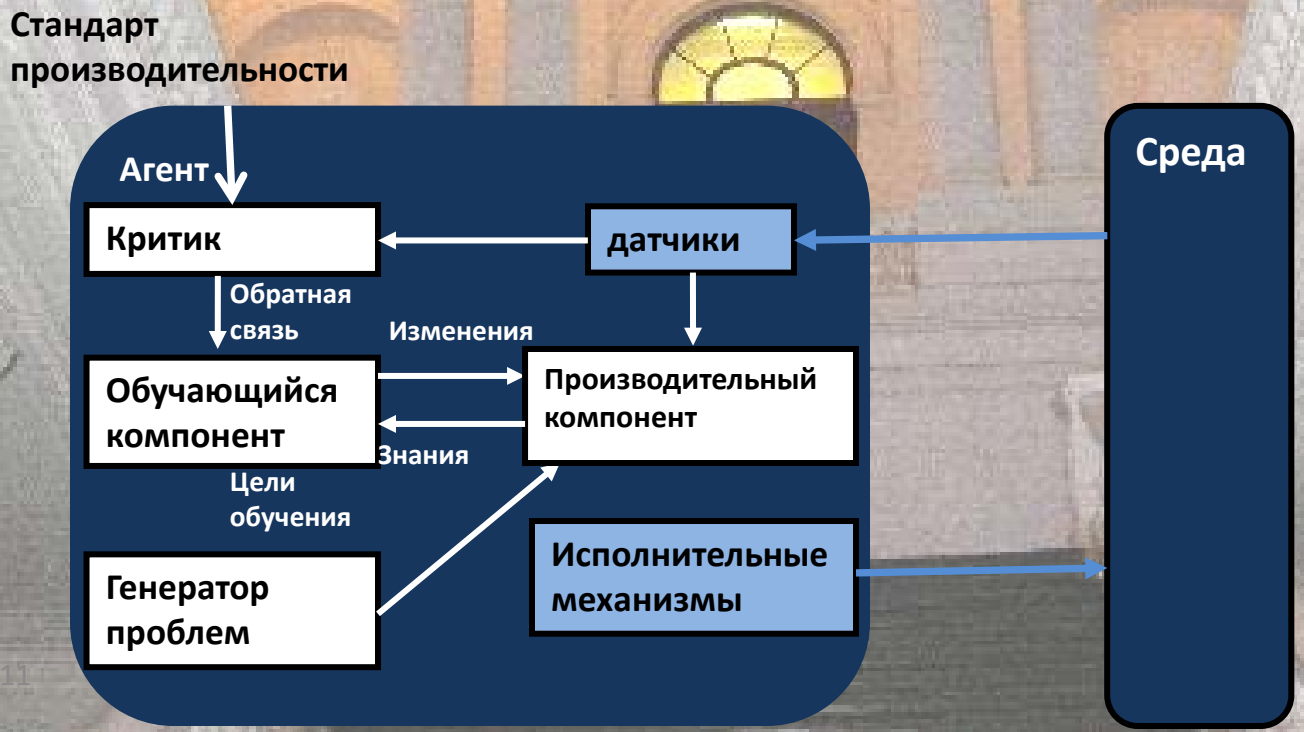
Тьюринг предложил создавать обучающиеся машины.

Обучающий компонент отвечает за внесение усовершенствований.

Производительный компонент обеспечивает выбор внешний действий. Получает измерительную информацию.

Критик оценивает, как действует агент.

Генератор проблем позволяет получить новый информационный опыт выполнив исследование.



Обучающиеся агенты

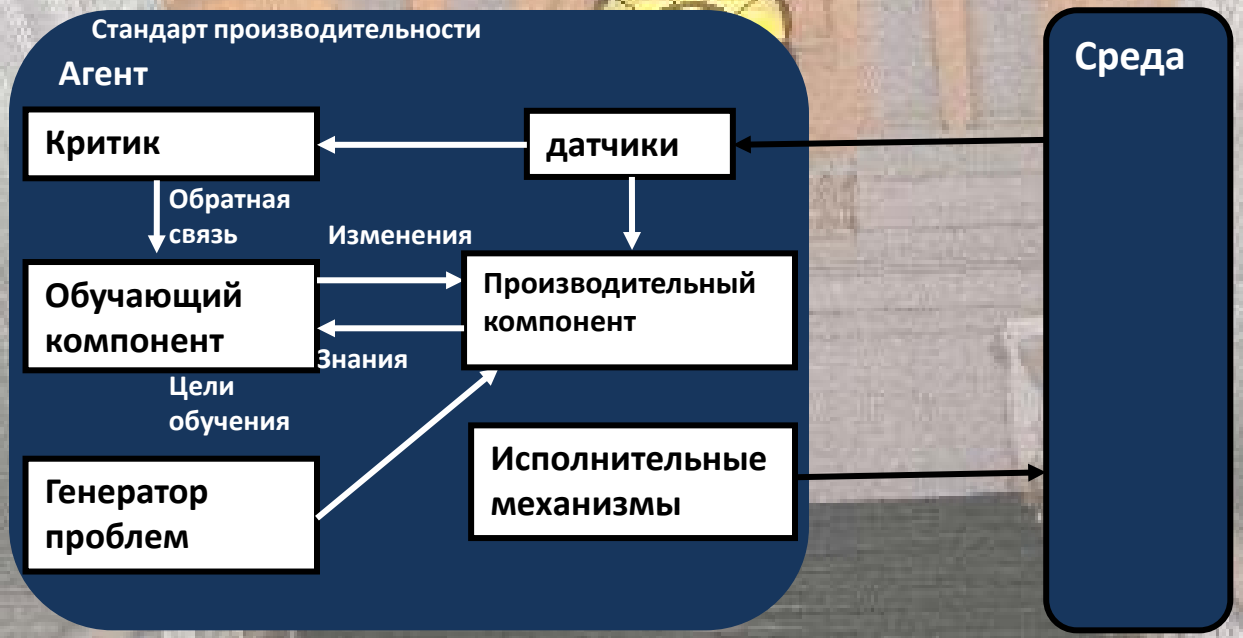
Производительный компонент управляет движением по дорогам.

Критик обнаруживает нарушение правил движения.

Обучающий компонент формулирует новое правило, чтобы избежать неправильного вождения.

Производительный компонент модифицируется.

Генератор проблем выполняет исследование тормозов на мокрой дороге.



Выводы

1. **Агентом** является объект, воспринимающий окружающую среду с помощью датчиков и действующий на среду с помощью исполнительных механизмов.
2. Поведение агента в среде оценивают **показатели производительности. Рациональный агент** действует так, чтобы максимизировать ожидаемые значения показателей производительности, с учетом полученной к данному моменту времени информацией в результате выполнения последовательности измерений.
3. **Проблемная среда** характеризуется показателями производительности, типом среды, возможностью использования датчиков, возможностью использования исполнительных механизмов.
4. **Варианты проблемной среды :**
   - Полностью наблюдаемая и частично наблюдаемая
   - Детерминированная и стохастическая.
   - Эпизодическая и последовательная.
   - Статическая и динамическая.
   - Дискретная и непрерывная.
   - Одноагентная и мультиагентная (конкурентная кооперативная).

Выводы

5. Существует множество **программ агентов,** соответствующих характеру явно воспринимаемой информации, которая используется в процессе принятия решений. Выбор подходящего проекта зависит от характера среды.
6. **Простые рефлексные агенты** используют принцип правило-условие-действие.
7. **Рефлексные агенты, основанные на модели,** поддерживают внутренне состояние, прослеживая аспекты среды, которые не наблюдаются в текущий момент времени.
8. **Агенты, действующие на основе цели,** организуют свои действия так, чтобы достигнуть своих целей.
9. **Агенты, действующие с учетом полезности**, пытаются максимизировать свою собственную ожидаемую «удовлетворенность».
10. **Обучающиеся агенты** позволяют совершенствовать производительный компонент

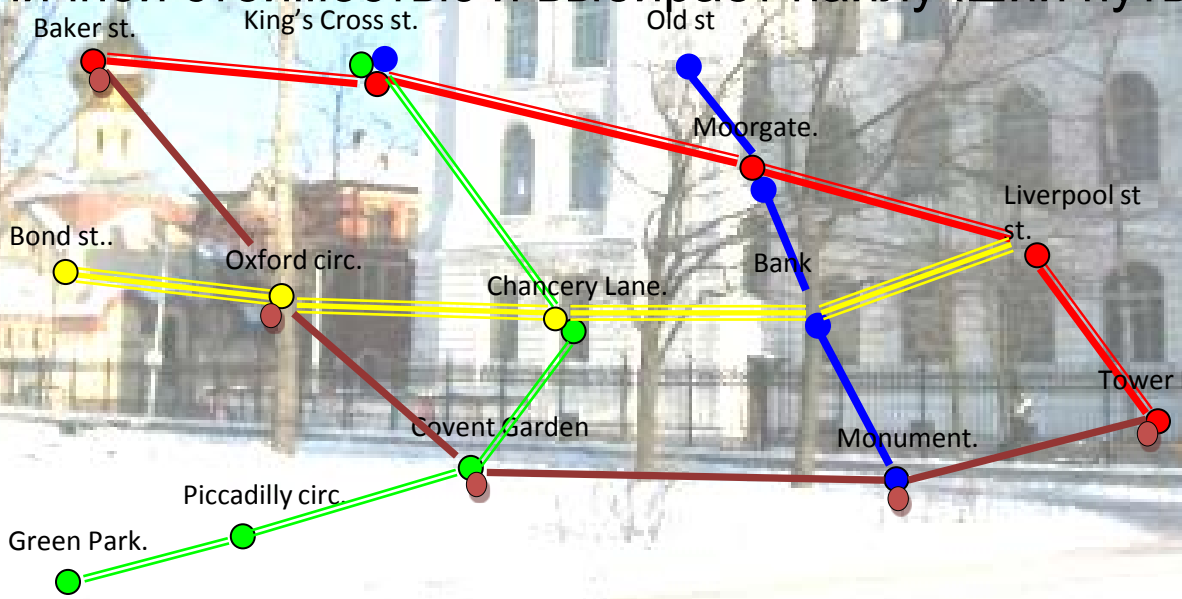
Решение задач посредством поиска

Решение посредством поиска

- Каким образом агент, решающий задачи, может найти последовательность действий, позволяющую достичь целей.
- Алгоритмы называются **не информированными**, если в них не используется какая-либо дополнительная информация кроме определения задачи.
- Алгоритмы называются **информированными,** если в них не используется какая-либо информация о том, как следует искать решение.

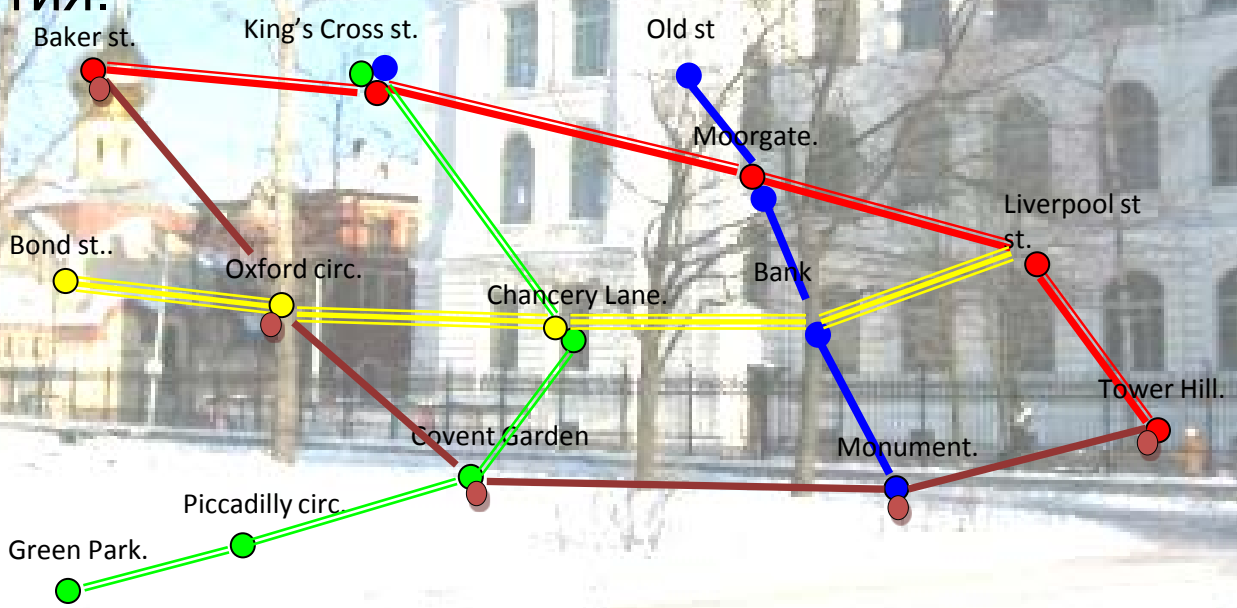
Агенты, решающие задачи поиска

- Формулировка задачи – процесс определения действий агента для **достижения цели.**
  - Перемещение по дорогам Лондона от Baker st. до Tower Hill на автобусе.
  - Агент имеет карту Лондона.
- Агент имеет несколько вариантов выбора. Агент вычисляет пути с различной стоимостью и выбирает наилучший путь.



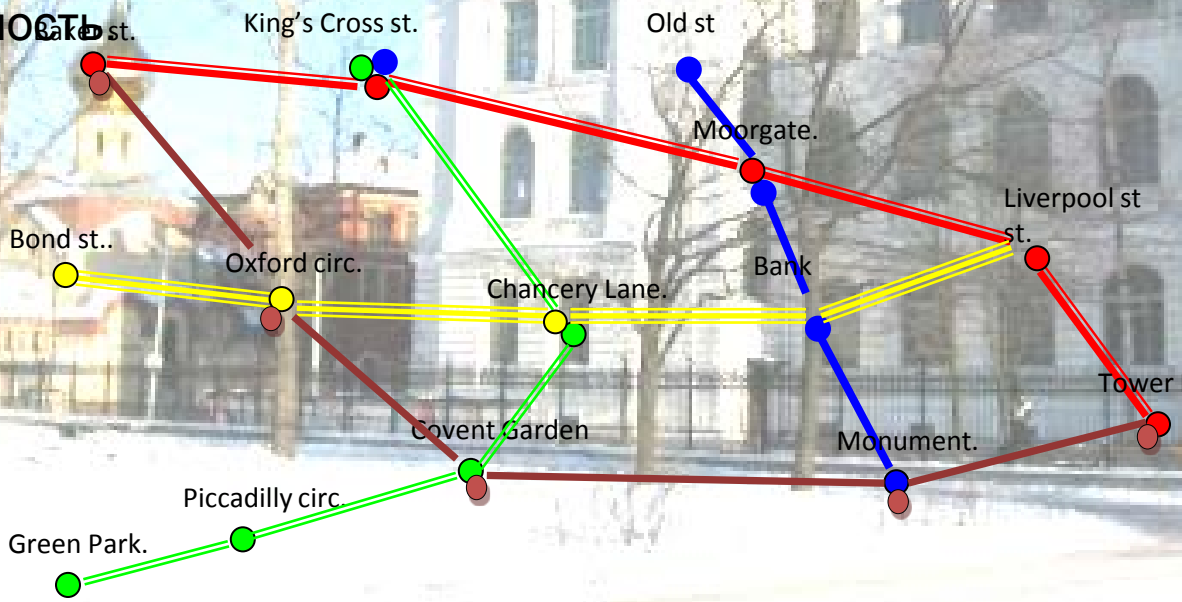
Проблемная среда

- **Статическая,** не претерпевает изменений.
- **Наблюдаемая,** известна начальная, конечная точка и карта.
- **Дискретная**, возможность перечислить все пути.
- **Детерминированная,** отсутствуют неожиданные события.



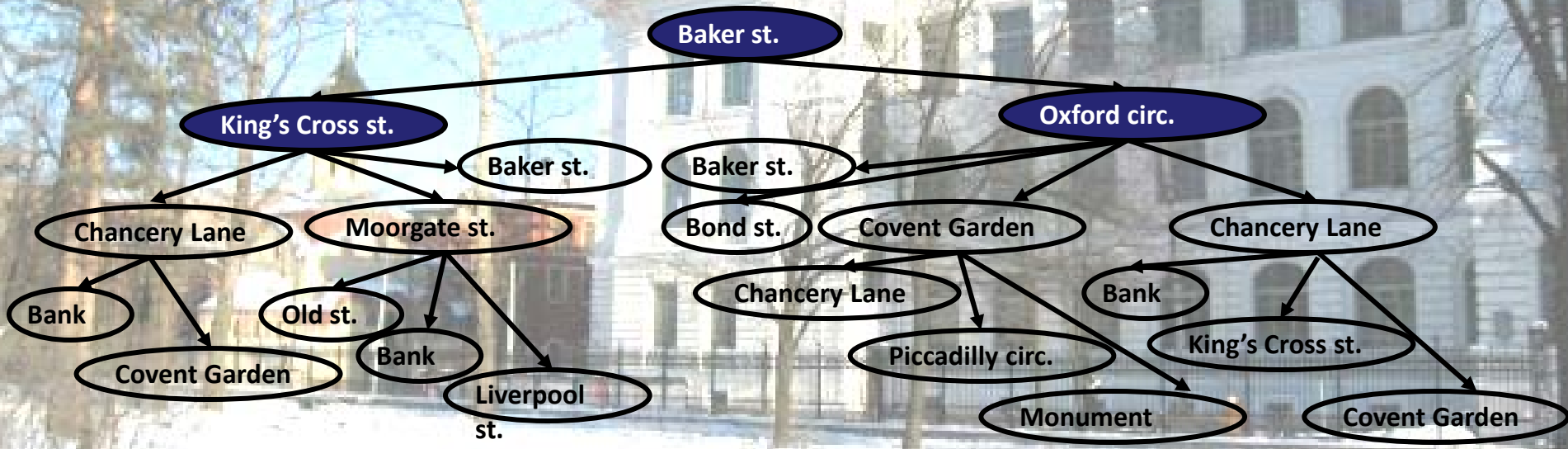
Хорошо структурированные задачи

- **Начальное состояние:** агент находится в Baker st.
- **Функция определения приемника:** возвращает множество пар <действие, преемник> .
- **Пространство состояний:** множество остановок, образующее граф.
- **Проверка цели:** достигнуто ли состояние Tower Hill.
- **Оптимальным решением** является путь, имеющий наименьшую стоимость.



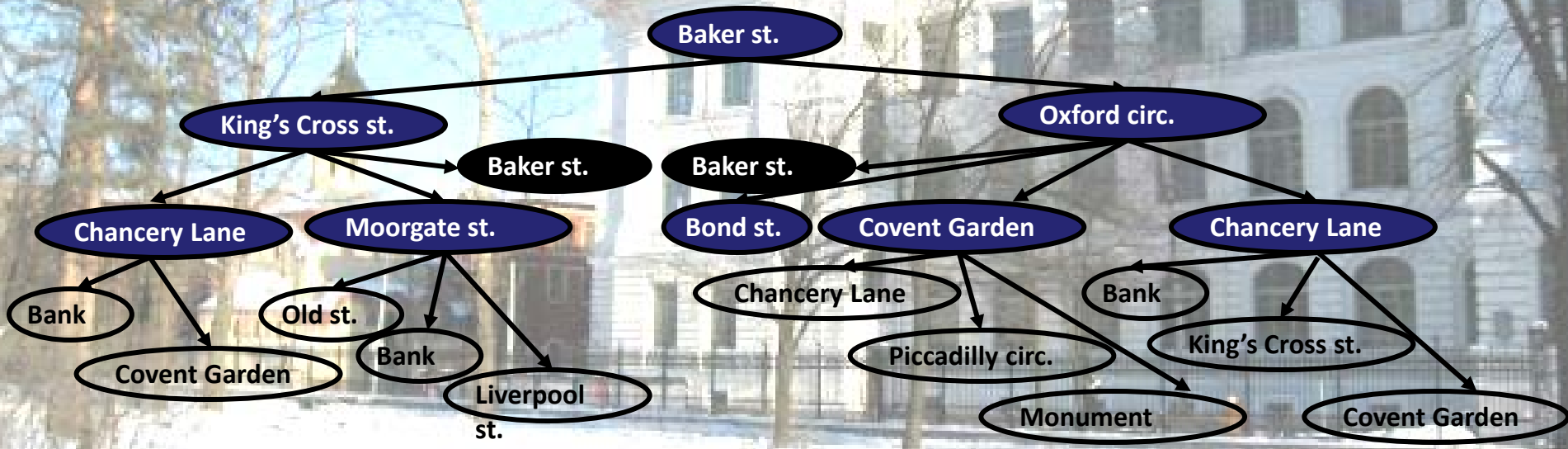
Поиск решений

- Строим дерево поиска.
- Корнем дерева является узел, соответствующий начальному состоянию.



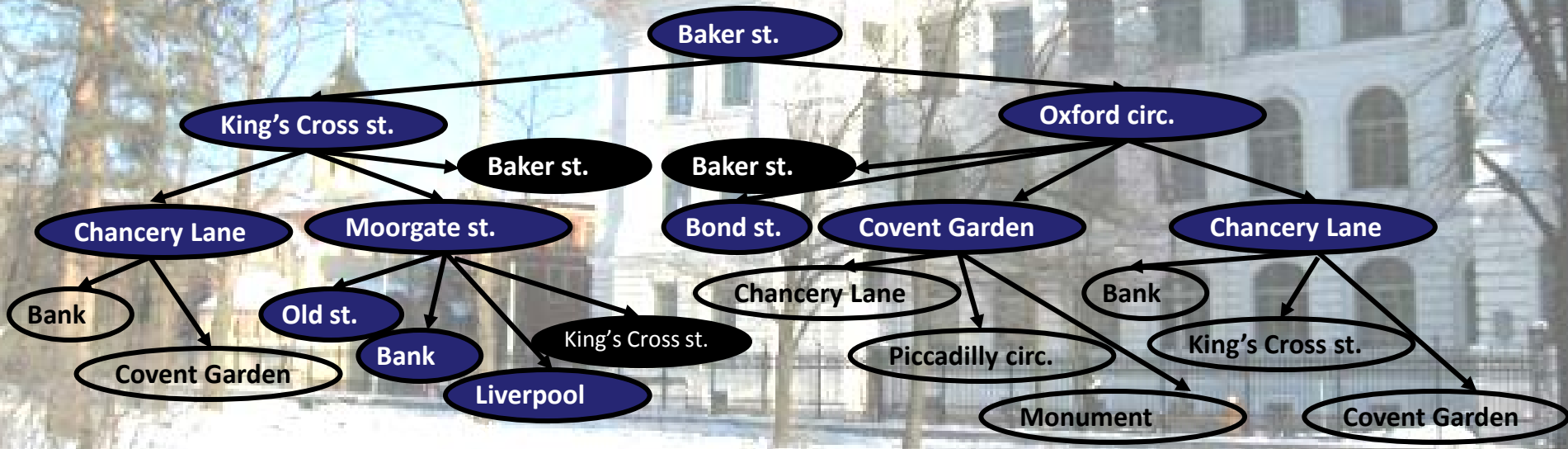
Поиск решений на графе

- Строим дерево поиска.
- Корнем дерева является узел, соответствующий начальному состоянию.



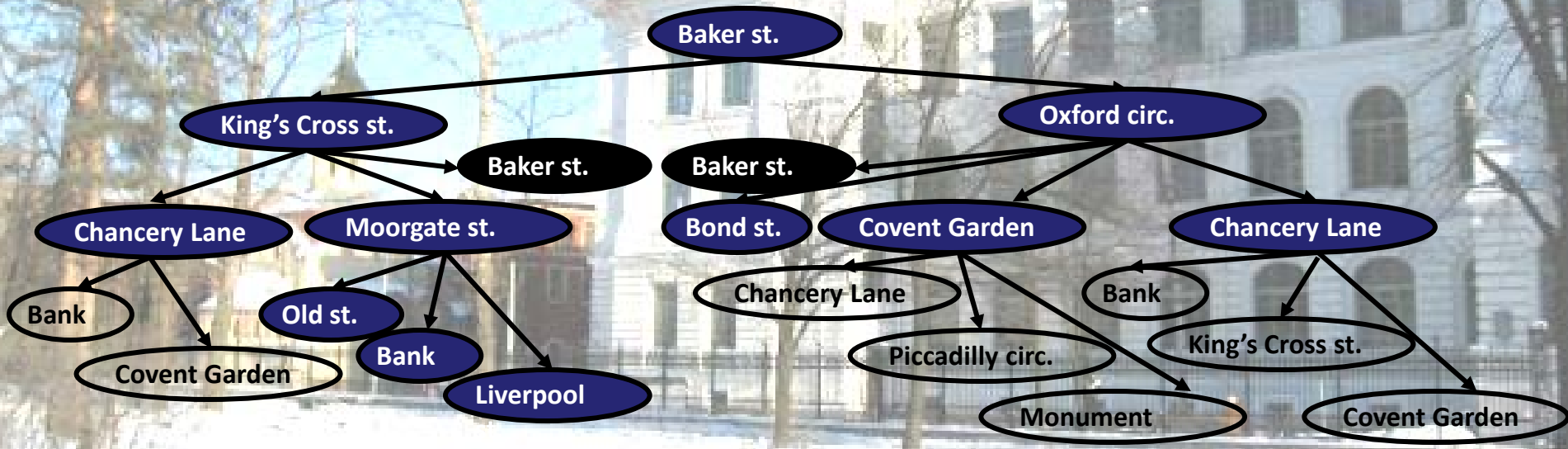
Поиск решений на графе

- Количество путей в этом пространстве состояний бесконечно.
- Дерево имеет бесконечное число узлов.



Поиск решений на графе

- Узел является структурой:
  - состояние, которому соответствует узел (конфигурация мира)
  - указатель родительский узел - данный узел
  - действие, приведшее к формированию данного узла
  - стоимость пути от начала до данного узла
  - количество пройденных узлов от начала пути



Задача игры в восемь

- **Состояния.** Местонахождение каждой из фишек.
- **Начальное состояние.** Любое состояние.
- **Функция определения преемника.** Осуществление действий <Влево, Вправо, Вверх, Вниз>.
- **Проверка цели.** Соответствие целевого состояния.
- **Стоимость пути:** каждый этап имеет стоимость 1.

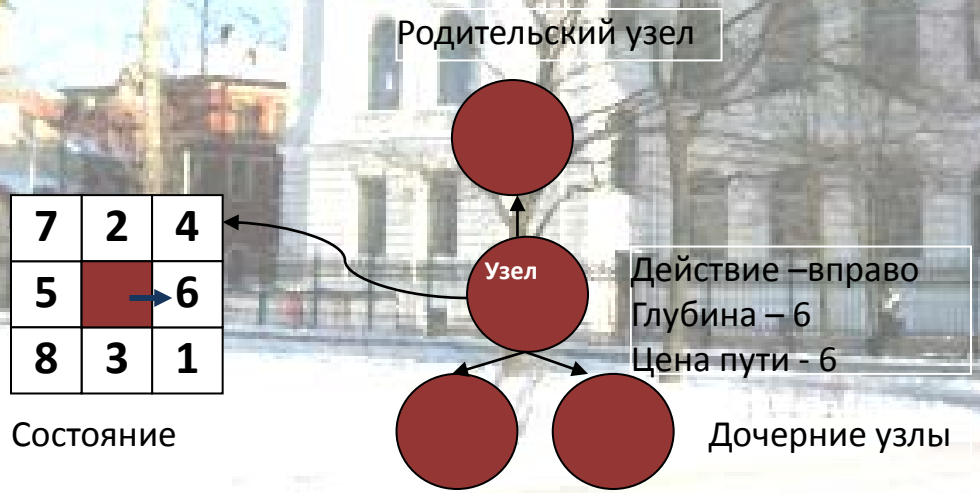
7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8

Количество достижимых состояний $9!/2 = 181440$

Поиск решений на графе

- Узел является структурой:
  - состояние, которому соответствует узел (конфигурация мира)
  - указатель родительский узел - данный узел
  - действие, приведшее к формированию данного узла
  - стоимость пути от начала до данного узла
  - количество пройденных узлов от начала пути



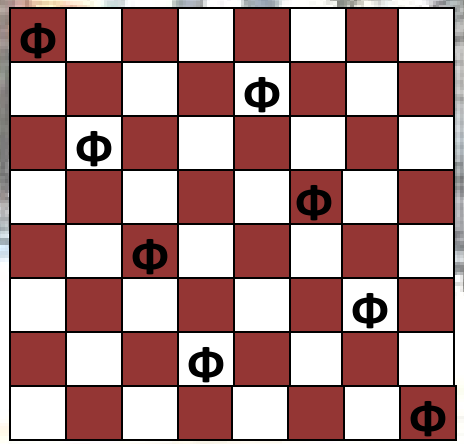
Показатель производительности

- **Полнота.** Гарантирует ли алгоритм получение решения.
- **Оптимальность.** Является ли решение оптимальным.
- **Временная сложность**. Время получения оптимального решения. Определяется количеством узлов, пройденных в процессе поиска.
- **Пространственная сложность.** Объем памяти, необходимый для решения задачи. Определяется коэффициентом ветвления и количеством дочерних узлов.
- **Стоимость поиска** зависит от времени поиска и объема памяти.

Задача игры в восемь

- **Состояния.** Место расположения ферзей.
- **Начальное состояние.** Отсутствие ферзей на доске.
- **Функция определения преемника.** Установка ферзя в пустую клетку.
- **Проверка цели.** Ни один из ферзей не атакован.
- **Стоимость пути:** каждый этап имеет стоимость 1.
- Количество достижимых состояний

$$64 \cdot 63 \cdot 62 \cdot \ldots \cdot 57 = 3 \cdot 10^{14}$$

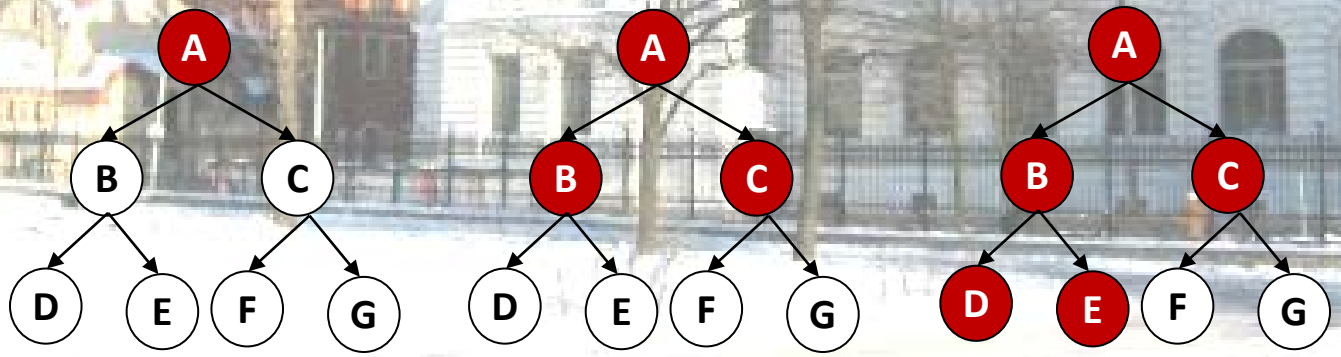


Стратегии неинформированного поиска

- **Неинформированный поиск** называется слепым поиском.
- Более перспективным является **информированный поиск**, который называется **эвристическим поиском**.

Неинформированный поиск в ширину

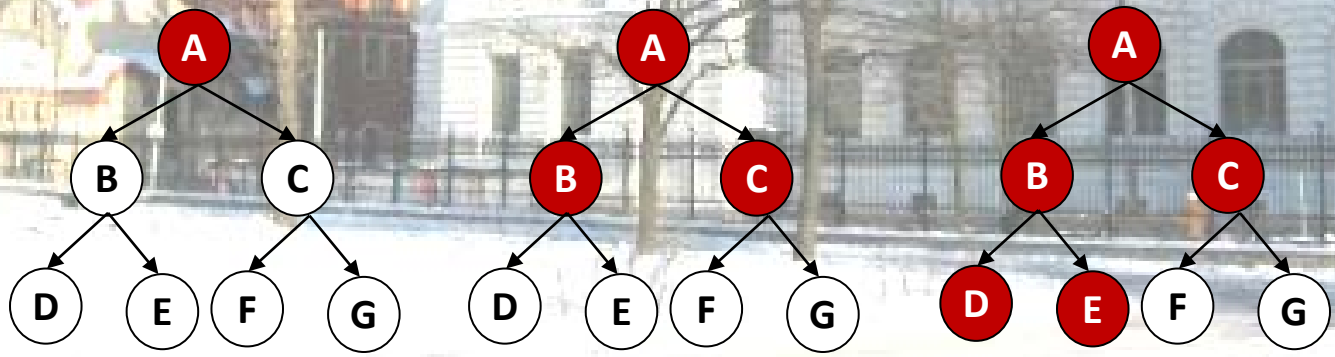
- Развертываются преемники корневого узла.
- Развертываются преемники преемников корневого узла.



Поиск в ширину

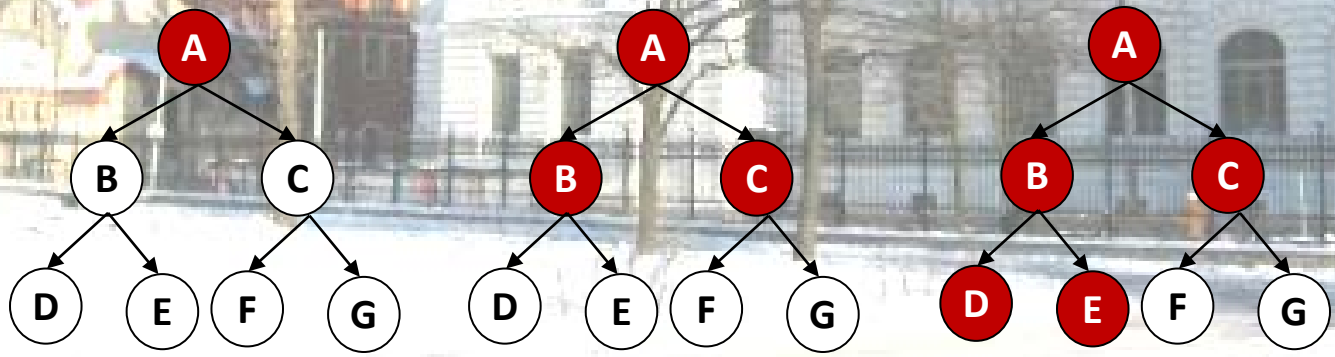
- **Условие полноты.** Самый первый целевой узел находится на конечной глубине d.
- **Оптимальность.** Самый первый целевой узел не обязательно является оптимальным.
- **Временная сложность**.
- Временная сложность определяется количеством обрабатываемых узлов.
- Если каждый узел имеет b преемников, то количество обрабатываемых узлов.

$$b + b^2 + b^3 + \ldots + \left(b^{d+1} - b\right) = O(b^{d+1})$$



Поиск в ширину

- **Пространственная сложность.** Каждый обработанный узел должен храниться в памяти, поэтому временная сложность имеет тот же порядок, что и временная сложность.
- Для числа ветвлений b=10 и глубине поиска d=8 требуемое количество памяти равно 1 терабайту (10^9 байт)
- Поиск в ширину является задачей с **экспоненциальной сложностью.**
- Задачи поиска с экспоненциальной сложностью **невозможно решить** с помощью неинформированных методов. Исключение составляют самые простые задачи.



Поиск по критерию стоимости

- Поиск в ширину является задачей с экспоненциальной сложностью.
- **Поиск по критерию стоимости**
- Поиск в ширину оптимален, если стоимости всех узлов равны.
- Поиск по критерию стоимости обеспечивает развертывание узла с наименьшей стоимостью.

Поиск по критерию стоимости

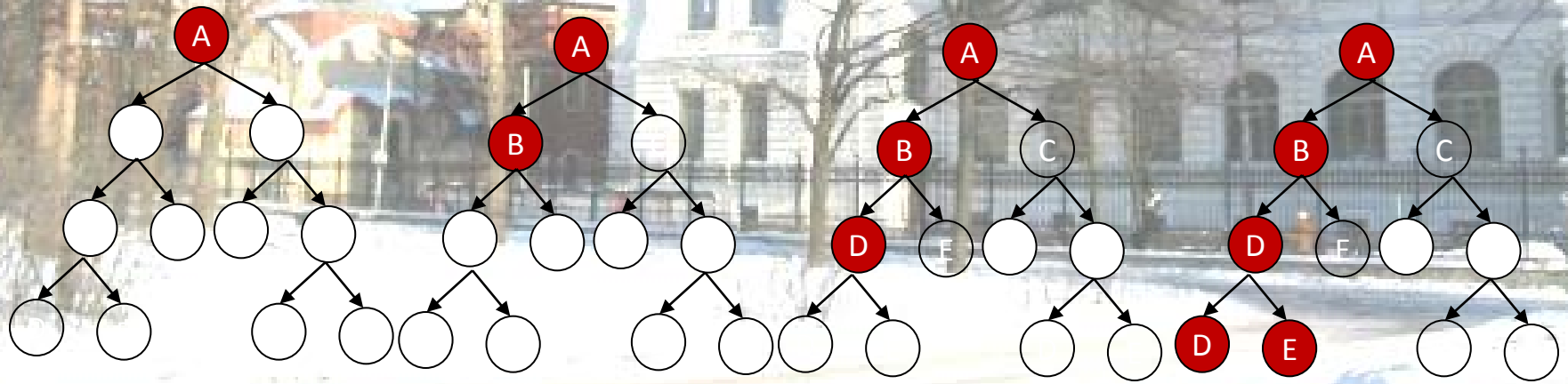
- Пространственная сложность при поиске по критерию стоимости.
- Оптимальная стоимость решения C^*
- Глубина поиска d
- Количество ветвлений b.
- Стоимость каждого действия e.
- Пространственная сложность

$$O\left(b^{\left\lfloor 1+C^*/e \right\rfloor}\right)$$

- Если все стоимости действий равны, то пространственная сложность такая же как при простом поиске в ширину.

Поиск в глубину

- Сначала развертывается самый глубокий узел.
- **Пространственная сложность.** Только родительские узлы должны храниться в памяти, поэтому уменьшается пространственная сложность.
- Для числа ветвлений b=10 и глубине поиска m>>d m=10 000 требуемое количество памяти равно bm+1 = 100 000 (вместо 1 терабайта).

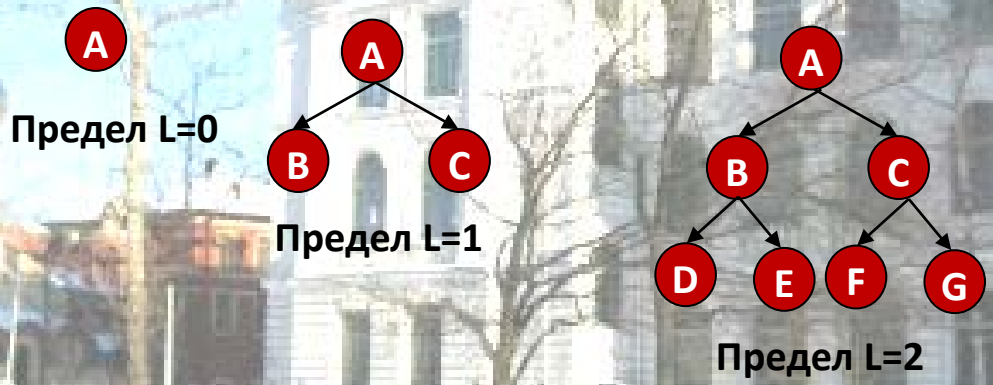


Поиск с ограничением глубины

- Для неограниченных деревьев следует примять поиск с ограничением глубины. Ограничение равно L.
- Если выбрано L<d, то решение выходит за пределы глубины.
- При поиске в глубину при L=∞.
- Если число остановок равно 15, то глубина поиска должна быть не менее L=14.
- Любая остановка может быть достигнута не менее чем за 5 шагов.
- Диаметр пространства состояний равно 5.

Поиск в глубину с итеративным углублением

- Пошаговое увеличение предела глубины L=0,1,2,…,d.
- d- глубина поверхностного узла.



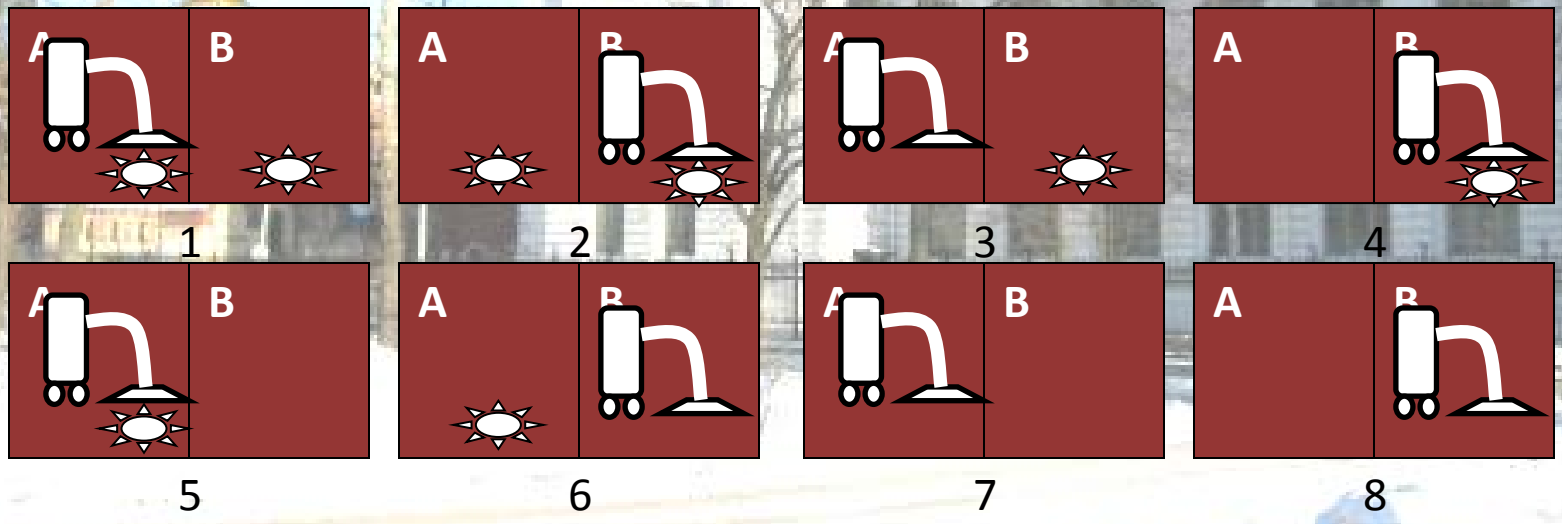
Поиск в глубину с итеративным углублением

- **Временная сложность** при поиске в глубину с итеративным углублением $O(b^d)$
- **Временная сложность** при поиске в ширину $O(b^{d+1}-b)$
- При b=10, d=5
- поиск в ширину формирует число узлов N=1 111 100
- поиск в глубину с итеративным углублением N=123 450

Поиск в глубину с итеративным углублением является лучшим методом неинформированного поиска при условии, когда глубина решения не известна и имеется большое пространство поиска.

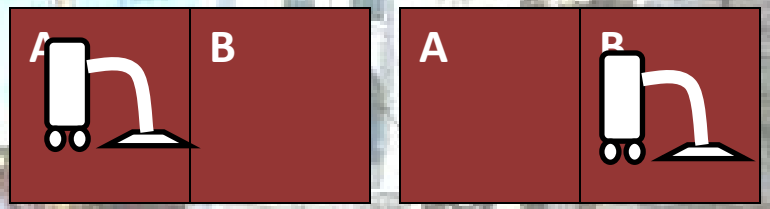
Поиск с частичной информацией

- Если агент **не имеет полной информации** о состоянии среды, то возникают проблемы:
- 1. Проблемы отсутствия датчиков.
- 2. Проблемы непредвиденных ситуаций или проблемы, обусловленные сторонним воздействием.
- 3. Проблемы исследования. Проблемы исследования рассматриваются как крайний случай непредвиденных ситуаций.



Проблема отсутствия датчиков

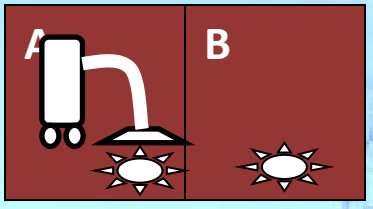
- Множество **доверительных состояний** – некоторые состояния, в которых, как предполагает агент, он находится.
- Действие <Right, Suck, Left,Suck> приводит к состоянию 7.
- Действие < Left, Suck, Right, Suck> приводит к состоянию 8.



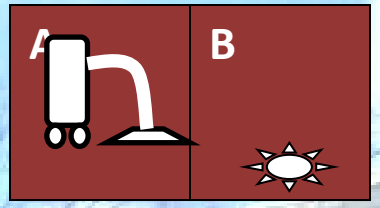
7 8

Проблема отсутствия датчиков

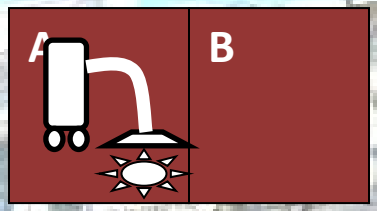
Доверительные состояния после выполнения команды <Left>



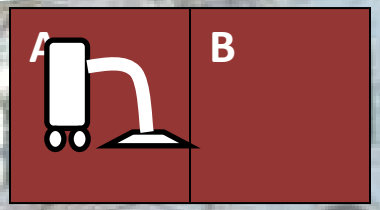
1



3

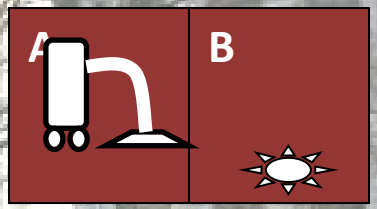


5

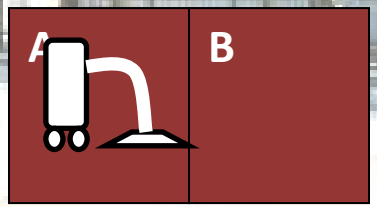


7

- **Доверительные состояния** после выполнения команд <Left,Suck>



3



7

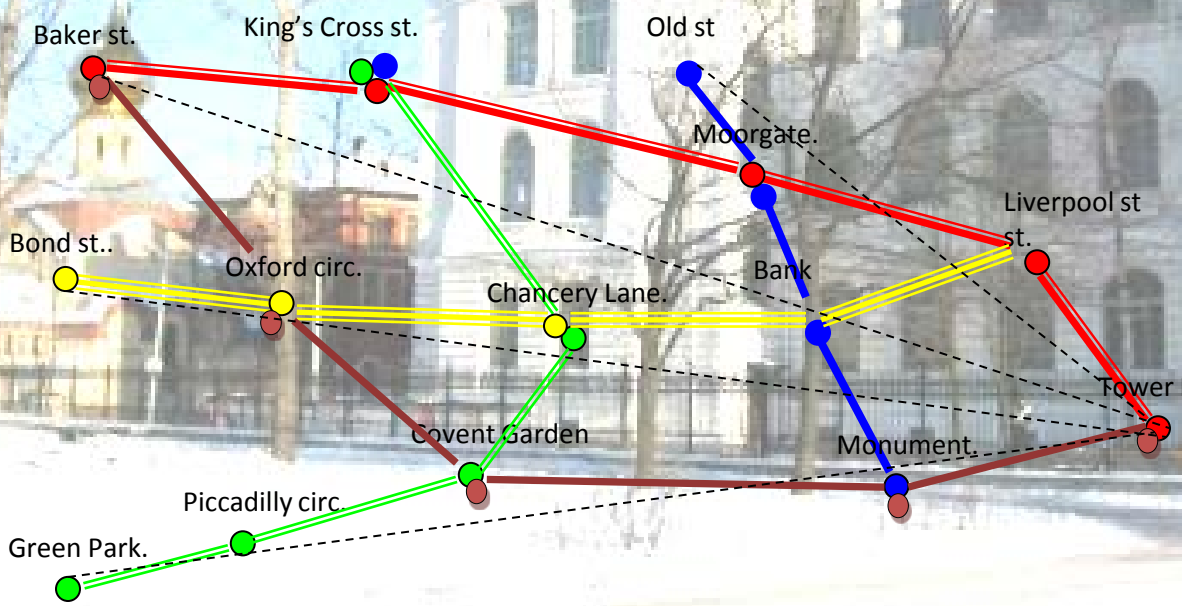
Информированный поиск и исследование пространства состояний

Стратегии эвристического поиска

- Поиск по первому наилучшему совпадению использует **функцию оценки f(n).**
- Для развертывания выбирается узел с наименьшей оценкой.
- Поиск по первому наилучшему совпадению наилучшего маршрута выполняется с использованием **эвристической функции h(n)**

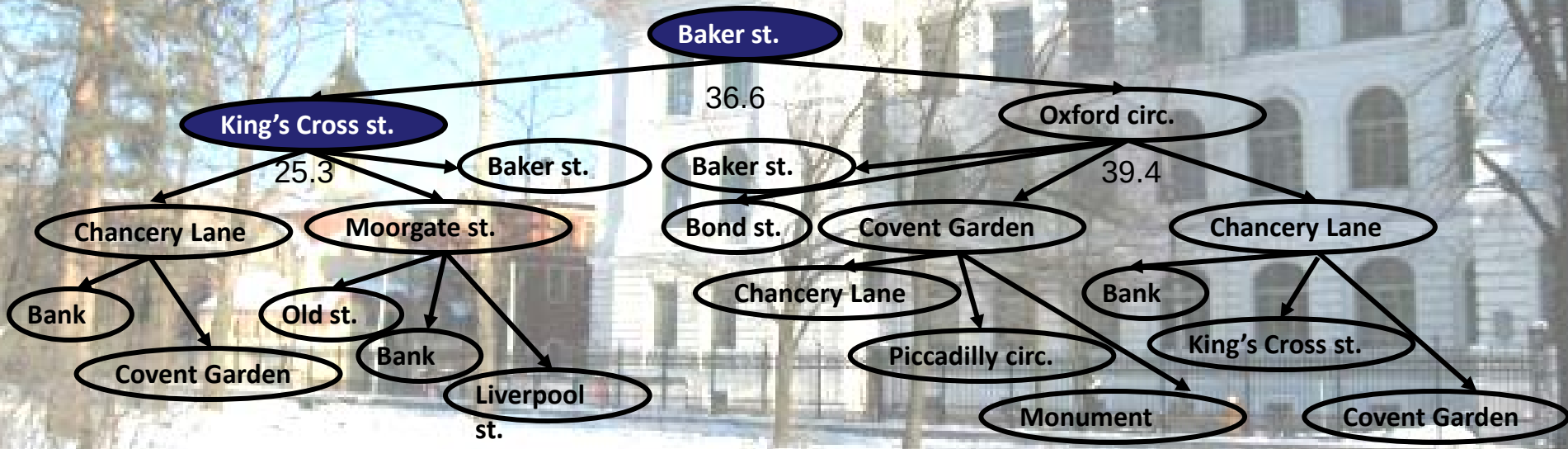
Жадный поиск по первому наилучшему совпадению

- Развертывается самый близкий к цели (Tower Hill) узел
- Эвристическая функция f(n)=h(n) задает расстояние по прямой до цели Tower Hill.



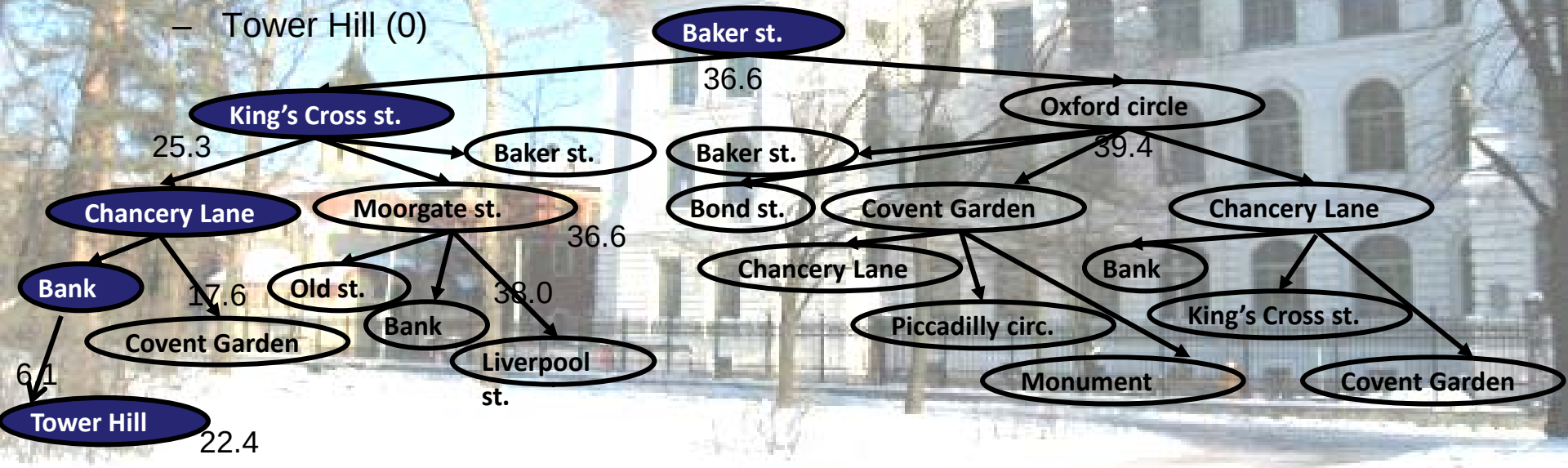
Жадный поиск по первому наилучшему совпадению

- После развертывания узла Baker str. наименьшая эвристическая функция у узла King’s Cross str.



Жадный поиск по первому наилучшему совпадению

- После развертывания узла King's Cross str. наименьшая эвристическая функция у узлов:
  - Chancery Lane (17.6)
  - Bank str.(6.1)
  - Tower Hill (0)

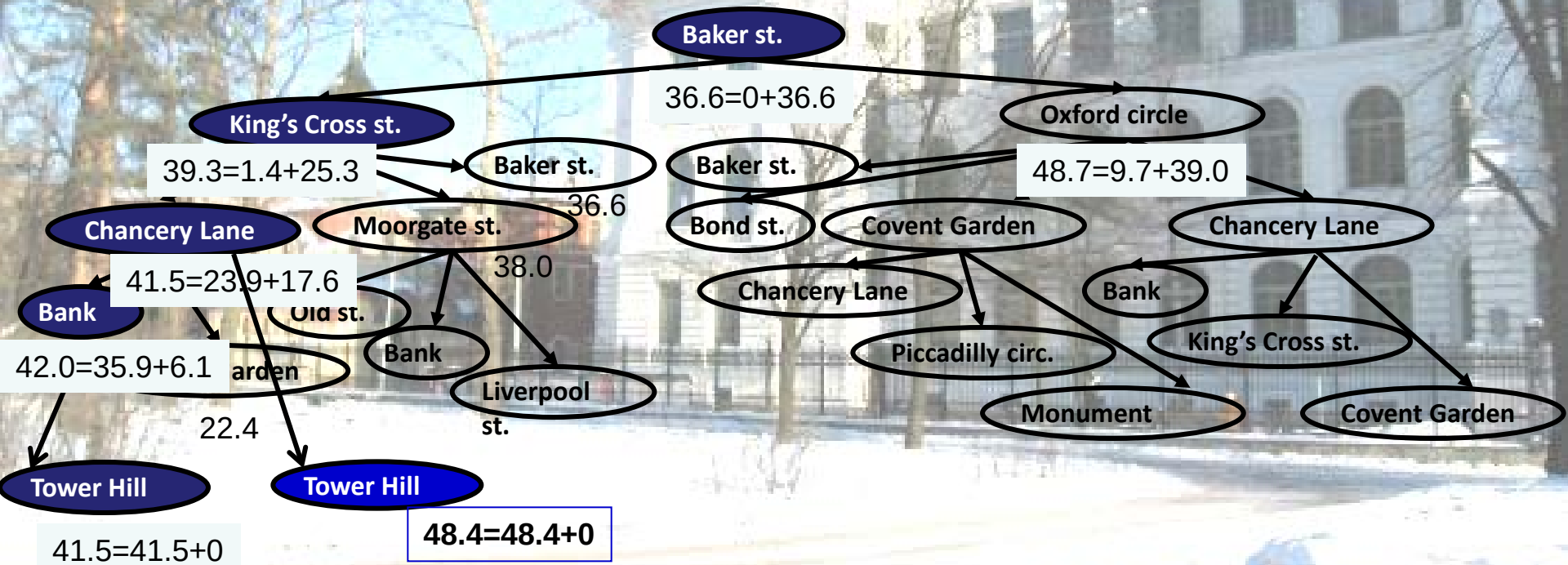


Поиск A* - разновидность поиска по первому наилучшему совпадению

- Эвристическая функция – оценка стоимости наименее дорогостоящего решения:
- f(n)=g(n)+h(n)
- g(n) - стоимость достижения узла n.
- h(n) – стоимость пути до цели от узла n, допустимая эвристическая функция.

Поиск A* - разновидность поиска по первому наилучшему совпадению

- Поиск A* оптимален, если функция h(n) допустимая.
- Если h(n) не переоценивает стоимость пути решения f(n)=g(n)+h(n)<C*



Поиск A* - разновидность поиска по первому наилучшему совпадению

- Поиск A* является **оптимальным**, если эвристическая функция h(n) является **допустимой**
- Пусть найден **неоптимальный узел** G2 (Tower Hill – 48.4), где h(G2)=0 и стоимость больше оптимальной C*:

$$f(G2)=g(G2)+h(G2)=g(G2)>C^*$$

- Если существует оптимальное решение, то

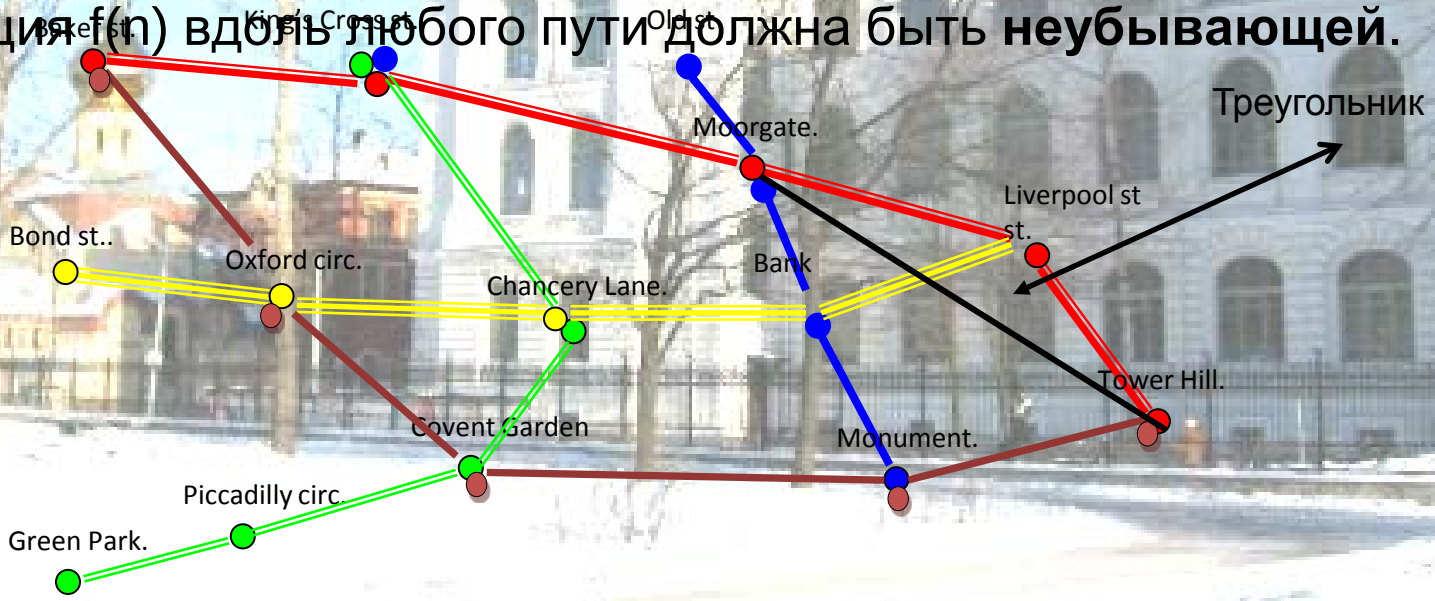
$$f(n)=g(n)+h(n)<=C^*$$

- Тогда узел G2 не развертывается.

Преемственность эвристической функции

- Функция h(n) является **преемственной**, если для любого преемника n’ узла n оценка стоимости

  h(n)<=c(n,n’)+h(n’) - **неравенство треугольника**
- Поиск A* является оптимальным, если **функция h(n)-преемственна**
- Функция f(n) вдоль любого пути должна быть **неубывающей**.



Эвристические функции

- Количество ветвлений в среднем - 3
- Количество состояний при исчерпывающем поиске $3^{22}=3*10^{10}$
- $H_1(n)$ – количество фишек, стоящих не на своем месте:

$$H_1=8$$

- $H_2(n)$ – сумма расстояний всех фишек от их целевых позиций (манхэттенское расстояние).
- $H_2=3+1+2+2+2+3+7+2=22$
- Функции $H_1(n)$ и $H_2(n)$ не переоценивают истинную стоимость решения

7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8

Составление допустимых эвристических функций

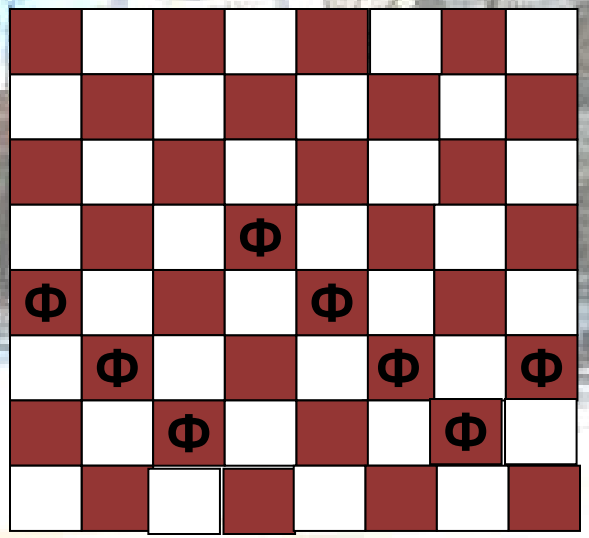
- **Ослабленная задача** является задачей с меньшим количеством ограничений.
- Стоимость оптимального решения ослабленной задачи представляет собой **допустимую эвристику** для первоначальной задачи.
- Три ослабленные задачи:
  - Фишка может быть передвинута из квадрата A в квадрат B, если квадрат A является смежным с квадратом B,
  - Фишка может быть передвинута из квадрата A в квадрат B, если квадрат B пуст.
  - Фишка может быть передвинута из квадрата A в квадрат B.
- **Подзадача** данной задачи имеет меньшее количество объектов.
- Подзадача пере я ч фишек 1,2,3,4.

Локальный поиск

- Алгоритмы локального поиска
  - Учитывают только **текущее состояние** и на его основе определяют переход к следующему состоянию,
  - Решают задачу оптимизации,
  - Ищут максимум целевой функции или минимум функции стоимости,
  - Учитывают ландшафт пространства состояний .

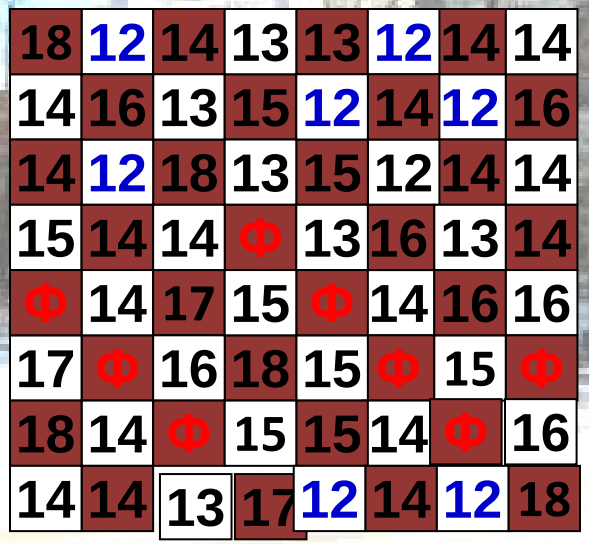
Алгоритмы локального поиска с восхождением к вершине

- Жадный локальный поиск выполняет захват самого хорошего соседнего состояния
  - Часто заходит в тупик по причинам
  - Сходится к локальному максимуму
  - Движется вдоль хребта
  - Находится на плато



Задача с восемью ферзями

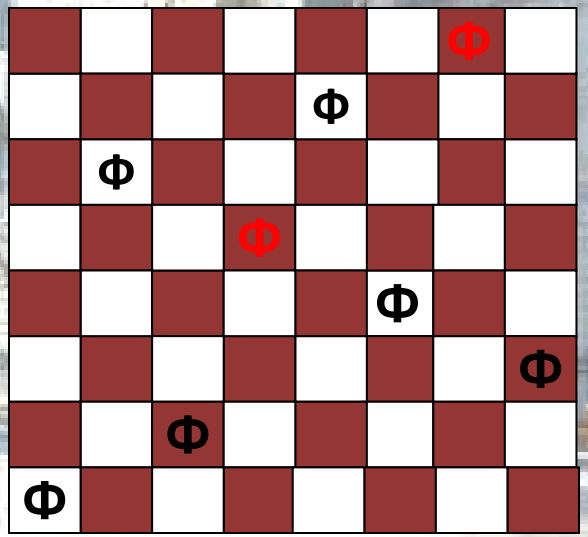
- **Эвристическая функция** определяет количество пар ферзей, которые атакуют друг друга прямо или косвенно
- Начальное значение h=17
- Наилучшие преемники h=12
- Случайный выбор на множестве наилучших преемников.



Задача с восемью ферзями

- Начальное значение h=17
- Решение получено в локальном минимуме h=1
- Количество этапов от 3 до 4
- Относительное количество тупиковых решений– 86%
- Относительное количество успешных решений – 14%

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	Ф	13	16	13	14
Ф	14	17	15	Ф	14	16	16
17	Ф	16	18	15	Ф	15	Ф
18	14	Ф	15	15	14	Ф	16
14	14	13	17	12	14	12	18

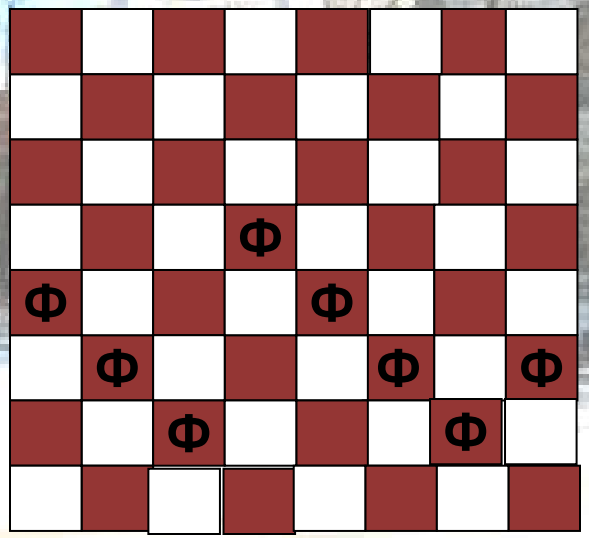


Другие алгоритмы локального поиска

- **Алгоритмы с движением в сторону** позволяют покидать «плато» имеют Относительное количество успешных решений – 94%, количество этапов – от 21 до 64.
- **Алгоритмы стохастического поиска** с восхождением к вершине случайным образом выбирает один из вариантов на каждом этапе.
- **Алгоритмы с перезапуском** выполняет новую попытку, если предыдущая попытка была неудачной. Относительное количество успешных решений –100%,

Генетические алгоритмы

- **Генетический алгоритм** - вариант стохастического алгоритма.
- **Популяция** – множество случайным образом сформированных состояний.
- **Индивидуум** – каждое конкретное состояние.
- Каждое состояние характеризуется **функцией пригодности.**



Генетические алгоритмы

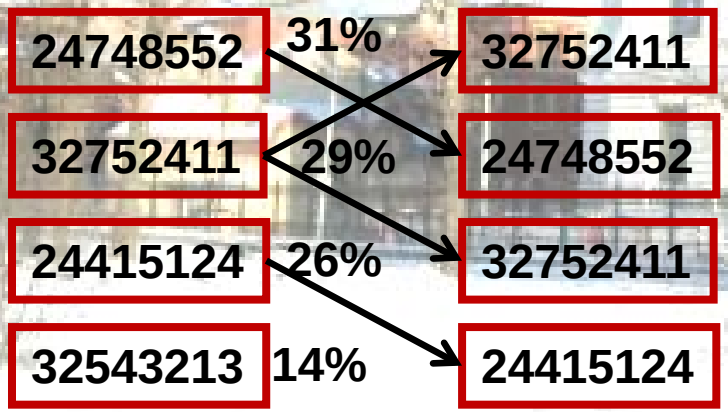
- **Функция пригодности** – эвристическая функция
- Функция пригодности возвращает количество пар **не атакующих** друг друга ферзей.
- Решение – h=28
- Значение функции пригодности для поколения 1: 24, 23, 20, 11:

24748552	24
32752411	23
24415124	20
32543213	11

Генетические алгоритмы

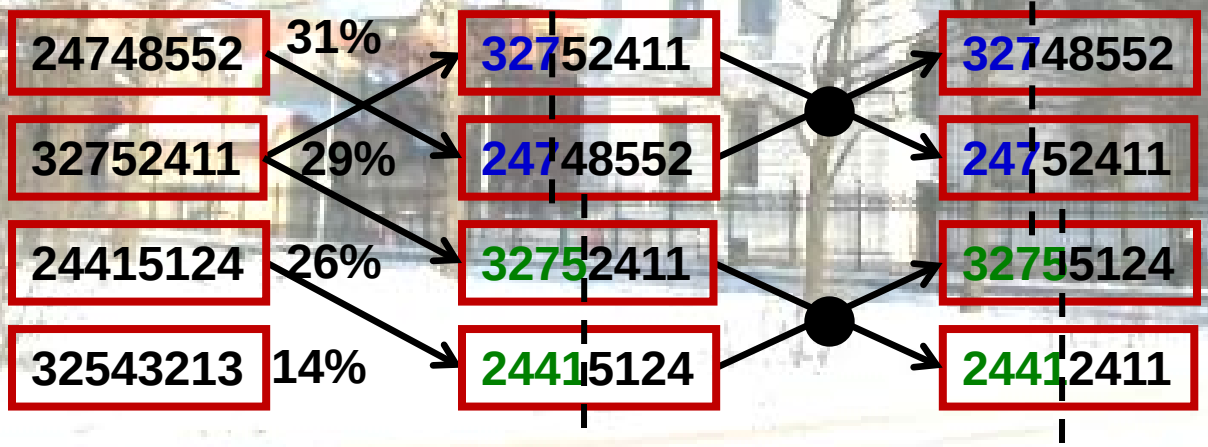
- **Для воспроизводства** выбираются пары в соответствии с вероятностями

  P1=24/(24+23+20+11)=31%, P2= 23/(24+23+20+11)=29%, и т.д.
- Один индивид может быть выбран дважды



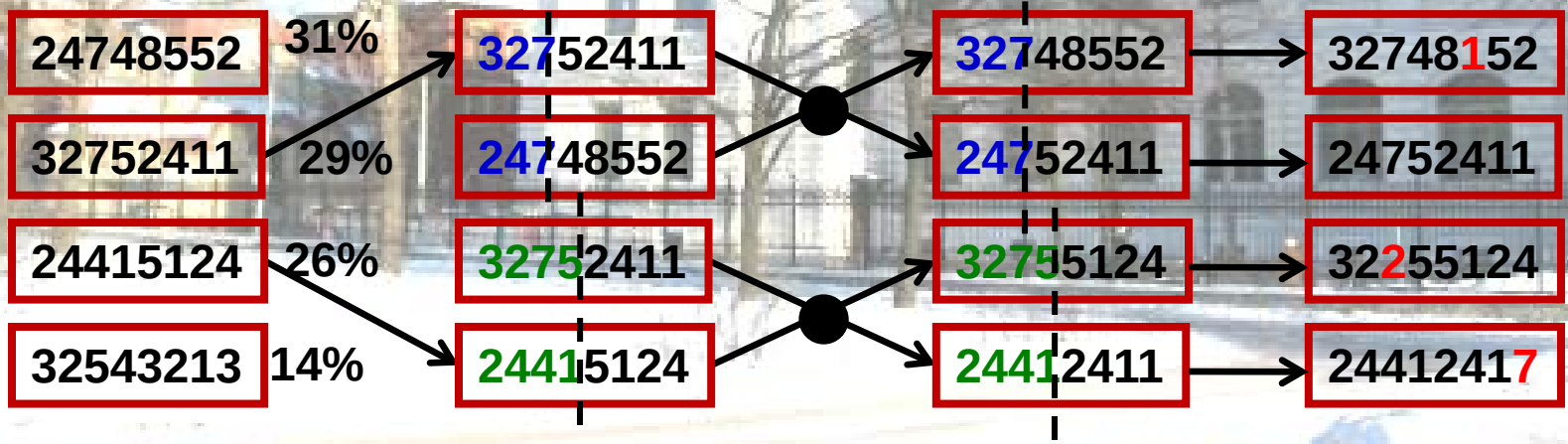
Генетические алгоритмы

- Для каждой пары случайным образом выбирается **точка скрещивания**
- **Образование нового поколения** путем скрещивания:



Генетические алгоритмы

- **Мутации** - случайным образом **с небольшой вероятностью** выполняются изменения одного из чисел



Эволюционные алгоритмы

- Теория эволюции изложена Чарльзом Дарвиным. Идея – в процессе эволюции возникают мутации, которые сохраняются в следующих поколениях в зависимости от пригодности.
- Грегор Мендель открыл вероятностные закону, управляющие процессом эволюции.
- Уотсон Крик выявил структуру молекул ДНК и ее алфавит, состоящий из аминокислот.

Поиск в оперативном режиме

Лекция 3

Задачи поиска в оперативном режиме

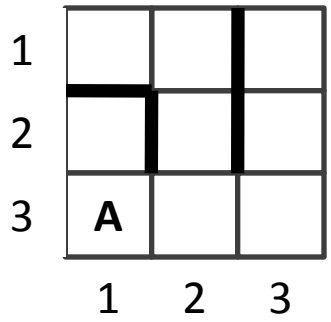
- Виды поиска:
- Поиск в автономном режиме
  - Поиск по первому наилучшему совпадению, для развертывания выбираются узлы с наименьшей стоимостью
  - Жадный поиск по первому наилучшему совпадению с развертыванием узлов с наименьшей h(n)
  - Поиск A* с развертыванием узлов с наименьшей g(n)+h(n)
  - Локальный поиск с восхождением к вершине
  - Генетический алгоритм поиска
- Поиск в оперативном режиме
- Поиск с ограничениями

Поиск в оперативном режиме

- Чередование вычислений и действий в стохастических и динамических проблемных средах,
- Наличие непредвиденных ситуаций
- Решение задач исследования среды,
  - составление карты,
  - нахождение цели.
- Пример:
- Робот исследует новое здание
- Используемый алгоритм: поиск выхода из лабиринта

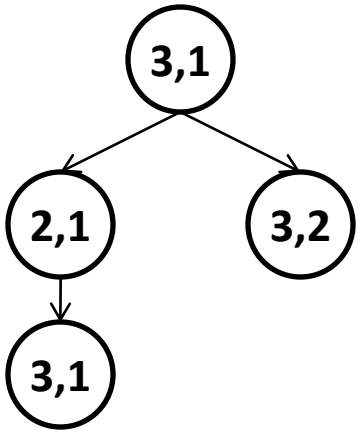
Задачи исследования в оперативном режиме

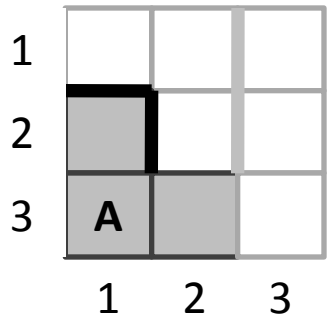
- Агент выполняет следующие действия:
  - Воспринимает среду и определяет состояние, которого достиг
  - Дополняет карту среды
  - Узнает состояния, которые он посещал
  - Решает, куда двигаться дальше
- Агент выполняет только детерминированные действия



Задачи исследования в оперативном режиме

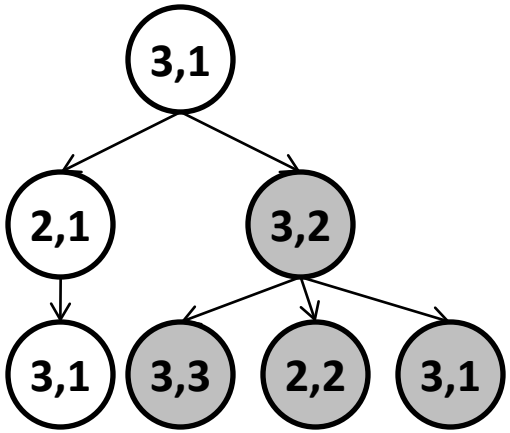
- Агент имеет датчики определения текущих координат
- Агент имеет датчики определения препятствий в текущем состоянии

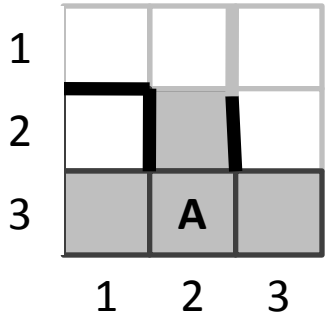




Задачи исследования в оперативном режиме

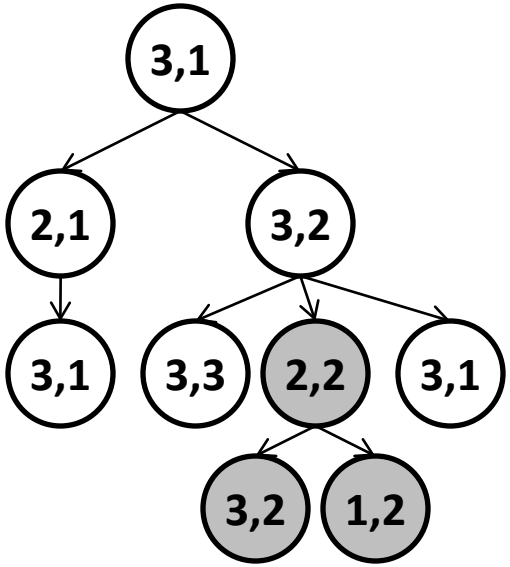
- Агент попадает в тупик s(2,1) и понимает, что должен двигаться назад в S(3,1)
- Агент двигается в S(3,2)

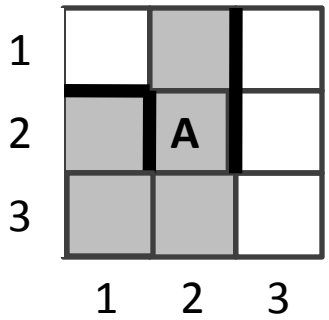




Задачи исследования в оперативном режиме

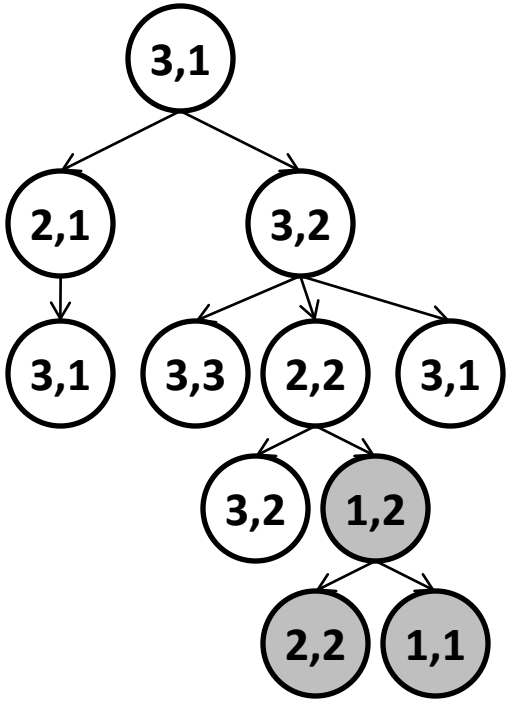
- Агент продолжает исследование, перемещаясь в S(2,2)

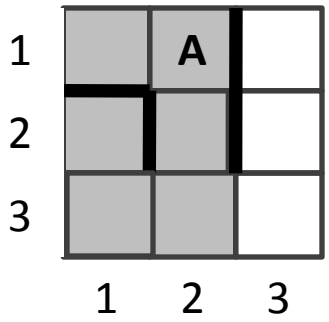




Задачи исследования в оперативном режиме

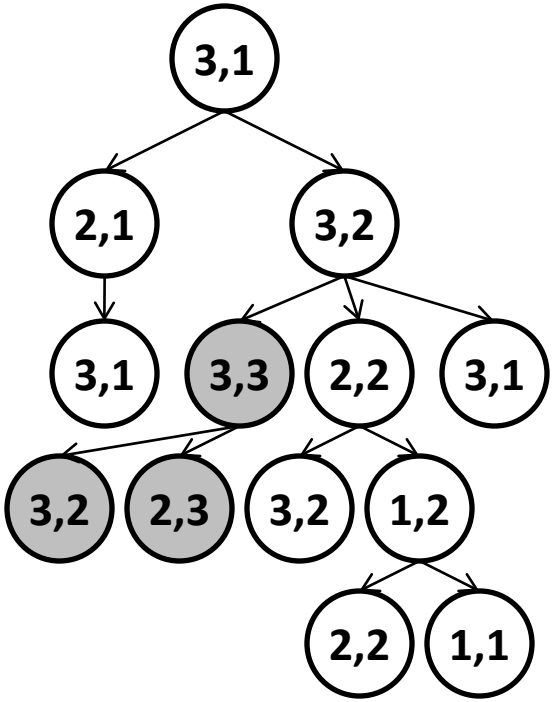
- Агент перемещается в S(2,1), обнаруживает позицию S(1,1)
- Агент посещает позицию S(1,1)
- Агент обнаруживает тупик и двигается обратно

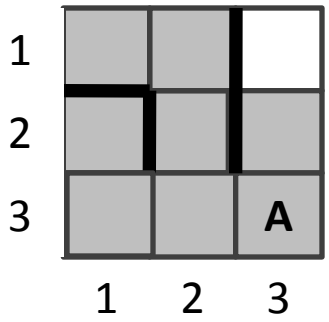




Задачи исследования в оперативном режиме

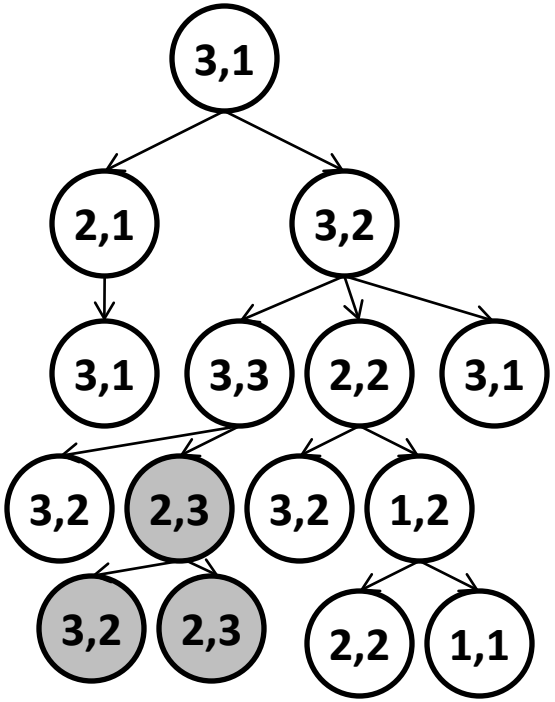
- Агент узнает позиции, которые он посещал
- Агент попадает в позицию S(3,3), где обнаруживает новый путь

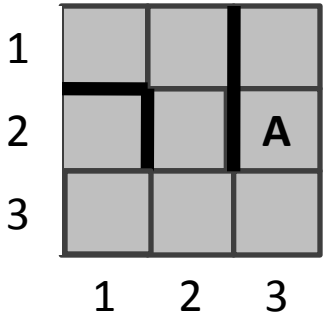




Задачи исследования в оперативном режиме

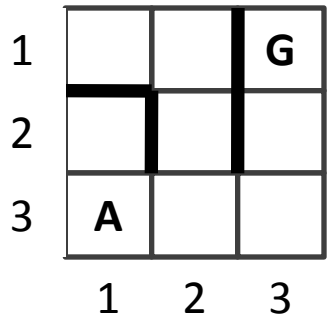
- Посетив позицию S(2,3) и затем S(1,3) агент завершает исследование проблемной среды
- Построена карта проблемной среды
- Построен граф





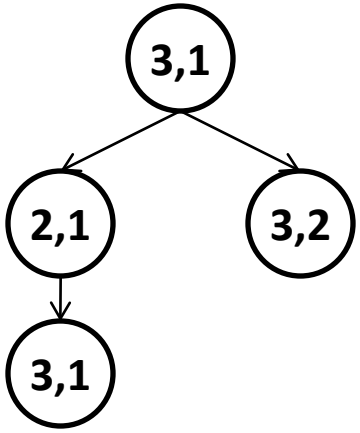
Задачи поиска в оперативном режиме

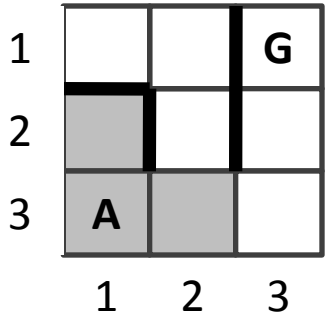
- Агент начинает перемещение из позиции S(3,1) и пытается найти цель Goal (G)
- Агент выполняет следующие функции:
  - Функцию действия
  - Функцию стоимости этапа **c(n,a,n')**
  - Функцию достижения цели - **G**
- Агенту доступна эвристическая функция **h(n)** – стоимость пути до цели, функция текущей стоимости пути **g(n)** и функция оценки **f(n)=g(n)+h(n)**
- Алгоритм поиска A*



Поиск по первому наилучшему совпадению

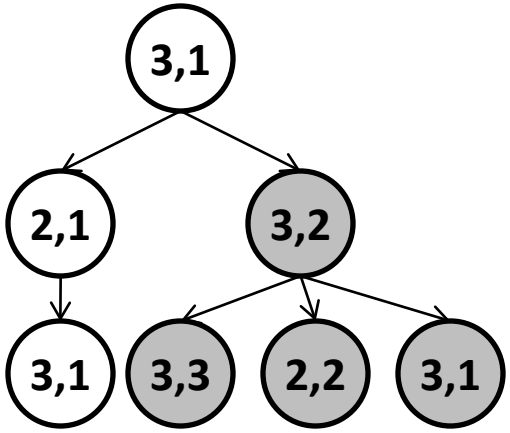
- Ни один алгоритм не позволит избежать тупиков во всех возможных пространствах состояний
- Безопасно исследуемые пространства – это пространства состояний, в которых цель может быть достигнута из любого текущего состояния.
- Даже в безопасных средах возможны пути с неограниченной стоимостью.

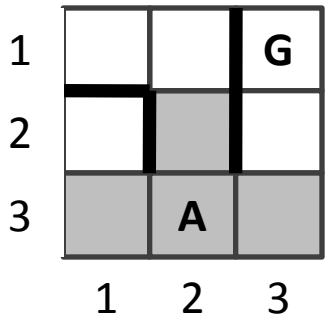




Поиск по первому наилучшему совпадению

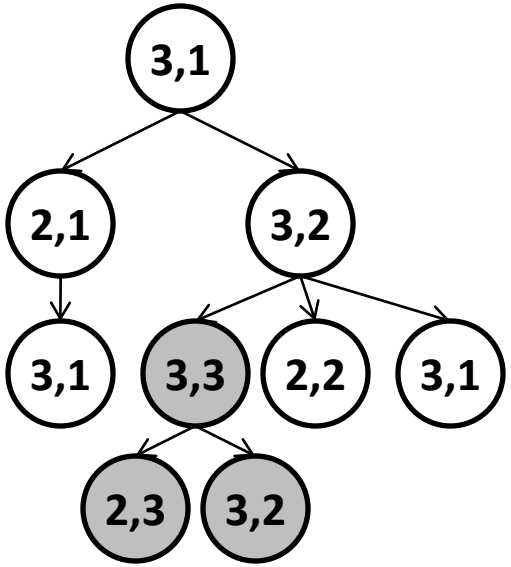
- Ни один алгоритм не позволит избежать тупиков во всех возможных пространствах состояний
- Безопасно исследуемые пространства – это пространства состояний, в которых цель может быть достигнута из любого текущего состояния.
- Даже в безопасных средах возможны пути с неограниченной стоимостью.

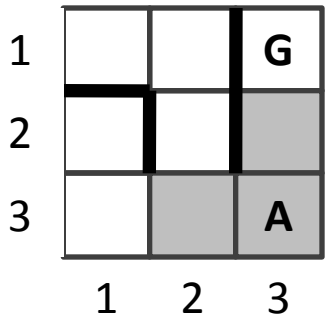




Поиск по первому наилучшему совпадению

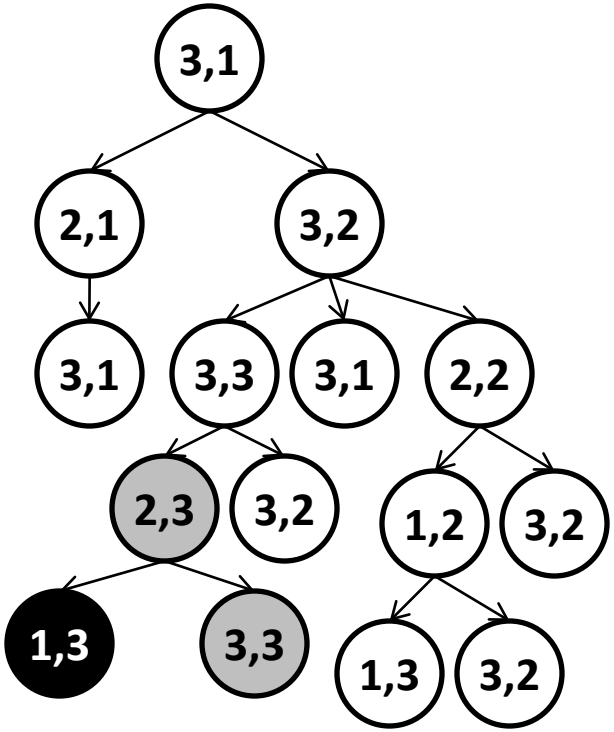
- Ни один алгоритм не позволит избежать тупиков во всех возможных пространствах состояний
- Безопасно исследуемые пространства – это пространства состояний, в которых цель может быть достигнута из любого текущего состояния.
- Даже в безопасных средах возможны пути с неограниченной стоимостью.

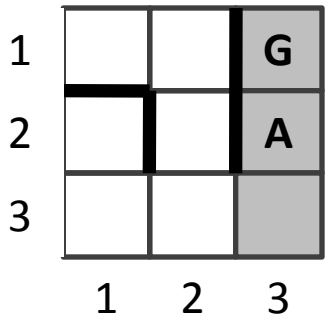




Поиск по первому наилучшему совпадению

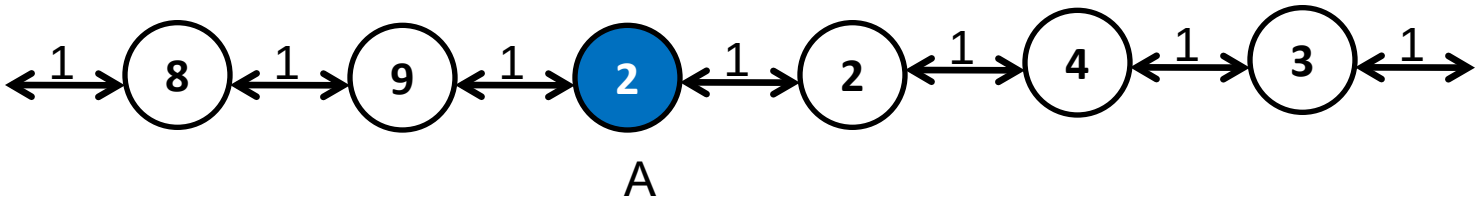
- Ни один алгоритм не позволит избежать тупиков во всех возможных пространствах состояний
- Безопасно исследуемые пространства – это пространства состояний, в которых цель может быть достигнута из любого текущего состояния.
- Даже в безопасных средах возможны пути с неограниченной стоимостью.





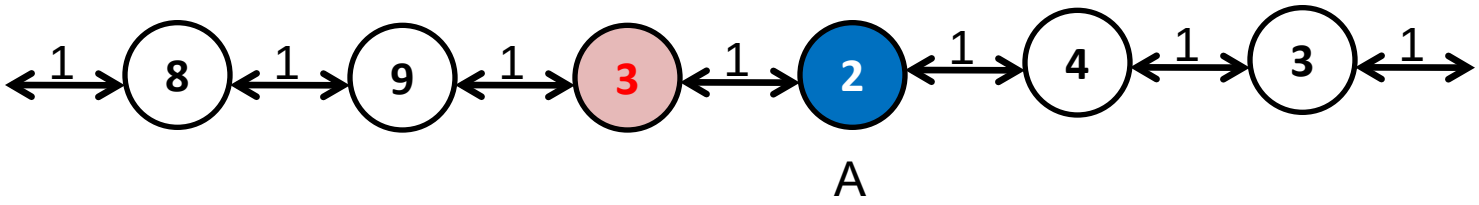
Локальный поиск в оперативном режиме

- Поиск с восхождением к вершине требует сохранения в памяти только одного текущего состояния и наилучшую оценку H(s).
- Не позволяет выполнять случайный перезапуск алгоритма.
- Вместо перезапуска используется случайное блуждание
- Эвристическая функция h(s) обновляется по мере исследования среды.
- H(s)+h(s) – обновленное значение эвристической функции
- Агент попал на плоский участок **локального минимума** h(s)=2



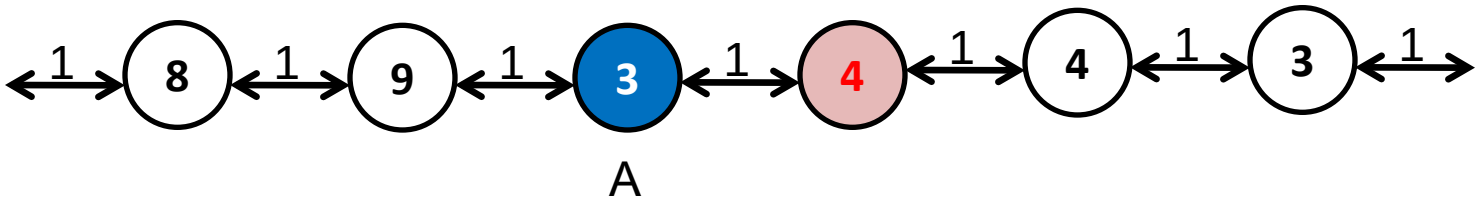
Локальный поиск в оперативном режиме

- Агент продолжает блуждание в наилучшем направлении
- Оценка стоимости достижения цели через состояние s’ складывается c(s,a,s’)+H(s), c(s,a,s’)=1
- Следующее движение выполняется в сторону с наименьшей стоимостью
- Новая стоимость достижения цели h(s)=2
- Оценка предыдущего состояния агента A оказалась слишком оптимистической
- Поэтому оценки увеличивается h(s)+H(s)=1+2=3



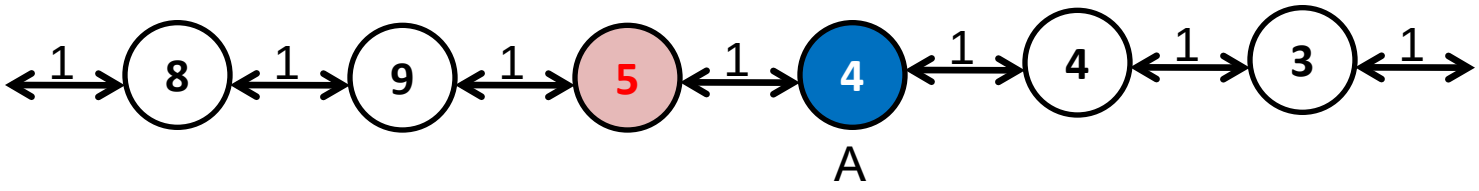
Локальный поиск в оперативном режиме

- Агент продолжает блуждание в наилучшем направлении
- Следующее движение выполняется в сторону с наименьшей стоимостью
- Новая стоимость достижения цели h(s)=3
- Оценка предыдущего состояния агента A оказалась слишком оптимистической
- Стоимость в новом состоянии равна 3
- Стоимость в предыдущем состоянии должна быть больше на один переход h(s)+H(s)=1+3=4



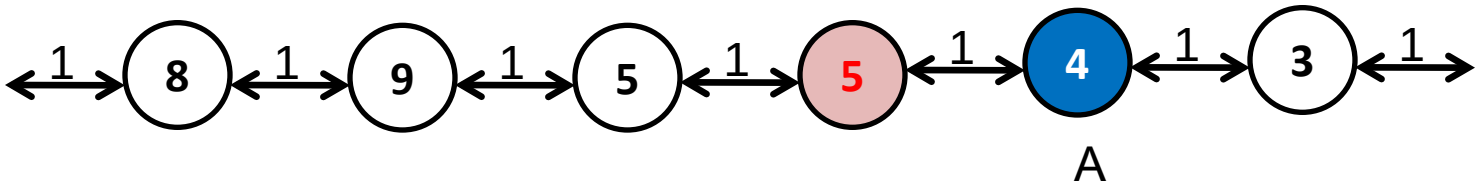
Локальный поиск в оперативном режиме

- Агент продолжает блуждание в наилучшем направлении
- Следующее движение выполняется в сторону с наименьшей стоимостью
- Новая стоимость достижения цели h(s)=4
- Оценка предыдущего состояния агента A оказалась слишком оптимистической
- Стоимость в новом состоянии равна 4
- Стоимость в предыдущем состоянии должна быть больше на один переход h(s)+H(s)=1+4=5.



Локальный поиск в оперативном режиме

- Агент продолжает блуждание в наилучшем направлении
- Следующее движение выполняется в сторону с наименьшей стоимостью
- Новая стоимость достижения цели h(s)=4
- Оценка предыдущего состояния агента A оказалась слишком оптимистической
- Стоимость в предыдущем состоянии должна быть больше на один переход, чем стоимость нового состояния h(s)+H(s)=1+4=5.
- Агент покинул **локальный минимум**



Обучение в ходе поиска в оперативном режиме

- Для безопасно исследуемой среды можно выполнять поиск в оперативном режиме.
- Если среда детерминированная, то для изучения каждого действия достаточно одного эксперимента
- Несколько экспериментов позволяют получить более точные оценки

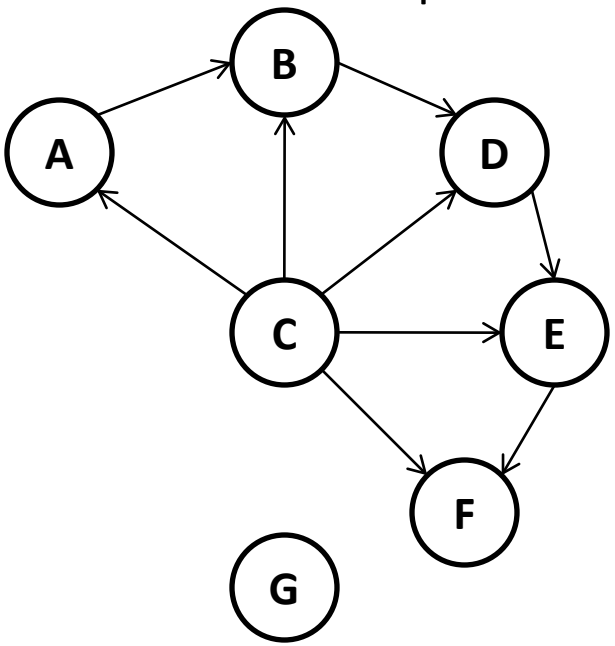
Задачи удовлетворения ограничений

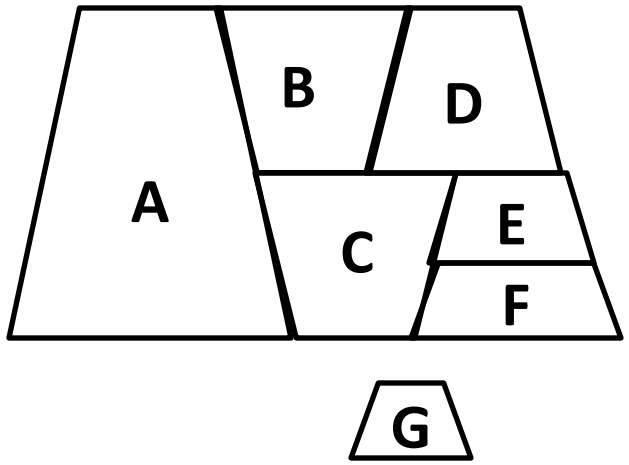
Задачи удовлетворения ограничений

- Задача удовлетворения ограничений (CSP - Constraint Satisfaction Problem) имеет множество переменных $<X_1,X_2,\ldots>$ и множество ограничений $<C_1,C_2,\ldots>$.
- Каждая переменная имеет область значений D
- Состояние задачи определяется путем присваивания значений переменным $X_1 = v_1$, $X_2 = v_2$, …
- Каждое состояние среды представляется как **черный ящик**
- Присваивание, которое не нарушает ограничений называется **совместным**
- **Решением** является совместное присваивание значений всем переменным
- Решение CSP максимизирует **целевую функцию**

Задача удовлетворения ограничений

- Имеются семь регионов <A, B, C, D, E, F, G>
- Имеется три цвета <Red, Green, Blue>
- Требуется закрасить регионы в три цвета таким образом, чтобы соседние регионы не имели одинаковый цвет
- Задача имеет множество решений





Формулировка задачи удовлетворения ограничений

- **Начальное состояние** – пустое присваивание {}
- **Функция определения преемника** – присваивание значения следующей переменной
- **Проверка цели** – является ли текущее присваивание полным
- **Стоимость пути** – постоянная величина для каждого этапа **c=1**
- **Глубина поиска** равна числу переменных **n**

- Путь, по которому достигается решение **не имеет интереса**
- **Размер задачи.** Если переменные могут принимать d значений, то количество полных присваиваний **$O(d^n)$.**
- **Тип ограничений –** унарные G>0, бинарные A=B и ограничения высокого порядка

Гиперграф ограничений высокого порядка для криптографической головоломки

$$\begin{array}{r} +\ TWO \\ \hline TWO \\ \hline FOUR \end{array}$$

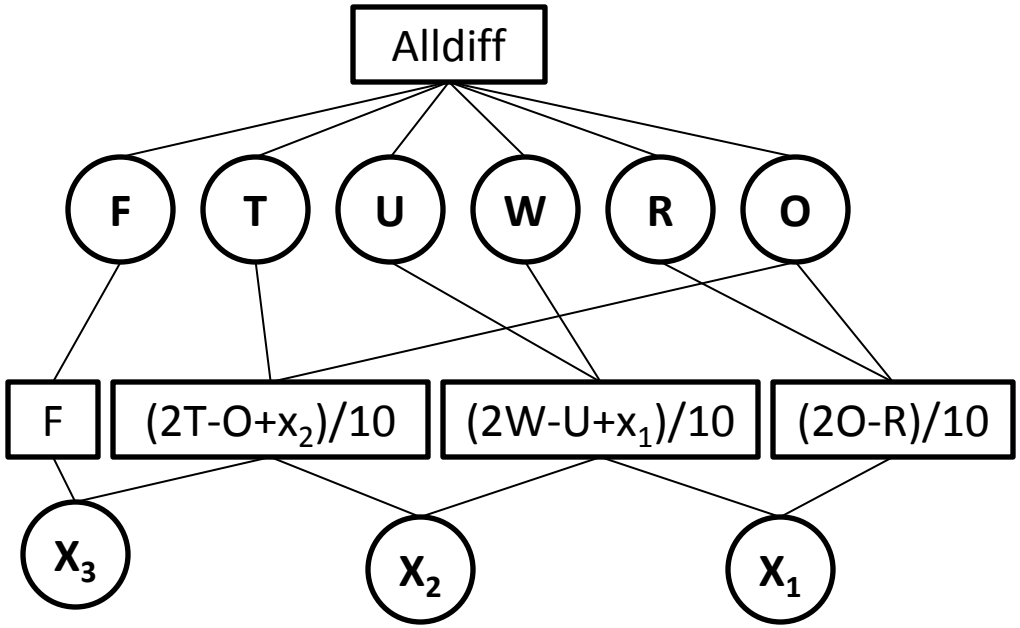
$$O + O = R + 10 \cdot x_1$$

$$x_1 + W + W = U + 10 \cdot x_2$$

$$x_2 + T + T = O + 10 \cdot x_3$$

$$x_3 = F$$

$$F \neq O, O \neq U, U \neq F, \ldots$$

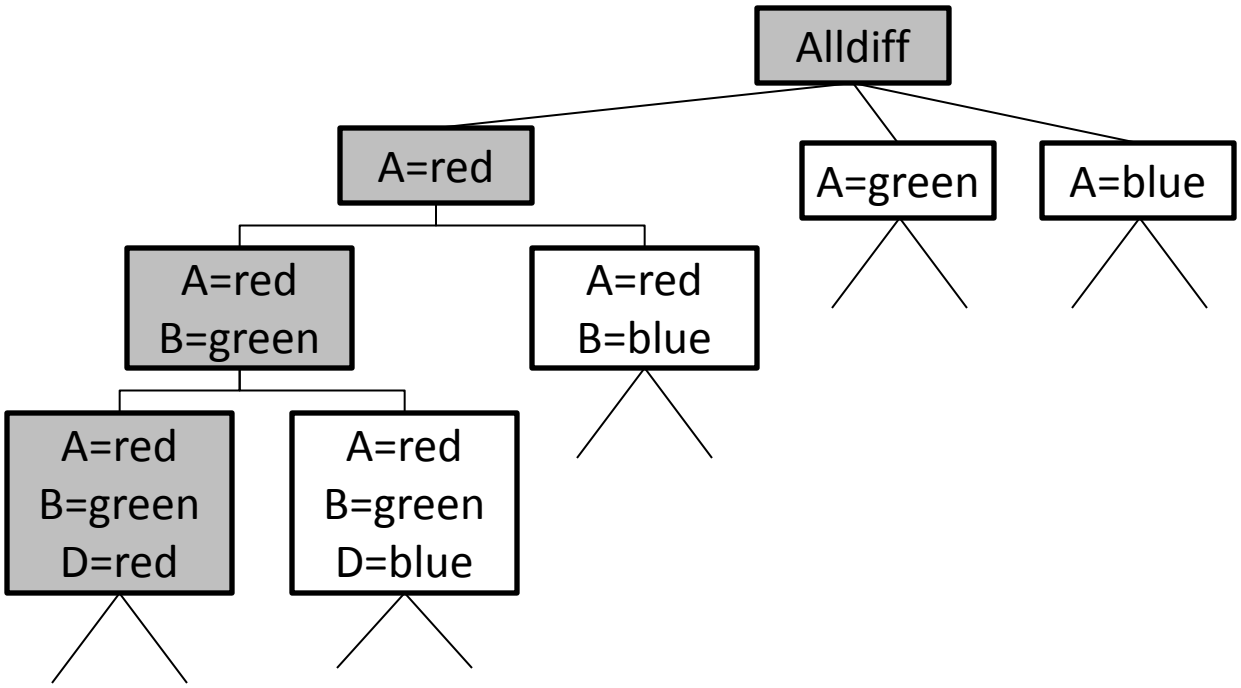


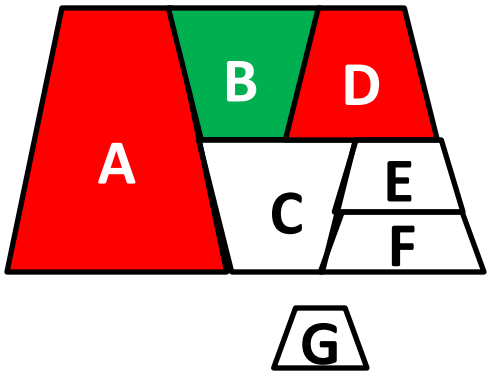
Поиск с возвратами для решения задач CSP

- CSP - задача коммутативная. **Коммутативность** – свойство задачи, когда порядок применения любого множества действий не влияет на результат
- Поиск в ширину неэффективен, поэтому для задач CSP применяется **поиск в глубину**.
- Если нет возможности присвоения значения переменной, то выполняется возврат. Поэтому для задач CSP применяется **поиск в глубину с возвратами**.

Поиск в глубину с возвратами

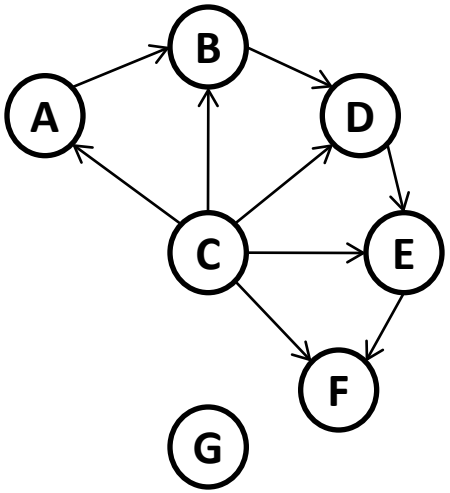
- Фрагмент дерева поиска в глубину с возвратами

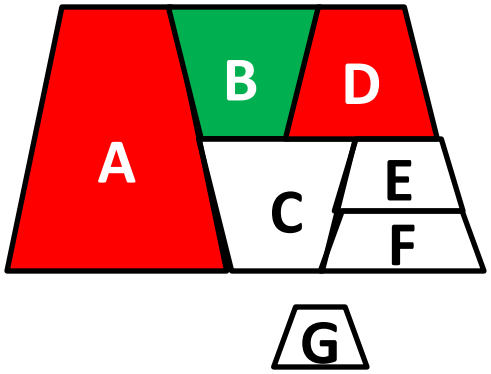




Эвристические функции

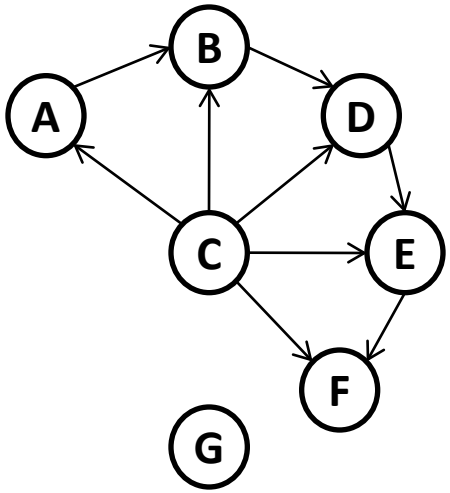
- **Эвристика с минимальным количеством оставшихся значений**. Выбор значения переменной, которое с наибольшей вероятность приведет к неудаче, чтобы усечь дерево поиска.
- **Степенная эвристика.** Выбор переменной, которая участвует в наибольшем количестве ограничений.
- **Эвристика с наименее ограничительным значением.** D=red – наименее ограничительное значение.





Метод предварительной проверки

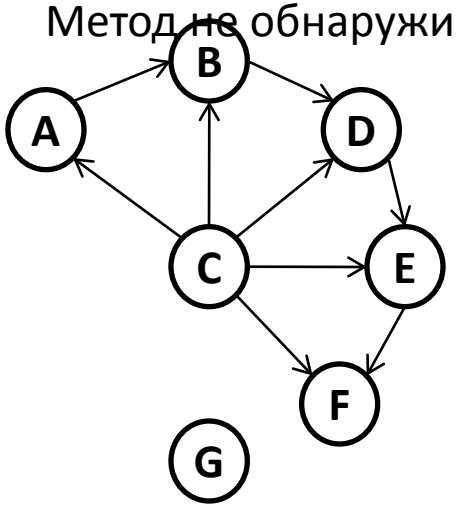
- Вначале выполняется присваивание любой переменной X.
- Затем просматриваются все переменные и для них оставляют только допустимые значения
- B и C назначаются значения **B=C=blue**



A	B	C	D	E	F	G
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	GB	RGB	RGB	RGB	RGB
R	**B**	**B**	**G**	**RB**	**RGB**	**RGB**
R	B	**нет**	**G**	R	**B**	RGB

Метод проверки совместимости дуг

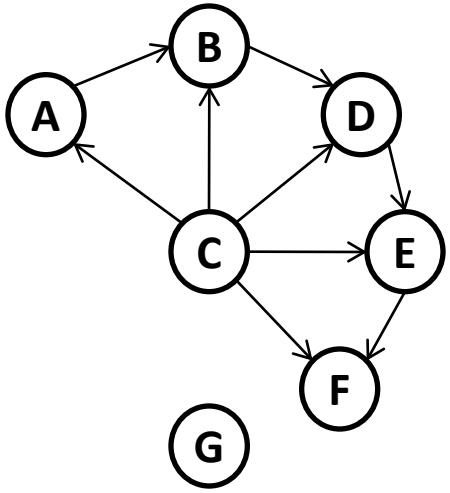
- Метод проверки совместимости дуг. Каждая дуга (Xi->Xj) проверяется по очереди
- Дуга, связывающая C и E, **совместима**, если C и E присвоены совместимые значения. Значение C=blue, E={blue, red}
- Дуга, связывающая C и B, **несовместима**, если C и B присвоены совместимые значения. Значение C={blue}, B={blue}
- Метод дает быстрое обнаружение
- Метод не обнаруживает A={red}, E={red}



A	B	C	D	E	F	G
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	GB	RGB	RGB	RGB	RGB
R	B	**B**	**G**	**RB**	RGB	RGB
R	B	**нет**	**G**	R	**B**	RGB

Метод k-совместимости

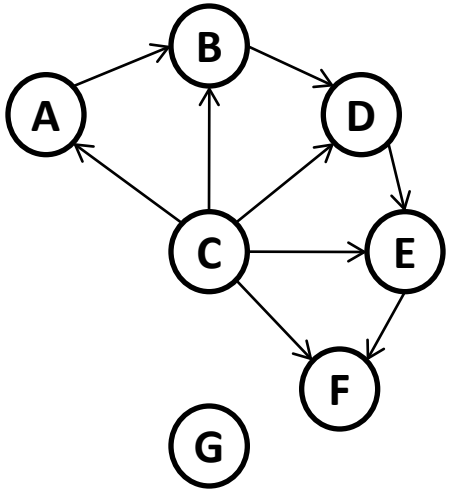
- Задача k-совместима, если для k-1 переменных, имеющих совместимые значения, можно найти совместимое значение для k-ой переменной.
- Задача 1-совместима – совместимость узлов,
- Задача 2-совместима – совместимость дуг.



A	B	C	D	E	F	G
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	GB	RGB	RGB	RGB	RGB
R	B	**B**	**G**	**RB**	RGB	RGB
R	B	**нет**	**G**	R	**B**	RGB

Ограничение AllDiff

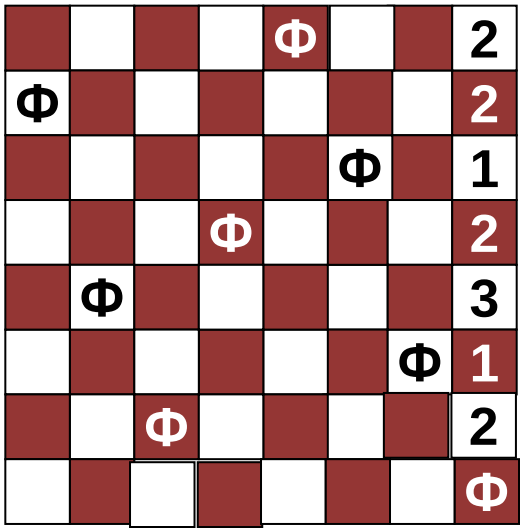
- Переменные B,C,D связаны отношением - AllDiff.

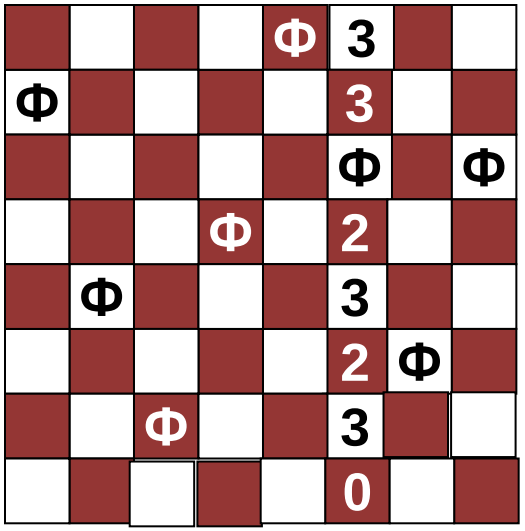


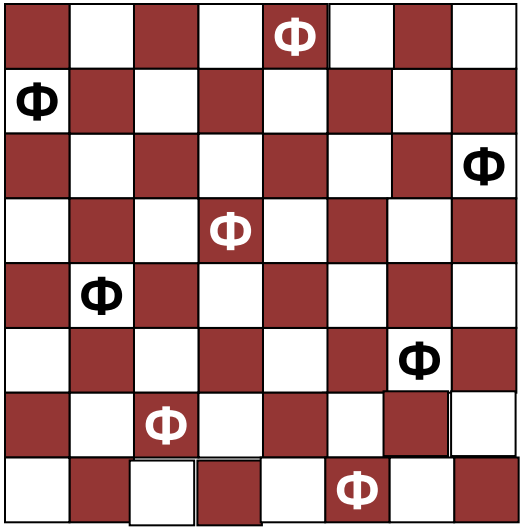
A	B	C	D	E	F	G
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	GB	RGB	RGB	RGB	RGB
R	B	**B**	**G**	**RB**	RGB	RGB
R	B	**нет**	**G**	R	**B**	RGB

Задача с восемью ферзями

- Ферзи расставлены случайно с каждом столбце
- В одном из столбцов определяем количество конфликтов.
- Перемещаем ферзь в позицию с минимальным числом конфликтов

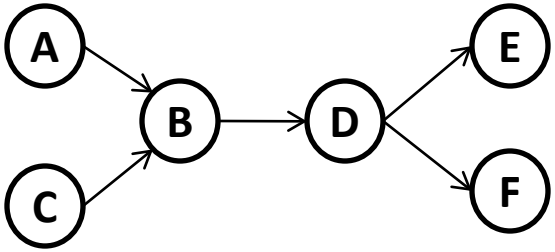


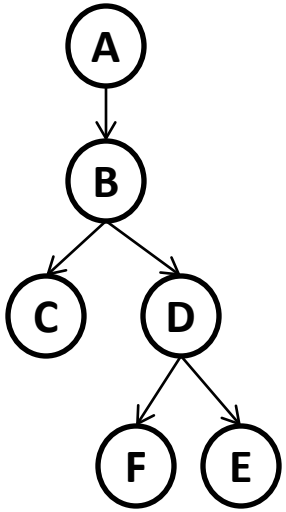




Структура задач

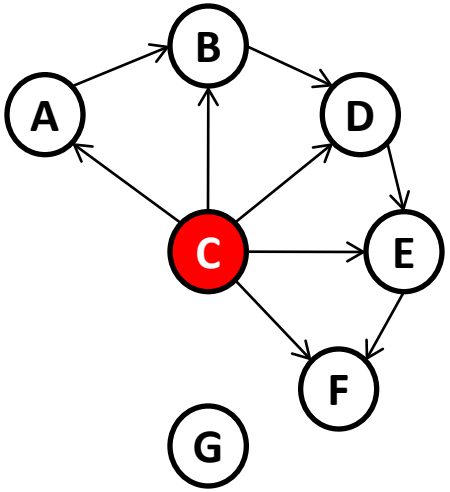
- Задача удовлетворения ограничениям может быть представлена в виде графа
- Задача удовлетворения ограничениям представленная в виде дерева имеет самую высокую скорость решения.
- В цикле выполняется проверка совмести дуг

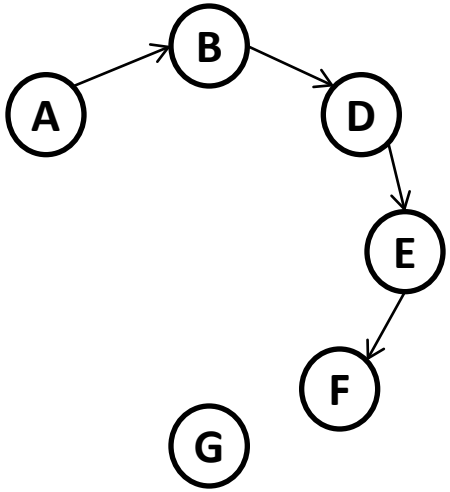




Структура задач

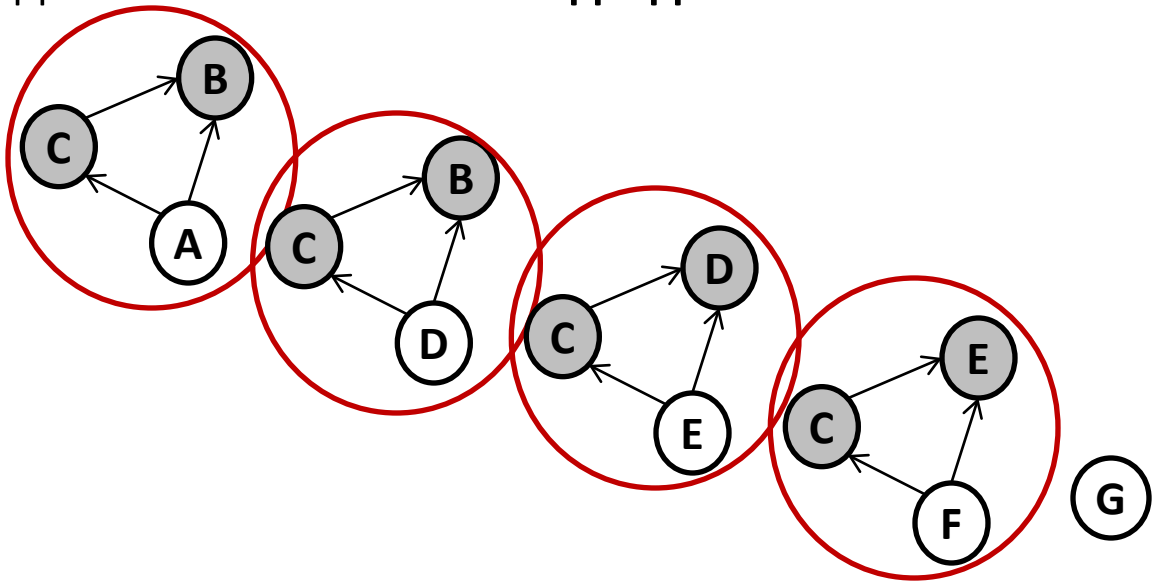
- Выполняется присваивание значений переменным так, чтобы оставшиеся переменные образовали дерево C=Red
- Остальным переменным присваиваются значения, совместимые со значением C=Red

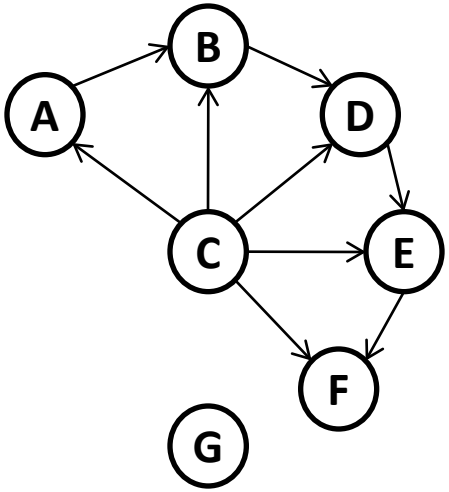




Древовидная декомпозиция

- Граф ограничений разбивается на **несколько связанных подзадач**.
- Каждая подзадача решается **независимо**.
- Каждая переменная из первой задачи появляется по меньшей мере **в одной из подзадач**.
- Если две переменные исходной задачи связаны ограничением, то они должны появляться **в подзадаче вместе**





Выводы

Выводы

Задания

- Решите криптографическую задачу вручную с помощью поиска с возвратами, предварительной проверки, а также на основе эвристик

$$\begin{array}{r} +\ TWO \\ \hline TWO \\ \hline FOUR \end{array}$$

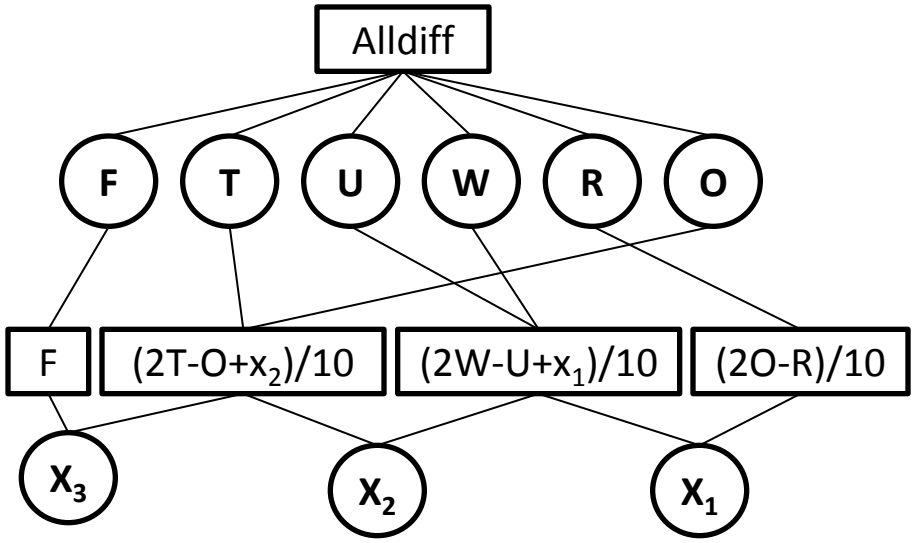
$$O + O = R + 10 \cdot x_1$$

$$x_1 + W + W = U + 10 \cdot x_2$$

$$x_2 + T + T = O + 10 \cdot x_3$$

$$x_3 = F$$

$$F \neq O, O \neq U, U \neq F, \ldots$$



Представьте в виде задачи с ограничениями

- Даны 5 домов,
  - дома имеют разный цвет,
  - в них живут лица разных национальностей,
  - которые предпочитают сигареты разных сортов,
  - пьют разные напитки,
  - держат разных питомцев.

  Вопрос:

  В каком доме держат зебру ?

  В каком доме пьют воду?

Представьте в виде задачи с ограничениями

1. Англичанин живет в доме красного цвета
2. Испанец держит собаку
3. Норвежец живет в первом доме слева
4. Сигареты «Кулс» курят в доме желтого цвета
5. Человек, курящий «Честерфилд», живет рядом с человеком, который держит лису
6. Норвежец живет рядом с домом синего цвета
7. Человек, курящий «Уинстон», держит улиток
8. Человек, курящий «Лаки страйк», пьет апельсиновый сок
9. Украинец пьет чай
10. Японец курит «Парламент»
11. Сигареты «Кулс» курят в доме, находящемся рядом с домом, где держат лошадь
12. Кофе пьют в доме зеленого цвета
13. Дом зеленого цвета находится непосредственно справа от дома слоновой кости
14. Молоко пьют в среднем доме

Принятие решений в условиях противодействия

Лекция 4

Игры

- Варианты мультиагентных сред:
  - Кооперативные,
  - Конкурентные, в которых цели игроков конфликтуют, называются **играми**.
- Конкурентные варианты среды с очень большим количеством агентов рассматривают как **экономики**.
- В реальных играх трудно найти решение. Шахматы имеют коэффициент ветвления = 35. Количество ходов каждого игрока = 50. Дерево поиска имеет число узлов $35^{100} = 10^{154}$
- В таких играх используют следующие методы:
  - Метод **усечения дерева** поиска,
  - **Эвристические функции оценки** для оценки полезности состояния без полного поиска

Теория игр

- **Теория игр** – это раздел прикладной математики, исследующий построение моделей принятия решений в условиях конфликта.
- **Конфликт** – противостояние нескольких сторон или коалиций, при котором каждый из участников желает нанести наибольший урон сторонам.
- **Игра** – формализованное описание конфликта.

$$\Gamma = \left\langle N, \{X_i\}_{i \in N}, X, \{R_i\}_{i \in N} \right\rangle$$

N – множество сторон,

$\{X_i\}$ - множество стратегий сторон,

$X \in \prod_{i \in N} X_i$ - произведение стратегий, который определяет ситуацию в игре.

$\{R_i\}$ – предпочтение каждой из сторон, например, функция полезности.

Примеры игр

- Орлянка – **антагонистическая игра** или игра **с нулевой суммой**. Интересы игроков противоположны. $X_1 = X_2 = \{орел, решка\}$
- Выигрыш одного ведет к проигрышу другого.

Игрок 2

Игрок 1		Орел	Решка
	Орел	**-1**	**1**
	Решка	**1**	**-1**

Примеры игр

- Дилемма заключенного – **биматричная игра**
- Интересы игроков **не противоположны**
- $X_1 = X_2$ ={сознаться, не сознаться}

Грабитель 2

Грабитель 1		Сознаться	Не сознаться
	Сознаться	-8 -8	0 -10
	Не сознаться	-10 0	-2 -2

Примеры игр

- Семейный спор– **биматричная не антагонистическая** игра.
- Интересы игроков не противоположны
- $X_1 = X_2 = \{$футбол, театр$\}$

Жена

Муж		Футбол	Театр
	Футбол	5 3	0 0
	Театр	0 0	3 5

Примеры игр

- **Дифференциальная игра** развертывается с течением времени
- Стратегии игроков является функцией, зависящей от времени
- Шофер-убийца управляет автомобилем, который движется со скоростью $v_ш$
- Пешеход движется со скоростью $v_п < v_ш$

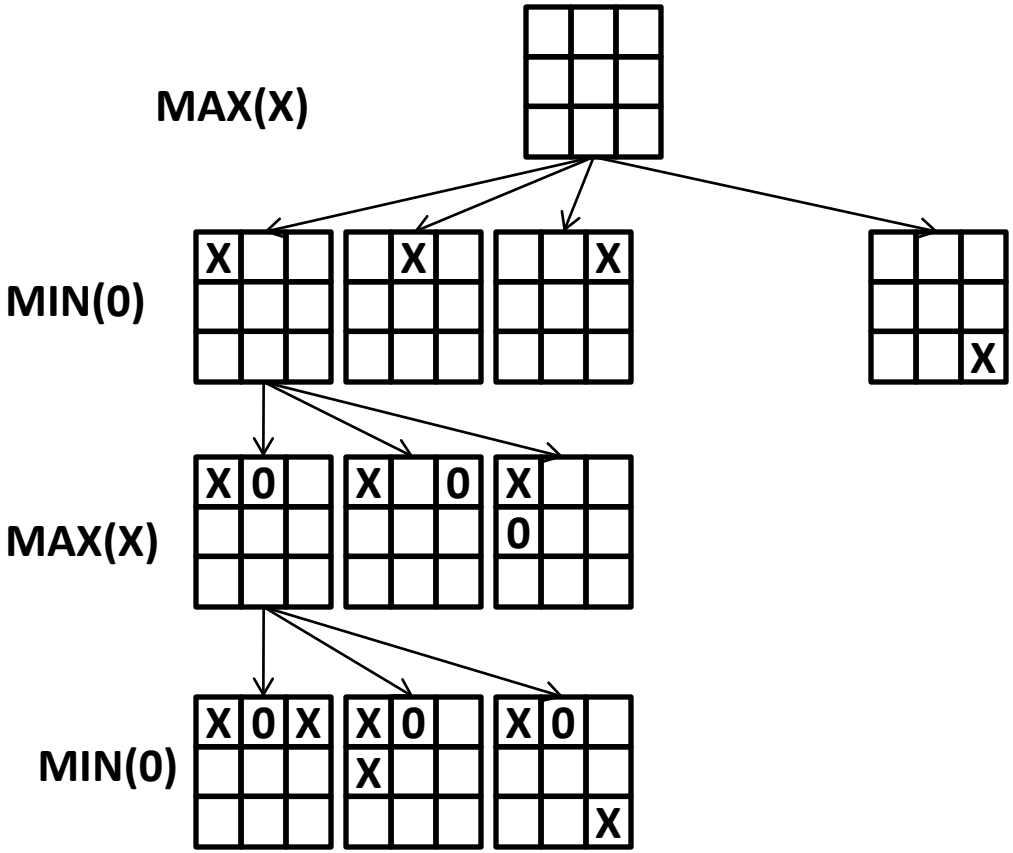
$$\dot{x}_ø = v_ø \sin\theta; \quad \dot{y}_ø = v_ø \cos\theta;$$

$$\dot{x}_ï = v_ï \sin\theta; \quad \dot{y}_ï = v_ï \cos\theta;$$

$$\theta = {v_ø}/{R} \cdot \varphi, \quad -1 \le \varphi \le 1;$$

Принятие оптимального решения в играх с двумя игроками

- **Начальное состояние** определяет состояние среды и игрока, который должен ходить первым.
- **Функция определения** преемника
- Проверка **терминального состояния** окончания игры.
- **Функция полезности** дает числовое значение терминальным состояниям
- **Дерево игры** определяет допустимые ходы.



Некооперативные игры. Предположения.

- **Игроки ведут себя рационально.** Полная рациональность требует точной оценки игроками долгосрочных последствий своих действий. Поэтому приемлема концепция **ограниченной рациональности**.

- Полная информированность. Информация о функциях выигрыша является **общим знанием**. Каждый игрок **знает** функции выигрыша, остальные игроки знают, что они знают.

- **Независимость стратегий.** Игрок не знает, какие стратегии выбрали остальные игроки. Но игрок знает функции выигрыша и может предполагать , каковы стратегии остальных.

- **Однократность взаимодействия.** Отсутствуют месть, благодарность и др.

Осторожные стратегии в антагонистических играх

- Игрок 1 не знает предпочтений других игроков. В этих условиях при выборе своей стратегии $\mathbf{x_1}$ он определяет наименьший выигрыш, для всех возможных стратегий $\mathbf{x_2}$ других игроков

$$\mathbf{v(x_1) = \min_{x2} v(x_1, x_2);}$$

- **Осторожный игрок** выбирает стратегию, приносящую наибольший гарантированный выигрыш:

$$\mathbf{A = \max_{x1} v(x_1) = \max_{x1} \min_{x2} v(x_1, x_2);}$$

- Никакое другое решение других игроков не может дать игроку 1 меньший выигрыш.
- А – гарантированный выигрыш

	X2(1)	X2(2)	X2(3)	$\min_y$
X1(1)	1	2	3	1
X1(2)	4	5	6	4
X1(3)	7	8	9	7
$\max_x$	7	8	9	7

Осторожные стратегии в антагонистических играх

- Игрок 1 имеет полную информацию о действиях других игроков. Другие игроки уже сделали свой ход. Игрок 1 определяет свою стратегию так, чтобы другие игроки получили наименьший выигрыш:

$$B_1 = \min_{x2} \max_{x1} v(x_1, x_2);$$

- Наличие информации о действиях других игроков дает возможность игроку 1 получить более высокий выигрыш:

$$B_1 = \min_{x2} \max_{x1} v(x_1, x_2) >= A_1 = \max_{x1} \min_{x2} v(x_1, x_2);$$

- Игрок 1 может получить выигрыш меньше A_1 только если ведет себя глупо или азартно.
- Игрок 1 может получить выигрыш в диапазоне $[A_1, B_1]$ в условиях противодействия за счет информированности других игроков.
- Игрок 1 может получить выигрыш B_1 в условиях полной информированности, когда игрок1 знает действия других игроков.

Антагонистические игры двух игроков

- Антагонистической называется игра двух лиц

$$\Gamma = \langle 1,2,\{X,Y\},\{v_1(x,y),v_2(x,y)\}\rangle$$

- В которой выигрыш одного игрока равен проигрышу второго игрока

$$v_1(x,y) = -v_2(x,y)$$

Антагонистическая игра полностью характеризуется одной функцией выигрыша *v(x,y)*

Осторожные стратегии в антагонистических играх

- Используя осторожные стратегии, игрок 1 может выиграть не более A:

$$A = \max_x \min_y v(x, y);$$

- Используя осторожные стратегии, игрок 2 может проиграть не более B:

$$B = \min_y \max_x v(x, y);$$

- Игра имеет единственный приемлемый для двух игроков исход, если A=B.

$$A = \max_x \min_y [v(x, y)] = -\min_y \max_x v(x, y) = B$$

- Значение **A=B** называется **ценой игры**.
- Если игра имеет цену, то осторожные стратегии игроков представляют для них **единственное разумное** решение.
- Никому не выгодно отклоняться от данных стратегий.

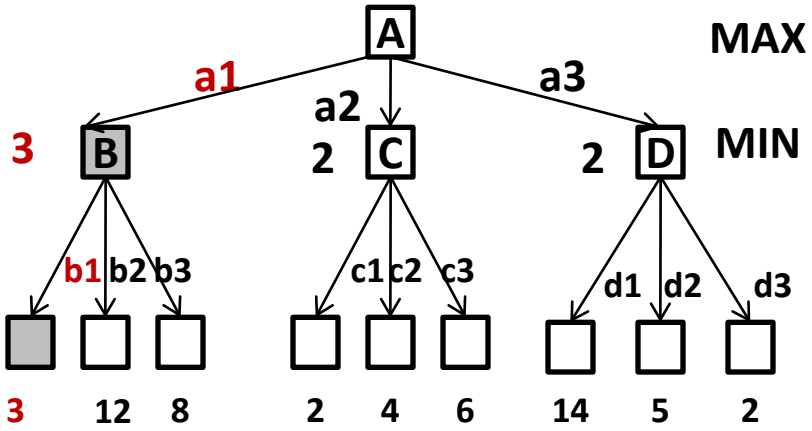
Осторожные стратегии в антагонистических играх

- Антагонистическая игра имеет цену тогда и только тогда, когда функция выигрыша имеет седловую точку
- Пример антагонистической игры с матрицей выигрышей
- **$A = \max_x \min_y [v(x, y)] = -\min_y \max_x v(x, y) = B$**

	y2	y3	y2	$\min_y$
x1	1	2	3	1
x2	4	5	6	4
x3	7	8	9	7
$\max_x$	7	8	9	7

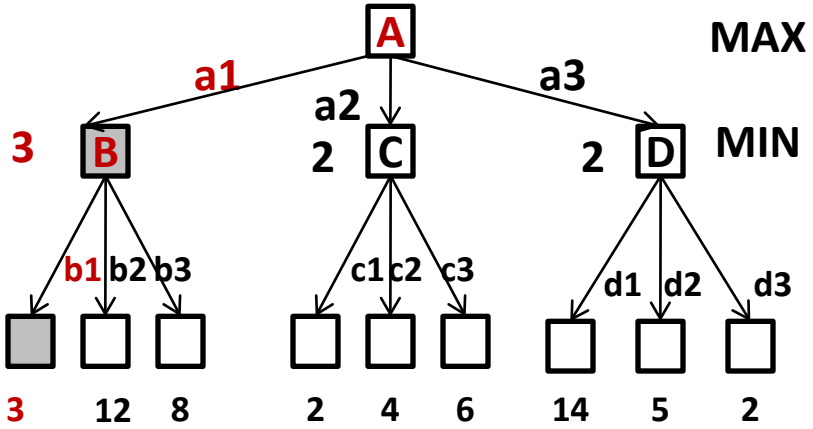
Минимаксные оптимальные стратегии

- **Стратегия** определяет для каждого игрока ходы,
- **Оптимальная стратегия** должна принести максимальную полезность,
- Каждый участник MIN и MAX делают **полуходы.**
- MAX выбирает ход из a1, a2, a3
- Лучший ход MAX – это a1 – **максимум полезности** для MAX.
- Полезность хода a1 равна 3.
- Лучший ответный ход MIN – это b1 - **минимум полезности** для MAX
- Оптимальная стратегия- **минимаксная**



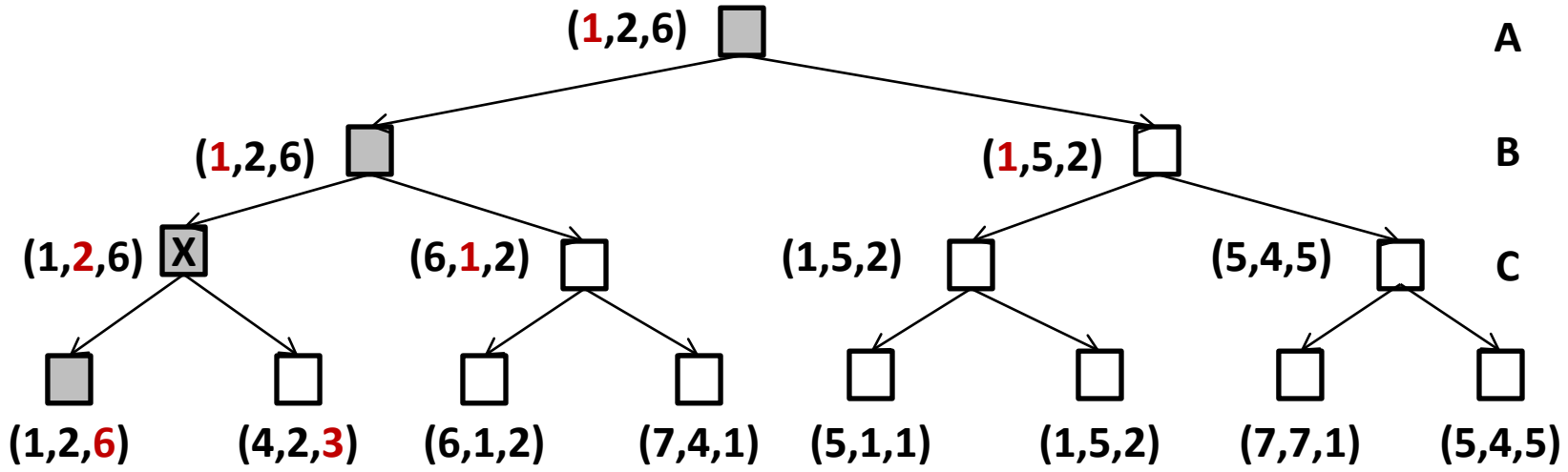
Минимаксный алгоритм выбора оптимальной стратегии

- Минимаксный алгоритм рекурсивно вычисляет **минимаксное значение** функции полезности,
- Минимаксный алгоритм выполняет **полное** исследование дерева игры,
- Сложность алгоритма **$O(b^m)$,**
  - где **b**-число допустимых ходов,
  - **m** – глубина поиска ,
- Такой алгоритм неприемлем в практике



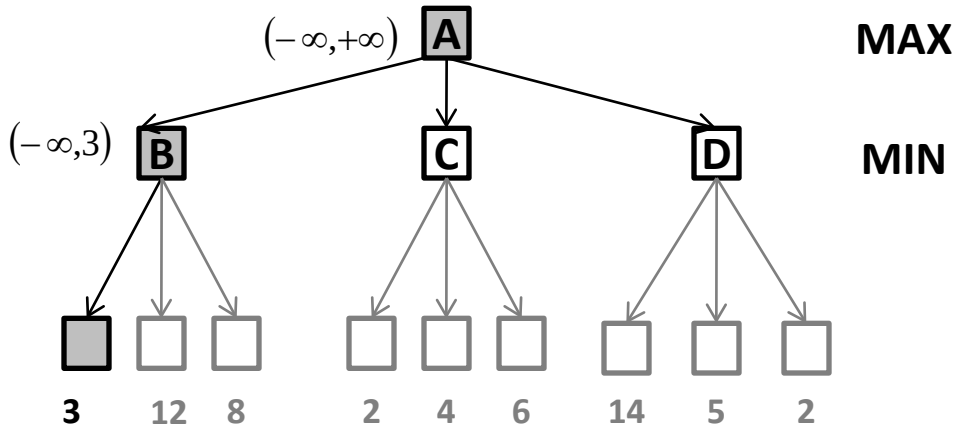
Игры с несколькими игроками

- Игроки **A,B,C.**
- Узлы дерева игры имеют **функцию полезности** (v_A,v_B,v_C)
- На каждом шаге игрок выбирает ход с **максимальным** значением функции полезности
- Возможно образование коалиций между игроками



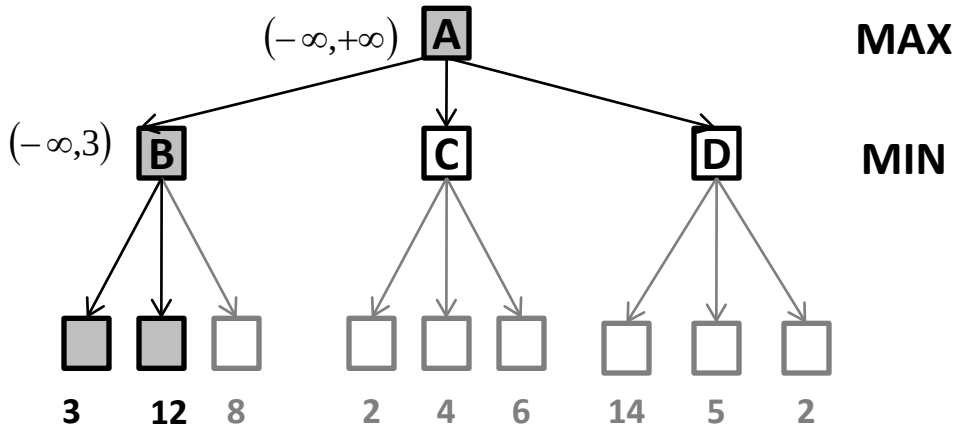
Альфа-бета отсечение

- Метод альфа-бета отсечения предназначен для быстрого поиска **минимаксного значения** по дереву решения
- Игрок MIN имеет самое большее значение полезности =3



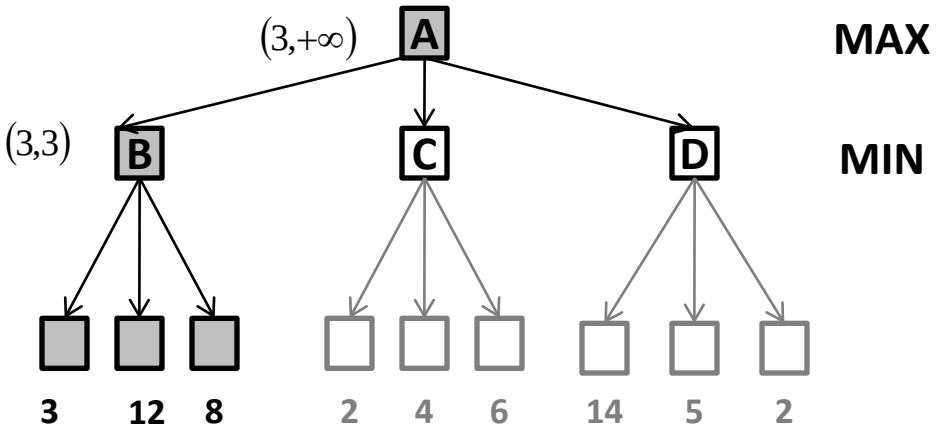
Альфа-бета отсечение

- Игрок MIN имеет самое большее значение полезности =3
- Игрок MIN избегает узла с полезностью =12



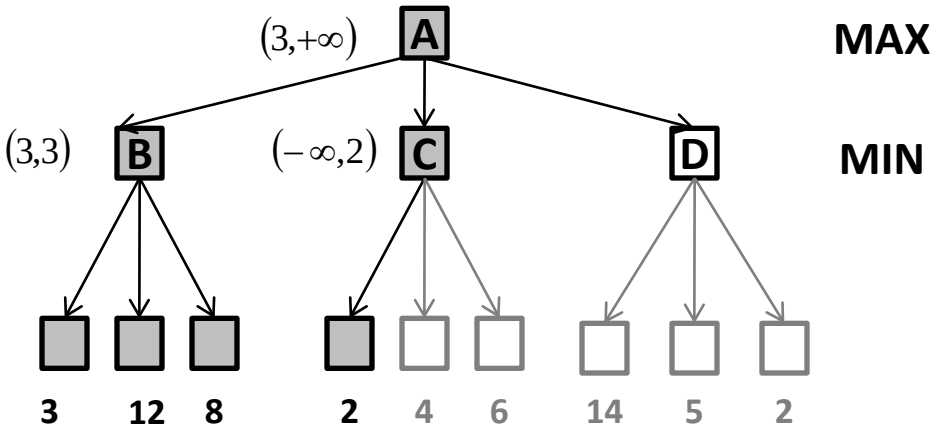
Альфа-бета отсечение

- Игрок MIN имеет самое большее значение полезности =3
- Игрок MIN избегает узла с полезностью =12
- Игрок MIN не выбирает узла с полезностью =8
- Проверили все узлы B.



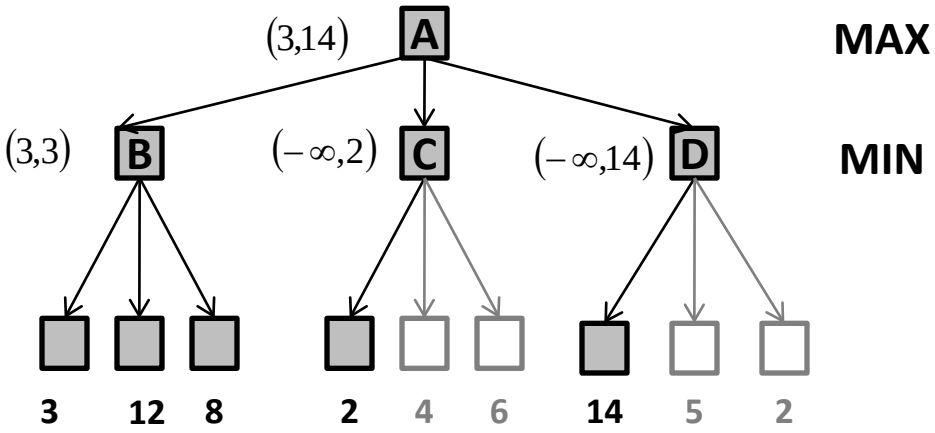
Альфа-бета отсечение

- Игрок MAX не будет выбирать узел C
- Нет смысла проверять другие терминальные вершины узла C



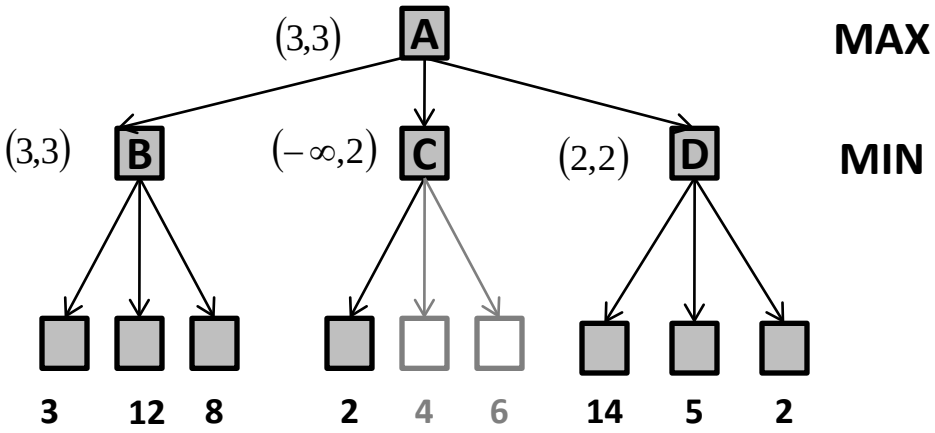
Альфа-бета отсечение

- Первый лист имеет полезность =14.
- Узел D имеет самое большое значение =14.
- Самое большее значение полезности корневого узла тоже = 14.



Альфа-бета отсечение

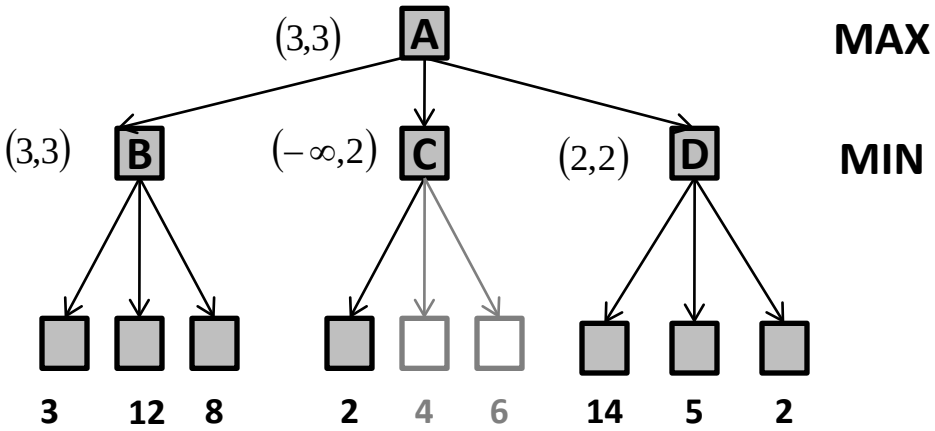
- Второй лист имеет полезность = 5
- Продолжаем исследование
- Третий лист имеет полезность = 2
- Делаем вывод, что MAX в корневом узле принимает решение B



Альфа-бета отсечение

Алгоритм альфа-бета отсечения:

- Minimax = max(min(3, 12, 8), min(2, x, y), min(14, 5, 2))
- = max(3, min(2, x, y), 2)
- = max(3, z, 2) где z<=2
- =3;



Альфа-бета отсечение

- **Альфа** – значение наилучшего варианта (с **наибольшим** значением), который до сих пор был найден в любой точке выбора вдоль пути игрока **MAX**.
- **Бета** – значение наилучшего варианта (с **наименьшим** значением), который до сих пор был найден в любой точке выбора вдоль пути игрока **MIN**.
- Алгоритм альфа-бета отсечения позволяет **отсекать ветви** и **поддеревья** алгоритма
- **Эффективность алгоритма** зависит от порядка проверки узлов.
- Алгоритм поиска имеет трудоемкость **$O(b^m)$.** (Для шахмат b=35).
- Алгоритм альфа-бета отсечения в наилучшем случае имеет трудоемкость **$O(b^{m/2})$.** (Для шахмат $b^{1/2}=6$).

Неидеальные решения, принимаемые в реальном мире

- Реальные программы должны
  - Оканчивать проверку раньше времени
  - Применять функцию оценки

Функции оценки

- Должна упорядочивать терминальные состояния
- Должна быстро вычисляться
- Прогнозировать шансы на выигрыш
- Останавливать поиск

Функции оценки в шахматной игре

- Количество пешек,
- Вероятность выигрыша в данной позиции в %,
- Стоимость материала (пешка = 1, конь и слон = 3, ладья = 5, ферзь = 9)

Смешанные стратегии

- Множество стратегий X называется **чистыми стратегиями.**
- Распределение вероятностей на множестве X называется **смешанной стратегией.**
- Игрок использует стратегии в соответствии с вероятностями.

- Множества смешанных стратегий

$$D(X_i)=\{p=(p_{x1}, p_{x2}, \ldots) \mid p_{x1} >= 0, p_{x2} >= 0, \ldots, \sum p_{xi} = 1\}$$

- Функции выигрыша

$$V(p_1,\ldots,p_N)= \sum_{X1} \ldots \sum_{XN} v_i(x_1,\ldots,x_N)\, p(x_1) \ldots p(x_N)$$

- $p(x_1)$ – вероятности использования игроком i-ой чистой стратегии

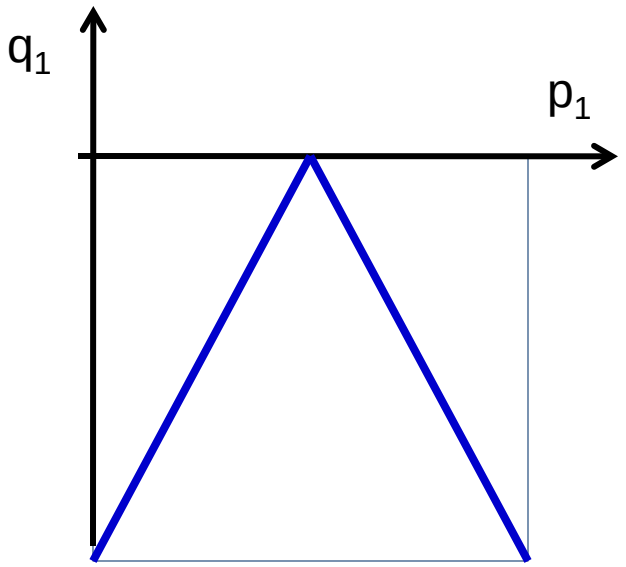
Смешанные стратегии

Игра в орлянку

- Множества смешанных стратегий
- $D(X) = \{p=(p_x, p_y) \mid p_x >= 0, p_y >= 0, p_x + p_y = 1\}$
- Ожидаемый выигрыш
- $V = p_1 q_1 - p_1 q_2 - p_2 q_1 + p_2 q_2$;

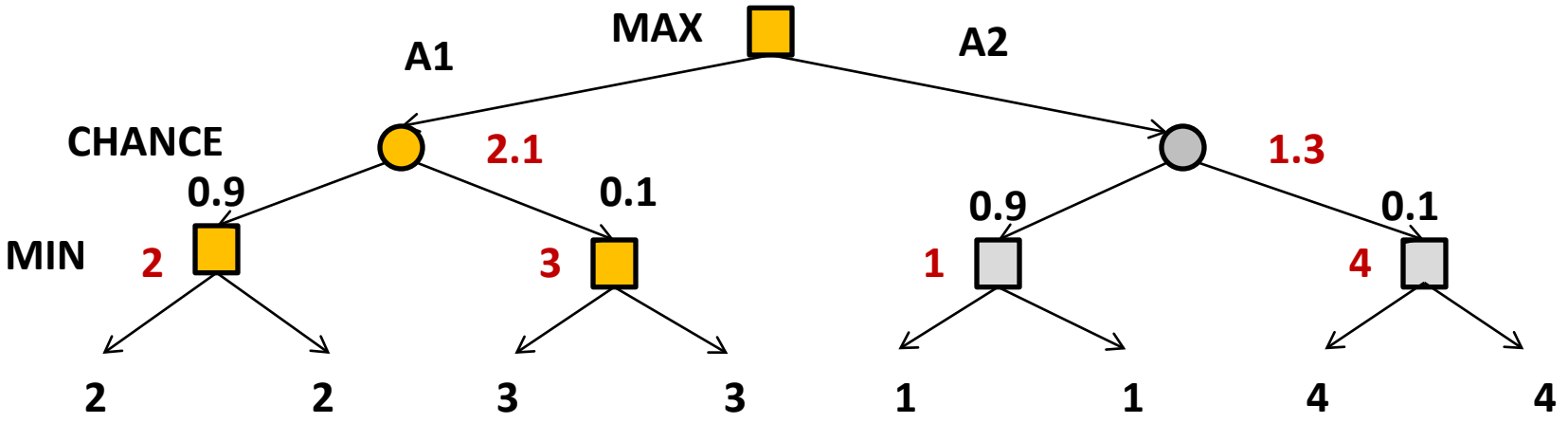
Игрок 2

Игрок 1		Орел	Решка
	Орел	$-p_1 q_1$	$p_1 q_2$
	Решка	$p_2 q_1$	$-p_2 q_2$



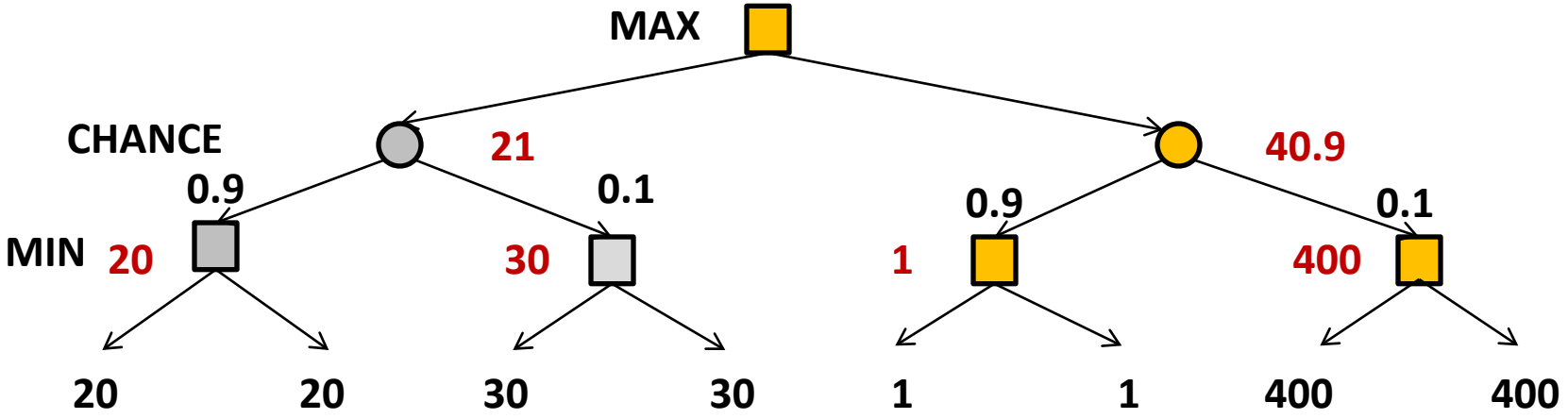
Поиск в условиях противодействия

- Дерево игры включает **узлы жеребьевки**, в которых случайным образом выбирается стратегия
- Это позволяет вычислить только ожидаемое значение выигрыша
- Решением является **ожидаемое минимаксное значение**
- Наилучший ход A1



Поиск в условиях противодействия

- Дерево игры включает **узлы жеребьевки**, в которых случайным образом выбирается стратегия
- Это позволяет вычислить только ожидаемое значение выигрыша
- Решением является **ожидаемое минимаксное значение**
- Наилучший ход A2



Логические агенты

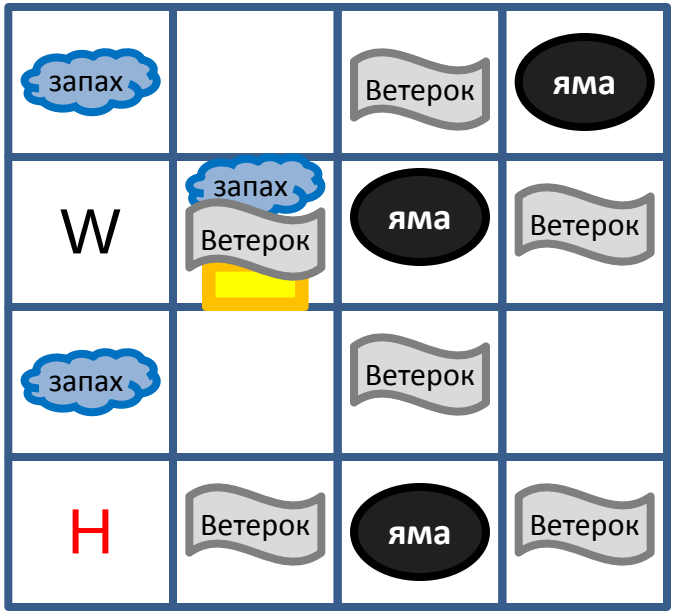
Лекция 5

Агенты, основанные на знаниях

- Агент имеет **базу знаний**.
- База содержит множество **высказываний**.
- Высказывание выражено на **языке представления знаний**.
- Извлечение знаний из базы называется **логическим выводом**.
- Первоначально база содержит **фоновые знания**. Фоновые (первоначальные) знания предоставляет проектировщик.
- **Декларативный подход** создания агентов основан на высказываниях, выраженных на языке представления знаний.
- **Процедурный подход** основан на реализации кода программ.

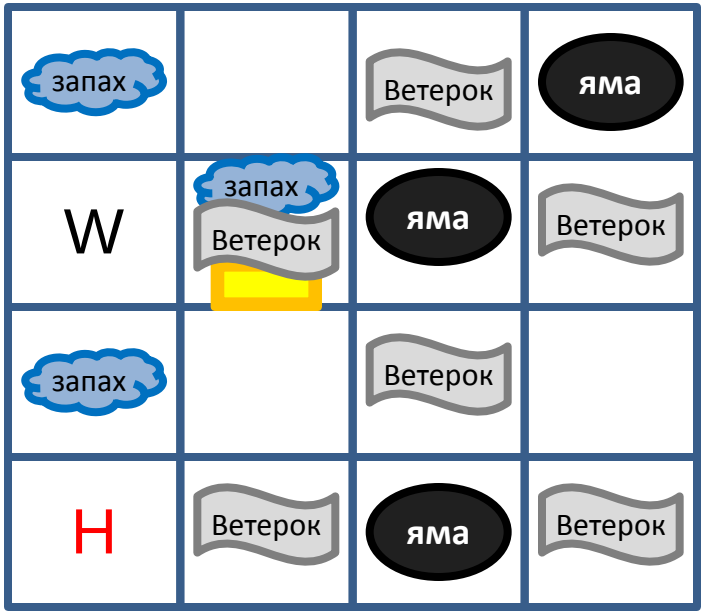
Мир Вампуса

- Агенты имеют базу знаний.
- База знаний содержит множество высказываний
- Мир Вампуса – это пещера, состоящая из залов.
- **W** - Вампус – зверь, поедает всех в своем зале.
- **H** - Охотник ищет в пещере золото.
- Охотник может убить Вампуса единственной стрелой в прямом направлении
- **Яма**, в которую может провалиться охотник.
- **Среда** – поле 4*4, в которой местоположение ямы, золота и Вампуса выбирается случайно.



Мир Вампуса

- **Датчики:**
  - Датчик Запаха **(Stench)**
  - Датчик Ветра **(Breeze)**
  - Датчик Блеска (золота) **(Glitter)**
  - Датчик удара (о стену) **(Bump)**
  - Датчик звука (крик пораженного Вампуса) **(Scream)**
- **Исполнительные механизмы:**
  - Движение вперед.
  - Поворот на 90 градусов.
  - Выстрел вперед (единственной стрелой)



Мир Вампуса

- Начало
  - Агент знает правила действующие в среде
  - Агент знает свои координаты (4,1)
  - Агент не знает конфигурацию среды.
- Агент выполняет измерения:
  - **(Stench, Breeze, Glitter, Bump, Scream)**
- Результат измерения:
  - **(none,none,none,none,none);**
- Логический вывод:
  - Соседние квадраты **безопасны**

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1 **(OK)**	3,2	3,3	3,4
H **(OK)**	4,2 **(OK)**	4,3	4,4

Мир Вампуса

- Агент переходит в (4,2)
- Агент выполняет измерения:
  - **(Stench, Breeze, Glitter, Bump, Scream)**
- Результат измерения:
  - **(none, Breeze, none, none, none);**
- Логический вывод:
  - В соседних квадратах 3,2 или 4,3 или в обоих вместе **может быть яма**

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1 **(OK)**	3,2 P?	3,3	3,4
4,1 (OK)	H breeze **(OK)**	4,3 P?	4,4

Мир Вампуса

- Агент переходит в (3,1), поскольку он боится перейти в 3,2 или 4,3 где может быть яма
- Агент выполняет измерения:
  - **(Stench, Breeze, Glitter, Bump, Scream)**
- Результат измерения:
  - **(Stench, none, none, none, none);**
- Логический вывод:
  - Отсутствует яма в 3,2
  - Квадрат 3,2 безопасен
  - Яма находится в 4,3
  - Вампус находится в 3,1

  Логический вывод объединяет знания, полученные на разных этапах

1,1	1,2	1,3	1,4
W! 2,1	2,2	2,3	2,4
H (OK) Stench	3,2 **(OK)**	3,3	3,4
4,1 (OK)	breeze **(OK)**	4,3 P!	4,4

Мир Вампуса

- Агент переходит в безопасный квадрат (3,2), затем в 2,2:
- Агент выполняет измерения:
  - **(Stench, Breeze, Glitter, Bump, Scream)**
- Результат измерения:
  - **(Stench, Breeze, Glitter, none, none);**
- Логический вывод:
  - Цель достигнута, можно забрать золото

Агент выводит заключение из доступной ему информации

1,1	1,2	1,3	1,4
W! 2,1	H Stench Breeze Glitter	2,3	2,4
3.1 (OK) Stench	3,2 (OK)	3,3	3,4
4,1 (OK)	breeze (OK)	4,3 P!	4,4

Логика

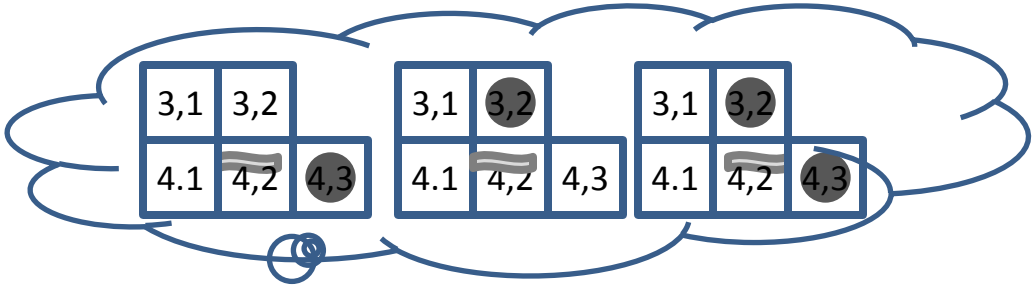
- Высказывания выражаются в соответствии с **синтаксисом языка**.
- Логика высказывания определяет **семантику языка**.
- **Семантика** определяет истинность высказывания относительно модели мира.
- **Логическое следствие** между высказываниями означает, что одно вытекает из другого.
- Высказывание α **влечет за собой** высказывание β:

$$\alpha \mapsto \beta$$

Если α истинно, то β тоже истинно.

Логический вывод с проверкой по моделям

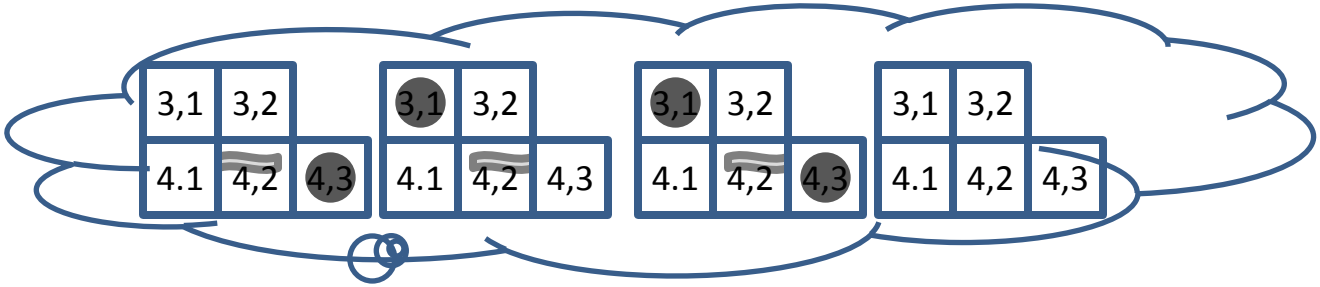
- **Модели базы знаний**
- **Исследование:** В квадрате (4,1) нет ничего, а в квадрате (4,2) чувствуется ветерок.
- **Высказывание α_1 :** «в квадрате 3,1 нет ямы»



- Модели базы знаний

• **Исследование:** В квадрате 3,1 не чувствуется ветерок, а в квадрате (4,2) чувствовался ветерок.

•**Высказывание α_2** - «в квадрате (3,2) нет ямы»



Алгоритм логического вывода

- Свойства алгоритма логического вывода:
  - **Непротиворечивость** (алгоритм вывода по моделям является непротиворечивым)
  - **Полнота –** способность вывести высказывание, которое следует из КВ.
- Если база знаний является истинной в реальном мире, то любое высказывание α, полученное непротиворечивой процедурой логического вывода, тоже **является истинным.**

Пропозициональная логика

- **Пропозициональная** логика (Дж. Буля) является способом определения истинности высказываний.

Синтаксис

- **Атомарные** высказывания состоят из символа логики.
- Например, $\mathbf{W_{2,1}}$ - Вампус находится в клетке (2,1).
- **Сложные** высказывания формируются из атомарных высказываний и **логических связок**.

Пропозициональная логика

$\neg W_{2,1}$ - отрицание (в клетке 2,1 нет Вампуса)

$W_{2,1} \wedge P_{1,3}$ - конъюнкция

$(W_{2,1} \wedge P_{1,3}) \vee W_{2,2}$ - дизъюнкция

$\Rightarrow$ - влечет за собой (импликация) , например, $(W_{2,1} \wedge P_{1,3}) \Rightarrow \neg W_{2,2}$

$\Leftrightarrow$ - если и только если (двухсторонняя импликация) $W_{3,1} \Leftrightarrow W_{2,2}$

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1 **(OK)**	3,2 **P?**	3,3	3,4
4,1 (OK)	H breeze **(OK)**	4,3 **P?**	4,4

Синтаксис пропозициональной логики

- Синтаксис определяет **порядок предшествования** в пропозициональной логике
- Синтаксис определяет **эквивалентные высказывания**

$$\begin{cases} \neg P \vee Q \wedge R \Rightarrow S \\ \left(\left(\neg P\right) \vee \left(Q \wedge R\right)\right) \Rightarrow S \end{cases}$$

Семантика пропозициональной логики

- **Семантика** определяет, как следует вычислять истинное значение.
- Правила определения истинности должны быть обобщены в **таблице истинности.**
- Высказывание, оцениваемое по модели:

$$\neg P_{4,2} \wedge \left(P_{3,2} \vee P_{4,3}\right)$$

- приводит к получению:

$$true \wedge \left(false \vee true\right)$$

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1 **(OK)**	3,2 **P?**	3,3	3,4
4,1 (OK)	H breeze **(OK)**	4,3 **P?**	4,4

Семантика пропозициональной логики

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

$P \Rightarrow Q$

Истинность P влечет за собой Q
Если P истинно, то утверждаем, что Q истинно.
Если P ложно, то нет никаких утверждений

$$(P \Rightarrow Q) = \overline{P} \vee Q$$

Простая база знаний

- В базе знаний рассматриваем только ямы. Высказывания базы знаний для каждого **(i,j):**

  В квадрате (4,1) отсутствует яма:

  $R_1: \quad \neg P_{4,1}$

  В квадрате чувствуется ветерок тогда и только тогда, когда в соседнем квадрате есть яма

  $R_2: \quad B_{4,1} \Leftrightarrow (P_{4,2} \vee P_{3,1})$

  $R_3: \quad B_{3,1} \Leftrightarrow (P_{4,1} \vee P_{3,2} \vee P_{2,1})$

  В квадрате (4,1) нет ветерка

  $R_4: \quad \neg B_{4,1}$

  В квадрате (3,1) нет ветерка

  $R_5: \quad \neg B_{3,1}$

1,1	1,2	1,3	1,4
2,1 **P?**	2,2	2,3	2,4
3,1 p?	3,2 **P?**	3,3	3,4
4,1 **P?**	4,2 **P?**	4,3	4,4

База знаний

$B_{4,1}$	$B_{4,2}$	$P_{4,2}$	$P_{4,3}$	$P_{3,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	false	false	false	false	false	true	true	true	true	false	false
false	false	false	false	false	true	true	true	false	true	false	false
false	false	false	false	true	false	true	true	false	true	true	false
…	…	…	…	…	…	…	…				
true	true	true	true	true	true	false	true	true	false	true	false

R_1 - в квадрате (4,3) отсутствует яма,
R_2 – ветерок в (4,1), следовательно яма в квадратах (3,1) или (4,2), (для каждого квадрата)
R_3 – ветерок в (4,2), следовательно яма в квадратах (3,1) или (4,1) или (4,3) (для каждого квадрата),
R_4 – в квадрате (4,1) не чувствуется ветерка
R_5 – в квадрате (4,2) есть ветерок

$$KB = R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$$

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1 **(OK)**	3,2 P?	3,3	3,4
4,1 (OK)	H breeze **(OK)**	4,3 P?	4,4

Правила логического вывода

Правила устранения импликации

$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \Rightarrow \neg\beta)$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$$
$$(\alpha \Leftrightarrow \beta) \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$

Правила де-Моргана

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$$
$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$$

Правила логического вывода

- Алгоритм логического вывода должен быть
  - **Непротиворечивым**
  - **Полным,** поскольку может применяться для любого высказывания

Правило удаления связки «и».

$$a \Rightarrow \frac{a \wedge b}{b}$$

Например, $(\text{WumpusAhead} \wedge \text{WampusAlive}) \Rightarrow \text{WampusAlive}$

Правило удаления двусторонней импликации

$$\frac{a \Leftrightarrow b}{(a \Rightarrow b) \wedge (b \Rightarrow a)}$$

$$\frac{(a \Rightarrow b) \wedge (b \Rightarrow a)}{a \Leftrightarrow b}$$

Правила логического вывода для мира Вампуса

Доказательство того, что в квадрате (4,2) нет ямы.

Применим правило R_2 – «в квадрате чувствуется ветерок тогда и только тогда, когда в соседнем квадрате есть яма» $R_2: \quad B_{4,1} \Leftrightarrow (P_{4,2} \vee P_{3,1})$

$$R_6: \quad (B_{4,1} \Rightarrow (P_{4,2} \vee P_{3,1})) \wedge ((P_{4,2} \vee P_{3,1}) \Rightarrow B_{4,1})$$

Применим правило связки «И»

$$R_7: \quad (P_{4,2} \vee P_{3,1}) \Rightarrow B_{4,1}$$

Применим правило логической эквивалентности отрицаний

$$R_8: \quad \neg B_{4,1} \Rightarrow \neg(P_{4,2} \vee P_{3,1})$$

Применим правило отделения к R8 и R4

$$R_9: \quad \neg(P_{4,2} \vee P_{3,1})$$

Применим правило Де Моргана

$$R_{10}: \quad \neg P_{4,2} \wedge \neg P_{3,1}$$

Доказано, что ямы нет ни в (4,2) ни в (3,1)

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1 **P?**	3,2	3,3	3,4
4,1 (OK)	4,2 **P?**	4,3	4,4

Правило резолюции для мира Вампуса

Правило резолюции – это правило логического вывода, который становится полным.

Введем факты в KB:
Агент возвращается из (4,2) в (4,1) и переходит в (3,1)

$$R_{11}: \quad \neg B_{4,1}$$

$$R_{12}: \quad B_{4,2} \Leftrightarrow \left(P_{4,1} \vee P_{3,2} \vee P_{4,3}\right)$$

С помощью предыдущего доказательства делаем заключение об отсутствии ям в квадратах (3,2) И (4,1)

$$R_{13}: \quad \neg P_{3,2}$$

$$R_{14}: \quad \neg P_{4,1}$$

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1 H	3,2 **P?**	3,3	3,4
4,1 **P?**	4,2 breeze	4,3 **P?**	4,4

Правило резолюции для мира Вампуса

Применим R3 правило удаления двухсторонней импликации
По меньшей мере в одном из квадратов (4,1),(3,2),(4,3) есть яма.

$$R_{15}: \quad (P_{4,1} \vee P_{3,2} \vee P_{4,3})$$

Используем правило устранения противоречия между R_{13} и R_{15}

$$R_{13}: \neg P_{3,2}$$

$$R_{15}: (P_{4,1} \vee P_{3,2} \vee P_{4,3})$$

Используем правило устранения противоречия между R_{14} и R_{15}

$$R_{14}: \neg P_{4,1}$$

$$R_{15}: (P_{4,1} \vee P_{4,3})$$

$$R_{16}: \quad P_{4,3}$$

1,1	1,2	1,3	1,4
2,1 **P?**	2,2	2,3	2,4
3,1 H	3,2 **P?**	3,3	3,4
4,1 **P?**	4,2 breeze	4,3 **P?**	4,4

Конъюнктивная нормальная форма

- Каждое высказывание пропозициональной логики можно представить в конъюнктивной нормальной форме
- CNF – это конъюнкция дизъюнкций логических переменных:

$$(\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n) \wedge (\beta_1 \vee \beta_2 \vee \ldots \vee \beta_m) \wedge \ldots$$

Выражение CNF представляется более сложным, но вычисление по нему становится более быстрым

Правило резолюции применяется только к CNF

Конъюнктивная нормальная форма

Представим в CNF:

$$B_{4,1} \Leftrightarrow (P_{3,1} \vee P_{4,2})$$

1. Устранение двусторонней импликации:

$$(B_{4,1} \Rightarrow (P_{3,1} \vee P_{4,2})) \wedge ((P_{3,1} \vee P_{4,2}) \Rightarrow B_{4,1})$$

2. Устранение связки $\rightarrow$:

$$(\neg B_{4,1} \vee P_{3,1} \vee P_{4,2}) \wedge (\neg (P_{3,1} \vee P_{4,2}) \vee B_{4,1})$$

3. Применим правило де Моргана:

$$(\neg B_{4,1} \vee P_{3,1} \vee P_{4,2}) \wedge ((\neg P_{3,1} \wedge \neg P_{4,2}) \vee B_{4,1})$$

4. Представление в CNF:

$$(\neg B_{4,1} \vee P_{3,1} \vee P_{4,2}) \wedge (\neg P_{3,1} \vee B_{4,1}) \wedge (\neg P_{4,2} \vee B_{4,1})$$

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1 **P?**	3,2	3,3	3,4
4,1 H	4,2 **P?**	4,3	4,4

Конъюнктивная нормальная форма для логического вывода из мира Вампуса

Доказать, что если агент находится в квадрате (4,1) и не чувствует ветерка, то в соседних квадратах нет ям

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1 **(ОК)**	3,2	3,3	3,4
H **(ОК)**	4,2 **(ОК)**	4,3	4,4

R_2 – яма в квадратах (3,1) или (4,2),

$$R_2 = \left(B_{4,1} \Leftrightarrow \left(P_{4,2} \vee P_{3,1}\right)\right)$$

R_4 – в квадрате (4,1) не чувствуется ветерка

$$R_4 = \neg B_{4,1}$$

В квадрате (4,1) не чувствуется ветерка, но яма в квадратах (4,2) или (3.1) имеется

$$KB = R_2 \wedge R_4 = \left(B_{4,1} \Leftrightarrow \left(P_{4,2} \vee P_{3,1}\right)\right) \wedge \neg B_{4,1}$$

Представим в CNF.

$$KB = \left(\neg P_{3,1} \vee B_{4,1}\right) \wedge \left(\neg B_{4,1} \vee P_{4,2} \vee P_{3,1}\right) \wedge \left(\neg P_{4,2} \vee B_{4,1}\right) \wedge \left(\neg B_{4,1}\right)$$

Добавляем условие, что в квадрате (4,2) имеется яма, при этом результат должен быть false

$$KB = \left(\neg P_{3,1} \vee B_{4,1}\right) \wedge \left(\neg B_{4,1} \vee P_{4,2} \vee P_{3,1}\right) \wedge \left(\neg P_{4,2} \vee B_{4,1}\right) \wedge \left(\neg B_{4,1}\right) \wedge \left(P_{4,2}\right) \quad \wedge \left(P_{4,2}\right)$$

$$KB = false$$

Хорновские выражения

Реальные KB включают выражения в CNF, в которых только **один положительный литерал.**

$$\left(\neg L_{1,1} \vee \neg Breeze \vee B_{1,1}\right)$$

Это выражение можно записать как импликация:

$$\left(L_{1,1} \vee Breeze\right) \Rightarrow B_{1,1}$$

Если агент находится в квадрате (1,1) и чувствует ветерок, то ветерок чувствуется в квадрате (1,1). Это выражение без отрицательных литералов называется **фактом.**

Полнота алгоритма резолюции

- Алгоритм резолюции должен быть **полным.**
- Множество выражений S, которые могут быть получены путем применения алгоритма резолюции к исходным выражениям образуют замыкание
- **Прямой логический вывод.** Формирование логических рассуждений, управляемых данными. Агент делает заключение на основе восприятия. Результаты восприятия добавляются в базу знаний.
- **Обратный логический вывод.** Формирование логических рассуждений на основе запроса к базе знаний. Агент выполняет рассуждения, **направляемые целями.**

Агенты, основанные на логике

1. Агенты, **применяющие базы знаний** и алгоритмы логического вывода. Агенты следят за миром и выявляют его скрытые свойства, делают логические выводы.
2. Агенты, **вычисляющие логические функции** с помощью логических схем. Высказывания представлены в виде битов в регистрах. На каждом шаге выполняется обновление регистров.

Проектирование агентов, основанных на логике

- В каждом квадрате агент может почувствовать ветерок и сделать заключения о ямах

$$B_{x,y} \Leftrightarrow \left(P_{x,y+1} \vee P_{x+1,y} \vee P_{x,y-1} \vee P_{x-1,y}\right)$$

- Для каждого квадрата известна причина запаха:

$$S_{x,y} \Leftrightarrow \left(W_{x,y+1} \vee W_{x+1,y} \vee W_{x,y-1} \vee W_{x-1,y}\right)$$

- Существует как минимум один Вампус:

$$E \Leftrightarrow \left(W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}\right)$$

- Отсутствуют два квадрата с двумя Вампусами:

$$E_1 \Leftrightarrow \left(\neg W_{1,1} \vee \neg W_{1,2}\right)$$

Эти правила дают описание среды

Проектирование агентов, основанных на логике

- Слежение за местонахождением агентов

$$L_{x,y} \wedge FacingRight \wedge Forward \wedge \neg Bump \Rightarrow L_{x,y+1}$$

- Слежения выполняется во времени:

$$L^{t}_{x,y} \wedge FacingRight^{t} \wedge Forward^{t} \wedge \neg Bump^{t} \Rightarrow L^{t+1}_{x,y+1}$$

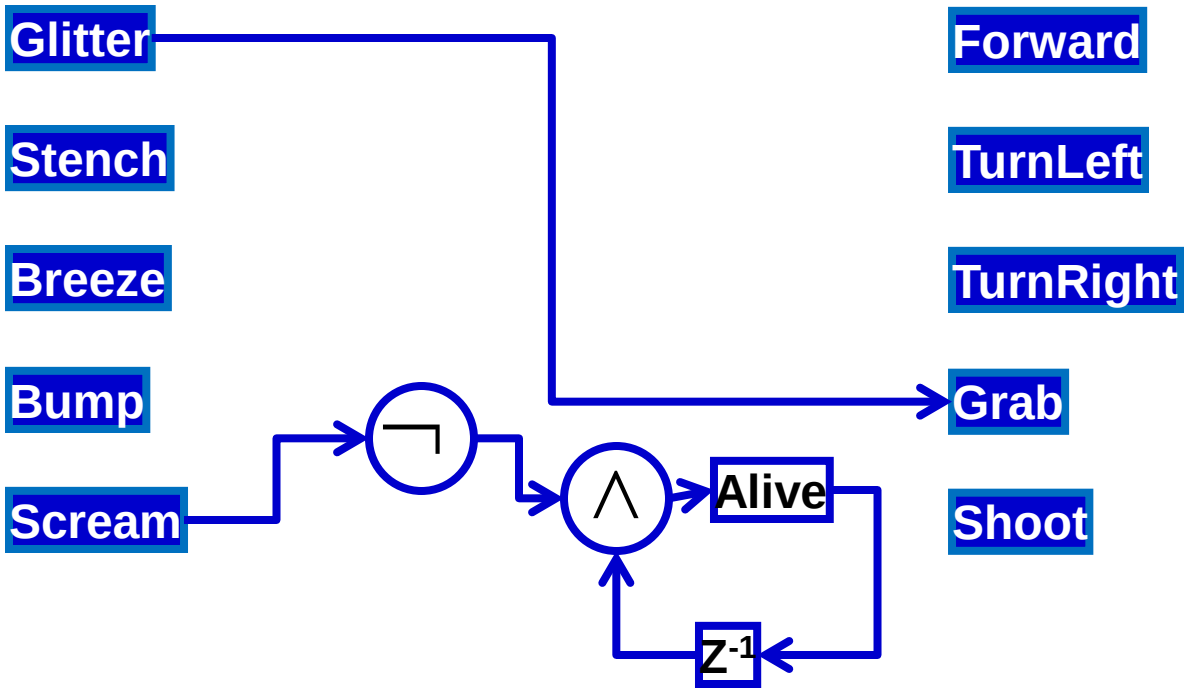
Эти правила дают описание местоположения агента

Агенты на основе логических схем

- Агенты, основанные на логических схемах, формируют высказывания в регистрах, содержание которых обновляется.

$$Grab^t \Leftrightarrow Glitter^t$$

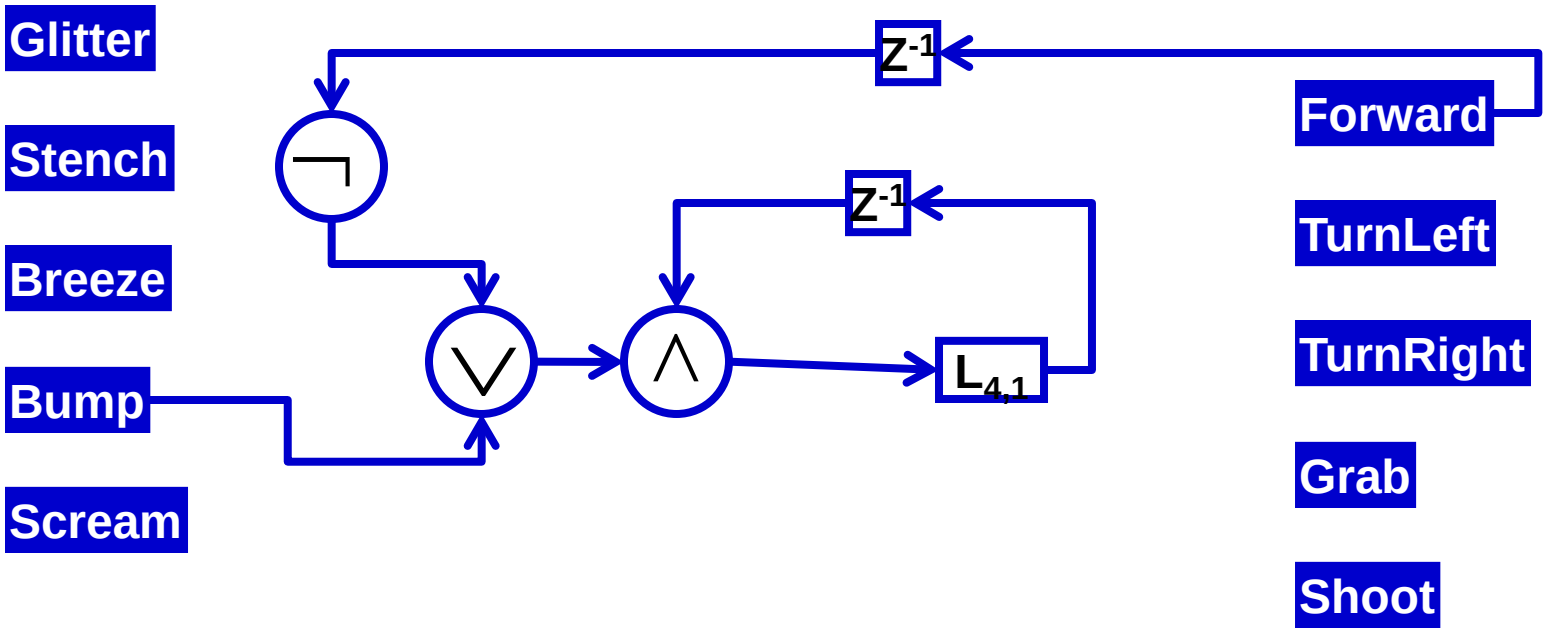
$$Alive^t \Leftrightarrow Alive^{t-1} \wedge \neg Scream^t$$



Агенты на основе логических схем

- Схема определения того, находится ли агент в квадрате (4,1).

$$L_{(4,1)}{}^{t} \Leftrightarrow \left(L_{(4,1)}{}^{t-1} \wedge \left(\neg Forward^{t-1} \vee Bump^{t}\right)\right)$$
$$\vee\left(L_{(4,2)}{}^{t-1} \wedge \left(FacingDown^{t-1} \wedge Forward^{t-1}\right)\right)$$
$$\vee\left(L_{(3,1)}{}^{t-1} \wedge \left(FacingLeft^{t-1} \wedge Forward^{t-1}\right)\right)$$



Логика первого порядка

Лекция 6

Языки представления знаний

Языки представления знаний должны быть декларативными и композиционными и непротиворечивыми

- Языки программирования (C++, Java, Lisp) используют **процедурный подход,** использующий структуры данных, классы.
- Пропозициональная логика использует **декларативный подход,** основанный на истинностных отношениях между миром и высказываниями.
- Пропозициональная логика имеет свойство **композиционности**. Истинность сложного высказывания зависит от истинности его компонент:

$$S = S_{1,4} \wedge S_{1,2}$$

$$S_{1,4} = \neg S_{1,3} \wedge W_{2,4}$$

Языки представления знаний

Языки представления знаний должны быть выразительными и независимыми от контекста

- Пропозициональная логика не обладает достаточной **выразительностью,** поскольку не позволяет описать среду с множеством объектов. Для каждого квадрата $L_{i,j}$ нужно записать свое логическое высказывание.
- В естественных языках смысл высказывания **зависят от контекста**. Например «Смотри!».
- **Язык мышления** отличается от естественного языка. Человек формирует несловесное представление, которое называется памятью.
- **Язык логики первого порядка** наследует от пропозициональной логики свойства декларативности и композиционности, непротиворечивости. Включают новые свойства **выразительности и независимости от контекста**

Язык логики первого порядка

- Язык **логики первого порядка** основан на объектах и отношениях между объектами.
- Язык **временной логики первого порядка** основан на том, что факты имеют место во временных интервалах.
- Язык **логики высокого порядка** основан на объектах и отношениях, причем отношения в логике первого порядка являются объектами в логике высокого порядка.
- Логика первого порядка использует **теорию вероятностей или нечеткую логику.**

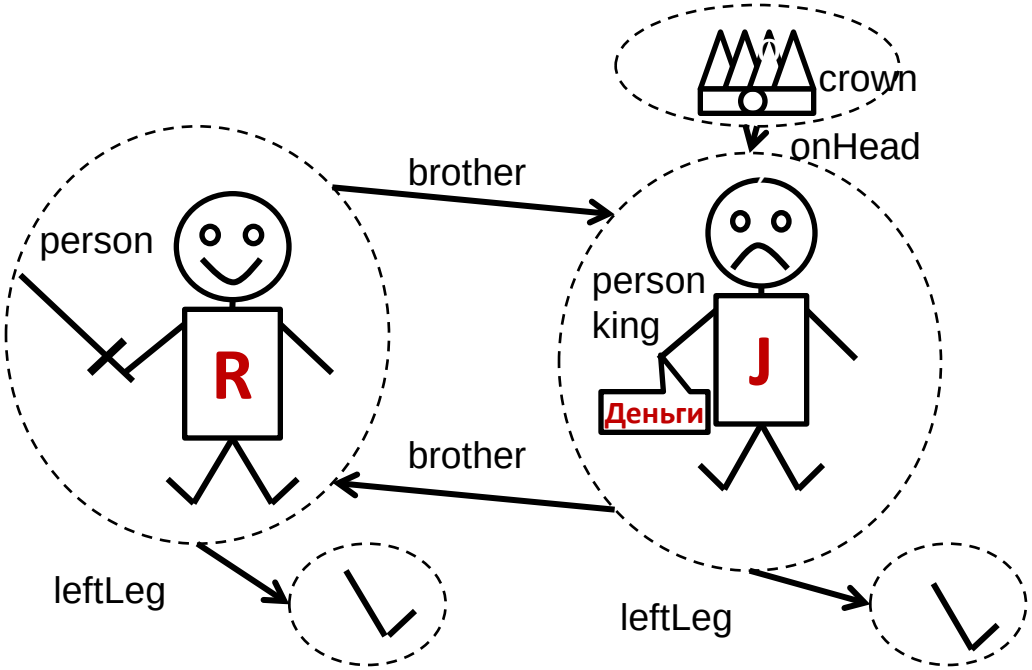
Язык	Онтологический вклад	Степень доверия
Пропозициональная логика	Факты	Истинно/ложно/неизвестно
Логика первого порядка	Факты, объекты, отношения	Истинно/ложно/неизвестно
Временная логика	Факты, объекты, отношения, интервалы	Истинно/ложно/неизвестно
Теория вероятностей	Факты	Степень доверия
Нечеткая логика	Факты со степенью истинности	Интервальное значение

Язык логики первого порядка

- Различные варианты логики отличаются друг от друга по своему **онтологическому вкладу** и эпистемологическому вкладу.
- **Пропозициональная логика** вносит вклад только в общий объем сведений, касающихся существования фактов.
- Каждый факт может быть истинным или ложным,
- **Логика первого порядка** позволяет наращивать объем сведений, касающихся существования объектов и отношений, поэтому приобретает значительную **выразительную мощь**.

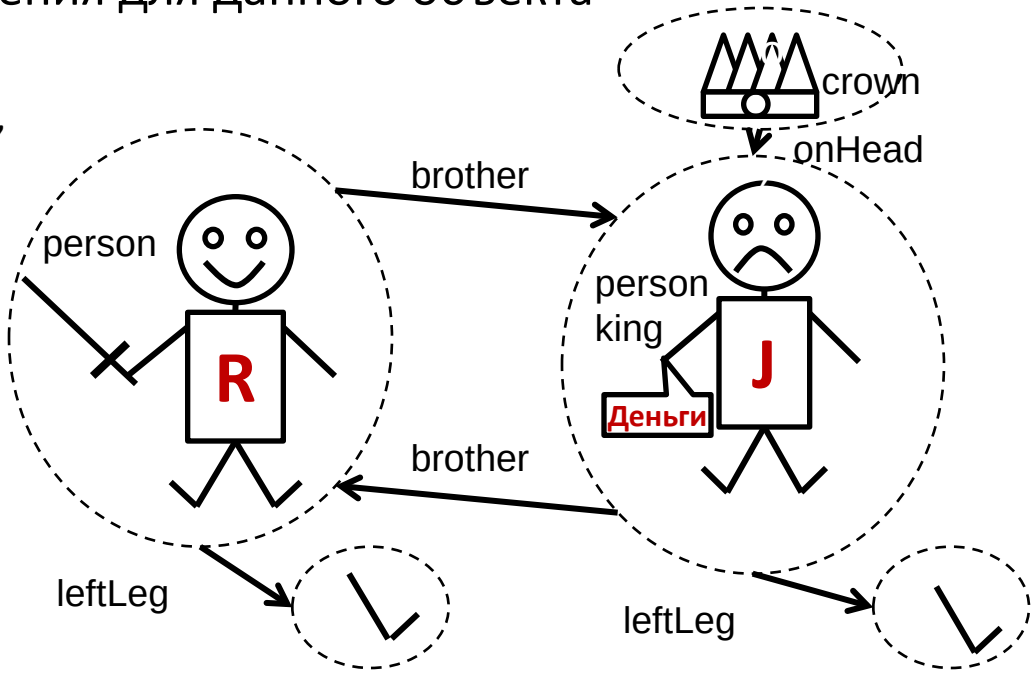
Модели для логики первого порядка

- Возможный мир, или модель, для логики первого порядка определяется **множеством объектов**, **отношениями** между ними и **функциями,** которые могут к ним применяться.
- **Проблемной областью** модели называется множеств, которые она содержит



Синтаксис и семантика логики первого порядка

- **Символы** – основные синтаксические единицы:
  - Константные, обозначающие объекты, (Richard, John)
  - Предикатные, обозначающие отношения (Brother, OnHead, Person)
  - Функциональные, обозначающие функции (leftLeg)
- **Термы** – логические выражения для данного объекта
  - Term → function(Term,…)
  - LeftLeg(Lohn), LeftLeg(Richard),



Синтаксис и семантика логики первого порядка

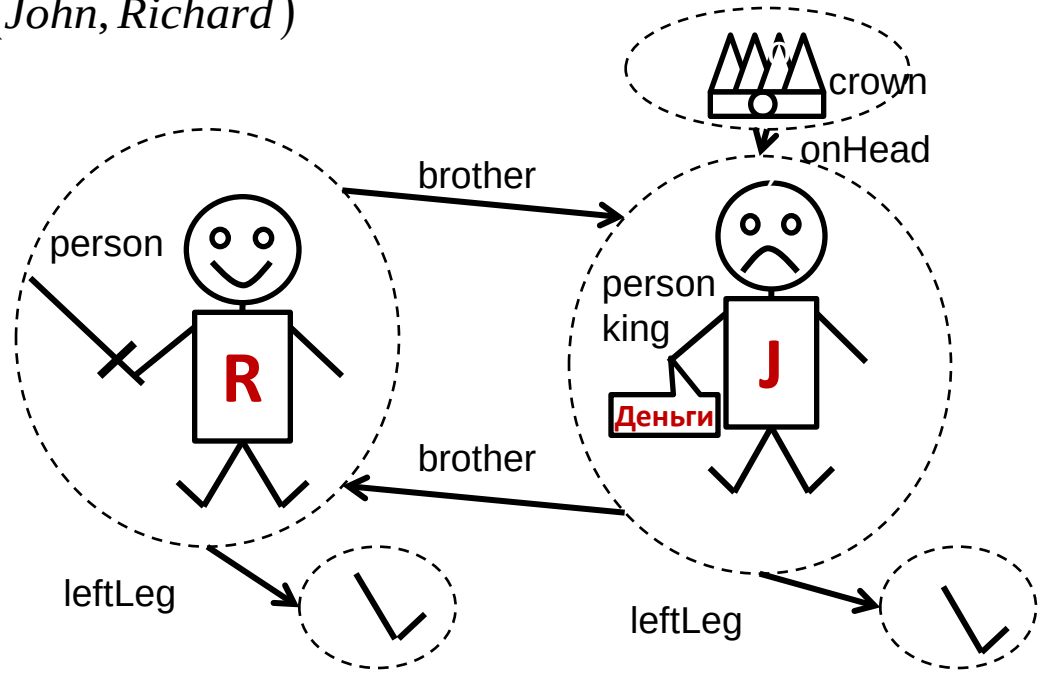
- **Атомарные высказывания** состоят из предиката и списка термов:
  - Brother(Richard, John)
- **Сложные высказывания** используют логические связки

$$\neg Brother\left(LeftLeg\left(Richard\right), John\right)$$

$$Brother\left(Richard, John\right) \wedge Brother\left(John, Richard\right)$$

$$King\left(Richard\right) \vee King\left(John\right)$$

$$\neg King\left(Richard\right) \Rightarrow King\left(John\right)$$



Кванторы. Квантор всеобщности

- **Высказывания с кванторами** позволяют выражать общие правила.
- Применяется **квантор всеобщности** и квантор существования.
- Высказывание «Все короли являются людьми»:

$$\forall x\, King(x) \Rightarrow Person(x)$$

- Высказывание имеет расширенную интерпретацию:

$$1).Richard - King \Rightarrow Richard - person$$
$$2).John - King \Rightarrow John - person$$
$$3).LeftLeg(Richard) - King \Rightarrow LeftLeg(Richard) - person$$
$$4).LeftLeg(John) - King \Rightarrow LeftLeg(John) - person$$
$$5).Crown - King \Rightarrow Crown - person$$

Все высказывания истинны, поскольку связка **x-King** справедлива только для John.

Квантор существования

- **Квантор существования** применяется в высказываниях о некотором объекте.
- Высказывание «на голову Джона возложена корона»:

$$\exists x \; Crown(x) \wedge OnHead(x, John)$$

- Высказывание имеет расширенную интерпретацию:

$1).Richard - Crown \wedge Richard \; - onHead(John)$

$2).John - Crown \wedge John - onHead(John)$

$3).LeftLeg(Richard) - Crown \wedge LeftLeg(Richard) \; - onHead(John)$

$4).LeftLeg(John) - Crown \wedge LeftLeg(John) - onHead(John)$

$5).Crown - Crown \wedge Crown - onHead(John)$

Истинно только пятое высказывание, поскольку истинно, что Crown-Crown.

Связь между кванторами

- Высказывание «Все не любят пастернак» или «Нет никого, кто бы любил пастернак»:

$$\forall x \quad \neg Likes(x, Parsnips)$$
$$\neg\exists x \quad Likes(x, Parsnips)$$

- Высказывание «Все любят мороженое» или «Нет никого, кто бы не любил мороженое»:

$$\forall x \quad Likes(x, IceCream)$$
$$\neg\exists x \quad \neg Likes(x, IceCream)$$

Использование логики первого порядка

- Высказывания вводятся в базу знаний

$$Tell(KB, King(John))$$
$$Tell(KB, \forall x\, King(x) \Rightarrow Person(x))$$

- На запрос к базе знаний

$$Ask(KB, King(John))$$

Будет получен ответ true

- На запрос к базе знаний

$$Ask(KB, Person(John))$$

Будет получен ответ true

Мир Вампуса

- Восприятие percept – бинарный предикат

$$Percept([Stench, Breeze, Glitter, None], 5)$$

- Действия в мире Вампуса выражаются в виде логических термов

$$Tern(Right), Turn(Left), Forward, Shoot, Grab$$

- Запрос к базе знаний. Какое действие является наилучшим?

$$\exists a \qquad BestAction(a, 5)$$

Мир Вампуса

- Из восприятия следуют факты о текущем состоянии среды

$$\forall t, s, g, m, c \;\; Percept([s, Breeze, g, m, c], t) \Rightarrow Breeze(t)$$

$$\forall t, s, g, m, c \;\; Percept([s, b, Glitter, m, c], t) \Rightarrow Glitter(t)$$

- Вариант поведения

$$\forall t \qquad Glitter(t) \Rightarrow BestAction(Grab, t)$$

- Представление среды в базе знаний. Высказывание о соседних клетках:

$$\exists x, y, a, b \;\; Adjacent([x, y], [a, b]) \Leftrightarrow [a, b] \in ([x+1, y], [x-1, y], [x, y+1], [x, y-1])$$

Мир Вампуса

- Синхронные правила, позволяющие сделать логические выводы
- **Диагностические правила** ведут к раскрытию скрытых причин
- «Если в квадрате чувствуется ветерок, то в соседнем квадрате находится яма».

$$\forall s \; Breezy(s) \Rightarrow \exists r \; Adjacent(r, s) \wedge Pit(r)$$

- «Если в квадрате не чувствуется ветерок, то в соседних квадратах нет ямы»

$$\forall s \; \neg Breezy(s) \Rightarrow \neg \exists r \; Adjacent(r, s) \wedge Pit(r)$$

Мир Вампуса

- Синхронные правила, позволяющие сделать логические выводы
- **Причинные правила** определяют причинно-следственные связи:
- «Яма вызывает ветерок в соседних квадратах».

$$\forall r \;\; Pit(r) \Rightarrow \left[\forall s \; Adjacent(r,s) \Rightarrow Breezy(s)\right]$$

- «Если в квадрате нет ямы, то в соседних квадратах не чувствуется ветерок»

$$\forall s \left[\forall r \; Adjacent(r,s) \Rightarrow \neg Pit(r)\right] \Rightarrow \neg Breezy(s)$$

Инженерия знаний

- Процесс инженерии знаний
- Идентификация задания.
- Сбор относящихся к делу знаний
- Определение словаря предикатов, функций и констант
- Регистрация общих знаний о проблемной области
- Составление описания данного конкретного экземпляра задачи
- Передача запросов процедуре логического вывода и получение ответов
- Отладка базы знаний

Логический вывод в логике первого порядка

Лекция 7

Правила логического вывода для квантора всеобщности

Все жадные короли – злые:

$$\forall x \qquad King(x) \wedge Greedy(x) \Rightarrow Evil(x)$$

Выведем высказывания:

$$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$$

$$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$$

$$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$$

Были использованы подстановки:

$$(x / John), (x, Richard), (x, Father(John))$$

Высказывание с квантором всеобщности может быть заменено множеством конкретизаций

Правила логического вывода для квантора существования

Логический вывод заключается в конкретизации выражения с квантором существования.
Существует убийца. Кто является убийцей?

$$\exists x \qquad KIll(x, Victim)$$

Конкретизация:

$$KIll(Murderer, Victim)$$

Используется правило логического вывода для конкретизации кванторов в целях преобразования задачи логического вывода в форму пропозициональной логики.
Это правило характеризуется низким быстродействием.

Высказывание с квантором существования может быть заменено единственной конкретизацией

Обобщенное правило отделения

- Конкретизация высказываний с квантором всеобщности (единственности) заключается в использовании подстановок из словаря базы знаний.
- Правило логического вывода называется обобщенным правилом отделения для атомарных высказываний p_i', p_i, q если существует подстановка

$$Subset(\theta, p_i') = Subset(\theta, p_i)$$

$$\frac{p_1', p_2', \ldots, p_n', \left(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q\right)}{Subset(\theta, q)}$$

$$p_1' - King(John) \qquad p_1 - King(x)$$

$$p_2' - Greedy(y) \qquad p_2 - Greedy(x)$$

$$\theta - \left(x / John, y / John\right) \qquad q - Evil(x)$$

$$Subset(\theta, q) - Evil(John)$$

Унификация и поднятие

Поднятие – преобразование правила отделения из пропозициональной логики в логику первого порядка

Унификация – поиск подстановок, в результате которых логические выражения становятся идентичными.

$$Unify\big(Knows(John, x)\big), Knows(John, Lane) = (x / Jane)$$

$$Unify\big(Knows(John, x)\big), Knows(y, Bill) = (x / Bill, y / John)$$

$$Unify\big(Knows(John, x)\big), Knows(y, Mother(y)) = (y / John, x / Mother(John))$$

Хранение и выборка

Хранение и выборка – на основе Tell() и Ask()

Индексирование фактов по предикатам:

$Segment\ 1: \quad Knows(John, x)$

$Segment\ 2: \quad Brother(Richard, John)$

Индексирование фактов по предикатам и по одному из параметров:

$Employs(AIMA.org, y)$

Запросы к факту занятости Ричарда:

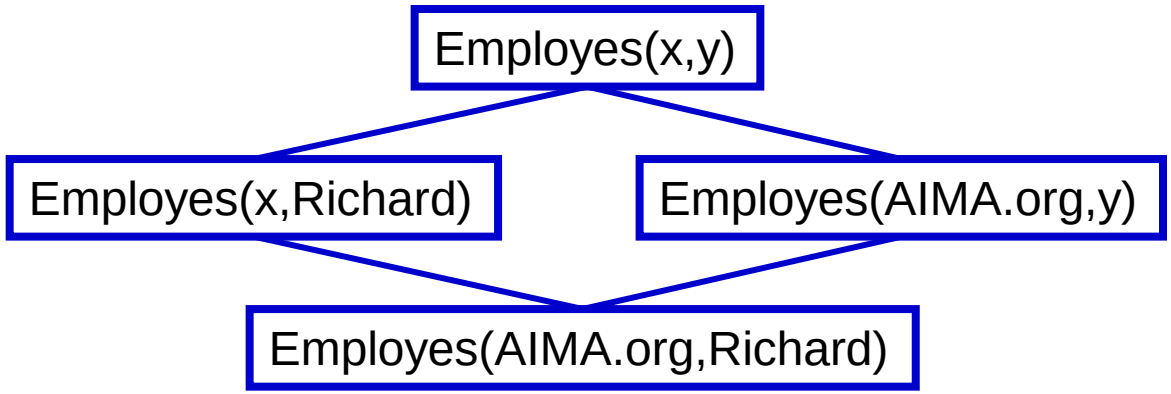
$Employs(AIMA.org, Richard)$

$Employs(x, Richard)$

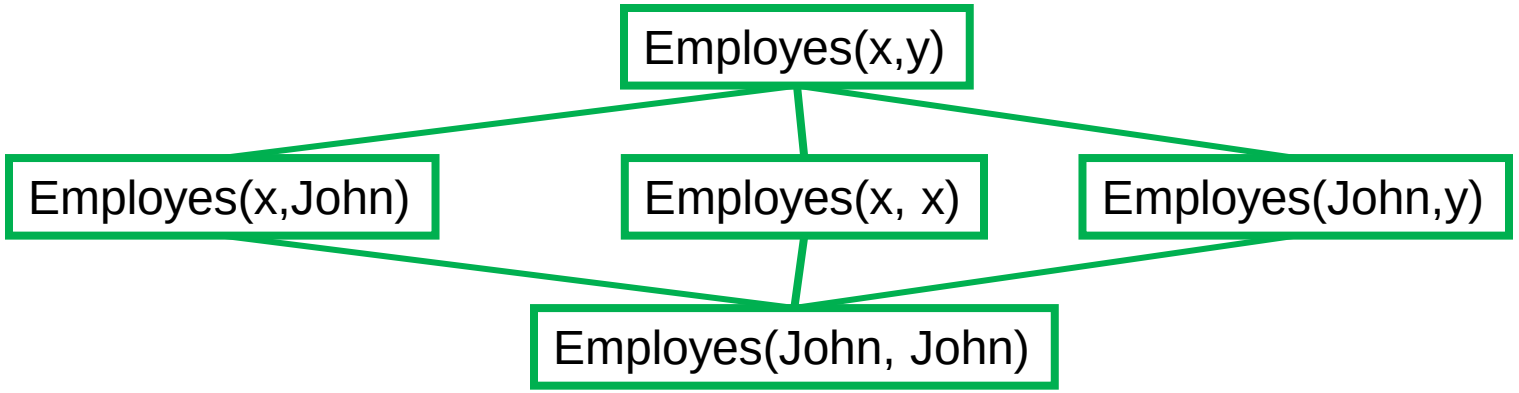
$Employs(AIMA.org, y)$

$Employs(x, y)$

Решетка обобщения



Высказывания с повторяющимися константами



Прямой логический вывод

- Прямой логический вывод используется для случая

Situation → Response

- Продажа оружия недружественным странам, осуществляемая гражданином России, считается преступлением. В государстве Unkown имеются ракеты, проданные гражданином России полковником Фоксом.
- Доказать, что полковник Фокс является преступником.

Продажа оружия враждебным странам является преступлением:

$$Russian(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Crime(x)$$

Является полковник Фокс преступником?

В государстве имеются ракеты:

$$\exists x \quad Owns(Unknown, x) \wedge Missile(x)$$

Все ракеты государства были проданы ему полковником Фоксом

$$Missile(x) \wedge Owns(Unknown, x) \Rightarrow Sells(Fox, x, Unknown)$$

Ракеты являются оружием:

$$Missile(x) \Rightarrow Weapon(x)$$

Государство Unknown не дружественно России:

$$Enemy(x, Russia) \Rightarrow Hostile(x)$$

Полковник Фокс является гражданином России

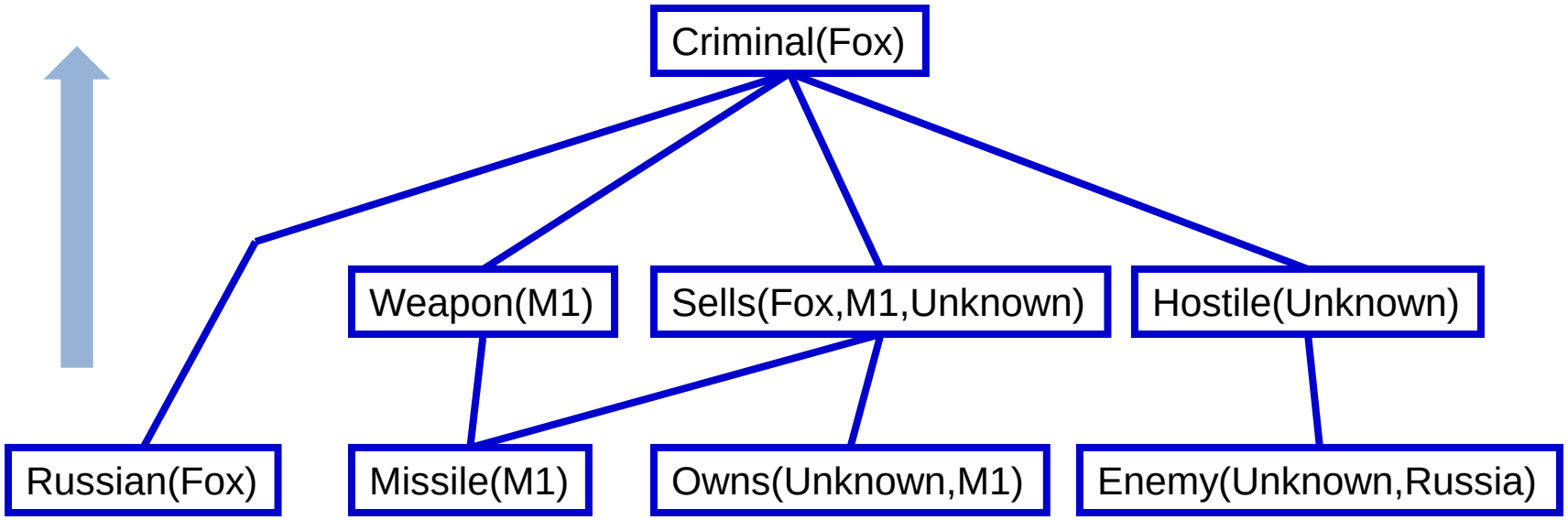
$$Russian(Fox)$$

Государство Unknown враждебно России:

$$Enemy(Unknown, Russia)$$

Прямой логический вывод

Алгоритм на каждой итерации добавляет КВ все новые атомарные высказывания, которые могут быть выведены из имеющихся атомарных высказываний



Эффективный прямой логический вывод

- Проблема согласования предпосылки правила с хранящимися в КВ фактами.

$$Missile(x) \wedge Owns(Unknown, x) \Rightarrow Sells(Fox, Unknown)$$

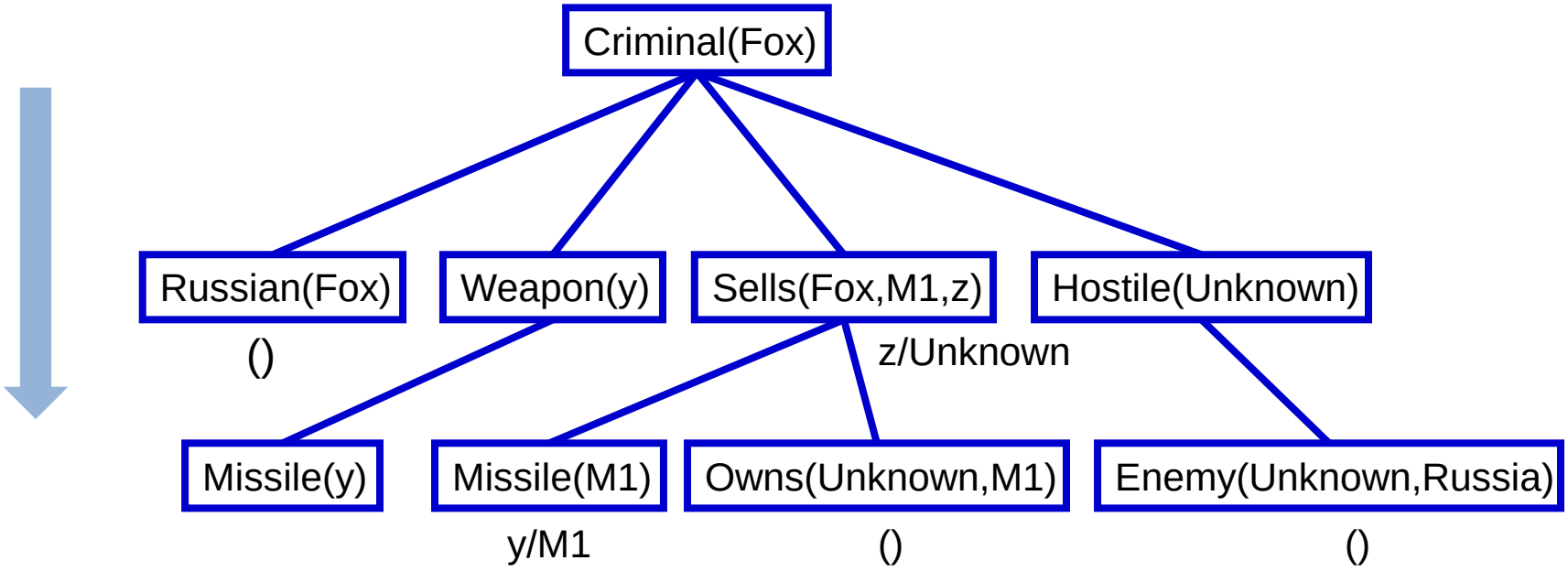
Проблема упорядочения конъюнктов.

1. Найти все объекты, принадлежащие Unknown.
2. Проверить, что объект является ракетой

Первой проверять наиболее ограниченную переменную.
(проверять ракеты, затем те из них, которые принадлежат Unknown)

Обратный логический вывод

- Алгоритмы действуют **в обратном направлении** от цели.
- Необходимо доказать 4 конъюнкта:
  - Russian(Fox) находится только в КВ
  - Остальные конъюнкты проверяются далее



Логическое программирование

- Логическое программирование – форма автоматического формирования рассуждений:

  `Алгоритм = Логика + Управление`

- Язык логического программирования **Prolog**

Резолюция

- **Резолюция** – это полная процедура логического вывода на основе опровержения.
- Резолюция использует конъюнктивную нормальную форму **CNF**

Доказать высказывание:

$$\forall x \quad Russian(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Crime(x)$$

Каждое высказывание в логике первого порядка м.б. преобразовано в эквивалентное высказывание в CNF.

$$\neg Russian(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Crime(x)$$

Использованы тождества:

$$\alpha \Rightarrow \beta \equiv (\neg\alpha \vee \beta)$$
$$\alpha \wedge \beta \equiv \neg(\neg\alpha \vee \neg\beta)$$

CNF

- «Каждого, кто любит всех животных, кто-то любит»

$$\forall x \left[\forall y \ Animal(y) \Rightarrow Loves(x, y)\right] \Rightarrow \left[\exists y \ Loves(y, x)\right]$$

- Устранение импликаций

$$\forall x \left[\forall y \ \neg Animal(y) \vee Loves(x, y)\right] \Rightarrow \left[\exists y \ Loves(y, x)\right]$$

$$\forall x \left[\neg \forall y \ \neg Animal(y) \vee Loves(x, y)\right] \vee \left[\exists y \ Loves(y, x)\right]$$

- Используем правила:

$$\neg \forall x \ p \equiv \exists x \ \neg p$$

$$\neg \exists x \ p \equiv \forall x \ \neg p$$

- «Либо существует какое-либо животное, которого **x** не любит, либо кто-то любит **x**»

$$\forall x \left[\exists y \neg \neg Animal(y) \wedge \neg Loves(x, y)\right] \vee \left[\exists y \ Loves(y, x)\right]$$

$$\forall x \left[\exists y \ Animal(y) \wedge \neg Loves(x, y)\right] \vee \left[\exists y \ Loves(y, x)\right]$$

Процесс устранения кванторов

- Устранение **квантора существования** выполняется путем конкретизации, где A – некоторая константа:

$$\exists x\, P(x) \qquad \hat{a} \qquad P(A)$$

- «Каждый либо не способен любить какое-то животное A, либо его любит некоторое животное B» (Смысл высказывания неправильный).

$$\forall x\left[\, Animal(A) \wedge \neg Loves(x, A)\right] \vee Loves(B, x)$$

- Чтобы смысл высказывания сохранялся, требуется, чтобы A и B зависели от x.
- «Для каждого x существует некоторое животное, на которое указывает функция F(x), которое он (т.е.x) не любит или существует животное, на которое указывает функция G(x), которое его (т.е. x) любит»

$$\forall x\left[\, Animal\big(F(x)\big) \wedge \neg Loves\big(x, F(x)\big)\right] \vee Loves\big(G(x), x\big)$$

$$\left[Animal\big(F(x)\big) \wedge \neg Loves\big(x, F(x)\big)\right] \vee Loves\big(G(x), x\big)$$

Процесс устранения кванторов

- Для преобразования в CNF используем правило:

$$A \wedge \neg B \vee C = (A \vee C) \vee (\neg B \vee C)$$

- Логический вывод, полученный в CNF с помощью резолюции:

$$[Animal(F(x)) \wedge Loves(G(x), x)] \wedge [\neg Loves(x, F(x)) \vee Loves(G(x), x)]$$

Высказывание полностью недоступно для восприятия

Функция F(x) указывает на животное, которое потенциально может быть нелюбимым для x, а G(x) указывает на животное, которое может быть любить x.

Резолюция: «Кто убил кота Туна?»

- Каждого, кто любит всех животных, кто-то любит.
- Любого, кто убивает животных, никто не любит.
- Джек любит всех животных.
- Кота по имени Тун убил либо Джек, либо Любопытство.
- Действительно ли кота убило любопытство?

$$A.\ \forall x\ \left[\forall y\ Animal(y) \Rightarrow Loves(x,y)\right] \Rightarrow \left[\exists y\ Loves(y,x)\right]$$

$$B.\ \forall x\ \left[\exists y\ Animal(y) \wedge Kills(x,y)\right] \Rightarrow \left[\forall z\ \neg Loves(z,x)\right]$$

$$\tilde{N}.\ \forall x\ Animal(x) \Rightarrow Loves(Jack,x)$$

$$D.\ Kills(Jack,Tun) \vee Kills(Curiosity,Tun)$$

$$E.\ Cat(Tun)$$

$$F.\ \forall x\ Cat(x) \Rightarrow Animal(x)$$

$$G.\ \neg Kills(Curiosity,Tun)$$

Резолюция: «Кто убил кота Туна?»

- Первое выражение преобразуем по полученной ранее схеме:

$$\left[Animal\left(F(x)\right)\wedge Loves\left(G(x),x\right)\right]\wedge\left[\neg Loves\left(x,F(x)\right)\vee Loves\left(G(x),x\right)\right]$$

- Преобразуем все выражения к CNF

$A1.\ Animal\left(F(x)\right)\wedge Loves\left(G(x),x\right)$

$A2.\ \neg Loves\left(x,F(x)\right)\vee Loves\left(G(x),x\right)$

$B.\ \neg Animal(y)\vee\neg Kills(x,y)\vee\neg Loves(z,x)$

$\tilde{N}.\ \neg Animal(x)\vee\neg Loves(Jack,x)$

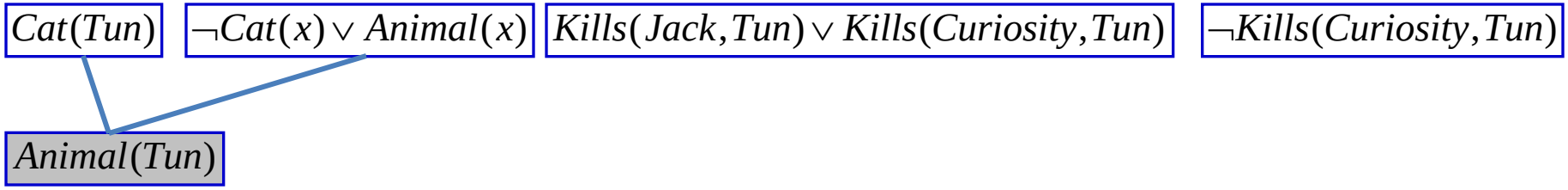
$D.\ Kills(Jack,Tun)\vee Kills(Curiosity,Tun)$

$E.\ Cat(Tun)$

$F.\ \neg Cat(x)\vee Animal(x)$

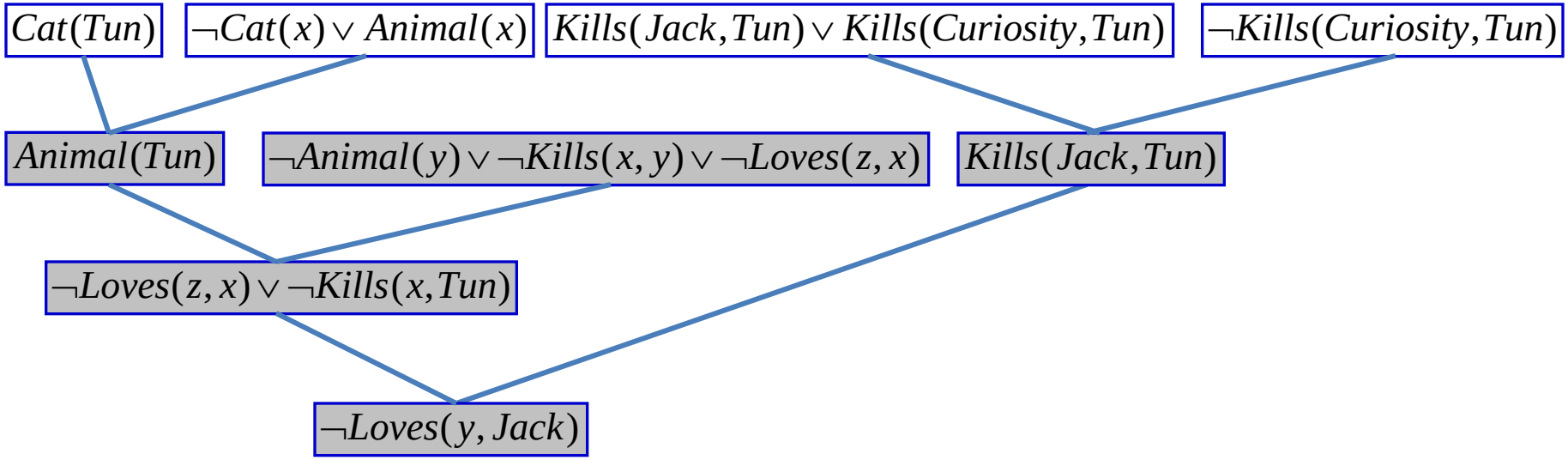
$\neg G.\ \neg Kills(Curiosity,Tun)$

Процедура доказательства



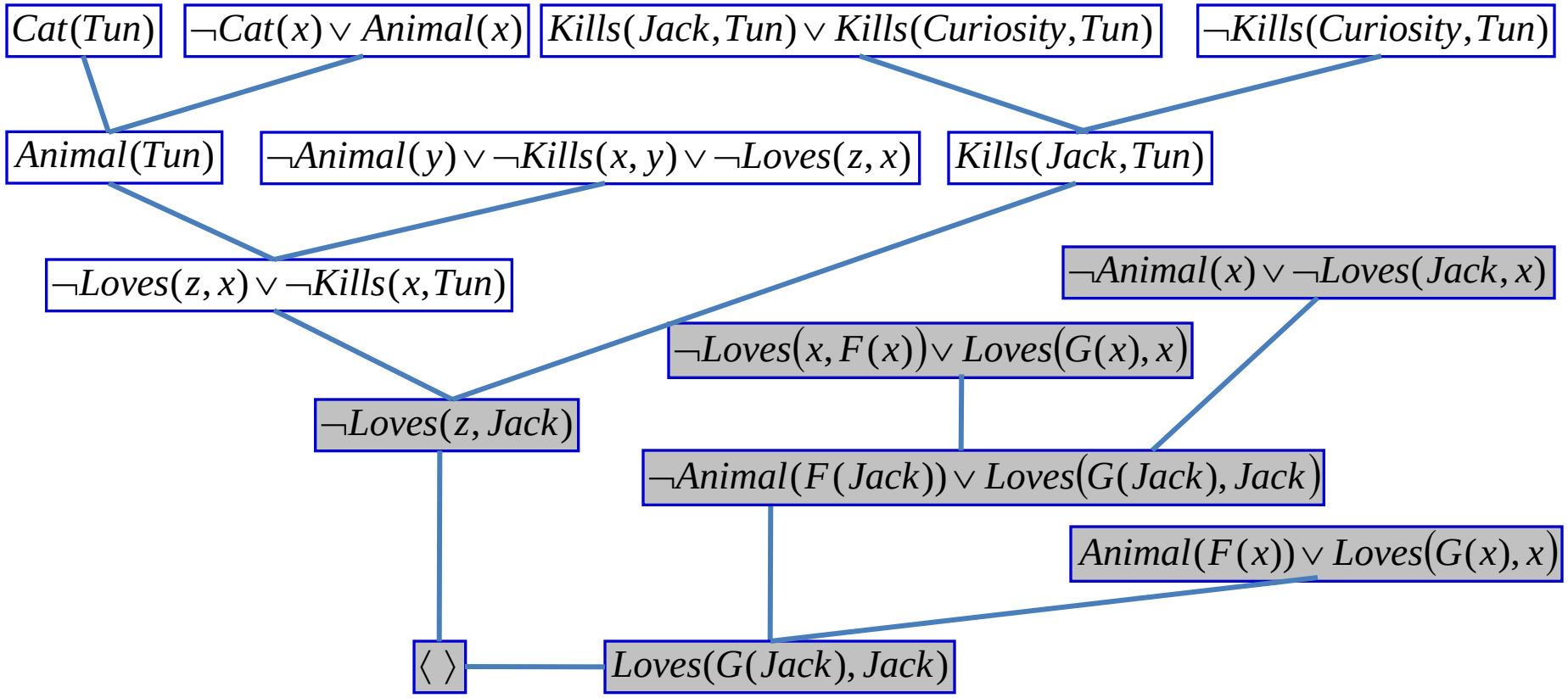
Тун – кот, а все коты – животные, следовательно, Тун - животное

Процедура доказательства



Любого, кто убивает животное, никто не любит, следовательно, никто не любит Джека

Процедура доказательства



Джек любит всех животных, поэтому кто-то его любит, следовательно, возникает противоречие (пустое высказывание)
Поэтому кота убило любопытство.

Резолюция

- **Резолюция** – это полная процедура логического вывода на основе опровержения.
- Резолюция использует конъюнктивную нормальную форму **CNF**

Доказать высказывание:

$$\forall x \quad Russian(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Crime(x)$$

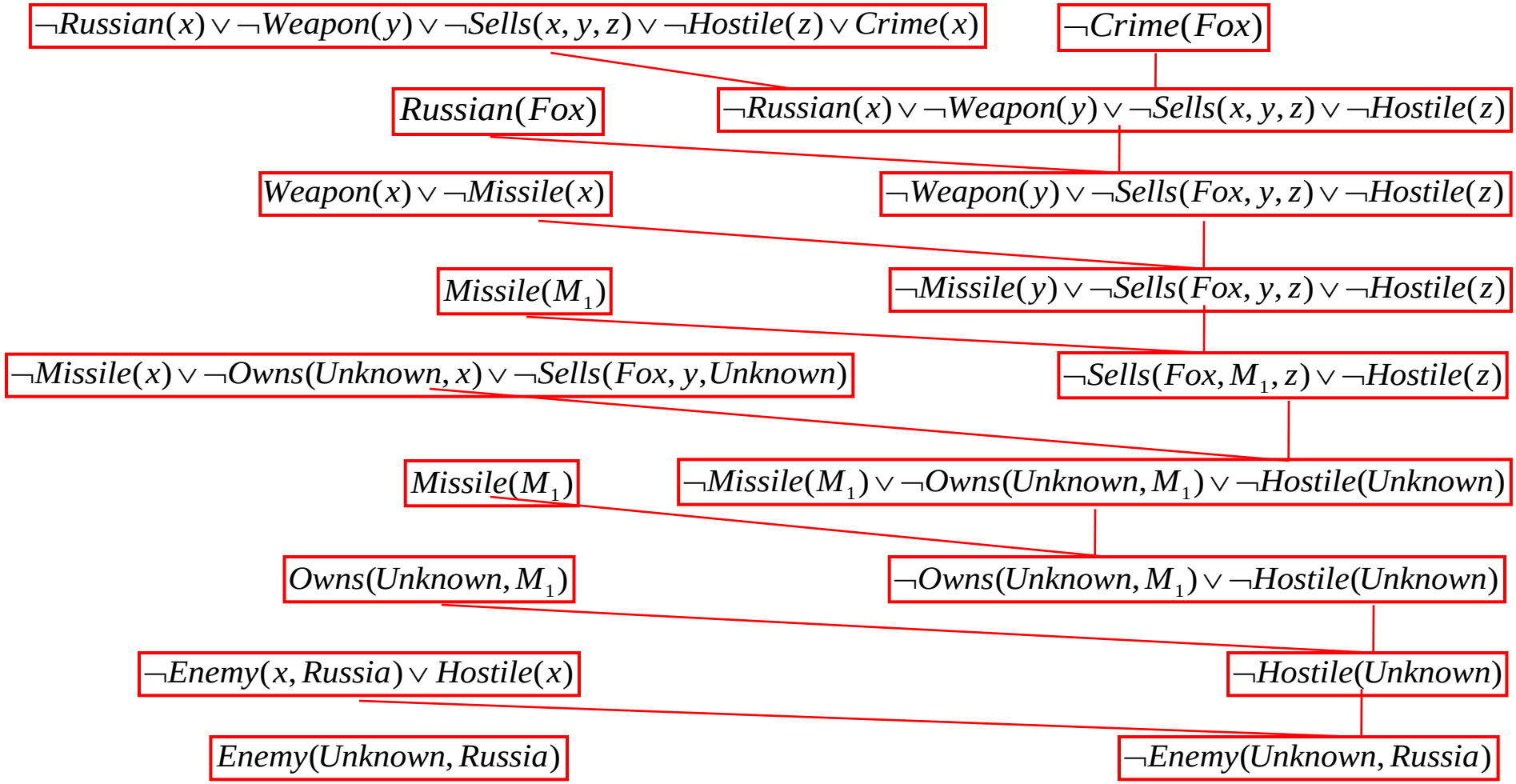
Каждое высказывание в логике первого порядка м.б. преобразовано в эквивалентное высказывание в CNF.

$$\neg Russian(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Crime(x)$$

Использованы тождества:

$$\alpha \Rightarrow \beta \equiv (\neg\alpha \vee \beta)$$
$$\alpha \wedge \beta \equiv \neg(\neg\alpha \vee \neg\beta)$$

Резолюция: «Является полковник Фокс преступником?»



Задания

- Составьте решетки обобщения
  - Employs(IBM,y)
  - Employs(Mother(John),Father(Richard))
- Запишите логические представления для приведенных высказываний:
  - Лошади, коровы, свиньи – млекопитающие
  - Рожденный лошадью – лошадь
  - Криз – лошадь
  - Криз – родитель Чарли
  - Отношения «быть рожденным» и «быть родителем» - обратные
  - Каждое млекопитающее имеет родителя
- Примените правило логического вывода к загадке: «братьев и сестер у меня нет, но отец этого человека – сын моего отца»

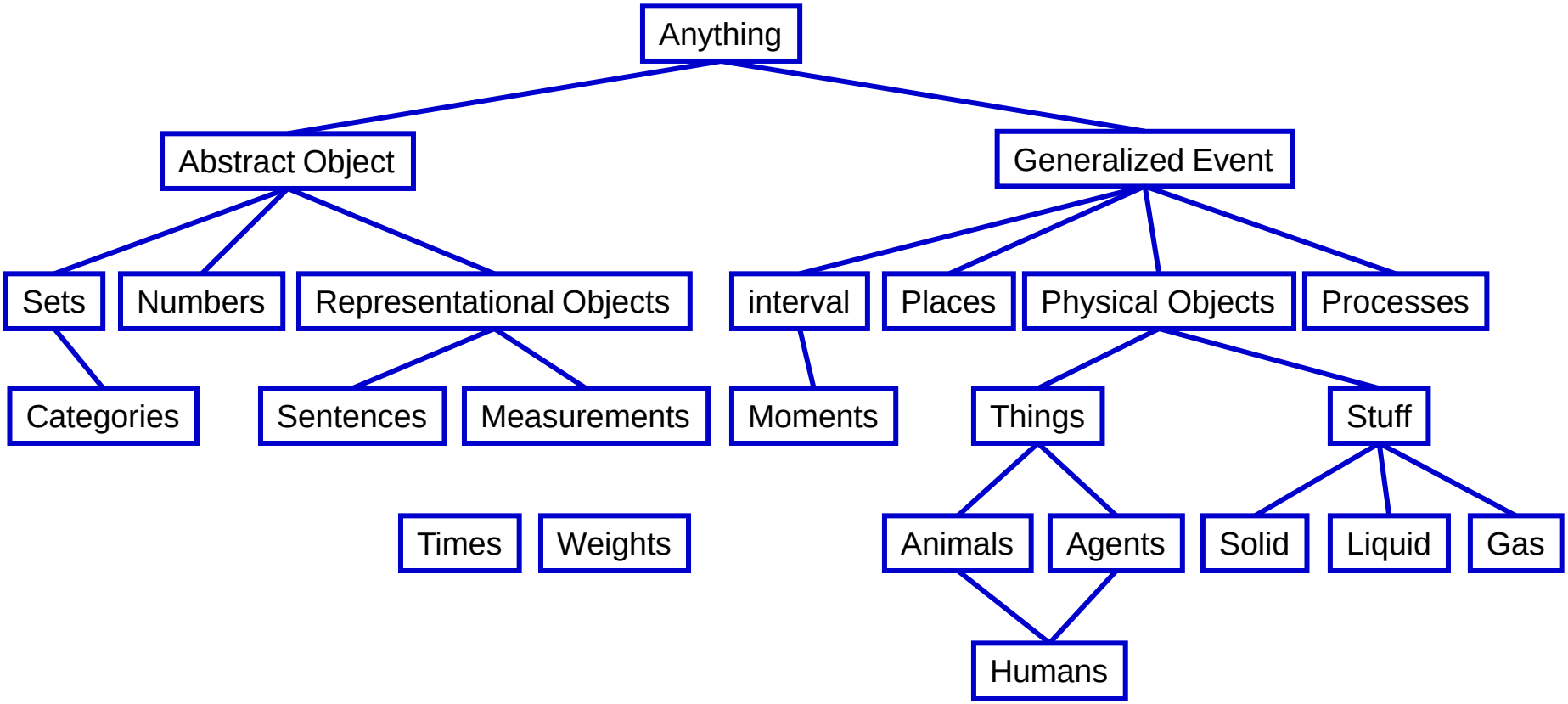
Представление знаний

Лекция 8

Онтологическая инженерия

- Способы представления абстрактных понятий, действий, времени, объектов и убеждений, называют **онтологической инженерией.**
- **Верхняя онтология** основана на категориях и исчислении событий. В частности использует объекты, время, и пространство, процессы, вещества и убеждения.
- Для крупномасштабного представления знаний требуется **онтология общего назначения**, позволяющая организовать и связать воедино различные специализированные области знаний.
- **Онтология общего назначения** должна быть применима в любой специализированной проблемной области с добавлением аксиом конкретной области.

Верхняя онтология мира



Категории и объекты

- Формирование рассуждений происходит на уровне **категорий.**
- Представление категорий в логике первого порядка (баскетбольный мяч):
- В виде объектов - **Basketballs**,
- С помощью предикатов **Basketball(b)** .
- Высказывание b – элемент категории баскетбольных мячей:

Member(b, Basketballs)

- Basketballs - подкатегория множества мячей balls:

Subset(Basketballs, balls)

- Категории служат для организации наследования. Например, наследование свойства съедобности:

Balls → Basketballs

Категории и объекты

- Факты о категориях сформированы в логике первого порядка.
- Элементы категории могут быть распознаны по некоторым свойствам:

$$Basketballs \subset Balls$$

$$x \in Basketballs \Rightarrow Round(x)$$

$$Orange(x) \wedge Round(x) \wedge Diameter(x) = "9.5" \wedge x \in Balls \Rightarrow x \in Basketballs$$

- Отношения между классами и подклассами можно организовать в виде **таксономии** или иерархии таксономий.
- Категории могут быть непересекающимися.

Физическая композиция объектов

Объект может составлять **часть другого**

$$PartOf(Russia, Europe);$$

$$PartOf(Europe, Earth);$$

Категория **составных объектов**. Совокупность трех яблок образует новый объект:

$$BunchOf(Apple1, Apple2, Apple3)$$

Меры

Значения, применяемые для оценки объектов называются **мерами.**
Длина отрезка:

$$Lenght(L_1) = Centimeters(3.58);$$

Аксиомы относительно мер длины:

$$Centimeters(2.54 \cdot d) = Inches(d)$$

Нечисловые меры:

$$e_1 \in Exercises \wedge e_2 \in Exercises \Rightarrow Difficulty(e_1) > Difficulty(e_2)$$

Качественная физика – направление искусственного интеллекта, в котором используются рассуждения о физических системах без погружения в числовые расчеты.

Вещества и объекты

- **Объекты** поддаются индивидуализации и исчислению,
- **Вещество** – не поддается исчислению,
- **Исчислимые** существительные (муравьи, агенты,…)
- **Неисчислимые** существительные (масло, энергия, …)
- Любая часть **вещества** тоже вещество.
- Любая часть **объекта** «масло» тоже «масло»:

$$x \in Batter \wedge PartOf(x, y) \Rightarrow y \in Batter$$

- Кусок масла разделили пополам:

$$x \in Batter \wedge \left(Lenght(x) = Centimaters(20)\right) \wedge PartOf(x, y, z)$$
$$\Rightarrow y \in Batter \wedge \left(Lenght(y) = Centimaters(10)\right) \wedge$$
$$z \in Batter \; z \wedge \left(Lenght(z) = Centimaters(10)\right)$$

Онтология ситуационного исчисления

- **Ситуация** – логический терм, состоящий из начальной ситуации S_0 и всех ситуаций, S_i, получаемых в результате действия **Do**.
- **Действия, события и время** могут быть представлены с помощью ситуационного исчисления,
- Пример **онтологии ситуационного исчисления :** «Для всех t результат, полученный в момент времени **t+1**,зависит от результата в момент **t**»
- **Флюентными** называются функции, изменяющиеся во времени.
- Агент на основе ситуационного исчисления способен определить результат последовательности своих действий.

Действия, события и время могут быть представлены также с применением **исчисления событий** и **вычисления флюентных высказываний**.
Такие представления дают возможность агенту составлять планы с помощью логического вывода.

Онтология ситуационного исчисления

Описание ситуации S_0 (агент находится в [4,1], золото – в [4,2]):

$$At(o, x, S_0) \Leftrightarrow [(o = Agent \wedge x = [4,1]) \vee (o = G \wedge x = [4,2])]$$

$$\neg Holding(o, S_0)$$

Золото находится в соседнем квадрате:

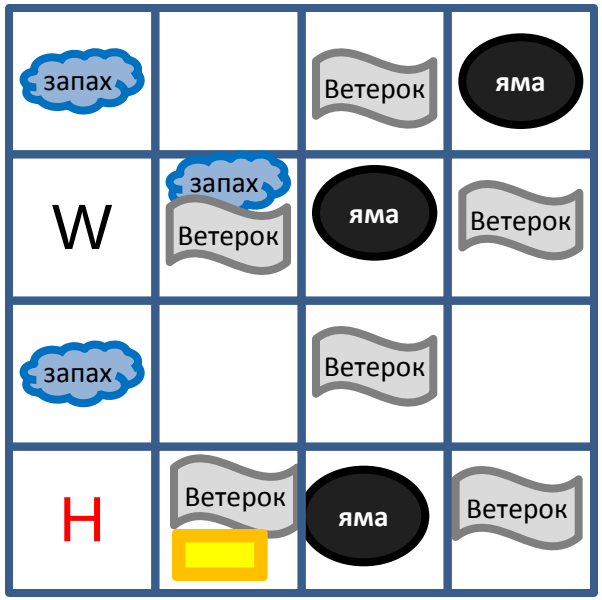
$$Gold(G) \wedge Adjucent([4,1],[4,2]) \wedge Adjucent([4,2],[4,1])$$

Агент планирует забрать золото и вернуться

$$At(G,[4,1], Result(Go[4,1],[4,2]), Grab(G), Go[4,2],[4,1], S_0))$$

Какая последовательность действий приведет к переносу золота в квадрат [4,1]?

$$\exists seq \;\; At(G,[4,1], \mathrm{Re}\, sult(seq, S_0))$$



Действия в ситуационном исчислении

- **Аксиомы возможности** – существует ли возможность выполнить действие: **`предусловия→Poss(a,s)`**

$$At(Agent, x, s) \wedge Adjacent(x, y) \Rightarrow Poss(Go(x, y), s)$$

$$Gold(g) \wedge At(Agent, x, s) \wedge At(g, x, s) \Rightarrow Poss(Grab(g), s)$$

$$Holding(g, s) \Rightarrow Poss(Release(g), s)$$

- **Аксиомы результата** –что произойдет после выполнения действия: **`Poss(a,s) → изменения после выполнения действия`**

$$Poss(Grab(g), s) \Rightarrow Holding(g, Result(Grab(g), s))$$

$$Poss(Release(g), s) \Rightarrow \neg Holding(g, Result(Release(g), s))$$

Действия в ситуационном исчислении

- Аксиомы результата указывают, что изменилась, но не указывают, что осталось неизменным
- **Аксиомы окружения** указывают, что осталось неизменным.
- При перемещении агента все объекты, которые не берет агент, остаются на месте:

$$At(o,x,s) \wedge (o \neq Agent) \wedge \neg Holding(o,s) \Rightarrow At\big(o,x,Result\big(Go(y,z),s\big)\big)$$

Проблема состоит в большом количестве аксиом окружения. Для F предикатов и A действий – O(AF) аксиом.

Проблема представительного окружения

- Проблему решает аксиома состояния – преемника:

Действие возможно→
(флюентное высказывание ↔
Высказывание было истинным в результате действия
ИЛИ
оно было истинным и действие его не изменило)

- Агент находится в квадрате y после выполнения действий:
  - при перемещении в квадрат y
  - при продолжении пребывания в квадрате y

$$Poss(a,s) \Rightarrow At\big(agent, y, Result(a,s)\big) \Leftrightarrow$$
$$a = Go(x,y) \vee \big(At\big(agent, y, s\big) \wedge a \neq Go(y,z)\big)$$

Количество аксиом окружения меньше. Для A действий и E результатов действий – O(AE) аксиом.

Проблема выводимого окружения

- Рассмотрим t-шаговый план p

$$S_t = Result(p, S_0)$$

- В каждой аксиоме рассматривается **несколько действий F_i**

$$Poss(a,s) \Rightarrow F_i\big(Result(a,s)\big) \Leftrightarrow \big(a = A_1 \vee a = A_2 \ldots\big) \vee F_1(s) \wedge \big(a \neq A_3\big) \wedge \big(a \neq A_4\big)$$

- Флюентное высказывание может быть истинным **PossEffect(a,F)**
- Флюентное высказывание может быть ложным **NegEffect(a,F)**

$$Poss(a,s) \Rightarrow F_i\big(Result(a,s)\big)$$
$$\Leftrightarrow PosEffect(a,F_i) \vee \big[F_i(s) \wedge \neg NegEffect(a,F_i)\big]$$

Исчисление времени и событий

- Исчисление событий основано на точках во времени.
- **Initiates(e,f,t)** означает, что высказывание **f** при возникновении события **e** в момент **t** стало истинным**.**
- **Terminates(w,f,t)** означает, что высказывание **f** при возникновении события **w** стало ложным.
- **Happens(e,t)** событие **e** произошло в момент **t**.
- **Clipped(f,t,t_2)** высказывание **f** стало ложным в интервале времени **t-t_2.**

Аксиома исчисления событий

$$T(f,t_2) \Leftrightarrow$$

$$\exists e,t\ Heppens(e,t) \wedge Initiates(e,f,t) \wedge (t < t_2) \wedge \neg Clipped(f,t,t_2) Clipped(f,t,t_2)$$

$$\Leftrightarrow \exists e,t_1\ Heppens(e,t_1) \wedge Terminates(e,f,t_1) \wedge (t < t_1) \wedge (t_1 < t_2)$$

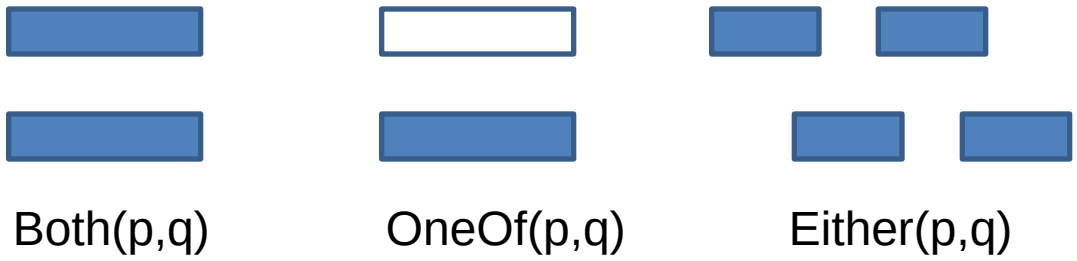
Обобщенные высказывания

- Обобщенное событие состоит из **пространственно-временного универсума.**
- Универсум имеет **три пространственных и два временных измерения**.
  - **«WorldWar II»** происходит в пространстве и времени
  - **«TwentiesCentury»** имеет максимальную пространственную **протяженность.**
  - **«Europe»** имеет максимальную временную протяженность
- Вторая мировая война – это событие, имеющее место в различных точках пространства и времени.
- Это событие может быть разбито на подсобытия:
  - **SubEvent(ButtleOfMoscow, WorldWarII);**
  - **SubEvent(WorldWarII, TwentiesCentury);**
  - **Duration(Period(WorldWarII))>Years(5);**

Процессы

- **Категория процессов**
- **`T(c,i)`** – некоторое событие типа ***c*** произошло в интервале времени **`i`**
- **Категория вневременных событий**
- **`In(Russia, Europe)`** - состояние или вневременное событие
- **Флюентные высказывания** - сложные высказывания о событиях: «Некто идет и жует резинку»

$$\exists p, i \left(p \in People\right) \wedge T\left(Both\left(Walk(p), ChewGum(p)\right), i\right)$$



Интервалы

- Интервалы времени имеют **продолжительность**

$$Interval(i) \Rightarrow Duration(i) = Time\big(End(i)\big) - Time\big(Start(i)\big)$$

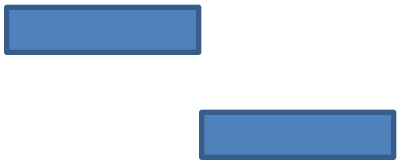
- Предикаты, задаваемые на временных интервалах:

$$Meet(i, j) \Leftrightarrow Time\big(End(i)\big) = Time\big(Start(i)\big)$$

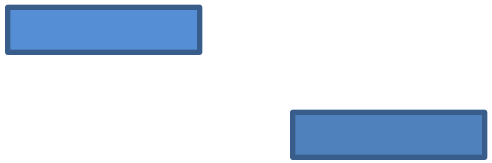
$$Before(i, j) \Leftrightarrow Time\big(End(i)\big) < Time\big(Start(i)\big)$$

$$After(i, j) \Leftrightarrow Before(j, i)$$

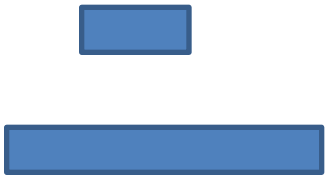
$$During(i, j) \Leftrightarrow \big(Time\big(Start(j)\big) \le Time\big(Start(i)\big)\big) \wedge \big(Time\big(End(i)\big) \le Time\big(End(j)\big)\big)$$



Meet(i,j)



Before(I,j)
After(j,i)



During(i,j)

Теория убеждений

- Агент может занять одну из позиций к некоторому высказыванию:

Агент убежден - Believes

$$Belives(Agent, Flies(Superman)) \Leftrightarrow Belives(Agent, Flies(Clark))$$

Агент знает – Knows.

Знания – это убеждения, истинность которых подтвердилась

Knows(a,p) – агент знает, что высказывание p истинно.

$$KnowsWhether(a, p) \Leftrightarrow Knows(a, p) \vee Knows(a, \neg p)$$

$$KnowsWhat(a, "Capital(Russia)") \Leftrightarrow KnowsWhat(a, "Capital(Russia)", ProperNames)$$

Агент желает – Wants

Мыслительные состояния агентов могут быть представлены строками, которые описывают убеждения.

Мир покупок в Интернет

- Агент принимает описание товара и готовит список Web-страниц, предлагающих товары.
- Пример Web-страницы

```
Select from our fine line of products:
•Computers
•Cameras
•Books
•Videos
•Music
<h1>Generic OnlineStore</h1>
<i>Select </i> from our fine line of products:
<ul>
<li><a href="http://gen-stor.com/compu">Computers</a>
<li><a href="http://gen-stor.com/camer">Cameras</a>
<li><a href="http://gen-stor.com/books">Books</a>
<li><a href="http://gen-stor.com/video">Videos</a>
<li><a href="http://gen-stor.com/music">Musics</a>
</ul>
```

Мир покупок в Интернет

- Найти страницу **page** с предложением товара по запросу *query*:

$$RelevantOffer(page, url, query) \Leftrightarrow Relevant(page, url, query) \wedge Offer(page)$$

- Страница должна содержать слова **“bye”, “price”, “a”, “/a”,** или **“from”**

$$Offer(page) \Leftrightarrow \big(InTag("a", str, page) \vee InTag("\ from", str, page)\big) \wedge$$
$$\big(In("bye", str) \vee In("\ price", str)\big)$$

- Агент обладает знаниями о магазинах:

$$Amazon \in OnlineStores \wedge Homepage(Amazon, "amazon.com")$$
$$GenStore \in OnlineStores \wedge Homepage(GenStore, "gen-store.com")$$

Мир покупок в Интернет

- Товары классифицируются **по категориям**. К младшим категориям можно перейти по ссылкам от старших категорий:

$$Relevant(page, url, query) \Leftrightarrow$$
$$\exists store, home \quad store \in OnlineStores \wedge Homepage(store, home) \wedge$$
$$\exists url_2 \quad RelevantChain(home, url_2, query) \wedge Links(url_2, url) \wedge$$
$$page = GetPage(url)$$

- Имеется ссылка между **`URL from`** и **`to,`** текстом анкера является **`text`**

$$RelevantChain(start, end, query) \Leftrightarrow (start = end) \vee$$
$$\exists u, text \; LinkText(start, u, text) \wedge$$
$$RelevantCategoryName(query, text) \wedge$$
$$RelevantChain(u, end, query)$$

Мир покупок в Интернет

Иерархия товаров

$Books \subset Products$

$MusicRecordings \subset Products$

$MusicCDs \subset MusicRecordings$

$Electronics \subset Products$

$DigitalCameras \subset Electronics$

$Computers \subset Electronics$

$LaptopComputers \subset Computers$

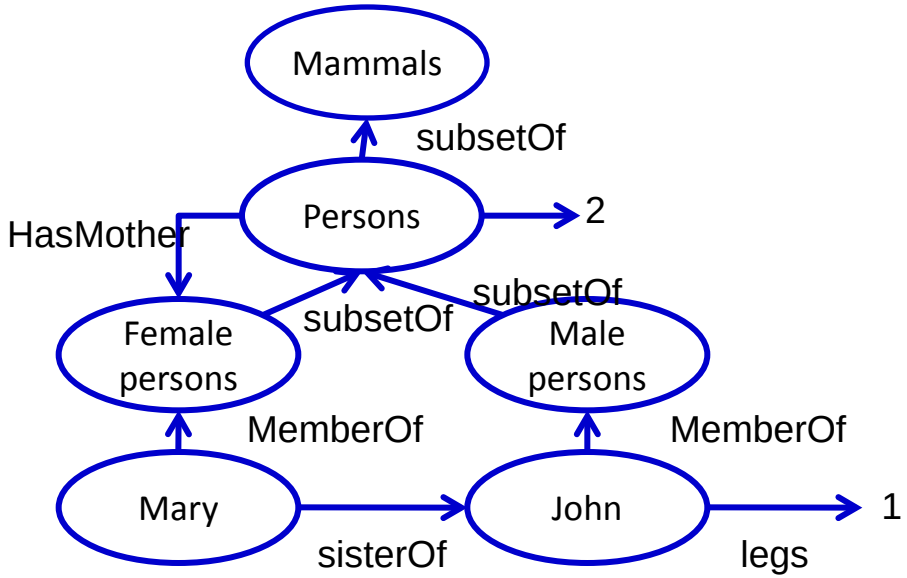
$DesktopComputers \subset Computers$

Проблема **синонимов** для одной категории товаров laptop и laptops computers

Проблема **омонимов** для разных категорий товаров CDs и CDs (Certificates of Deposit)

Семантические сети

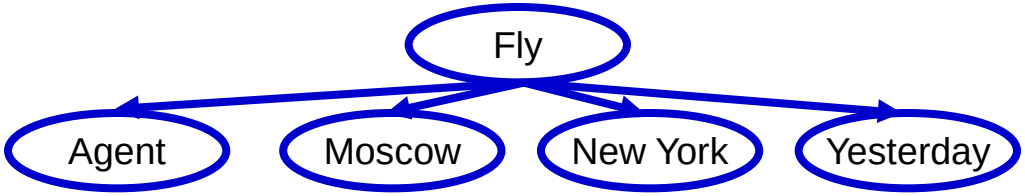
- Семантические сети позволяют **визуализировать KB**
- Семантические сети включают **категории и связи.**
- Связи могут иметь тип **`SubsetOf`**.
- Семантические сети используют принцип **наследования**.
- Разрешено **множественное наследование**.
- Использование **инверсных связей** HasSister и SisterOf



Достоинства и недостатки семантических сетей

Недостатки семантических сетей

- Связь предусмотрена только для двух категорий, поэтому возможны **бинарные отношения**
- Невозможны высказывания **с тремя** предикатами:
- `Fly(Agent, Moscow, NewYork, Yesterday)`
- Сложные связи **овеществляют** (представляют множеством бинарных связей)
- Отсутствуют **отрицание, дизъюнкция, кванторы существования**



Достоинства семантических сетей

- **Простота** логического вывода
- Использование **наследования**
- Использование **процедур** при выполнении запроса
- Представление заданных **по умолчанию** значений для категорий (автомобиль имеет 4 колеса)

Семантические сети

- Системы представления общего назначения, такие как **семантические сети** и **описательные логики**, позволяют организовать иерархии категорий.
- С этими системами связана важная форма логического вывода, называемого **наследованием**, позволяющая определять логическим путем свойства объектов на основании данных об их принадлежности к категориям.

Открытые и закрытые миры

- В логике первого порядка отсутствуют соглашения:
  - **О замкнутости мира** - предполагается, что базовые атомарные высказывания, не обозначенные как истинные, являются ложными.
  - **Об уникальности имен** - разные имена относятся к разным объектам

- В логике первого порядка объявлены университетские курсы:

`Course(10,”English”),Course(12,”Math”),Course(14,”Phisics”)`

- По умолчанию предполагается, что в университете имеются еще много курсов.
- Этот факт требуется дополнения в логике и первого порядка:

`Course(10,”English”),Course(12,”Math”),Course(14,”Phisics”),`
`Course(i,name)<-Integer(i)`

Открытые и закрытые миры

- **Предположение о замкнутом мире**, будучи реализованными в логических программах, предоставляют простой способ избежать необходимости задавать большие объемы отрицательной информации.
- Такую информацию интерпретируют как **заданную по умолчанию,** которая может быть переопределена с помощью дополнительной информации.

Логика косвенного описания

- **Немонотонность** логики – множество убеждений не возрастает монотонно со временем.
- Разработаны немонотонные логики: логика косвенного описания и логика умолчания.
- Косвенное описание использует предположение о замкнутом мире. Вводятся предикаты (**`Abnormal`**), которые ложны для всех объектов кроме явно указанных.

$$Bird(x) \wedge \neg Abnormal(x) \Rightarrow Flies(x)$$

- Косвенное описание - пример логики предпочтения моделей. Одна модель является более предпочтительной, чем другая.

Логика умолчания

- Запишем правило, применяемое по умолчанию: если выражение `Bird(x)` истинно, то выражение `Flight(x)` задано по умолчанию.

$$Berd(x):\frac{Flies(x)}{Flies(x)}$$

- Для формирования рассуждений по умолчанию предназначены **немонотонные логики**, такие как **логика косвенного описания** и **логика умолчания**.
- Применение **программирования** множества ответов позволяет ускорить немонотонный логический вывод.

Системы поддержки истинности

- **Пересмотр убеждений** при получении новых знаний выполняет система TMS (Truth Maintenance System)
- Олимпийские игры 2048 года в Румынии.

$$Site(Swimmong, Pitesty)$$

$$Site(Athletics, Bucharest)$$

$$Site(Equestrian, Arad)$$

- Изменили решение по легкой атлетике. TMS пересматривает всю KB

$$Site(Athletics, Sibiu)$$

- **Системы поддержки истинности** позволяют эффективно выполнять обновление и пересмотры баз знаний

Упражнение 1

- Запишите высказывание, позволяющее определить результаты Shot в мире вампуса. После выстрелы у агента нет стрелы.

$$At(o, x, s)$$
$$Shot(o, w, s)$$
$$HasArrow(o)$$
$$At(o, x, S_0) \Leftrightarrow \left[\left(o = Agent \wedge x = [4,1]\right) \vee \left(o = W \wedge x = [4,2]\right)\right]$$
$$\wedge\, HasArrow(o, S_0) \wedge Alive(w, S_0)$$
$$Wampus(W) \wedge Adjacent\left([4,1],[4,2]\right) \wedge Adjacent\left([4,2],[4,1]\right)$$
$$Result\left(Shot(o, w), \neg Alive(w), \neg HasArrow(o), S_0\right)$$

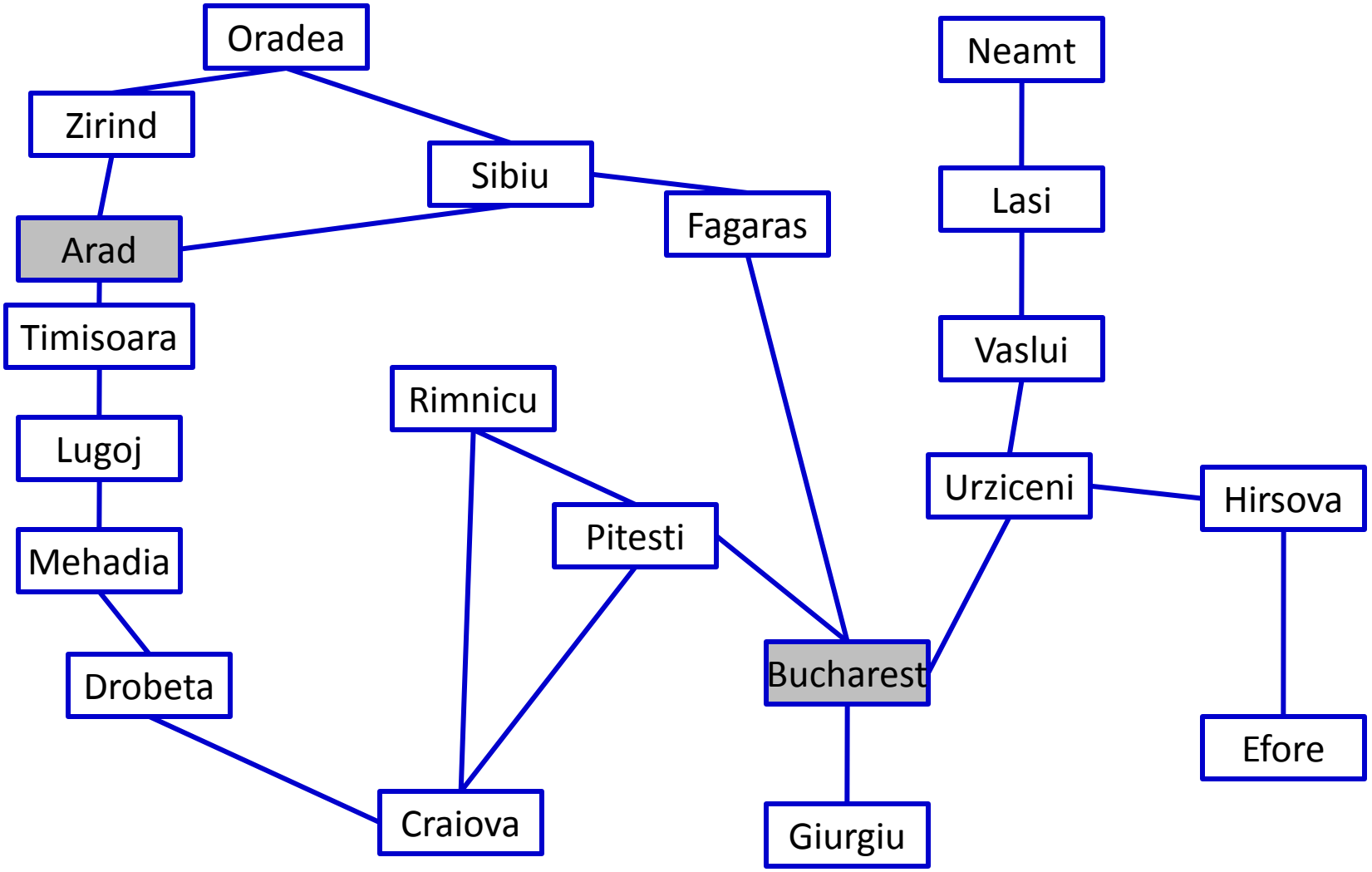
Упражнение 2

- Робот движется между городами. Действие **`Go(x,y)`** возможно, если имеется прямой путь **`DirectRoute(x,y).`** Робот начинает движение из Арада.
- Логическое описание начальной ситуации для робота.
- Запрос на описание возможных путей для робота.
- Описание действия Go.

- Робот стартует с полным баком и тратит топливо пропорционально пути.
- Получите новое описание Go.

- Имеются заправочные стации в пути. Дополните правила движения с учетом Fillup.

Дорожная карта Румынии



Планирование

Лекция 9

Язык планирования

- **Планирование** – это последовательность действий, которая из начального состояния приводит к состоянию, удовлетворяющему цели.
- Язык планирования – **`Strips.`**

Представление состояний

- Литералы для описания состояний не должны содержать функции.
- Используют только положительные литералы.
- Используют предположение о замкнутом мире.

Представление целей

Цель представляется в виде конъюнкции литералов

$$Rich \wedge Famos$$

- Достигнутое состояние соответствует достижению цели:

$$Rich \wedge Famos \wedge Miserable$$

Схема действия

Представление действий

- Действие представляется в виде предусловий, действий и результатов (Effect)

$$Action(Fly(p, from, to):$$

$$Precond: At(p, from) \wedge Plane(p)) \wedge AirPort(from) \wedge AirPort(to),$$

$$Effect: \neg At(p, from) \wedge At(p, to)$$

- **Имя действия** `Fly(p,from,to)`
- **Предусловие** задано конъюнкцией, не содержащих функций
- **Результатом** является конъюнкция, не содержащая функций. Подтверждается положительный и отрицательный литерал.

$$P(x, y) \wedge \neg P(x, z)$$

Схема действия

- Текущее состояние:

$$At(P_1, Moscow) \wedge At(P_2, NewYork) \wedge Plane(P_1) \wedge Plane(P_2) \wedge$$
$$Airport(Moscow) \wedge Airport(NewYork)$$

- Состояние удовлетворяет предусловию:

$$At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$$

с подстановкой: $\frac{p}{P_1}; \frac{from}{Moscow}; \frac{to}{NewYork}$

- После выполнения действия $Fly(P_1, Moscow, NewYork)$

- После выполнения действия:

$$At(P_1, NewYork) \wedge At(P_2, NewYork) \wedge Plane(P_1) \wedge Plane(P_2) \wedge$$
$$Airport(Moscow) \wedge Airport(NewYork)$$

Язык планирования ADL (action description Language)

- **Язык описания действий ADL**
- Использует **положительные** и **отрицательные** литералы

$$\neg Rich \wedge \neg Famous$$

- Предположение **об открытом мире** (неупомянутые литералы считаются неизвестными)
- Использует **кванторы** в целях

$$\exists x \;\; At(P_1, x) \wedge At(P_2, x)$$

- В целях допускается и **конъюнкции и дизъюнкции**

$$\neg Poor \wedge (Famous \vee Smart)$$

- Разрешены условные результаты **when**.
- Переменные могут иметь типы

Воздушный транспорт

Задача Strips.

- Транспортировка грузов между аэропортами.

- Груз C1 находится в Москве, груз C2 – в Нью-Йорке.
- Самолет P1 находится в Москве, груз P2 – в Нью-Йорке.

$$Init\big(At(C_1, Moscow) \wedge At(C_2, NewYork) \wedge At(P_1, Moscow) \wedge At(P_2, NewYork) \wedge$$
$$Cargo(C_1) \wedge Cargo(C_2) \wedge Plane(P_1) \wedge Plane(P_2) \wedge$$
$$Airport(NewYork) \wedge Airport(Moscow)\big)$$

- Цель – обмен грузами

$$Goal\big(At(C_1, NewYork) \wedge At(C_2, Moscow)\big)$$

Воздушный транспорт

- Действие – погрузка в самолет в аэропорту

$$Action\big(Load(c, p, a),$$
$$Precond: At(c, a) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$$
$$Effect: \neg At(c, a) \wedge In(c, p)\big)$$

- Действие – выгрузка из самолета в аэропорту

$$Action\big(Unload(c, p, a),$$
$$Precond: In(c, p) \wedge At(p, a) \wedge Cargo(c) \wedge Plane(p) \wedge Airport(a)$$
$$Effect: At(c, a) \wedge \neg In(c, p)\big)$$

- Действие – полет

$$Action\big(Fly(p, from, to),$$
$$Precond: At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$$
$$Effect: \neg At(p, from) \wedge At(p, to)\big)$$

Задача с запасным колесом

Задача ADL «Смена колеса со стертой покрышкой»

- Запасное колесо находится в багажнике.

$$InitAt\big((Flat, Axle) \wedge At(Spare, Trunk)\big)$$

•Цель – установить запасное колесо.

$$Goal\big(At(Spare, Axle)\big)$$

- Достать запасное колесо и положить на землю

$$Action\big(Remove(Spare, Trunk),$$
$$Precond: At(Spare, Trunk)$$
$$Effect: \neg At(Spare, Trunk) \wedge At(Spare, Ground)\big)$$

Задача с запасным колесом

- Снять колесо со стертой покрышкой и положить его на землю

$$Action\big(Remove(Flat, Axle),$$
$$Precond : At(Flat, Axle)$$
$$Effect : \neg At(Flat, Axle) \wedge At(Flat, Ground)\big)$$

- Поднять с земли и установить запасное колесо

$$Action\big(PutOn(Spare, Axle),$$
$$Precond : At(Spare, Ground) \wedge \neg At(Flat, Axle)$$
$$Effect : \neg At(Spare, Ground) \wedge At(Spare, Axle)\big)$$

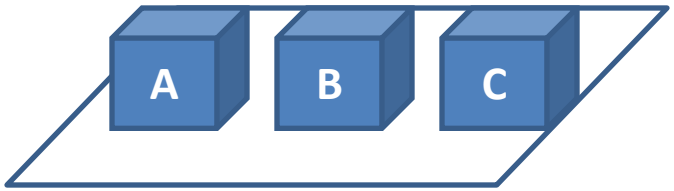
Планирование на языках Strips и ADL

- В языке Strips применяются описания действий в терминах их предусловий и ADL некоторые ограничения языка Strips ослаблены и допускается использование дизъюнкции, отрицания и кванторов.
- В языке Strips применяются описания действий в терминах их предусловий и ADL некоторые ограничения языка Strips ослаблены и допускается использование дизъюнкции, отрицания и кванторов.

Мир блоков

- Блоки A,B,C находятся на поверхности стола.

$$Init\big(On(A,Table) \wedge On(B,Table) \wedge On(C,Table)$$
$$\wedge Block(A) \wedge Block(B) \wedge Block(C)$$
$$\wedge Clear(A) \wedge Clear(B) \wedge Clear(C)\big)$$

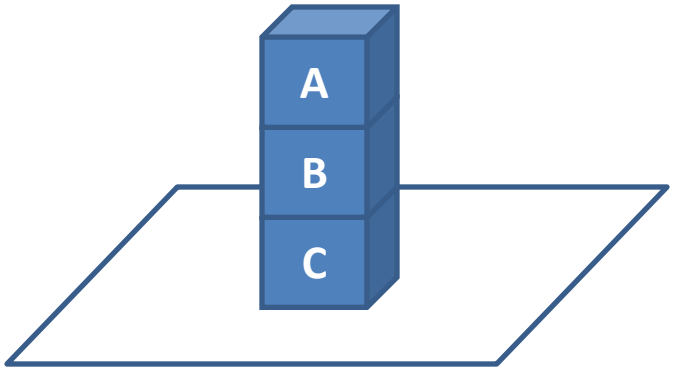


- Цель – построить из блоков столбик

$$Goal\big(On(A,B), On(B,C)\big)$$

- Последовательность действий

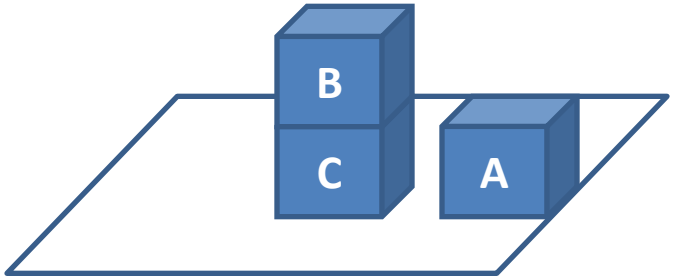
$$Move(B, Table, C),$$
$$Move(A, Table, B)$$



Мир блоков

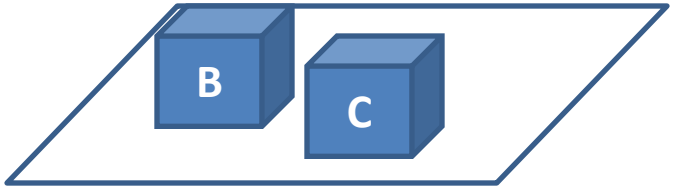
- Переместить блок ***b*** с верхней поверхности ***x*** на верхнюю поверхность ***y***

$$Move(B, Table, C),$$
$$Action(Move(b, x, y),$$
$$Precond: On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge$$
$$(b \neq x) \wedge (b \neq y)(x \neq y),$$
$$Effect: On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y))$$



- Перемещение блока b с блока x на стол.

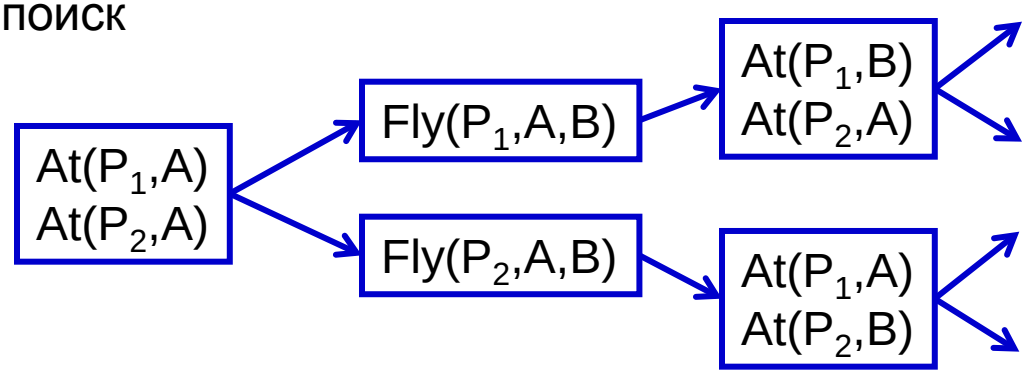
$$MoveToTable(B, C)$$
$$Action(MoveToTable(b, x),$$
$$Precond: On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x),$$
$$Effect: On(b, Table) \wedge Clear(x) \wedge \neg On(b, x))$$



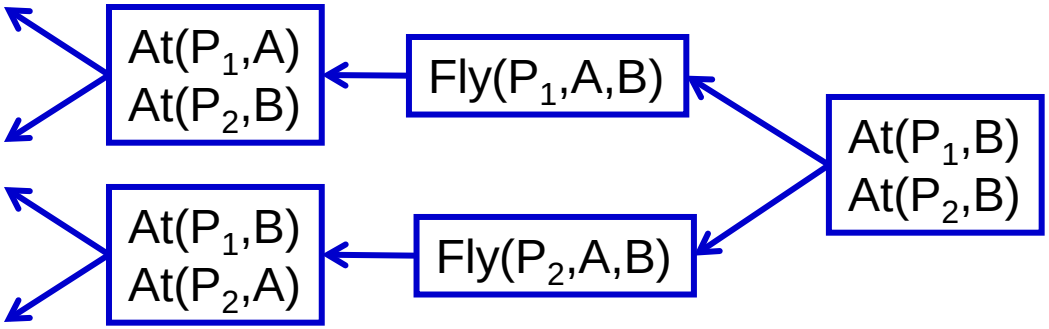
Планирование с помощью поиска в пространстве состояний

Два подхода к организации поиска

- Прямой (прогрессивный) поиск



- Обратный (регрессивный) поиск



Планирование с помощью поиска в пространстве состояний

Прямой (прогрессивный) поиск

- Начальное состояние
- Действия
- Проверка цели

Недостаток – большое число узлов поиска.

Пример: Требуется доставить 20 единиц груза в аэропорт В. Всего 10 аэропортов, 5 самолетов в каждом аэропорту.

5*10 (самолетов) каждый летит в 9 аэропортов, перевозит 20*10 (единиц груза)

Обратный (регрессивный) поиск

- Конечное состояние
- Только релевантные и совместимые действия
- Достоинство – меньшее число узлов поиска.

Планирование с помощью поиска в пространстве состояний

Обратный (регрессивный) поиск

- Рассматривает только **релевантные действия.**
- Релевантные действия – один из конъюнктов CNF достигает цели.

$$At(C_1, B) \wedge At(C_2, B) \wedge ... \wedge At(C_{20}, B)$$

- Пусть $At(C_1, B) = true$
- Предусловия $In(C_1, p) \wedge At(p, B) = true$
- Состояние - приемник $In(C_1, p) \wedge At(p, B) \wedge At(C_2, B) \wedge ... \wedge At(C_{20}, B)$
- Тогда груз можно доставить на любом самолете **p**

$$Unload(C_1, p, B) = true$$

Действия должны быть релевантными и совместимыми.

Пример несовместимого действия: $Load(C_2, p)$

Эвристики

- Вывести ослабленную задачу
- Удаление из действий всех предусловий
- Предположение о независимости подцелей – достижение отдельных подцелей.

$$Goal(A \wedge B \wedge C)$$
$$Action(X, Effect: A \wedge P)$$
$$Action(Y, Effect: B \wedge C \wedge Q)$$
$$Action(Z, Effect: B \wedge P \wedge Q)$$

Минимум действий для достижения цели {X,Y}
Эвристика дает 2

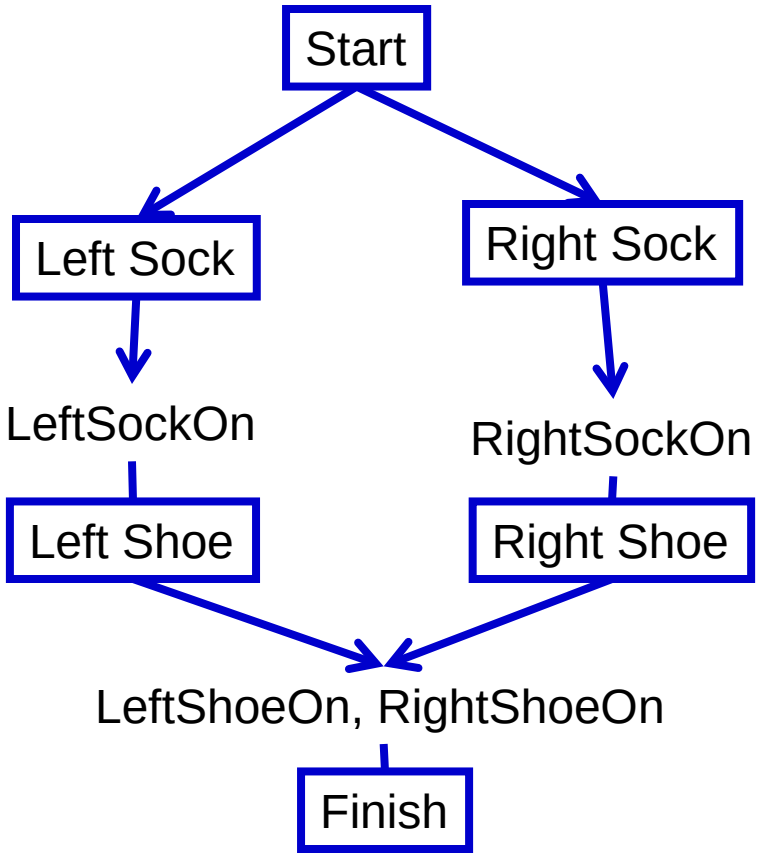
Планирование с частичным упорядочением

- Для выполнения прямого и обратного поиска необходимо выполнить упорядочение плана
- Задача надевания пары туфель

$$Goal(RightShoeOn \wedge LeftShoeOn)$$
$$Init()$$
$$Action(RightShoe, Precond: RightSockOn, Effect: RightShoeOn)$$
$$Action(RightSock, Effect: RightSockOn)$$
$$Action(LeftShoe, Precond: LeftSockOn, Effect: LeftShoeOn)$$
$$Action(LeftSock, Effect: LeftSockOn)$$

- **Планированием с частичным упорядочением** называется алгоритм способный включить в план два действия независимо (в любой последовательности)

План с частичным упорядочением



Планирование с частичным упорядочением

План включает:

- Множество действий LeftSock, RightSock,…, (действия Start и Finish – пустые)
- Множество ограничений упорядочения

$$LeftSock \prec LeftShoe$$

$$RightSock \prec RightShoe$$

- Множество причинных связей «A достигает p для B» $A \xrightarrow{p} B$

  Надетый носок является результатом действия LeftSock и предусловием для действия LeftShoe

$$LeftSock \xrightarrow{LeftSockOn} LeftShoe$$

$$LeftShoe \xrightarrow{LeftShoeOn} Finish$$

- Множество открытых предусловий. Предусловие является открытым, если оно не достигнуто в плане. Это множество следует делать пустым.

Согласованный план

- **Согласованным** является план, который **не имеет циклов** типа:

$$A \prec B,\ B \prec A$$

- В ограничениях **нет конфликтов** типа:

$$\begin{cases} A \xrightarrow{p} B \\ A \xrightarrow{\neg p} C \\ A \prec C, C \prec B \end{cases}$$

- **Решение** – это согласованный план без открытых предусловий

Задача с запасным колесом

$$Init\big(At(Flat, Axle), At(Spare, Trunk)\big)$$

$$Goal\big(At(Spare, Axle)\big)$$

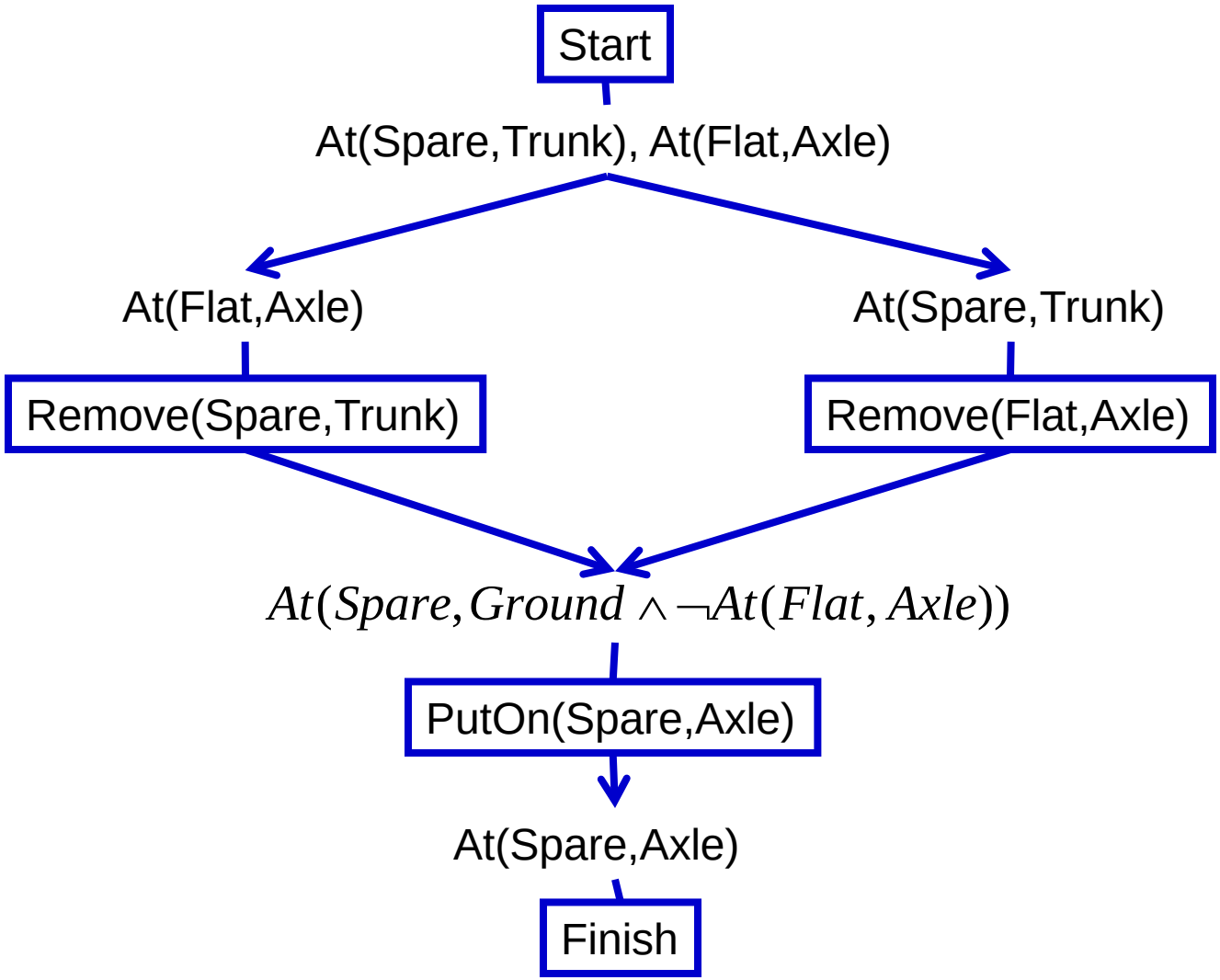
$$Action\begin{pmatrix} Remove(Spare, Trunk), \\ Precond: At(Spare, Trunk), \\ Effect: \neg At(Spare, Trunk) \wedge At(Spare, Ground) \end{pmatrix}$$

$$Action\begin{pmatrix} Remove(Flat, Axle), \\ Precond: At(Flat, Axle), \\ Effect: \neg At(Flat, Axle) \wedge At(Flat, Ground) \end{pmatrix}$$

$$Action\begin{pmatrix} PutOn(Spare, Axle), \\ Precond: At(Spare, Ground) \wedge \neg At(Flat, Axle), \\ Effect: \neg At(Spare, Ground) \wedge At(Spare, Acle) \end{pmatrix}$$

$$Action\begin{pmatrix} LeaveOverNight, \\ Precond: \\ Effect: \neg At(Spare, Ground) \wedge \neg At(Spare, Axle) \wedge \\ \neg At(Flat, Ground) \wedge \neg At(Flat, Axle) \end{pmatrix}$$

Задача с запасным колесом



Графы планирования

Свойства графа планирования:

- дает лучшие решения для эвристик
- состоит из уровней
- представляет не только действие, но и бездействие
- бездействие представляется **множеством сохраняющих действий**

Задача с кексом

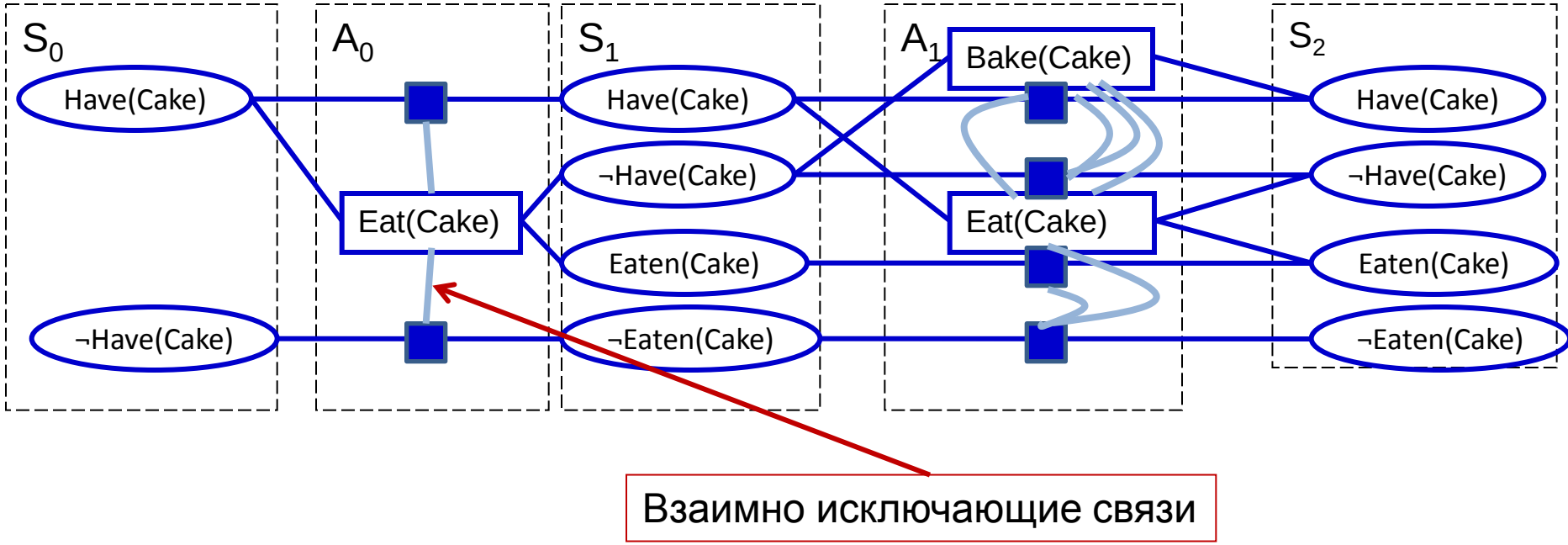
$$Init(Have(Cake))$$

$$Goal\big(Have(Cake) \wedge Eaten(Cake)\big)$$

$$Action\big(Eat(Cake), Precond : Have(Cake)\ Efect : \neg Have(Cake) \wedge Eaten(Cake)\big)$$

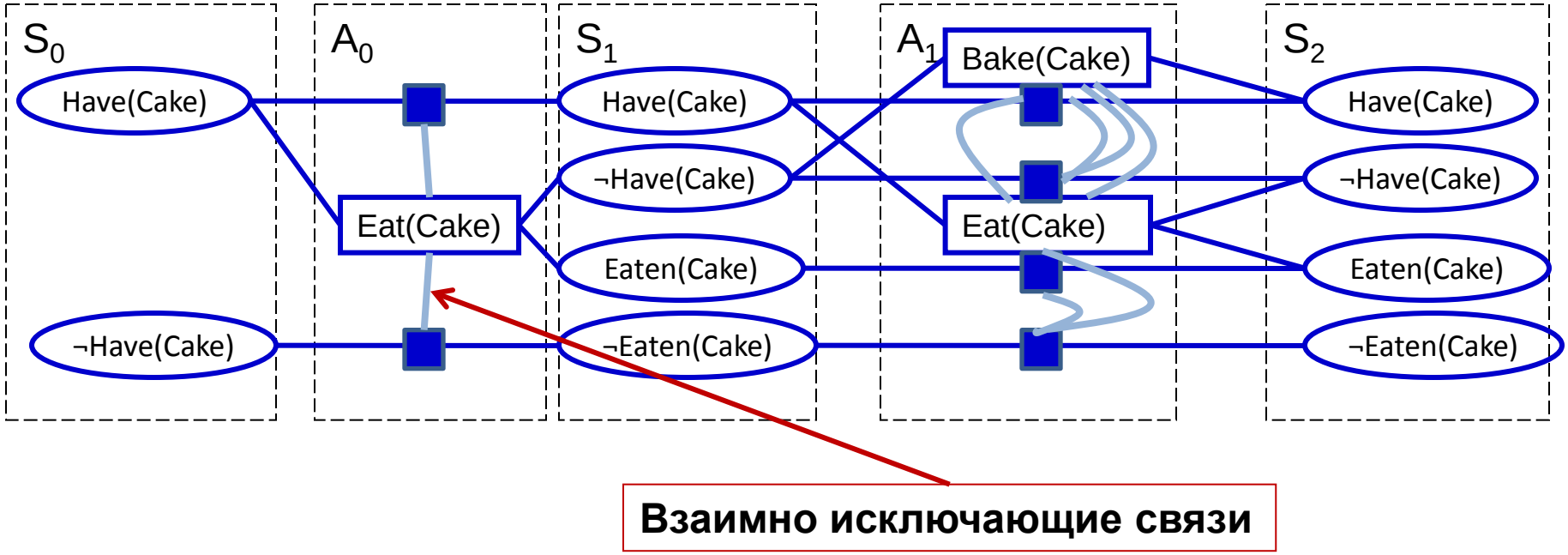
$$Action\big(Bake(Cake), Precond : \neg Have(Cake)\ Efect : Have(Cake)\big)$$

Графы планирования



S_0,S_1,S_2 – состояния
A_0,A_1 – действия уровня 0 и 1
Конечный уровень: на следующем уровне состояния не изменяются
Состояния не изменяются в точке выравнивания

Графы планирования

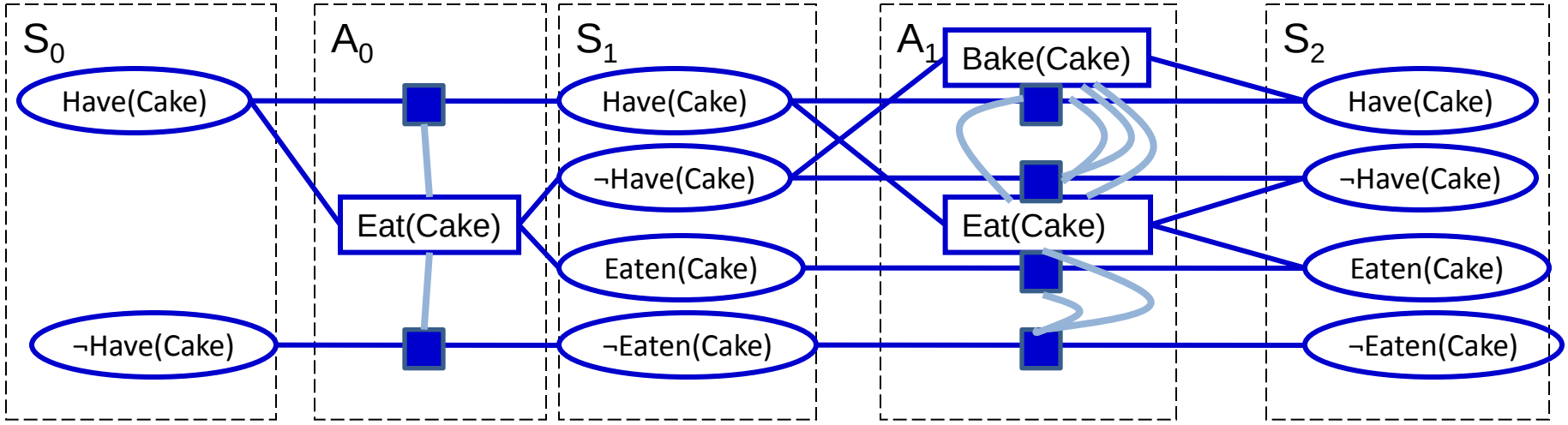


1.**Несогласованные результаты .** Одно из действий отрицает результат другого: Eat(Cake) и Have(cake)

2. **Вмешательство.** Один из результатов действия отрицает предусловие другого: Eat(Cake) и Have(cake)

3. **Конкурирующие потребности.** Предусловие одного действия исключает предусловие другого. Bake(Cake),Eat(Cake) конкурируют за Have(Cake)

Эвристики



- **Эвристика уровневая стоимость.** Have(Cake) имеет уровневую стоимость **=0,** Eaten(Cake)имеет стоимость **=1**
- **Эвристика максимального уровня** – максимальная стоимость цели **= 1**
- **Эвристика уровневой суммы** – сумма стоимостей целей. Суммарная стоимость Have(Cake)^ Eaten(Cake) равна **0+1=1**
- **Эвристика множественного уровня** – уровень, на котором все литералы CNF не имеют взаимных исключений . Have(Cake)^ Eaten(Cake) равна **=2,**

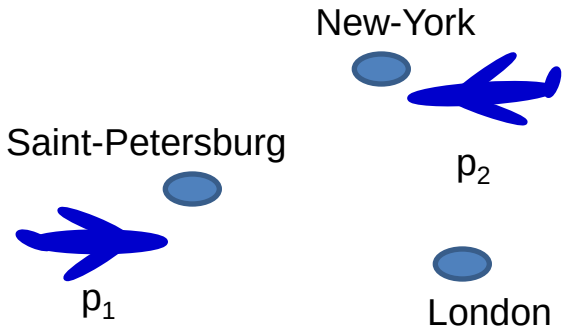
Планирование в пространстве состояний

- **Поиск в пространстве состояний** может действовать в прямом или в обратном направлении. Эффективные эвристики могут быть получены путем принятия предположения о независимости подцелей, а также с помощью ослабления задачи планирования.
- **Граф планирования** может формироваться инкрементно, начиная с начального состояния. Каждый уровень содержит действия, которые могут обнаруживаться на данном временном этапе. На каждом уровне задаются **взаимно-исключающие отношения** между действиями, которые не могут происходить одновременно.
- Графы планирования позволяют получить полезные **эвристики** для планирования.

Пропозициональная логика

Проверка выполнимости высказывания:

$$Initialization \wedge Actions \wedge Goal$$



Задача воздушной транспортировки

Инициализация на этапе 0 $\mathrm{At}(p_1, \mathrm{Saint-Petersburg})^0 \wedge \mathrm{At}(p_2, \mathrm{New-York})^0$

Используется модель **открытого мира**.
Необходимо добавить при инициализации:

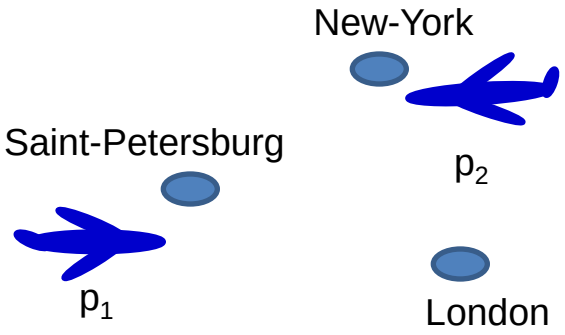
$$\neg\mathrm{At}(p_1, \mathrm{New-York})^0 \wedge \neg\mathrm{At}(p_2, \mathrm{Saint-Petersburg})^0$$

Цель достигается на этапе T: $\mathrm{At}(p_2, \mathrm{Saint-Petersburg})^T \wedge \mathrm{At}(p_1, \mathrm{New-York})^T$

Описание действий в пропозициональной логике

Задание аксиом состояния-преемника

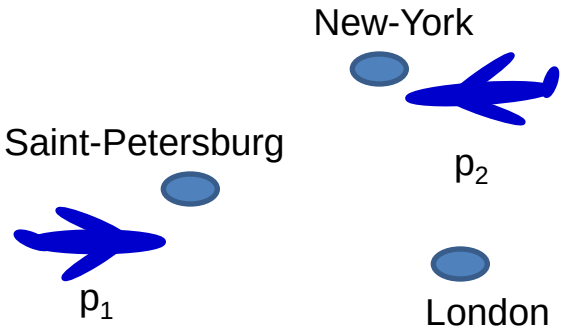
Самолет p_1 при T=1 будет в Нью-Йорке,
1. Если он уже находился в Нью-Йорке при T=0 и не улетел в Санкт-Петербург,
2. Если он находился в Санкт-Петербурге и перелетел в Нью-Йорк.



$$At(p_1,New\text{-}York)^1 \Leftrightarrow At(p_1,New\text{-}York)^0 \wedge$$
$$\neg\left(Fly(p_1,New\text{-}York,Saint\text{-}Petersburg)^0 \wedge At(p_1,New\text{-}York)^0\right)\vee$$
$$\left(Fly(p_1,Saint\text{-}Petersburg,New\text{-}York)^0 \wedge At(p_1,Saint\text{-}Petersburg)^0\right)$$

Число таких аксиом равно:
Число самолетов * Число городов * Число временных этапов

Планирование в пропозициональной логике



Наилучший план очевиден:

$$Fly(p_1, New\text{-}York, Saint\text{-}Petersburg)^0 \wedge$$

$$Fly(p_2, Saint\text{-}Petersburg, New\text{-}York)^0$$

Аксиомы предусловий

$$Fly(p_1, New\text{-}York, Saint\text{-}Petersburg)^0 \Rightarrow At(p_1, New\text{-}York)^0$$

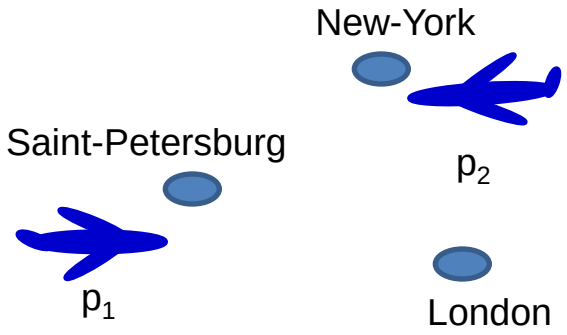
Аксиомам предусловий удовлетворяют действия, позволяющие вылететь в два аэропорта одновременно

$$Fly(p_2, New\text{-}York, Saint\text{-}Petersburg)^0$$

$$Fly(p_1, Saint\text{-}Petersburg, New\text{-}York)^0$$

$$Fly(p_2, New\text{-}York, London)^0$$

Планирование в пропозициональной логике



Аксиомы исключений

Предотвращают несовместимые действия

$$Fly(p_2, New\text{-}York, Saint\text{-}Petersburg)^0 \wedge$$
$$\neg Fly(p_2, New\text{-}York, London)^0$$

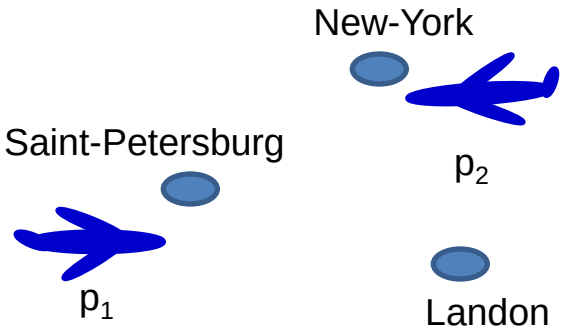
Аксиомы состояний

В более общем виде: ни один объект не мог находиться в двух местах одновременно:

$$\forall p, x, y, t \qquad x \neq y \Rightarrow \neg\left(At(p, x)^t \vee At(p, y)^t\right)$$

Описание действий в пропозициональной логике

Задание аксиом состояния-преемника



$$At(p_1,New\text{-}York)^1 \Leftrightarrow$$
$$At(p_1,New\text{-}York)^0 \wedge \neg Fly(p_1,New\text{-}York,Saint\text{-}Petersburg)^0 \wedge$$
$$\neg Fly(p_1,New\text{-}York,London)^0 \vee \neg Fly(p_1,Saint\text{-}Petersburg, New\text{-}York)^0 \vee$$
$$Fly(p_1,London, New\text{-}York)^0$$

Число таких аксиом равно:
Число самолетов * Число городов * Число временных этапов

Неопределенность

Лекция 10

Действия в условиях неопределенности

- Агенты никогда не имеют доступа ко всем необходимым сведениям о среде
- Агенты должны действовать в условиях **неопределенности**
  - Датчики в мире Вампуса имеют только локальную информацию
  - Агент-такси доставит пассажира в аэропорт согласно плану P, если не будет поломки, не закончится топливо, не будет аварий на трассе и т.д.
- Выбор правильной стратегии зависит от важности различных целей (доставить пассажира, избежать аварии, …), от степени уверенности, что они будут выполнены.
- В медицинской диагностике решения принимают в условиях неопределенности:
  - Слишком много усилий требуется, чтобы составить правила, учитывающие все факторы
  - Отсутствие полных теоретических знаний в медицине
  - Отсутствие практических знаний о конкретном пациенте

Вероятность

- Знания агента позволяют иметь некоторую **степень уверенности**
- **Теория вероятностей** позволяет оценить степень уверенности
- **Нечеткая логика** позволяет оценить степень истинности знаний агента
- Степень уверенности зависит от результатов восприятия среды - измерений
- По мере получения новых данных вероятностные оценки обновляются

Неопределенность и рациональные решения

- Задача – доставить пассажира в аэропорт
  - План $A_{0.5}$ (выехать за 0.5 часа) имеет вероятность успеха 95%
  - План A_{24} (выехать за 24 часа) имеет вероятность успеха почти 100%
- Агент должен получить информацию о **предпочтениях** между разными планами. Какой план более полезный?
- **Теория полезности** позволяет оценить функцию полезности
- **Теория решений** = Теория вероятностей + Теория полезности

Принцип максимальной ожидаемой полезности

- Любой агент является **рациональным** тогда и только тогда, когда он выбирает действие, позволяющее достичь **наибольшей ожидаемой полезности**, усредненной по всем возможным результатам этого действия

Высказывания

- **Случайная переменная,** которая ссылается на состояние мира имеет тип и область определения
- **Булевы** случайные переменные имеют область определения **<true,false>**
- **Дискретные** случайные переменные принимают значения из счетной области, например, **Weather <sunny,rainy,cloudy,snow>**
- **Непрерывные** случайные переменные, принимают значения из некоторого диапазона, например, [0,1]

Атомарные события

Атомарные события являются взаимно исключающими

$$sunny \wedge rainy \wedge cloudy \wedge snow = false$$

Множество всех возможных событий является исчерпывающим

$$sunny \vee rainy \vee cloudy \vee snow = true$$

Из атомарного события следует истинность или ложность высказывания

$$sunny \wedge \neg cloudy \Rightarrow sunny$$

$$sunny \wedge \neg cloudy \Rightarrow \neg cloudy$$

Любое высказывание эквивалентно конъюнкции всех атомарных событий. Высказывание `cloudy` эквивалентно:

$$cloudy \wedge rainy \vee cloudy \wedge \neg rainy$$

Априорная вероятность

Вероятности всех возможных атомарных событий состояний погоды

$$P(Weather = sunny) = 0.7$$

$$P(Weather = rainy) = 0.2$$

$$P(Weather = cloudy) = 0.08$$

$$P(Weather = snow) = 0.02$$

Распределение априорных вероятностей

$$Weather = \langle sunny, rainy, cloudy, snow \rangle$$

$$P(Weather) = \langle 0.7, 0.2, 0.08, 0.02 \rangle$$

Дискретные случайные переменные

$$LevelOfHealth = \langle healthy, headache, toothache, influenza \rangle$$

$$P(LevelOfHealth) = \langle 0.7, 0.2, 0.08, 0.02 \rangle$$

Совместное распределение вероятностей

Совместное распределение вероятностей Weather и LevelOfHealth:

$$P(Weather, LevelOfHealth)$$

	sunny	rain	cloudy	snow	P(LevelOfHealth)
healthy	0.49	0.14	0.056	0.014	**0.7**
headache	0.14	0.04	0.016	0.004	**0.2**
toothache	0.056	0.016	0.0064	0.0016	**0.08**
influenza	0.014	0.004	0.0016	0.0004	**0.02**
P(Weather)	**0.7**	**0.2**	**0.08**	**0.02**	

Для **непрерывной переменной t** (температура воздуха) выполним прогноз:

$$P(T = t) = U[18,22]t$$

U[18,22]- значение температуры распределено равномерно между 18 и 22 градусами

Условная вероятность

- После того, как агент получает информацию о среде, априорные вероятности становятся неприемлемыми.
- Апостериорные или условные вероятности дают диагностическую информацию:

$$P(headache/rain) = 0.4$$

$$P(headache/rain) = \frac{P(headache \wedge rain)}{P(rain)} = \frac{0.04}{0.2} = 0.2$$

- Правило произведения вероятностей:

$$P(a \wedge b) = P(a/b) \cdot P(b);$$

$$P(headache \wedge rain) = P(headache/rain) \cdot P(rain)$$

Истоки понятия вероятности

- **Частотная** интерпретация. Числовые оценки могут быть получены только в эксперименте. Например из 100 человек у 5 наблюдается головная боль. Вероятность головной боли равна 5%, если в пределе обследовать бесконечное число людей.
- **Объективистская** интерпретация. В поведении объектов имеются закономерности, которые мы не учитываем. Например, при бросании монеты случайность выпадения герба (P=0.5) обусловлена незнанием начальных условий.
- **Субъективистская** интерпретация. Способ описания уверенности агента. Например, я на 90% уверен, что мы успеем вовремя попасть в аэропорт.
- Использование **референтного класса**. Врач оценивает вероятность наличия болезни гриппа у пациента **N** на основании совпадения симптомов у этого пациента с симптомами группы, состоящей из **M** пациентов.

Аксиомы вероятностей

Аксиомы вероятностей:

$$0 \le P \le 1;$$

$$P(true) = 1; \;\; P(\, false) = 0;$$

$$P(a \vee b) = P(a) + P(b) - P(a \wedge b)$$

Рациональный агент должен действовать в соответствии с аксиомами вероятностей.

Утверждения относятся не к самому миру, а к состоянию знаний агента.

$$P(a) = 0.4$$

$$P(a \wedge b) = 0.0$$

$$P(b) = 0.3$$

$$P(a \vee b) = 0.8$$

Теорема де Финетти

Если агент 1 руководствуется множеством степеней уверенности, которое **нарушает** аксиомы теории вероятностей, то существует такая комбинация ставок агента 2, которая гарантирует, что агент 1 будет терять деньги при каждой ставке.

$$P(a) = 0.4$$
$$P(a \wedge b) = 0.0$$
$$P(b) = 0.3$$
$$P(a \vee b) = 0.8$$

Агент 1		**Агент2**			**Результат для Агента 1**			
Высказывание	Степень уверенности	Ставка	Сумма ставки		**a^b**	**a^¬b**	**¬a^b**	**¬a^¬b**
a	0.4	a	4	-6	-6	-6	4	4
b	0.3	b	3	-7	-7	3	-7	3
a or b	0.8	¬(a or b)	2	-8	2	2	2	-8
					-11	**-1**	**-1**	**-1**

Вероятностный вывод

X – переменная запроса
E – множество измеряемых переменных
У – множество ненаблюдаемых (не измеряемых) переменных
{X,E,У} – полное множество переменных
e – наблюдаемое значение этих переменных
y –значения не наблюдаемых переменных
Запрос:

$$P(X/\mathbf{e}) = \frac{P(X,\mathbf{e})}{P(\mathbf{e})} = \frac{1}{P(\mathbf{e})} \cdot \sum_{\mathbf{y}} P(X,\mathbf{e},\mathbf{y})$$

Правило Байеса

Правило произведения:

$$P(a \wedge b) = P(a/b) \cdot P(b)$$
$$P(a \wedge b) = P(b/a) \cdot P(a)$$

Правило Байеса:

$$P(b/a) = \frac{P(a/b) \cdot P(b)}{P(a)}$$

Диагностические знания: «головная боль является признаком гриппа в 20 случаях из 100».

$$P(influence/headache) = 0.2$$

Причинные знания: «во время гриппа болит голова в 90 случаях из 100»

$$P(headache/influence) = 0.9$$

Диагностические знания получить труднее, чем причинные

Мир Вампуса

Агент не может перейти в следующий квадрат без опасений.

[4,1] [4,2] [3,1] – известные квадраты,
[2,1],[3,2],[4,3] - могут содержать яму
Вероятность наличия ямы ***P**= 3/(16-1)=0.2*
Полное совместное распределение вероятности наличия ям и ветерков:
***P**(Pit[1,1],…,Pit[4,4],Br[4,1], Br[3,1],Br[4,2])*

$$\mathbf{P}\left(Pit[1,1],\ldots,Pit[4,4],Br[4,1],Br[3,1],Br[4,2]\right)=$$
$$\mathbf{P}\left(Br[4,1],Br[3,1],Br[4,2]/Pit[1,1],\ldots,Pit[4,4]\right)\cdot$$
$$\mathbf{P}\left(Pit[1,1],\ldots,Pit[4,4]\right)$$

Мир Вампуса

Каждый квадрат содержит яму независимо от других квадратов

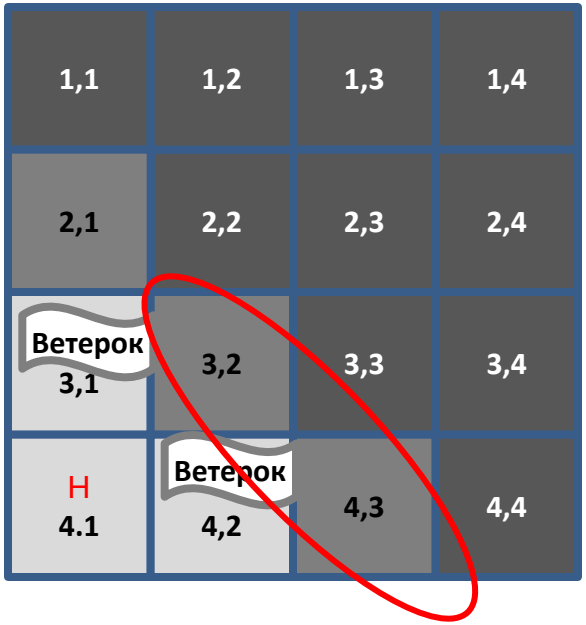
$$\mathbf{P}(Pit[1,1],\ldots,Pit[4,4]) = \sum_{i=1}^{4}\sum_{j=1}^{4}\mathbf{P}(Pit[i,j])$$

Исследовано, что в квадратах нет ямы

$$known = \neg Pit[4,1] \wedge \neg Pit[3,1] \wedge \neg Pit[4,2]$$

Исследовано, что в квадратах чувствуется ветерок

$$br = \neg br[4,1] \wedge br[3,1] \wedge br[4,2]$$

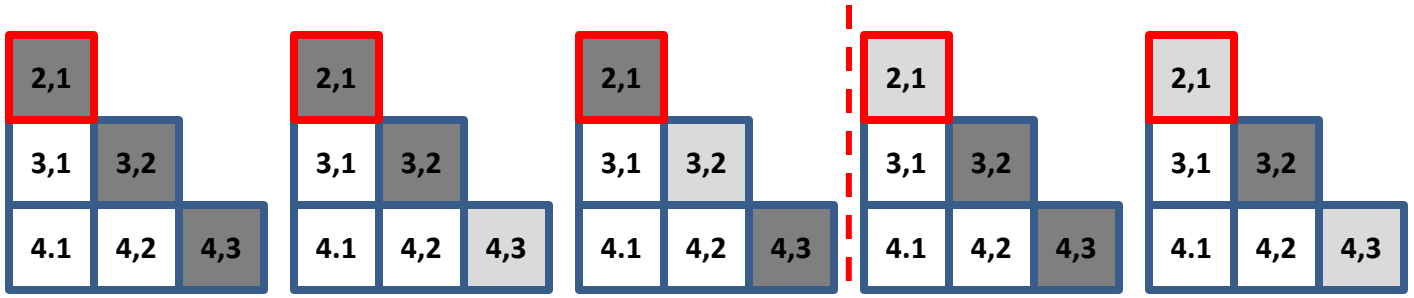


Вероятность ямы в квадрате [2,1]:

$$P(Pit[2,1]/known,br) = \beta \cdot P(Pit[2,1]) \sum_{fringe} P(br/known, Pit[2,1], fringe) \cdot P(fringe)$$

Мир Вампуса

Вероятность наличия ямы в квадрате [2,1]



$$\mathbf{P}(Pit[2,1]/known,br) = \beta \cdot 0.2 \cdot (0.04 + 0.16 + 0.16) = \beta \cdot 0.072$$

$$\mathbf{P}(\neg Pit[2,1]/known,br) = \beta \cdot 0.8 \cdot (0.04 + 0.16) = \beta \cdot 0.16$$

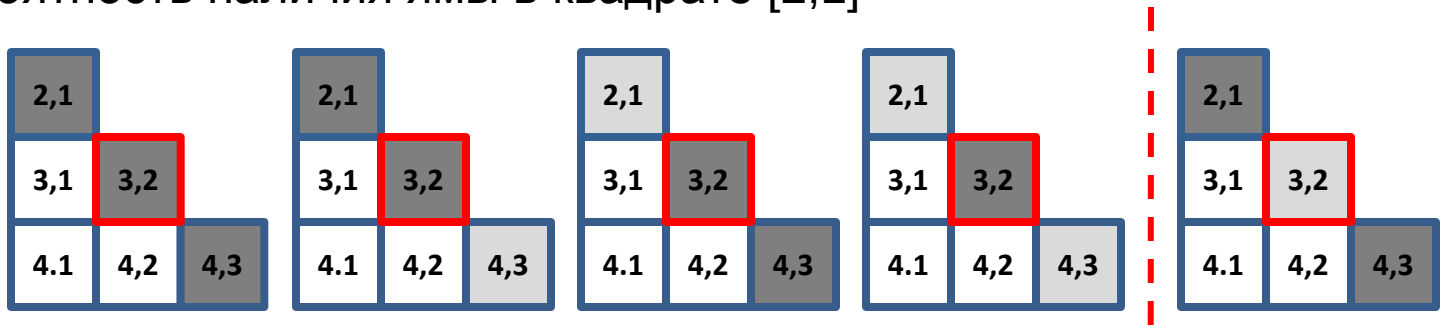
$$\beta = 1/(0.072 + 0.16) = 1/0.232 = 4.31$$

$$\mathbf{P}(Pit[2,1]/known,br) = 4.31 \cdot 0.072 = 0.31$$

$$\mathbf{P}(\neg Pit[2,1]/known,br) = 4.31 \cdot 0.16 = 0.69$$

Мир Вампуса

Вероятность наличия ямы в квадрате [2,1]



$$\mathbf{P}(Pit[3,2]/known,br) = \beta \cdot 0.2 \cdot (0.04 + 0.16 + 0.16 + 0.64) = \beta \cdot 0.2$$

$$\mathbf{P}(\neg Pit[3,2]/known,br) = \beta \cdot 0.8 \cdot (0.04) = \beta \cdot 0.032$$

$$\beta = 1/(0.2 + 0.032) = 1/0.232 = 4.31$$

$$\mathbf{P}(Pit[3,2]/known,br) = 4.31 \cdot 0.2 = 0.86$$

$$\mathbf{P}(\neg Pit[3,2]/known,br) = 4.31 \cdot 0.032 = 0.14$$

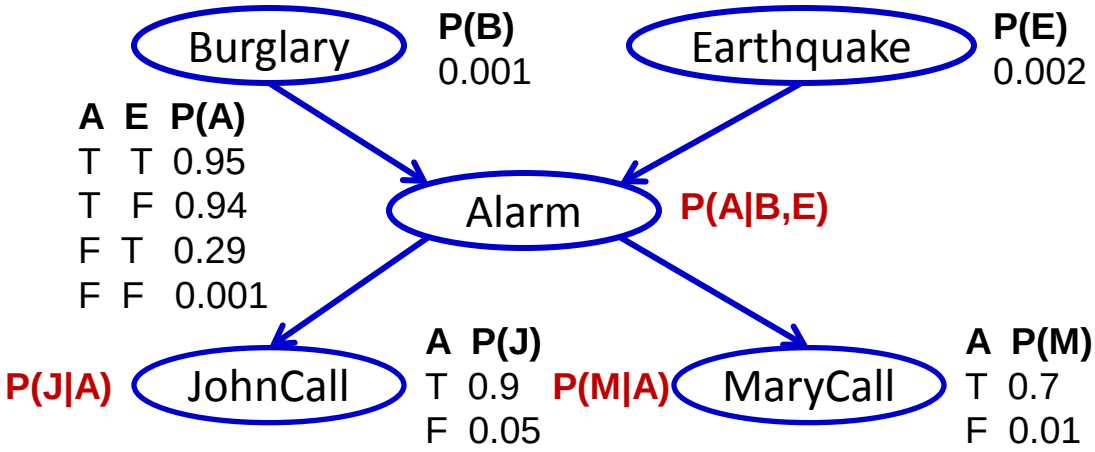
Агенту следует перемещаться в квадрат [2,1] или в квадрат [4,3] ***P(Pit)=0.31***
Перемещение в квадрат [3,2] более опасно ***P(Pit)=0.86.***

Байесовские сети

- Вершинами сети являются множество случайных переменных
- Вершины **X** соединяются ребрами
- Каждая вершина характеризуется условным распределением **`P(X|Parents(X))`**
- Граф не имеет циклов

- Дом имеет сигнализацию.
- Возможен взлом дома **P(B)=0.001** и землетрясение **P(E)=0.002.**
- Сигнализация в доме срабатывает при взломе и при землетрясении
- Мэри звонит хозяину дома, если слышит тревожный сигнал
- Джон тоже звонит, если слышит сигнал

Байесовские сети



Вероятность того, что прозвучал тревожный сигнал,
но не произошел ни взлом (**P= 1-0.001**), ни землетрясение (**P= 1-0.002**),
Хозяину дома позвонили и Мэри (**P=0.7**), и Джон (**P=0.9**).

$$P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) =$$
$$= P(j/a) \cdot P(m/a) \cdot P(a/\neg b \wedge \neg e) \cdot P(\neg b) \cdot P(\neg e) =$$
$$= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.00062$$

Составление байесовских сетей

Цепное правило вычисления совместной вероятности:

$$P(x_1, x_2, ..., x_n) = P(x_1) \cdot P(x_2 / x_1) \cdot P(x_3 / x_2, x_1) \cdot ... \cdot P(x_n / x_{n-1}, x_2, x_1)$$

Условная вероятность определяется только для величин $\mathtt{Parents}(x_k)$, которые непосредственно влияют на $\mathtt{P}(x_k)$:

$$P\left(x_k / x_1, x_2, ..., x_{k-1}\right) = P\left(x_k / Parents\left(x_k\right)\right)$$

Например,

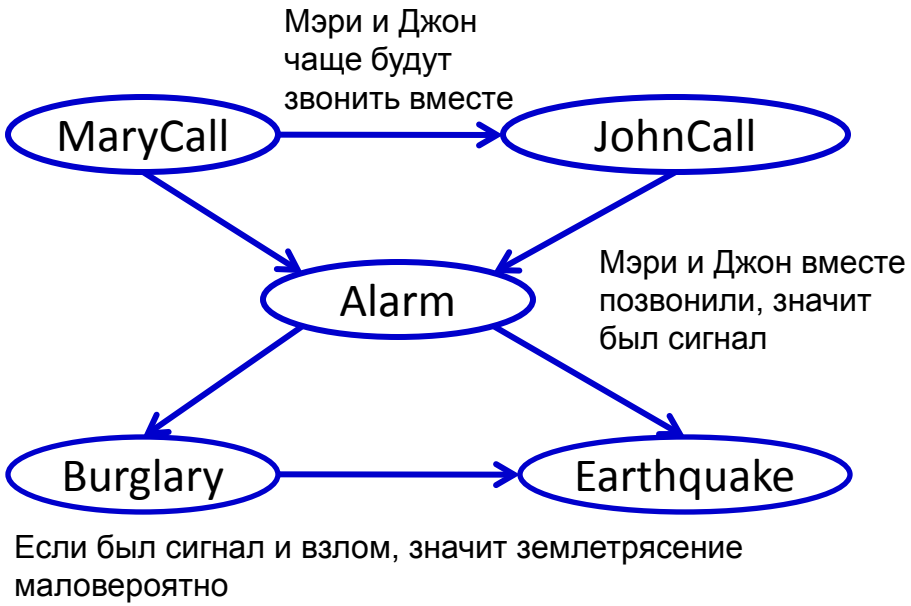
$$P\left(MaryCalls / JohnCalls, Alarm, Earthquake, Burglary\right) =$$
$$= P\left(MaryCalls / Alarm\right)$$

Компактность байесовских сетей

- Байесовские сети являются разреженными системами.

- Поэтому правильный порядок ввода вершин следующий: вначале необходимо вводить вершины **коренных причин**, затем вершины переменных, на которые они влияют.
- Порядок ввода вершин:

MaryCalls → JohnCalls → Alarm → Burglary → Earthquake



Вероятностный вывод в байесовских сетях

X – переменная запроса
E – множество измеряемых переменных
У – множество ненаблюдаемых (не измеряемых) переменных
{X,E,У} – полное множество переменных
e – наблюдаемое значение этих переменных
y –значения не наблюдаемых переменных

Запрос:

$$\mathbf{P}\left(Burglary / JohnCall = true, MaryCall = true\right) = \left\langle 0.284, 0.716 \right\rangle$$

Вероятность взлома при условии, что и Мэри и Джон позвонили

$$P\left(B / j, m\right) = \frac{P\left(B, j, m\right)}{P\left(j, m\right)} = \alpha \cdot P\left(B, j, m\right) = \alpha \cdot \sum_{e} \sum_{a} P\left(B, j, m, e, a\right)$$

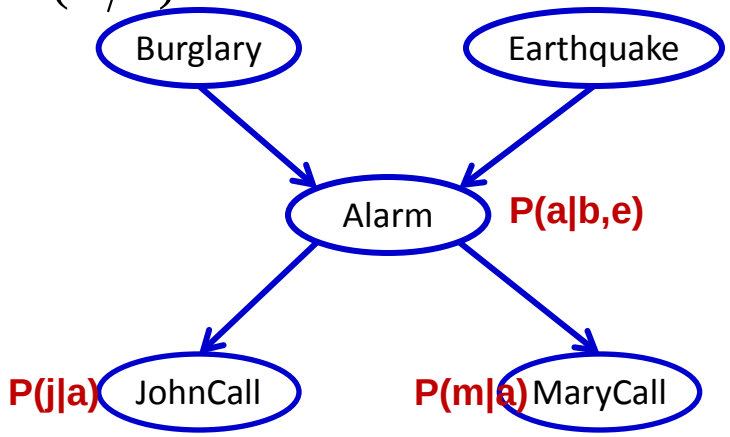
Вероятностный вывод в байесовских сетях

Вероятность взлома при условии, что и Мэри и Джон позвонили

$$P(B/j,m) = \alpha \cdot \sum_e \sum_a P(B, j, m, e, a)$$

$$P(B/j,m) = \alpha \cdot \sum_e \sum_a P(b) \cdot P(e) \cdot P(a/b,e) \cdot P(j/a) \cdot P(m/a)$$

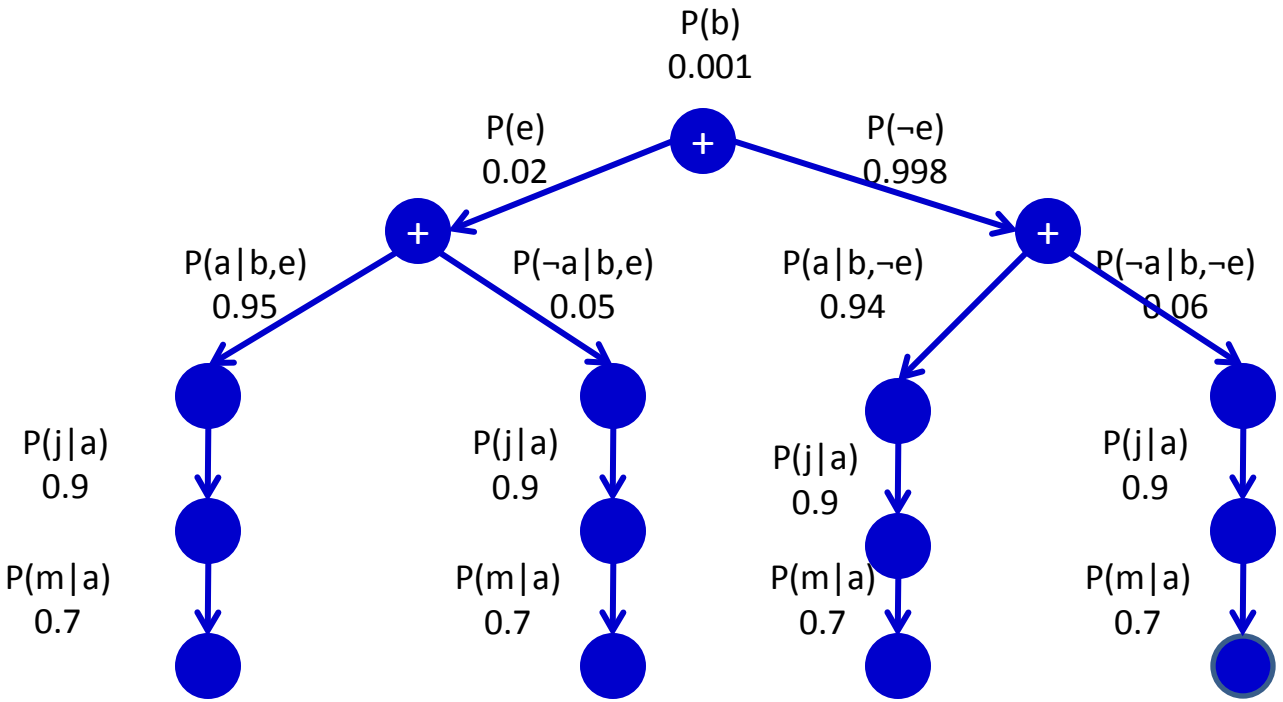
Байесовская сеть позволяет уменьшить размеры вероятностного представления запроса по сравнению с полным распределением



$$P(B/j,m) = \alpha \cdot P(b) \sum_e P(e) \cdot \sum_a P(a/b,e) \cdot P(j/a) \cdot P(m/a)$$

Вычисление вероятностей

$$P(B/j,m) = \alpha \cdot P(b) \sum_e P(e) \cdot \sum_a P(a/b,e) \cdot P(j/a) \cdot P(m/a)$$



Эффективное представление условных распределений

- Неопределенные отношения характеризуются зашумленными логическими операциями.

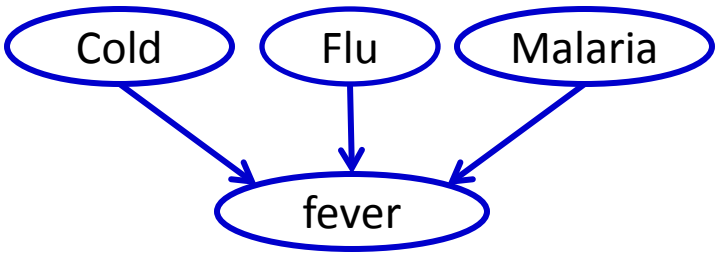
Зашумленное OR

- Жар возникает тогда и только тогда, когда имеет место простуда, или грипп, или малярия

$$Fever \Leftrightarrow (Cold \vee Flu \vee Malaria)$$

- Вероятности блокировки вершины `fever P(¬fever)`:

$$P(\neg fever / cold, \neg flu, \neg malaria) = 0.6$$
$$P(\neg fever / \neg cold, flu, \neg malaria) = 0.2$$
$$P(\neg fever / \neg cold, \neg flu, malaria) = 0.1$$

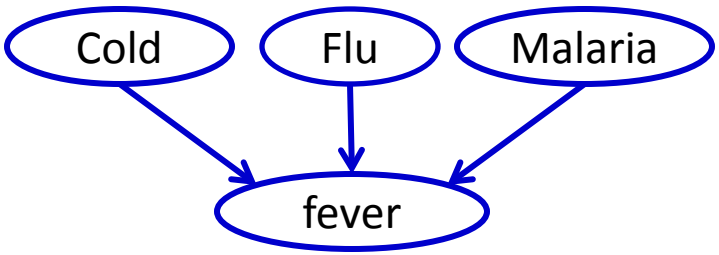


Эффективное представление условных распределений

- Таблица условных вероятностей:

Cold	Flu	Malaria	P(fever)	P(¬fever)
F	F	F	0.0	1.0
F	**F**	**T**	**0.9**	**0.1**
F	**T**	**F**	**0.8**	**0.2**
F	T	T	0.98	0.2×0.1=0.002
T	**F**	**F**	**0.4**	**0.6**
T	F	T	0.94	0.1×0.6=0.06
T	T	F	0.88	0.6×0.2=0.12
T	T	T	0.988	0.6×0.2×0.1=0.012

$$P(\neg fever / cold, \neg flu, \neg malaria) = 0.6$$
$$P(\neg fever / \neg cold, flu, \neg malaria) = 0.2$$
$$P(\neg fever / \neg cold, \neg flu, malaria) = 0.1$$



Байесовские сети
Выводы

- Байесовская сеть – это **ориентированный граф**, вершины которого соответствуют случайным переменным
- С каждой вершиной связано **распределение условных вероятностей** для этой вершины.
- Байесовские сети лежат в основе удобного способа представления **отношений условной независимости** в рассматриваемой проблемной области.
- Любая байесовская сеть задает **совместное распределение**
- Каждый элемент совместного распределения определяется как **произведение** соответствующих элементов в локальных условных распределениях.
- Байесовская сеть позволяет **экспоненциально уменьшить** размеры вероятностного представления по сравнению с полным распределением

Теория незнания

- Показатель уверенности – доверительная функция **Bel(X)**

- Подбрасывание монеты.
- Если неизвестна подлинность монеты, то:

$$Bel(\neg Heads) = 0$$
$$Bel(Heads) = 0$$

- Подбрасывание монеты.
- Пусть подлинность монеты дана с уверенностью 0.9:

$$Bel(\neg Heads) = 0.9 \times 0.5 = 0.45$$
$$Bel(Heads) = 0.9 \times 0.5 = 0.45$$

Нечеткие множества и нечеткая логика

- Нечеткая логика дает представление о неосведомленности.
- **Теория нечетких множеств** определяет расплывчатое описание объекта. Объект принадлежит к множеству. Множество задано нечетко.
- «Человек, имеющий рос т 180 см - стройный» Tall(fellow) – это число [0…1].

$$Tall(\ fellow) = 0.6$$

- **Нечеткая логика** – это метод формирования рассуждений с помощью логических выражений, описывающих принадлежность к нечетким множествам

$$T(A \wedge B) = \min(T(A), T(B))$$
$$T(A \vee B) = \max(T(A), T(B))$$
$$T(\neg A) = 1 - T(A)$$

Нечеткие множества и нечеткая логика

- Нечеткая логика – система с **истинностной функциональностью**.
- Являются обоснованными утверждения:

$$T\big(Tall(\ fellow)\big) = 0.6$$
$$T\big(Heavy(\ fellow)\big) = 0.4$$

- Представляются необоснованными утверждения:

$$T\big(Tall(\ fellow)\big) \wedge T\big(Heavy(\ fellow)\big) = 0.4$$
$$T\big(Tall(\ fellow)\big) \wedge \neg T\big(Tall(\ fellow)\big) = 0.4$$

- **Нечеткое управление** успешно потому, что использует небольшие базы правил.
- Управляемая система характеризуется **нечеткими операторами**.

Нечеткая логика и теория вероятностей

- Утверждения в нечеткой логике

$$T\big(Tall(\,fellow)\big)=0.6$$
$$T\big(Heavy(\,fellow)\big)=0.4$$

- Трактуются в терминах теории вероятностей:
  - Имеется наблюдатель, который сообщает что человек стройный или нет.
  - Условное распределение сообщения наблюдателя о человеке с данным весом и ростом.

$$P\big(Tall_{Observer}\,/\,Height,Weight\big)$$

- Системы с **истинностной функциональностью** не очень хорошо подходят для проведения рассуждений

Сети Маркова

Лекция 11

Состояния и наблюдения

- Состояние мира изменяется во времени.
- Состояние – это мгновенный снимок динамической среды
- Каждый временной срез содержит
- $\mathbf{E_t}$ - множество наблюдаемых параметров $\mathbf{E_t=e_t}$
- $\mathbf{X_t}$ – множество не наблюдаемых параметров

- Охранник в подземной комнате хочет узнать, идет ли дождь по наличию зонтика у директора.

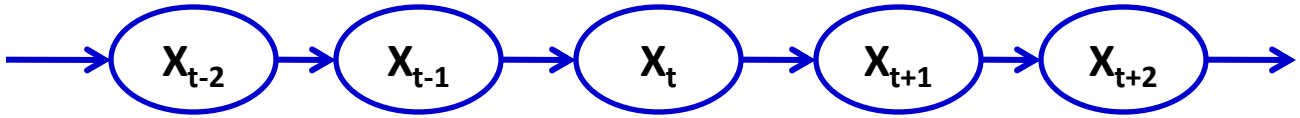
$$\mathbf{e_t=umbrella(t);\quad x_t=rain(t)}$$

- **Предположение о марковости** – текущее состояние зависит лишь от конечной истории предыдущих состояний.

Марковское предположение

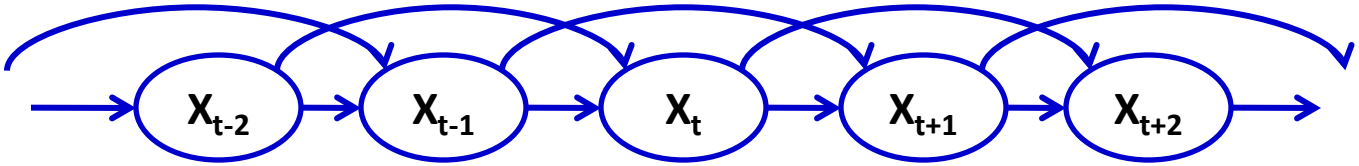
- Марковский процесс (Марковская сеть) первого порядка

$$P\left(\mathbf{X}_t / \mathbf{X}_0, \mathbf{X}_1, ..., \mathbf{X}_{t-1}\right) = P\left(\mathbf{X}_t / \mathbf{X}_{t-1}\right)$$



- Марковский процесс второго порядка

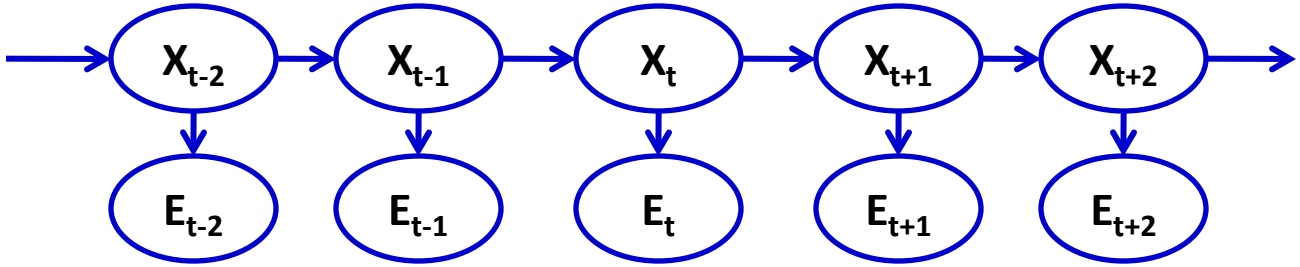
$$P\left(\mathbf{X}_t / \mathbf{X}_0, \mathbf{X}_1, ..., \mathbf{X}_{t-1}\right) = P\left(\mathbf{X}_t / \mathbf{X}_{t-1}.\mathbf{X}_{t-2}\right)$$



Марковское предположение

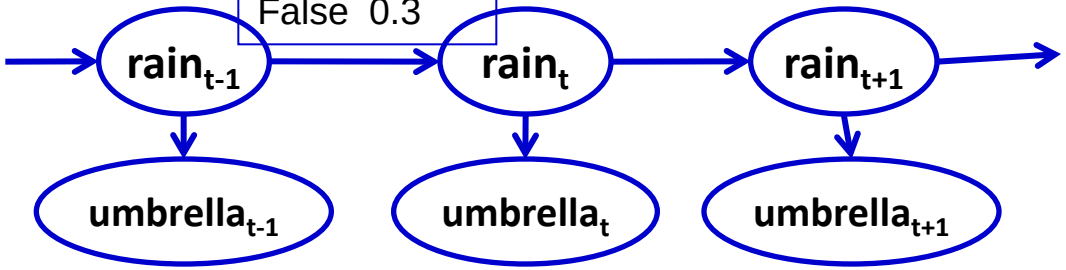
- Модель восприятия среды:

$$P(\mathbf{E}_t / \mathbf{X}_0, \mathbf{X}_1, ..., \mathbf{X}_t, \mathbf{E}_0, \mathbf{E}_1, ..., \mathbf{E}_t) = P(\mathbf{E}_t / \mathbf{X}_t)$$



- Байесовская сеть для мира задачи с зонтиком

rain	P(rain)
True	0.7
False	0.3



rain	P(umbrella)
True	0.9
False	0.2

Марковское предположение

- Для получения полного совместного распределения вероятностей модели необходимо знать:
  - Распределение априорных вероятностей в момент времени t=0 **$P(x_0)$**
  - Условное распределение **$P(E_t|X_t)$** – модель наблюдения
  - Условное распределение **$P(X_t|X_{t-1})$** – модель перехода

$$P(\mathbf{X_0}, \mathbf{X_1}, ..., \mathbf{X_t}, \mathbf{E_1}, ..., \mathbf{E_t}) = P(\mathbf{X_0}) \cdot \prod_{i=1}^{t} P(\mathbf{X_t}/\mathbf{X_{t-1}}) \cdot P(\mathbf{E_t}/\mathbf{X_t})$$

- Иногда марковская модель вызывает случайные блуждания.
- Методы корректировки модели :

- Повышение порядка модели Марковского процесса: **rain(t-2), rain(t-1), rain(t)**
- Расширения множества переменных состояния**: season(t), temperature(t), pressure(t), humidity (t).**

- Необходимо знать физику моделируемого процесса

Вероятностный вывод

- **Фильтрация или текущий контроль.** Вычисляется доверительное состояние- распределение апостериорных вероятностей, относящихся к данному моменту времени.

$$P(X_t|e_1,e_2,\ldots,e_t)$$

- **Предсказание** состоит в вычислении распределения апостериорных вероятностей значений переменных в будущем состоянии.

$$P(X_{t+k}|e_1,e_2,\ldots,e_t)$$

- **Сглаживание** или **ретроспективный анализ** состоит в вычислении апостериорных вероятностей значений переменных, относящихся к прошлому состоянию

$$P(X_k|e_1,e_2,\ldots,e_t) \text{ где } 0\leq k\leq t$$

Фильтрация и предсказание

- Есть результаты фильтрации до момента t, тогда результат в момент t+1:

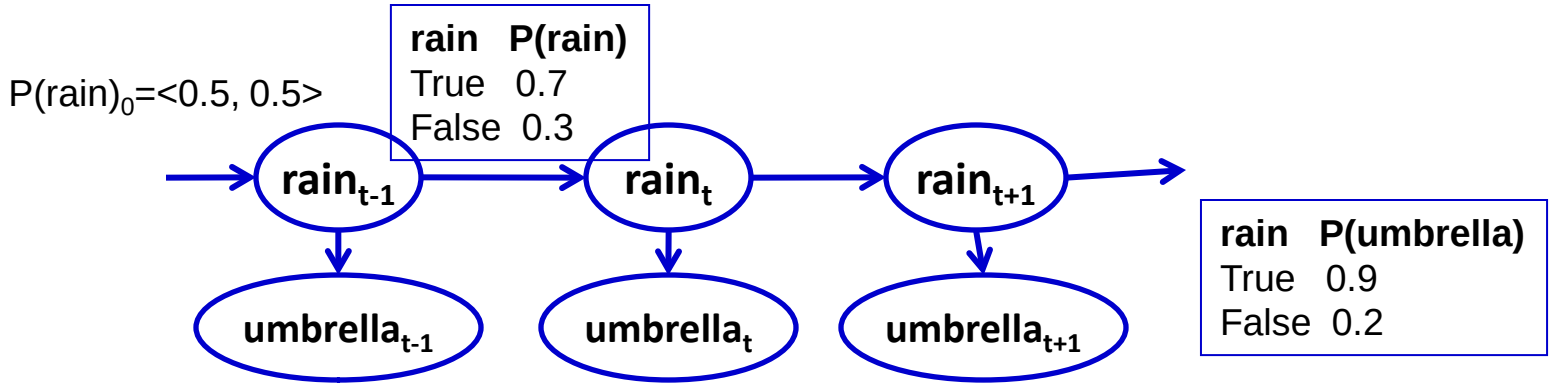
$$\mathbf{P}\left(\mathbf{X}_{t+1}/\mathbf{e}_1,\mathbf{e}_2,\ldots,\mathbf{e}_{t+1}\right)=f\left(\mathbf{e}_{t+1},P\left(\mathbf{X}_t/\mathbf{e}_1,\mathbf{e}_2,\ldots,\mathbf{e}_{t+1}\right)\right)$$

С учетом марковской модели получаем одношаговое предсказание

$$\mathbf{P}\left(\mathbf{X}_{t+1}/\mathbf{e}_1,\mathbf{e}_2,\ldots,\mathbf{e}_{t+1}\right)=\alpha\cdot\mathbf{P}\left(\mathbf{e}_{t+1}/\mathbf{X}_{t+1}\right)\cdot\sum_{\mathbf{x}_t}\mathbf{P}\left(\mathbf{X}_{t+1}/\mathbf{x}_t\right)P\left(\mathbf{x}_t/\mathbf{e}_1,\mathbf{e}_2,\ldots,\mathbf{e}_t\right)$$

Фильтрация использует одношаговое предсказание

Пример с зонтиком



Результаты предсказания дождя на следующий день:

$$P(Rain_1) = \sum_{rain} P(Rain_1 / rain_0) \cdot P(rain_0) = \langle 0.7, 0.3 \rangle \times 0.5 + \langle 0.3, 0.7 \rangle \times 0.5 = \langle 0.5, 0.5 \rangle$$

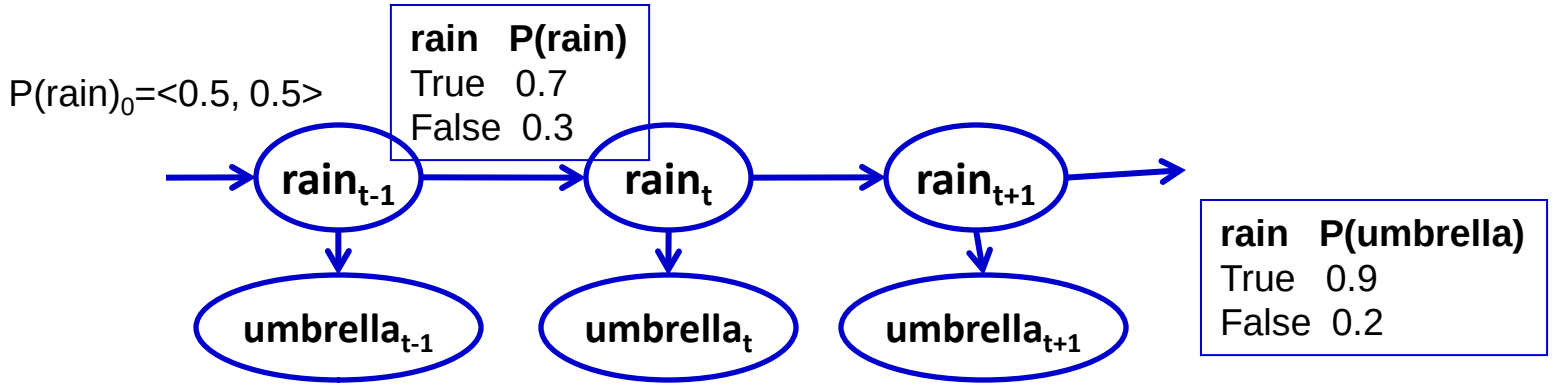
В первый день имеем наблюдение «директор пришел с зонтиком»

$umbrella_1$ =true.

Вероятность дождя с учетом наблюдения стала выше.

$$P(Rain_1 / umbrella_1) = \alpha \cdot P(umbrella_1 / Rain_1) \cdot P(Rain_1) =$$
$$= \alpha \cdot \langle 0.9, 0.2 \rangle \cdot \langle 0.5, 0.5 \rangle = \alpha \cdot \langle 0.45, 0.1 \rangle = \langle 0.818, 0.182 \rangle$$

Пример с зонтиком



Вероятность дождя **во второй день** без наблюдения за зонтом директора:

$$P(Rain_2/umbrella_1) = \sum_{rain_1} P(Rain_2/rain_1)\cdot P(rain_1/umbrella_1) =$$

$$= \langle 0.7,0.3\rangle \times 0.818 + \langle 0.3,0.7\rangle \times 0.182 = \langle 0.627,0.373\rangle$$

Во второй день опять имеем наблюдение «директор пришел с зонтиком»

$umbrella_2$ =true.

Вероятность дождя с учетом наблюдения стала выше.

$$P(Rain_2/umbrella_1, umbrella_2) = \alpha \cdot P(umbrella_2/Rain_2)\cdot P(Rain_2/umbrella_1) =$$

$$= \alpha \cdot \langle 0.9,0.2\rangle \cdot \langle 0.627,0.373\rangle = \alpha \cdot \langle 0.565,0.075\rangle = \langle 0.883,0.117\rangle$$

Предсказание

- Фильтрация выполняет одношаговое предсказание.
- Предсказание выполняется на число шагов k , большее 1. т.е. k>1

$$\mathbf{P}\left(\mathbf{X}_{t+k+1}/\mathbf{e_1},\mathbf{e_2},...,\mathbf{e_t}\right)=\sum_{\mathbf{x_{t+k}}}\mathbf{P}\left(\mathbf{X}_{t+k+1}/\mathbf{x}_{t+k}\right)P\left(\mathbf{x}_{t+k}/\mathbf{e_1},\mathbf{e_2},...,\mathbf{e_t}\right)$$

- Для предсказания на k шагов модель **восприятие** не может использоваться
- Прогнозируемое значение при увеличении числа шагов k сходится к **фиксированной точке** <0.5, 0.5> - стационарное распределение для марковского процесса.
- **Продолжительность смешивания** – это время, необходимое для достижения точки <0.5, 0.5> .
- Чем более неопределенная модель, тем **короче время** смешивания.

Сглаживание

- Сглаживание – процесс вычисления вероятностей событий в прошлом, при наличии наблюдений вплоть до нынешнего состояния

$$P(\mathbf{X}_k / \mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_t) \qquad 1 \le k \le t$$

- Формула Байеса

$$P(X/E) = \frac{P(E/X) \cdot P(X)}{P(E)}$$

- Применяем формулу Байеса для сглаживания

$$P(\mathbf{X}_k / \mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_t) = P(\mathbf{X}_k / \mathbf{e}_1 : \mathbf{e}_k, \mathbf{e}_k : \mathbf{e}_t) =$$
$$= \frac{P(\mathbf{e}_k : \mathbf{e}_t / \mathbf{X}_k, \mathbf{e}_1 : \mathbf{e}_k) \cdot P(\mathbf{X}_k / \mathbf{e}_1 : \mathbf{e}_k)}{P(\mathbf{e}_k : \mathbf{e}_t)} =$$
$$= \alpha \cdot P(\mathbf{e}_k : \mathbf{e}_t / \mathbf{X}_k, \mathbf{e}_1 : \mathbf{e}_k) \cdot P(\mathbf{X}_k / \mathbf{e}_1 : \mathbf{e}_k)$$

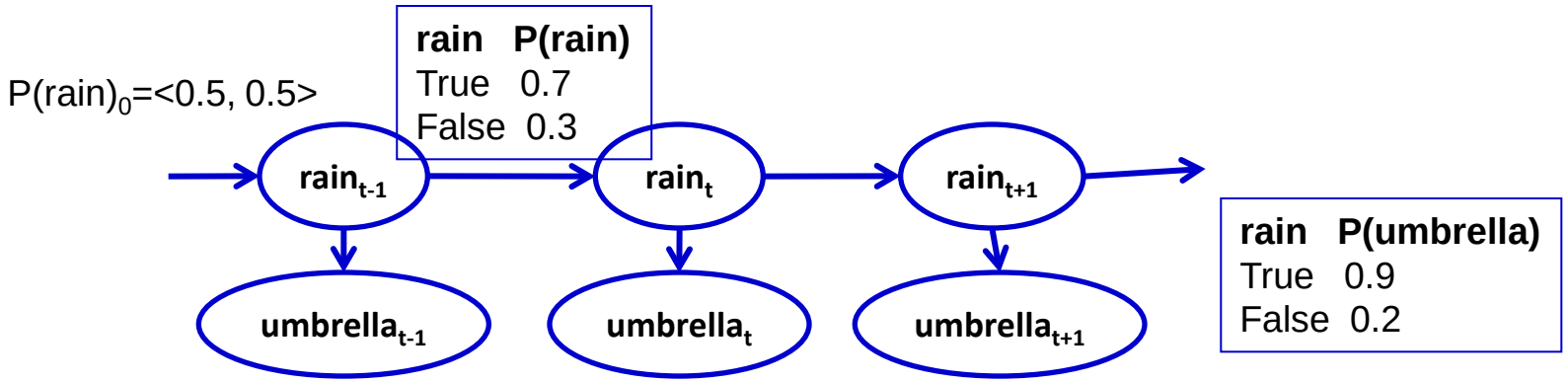
Сглаживание

- Сглаживание – процесс вычисления вероятностей событий в прошлом, при наличии наблюдений вплоть до нынешнего состояния

$$P(\mathbf{X}_k / \mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_t}) =$$
$$= \alpha \cdot P(\mathbf{X_k} / \mathbf{e_1 : e_k}) \cdot P(\mathbf{e_k : e_t} / \mathbf{X_k}) =$$
$$\alpha \cdot forward(1:k) \cdot backward(k+1:t)$$

- **Прямое сообщение forward(1:k)** может быть выполнено путем фильтрации в прямом направлении от 1 до k.
- **Обратное сообщение backward(k+1:t)** может быть выполнено путем рекурсии в обратном направлении от e до k+1.

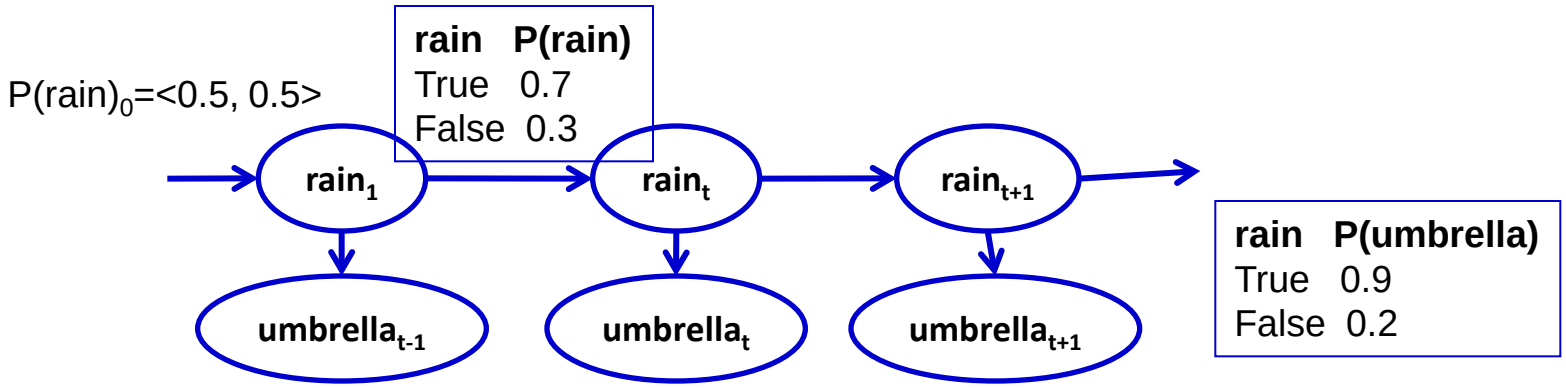
Пример с зонтиком



Вычислим **сглаженную вероятность** дождя в момент времени t=1 по данным наблюдения за два дня.
Была вычислена вероятность дождя во второй день:

$$P(Rain_2/umbrella_1, umbrella_2) = \alpha \cdot P(umbrella_2/Rain_2) \cdot P(Rain_2/umbrella_1) =$$
$$= \alpha \cdot \langle 0.9, 0.2 \rangle \cdot \langle 0.627, 0.373 \rangle = \alpha \cdot \langle 0.565, 0.075 \rangle = \langle 0.883, 0.117 \rangle$$

Пример с зонтиком

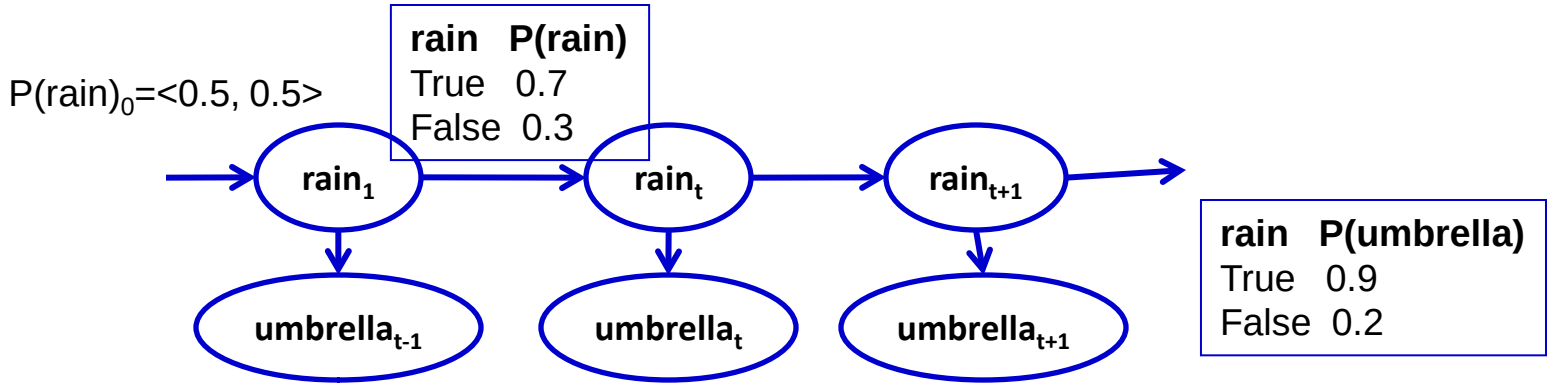


Инициализируем алгоритм <1.0,1.0>
Выполним обратную рекурсию:

$$P(umbrella_2/Rain_1)=\sum_{rain_2}P(umbrella_2/rain_2)\cdot P(\ /rain_2)\cdot P(rain_2/Rain_1)=$$

$$=\left(0.9\times1\times\langle0.7,0.3\rangle\right)+\left(0.2\times1\times\langle0.3,0.7\rangle\right)=\langle0.69,0.41\rangle$$

Пример с зонтиком



При **прямом** ходе алгоритма получили:

$$P(Rain_1/umbrella_1) = \alpha \cdot P(umbrella_1/Rain_1) \cdot P(Rain_1) =$$
$$= \alpha \cdot \langle 0.9,0.2 \rangle \cdot \langle 0.5,0.5 \rangle = \alpha \cdot \langle 0.45,0.1 \rangle = \langle 0.818,0.182 \rangle$$

При **обратном** ходе алгоритма получили:

$$P(umbrella_2/Rain_1) = \langle 0.69,0.41 \rangle$$

Сглаженная оценка вероятности дождя в момент времени t=1:

$$P(Rain_1/umbrella_1, umbrella_2) =$$
$$= \alpha \cdot \langle 0.818,0.182 \rangle \times \langle 0.69,0.41 \rangle = \langle 0.883,0.117 \rangle$$

Скрытые модели Маркова

- СММ - модель, в которой наблюдения являются вероятностной функцией состояния.
- СММ является **дважды стохастической**.
- СММ включает пару случайных процессов:
  - ненаблюдаемый **скрытый случайный процесс,** который является основным,
  - **открытый случайный процесс**, который порождает последовательность наблюдаемых значений
  - **наблюдаемые значения** характеризуют скрытый процесс
  - о скрытом процессе можно судить с помощью наблюдений, произведенных открытым процессом.

Модель урн и шаров

Из **какой урны** вынули шар?

Модель урн и шаров

Наблюдаемые цвета шаров:

O={красный, оранжевый, желтый, зеленый, синий, фиолетовый}

$e_i(j)$ - вероятность вынуть шар цвета j из урны номер i

j=1	$e_1(1)$	$e_2(1)$	…	$e_N(1)$
j=2	$e_1(2)$	$e_2(2)$	…	$eb_N(2)$
…	…	…	…	…
j=M	$e_1(7)$	$e_2(7)$		$e_N(7)$

Элементы СММ

Скрытая модель Маркова содержит следующие элементы:

- Состояния системы. Число состояний равно *N.*

$$\mathbf{X} = \{X_1, X_2, ..., X_N\}$$

- Вероятности переходов между состояниями системы. Матрица ***T*** вероятностей переходов:

$$\mathbf{T} = \{t_{i,j}\}, \qquad 1 \le i \le N, 1 \le j \le N$$

$$t_{i,j} = P(X_t = j / X_{t-1} = i)$$

- Начальное распределение вероятностей первого состояния в начале эксперимента:

$$\boldsymbol{\pi} = \{\pi_i\}, \qquad 1 \le i \le N$$

Элементы СММ

- Число ***M*** символов наблюдения, характеризующее размер алфавита символов

$$\mathbf{V} = \{V_1, V_2, ..., V_M\}$$

для набора цветов M=7

V={красный, оранжевый, желтый, зеленый, синий, фиолетовый}

- Распределение вероятностей появления символов V_j из алфавита V в каждом состоянии X_i:

$$\mathbf{E} = \{\mathbf{e}_i(1), \mathbf{e}_i(2), ..., \mathbf{e}_i(M)\}, \qquad i = 1, .., N$$

Элементы СММ

Матрица переходов:

$$T = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1N} \\ t_{21} & t_{22} & \dots & t_{2N} \\ \dots & \dots & \dots & \dots \\ t_{N1} & t_{N2} & \dots & t_{NN} \end{bmatrix}$$

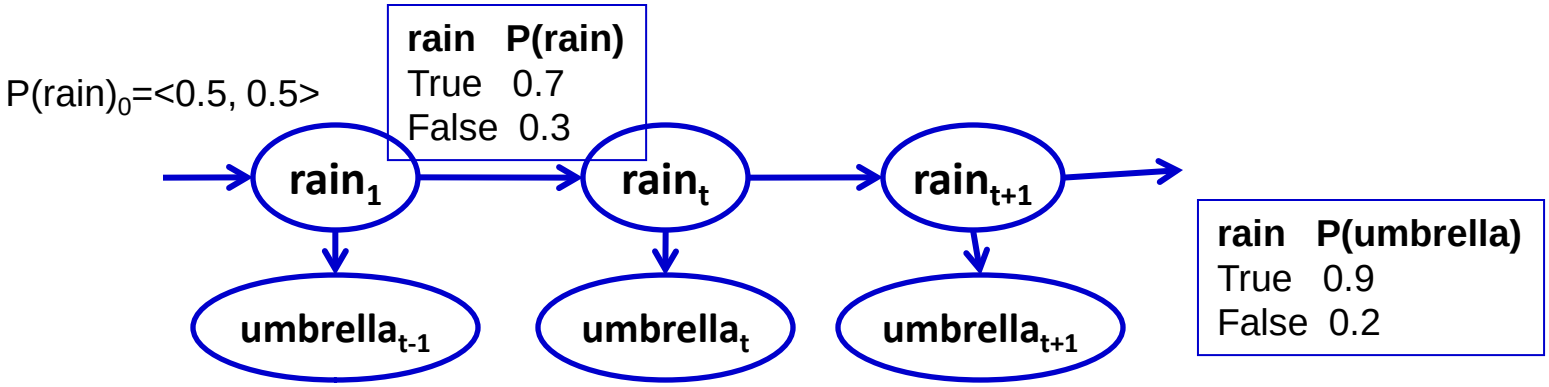
Вектор начальных значений

$$\pi = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \dots \\ \pi_N \end{bmatrix}$$

Матрица наблюдений:

$$E = \begin{bmatrix} e_{11} & e_{12} & \dots & e_{1N} \\ e_{21} & e_{22} & \dots & e_{2N} \\ \dots & \dots & \dots & \dots \\ e_{M1} & e_{M2} & \dots & e_{MN} \end{bmatrix}$$

Пример с зонтиком



- Матрица вероятностей перехода состояния погоды из X_i в X_j

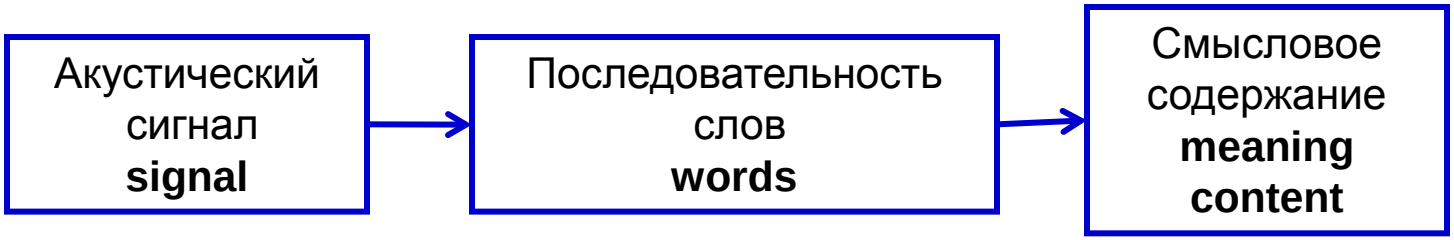
$$\mathbf{T} = \mathbf{P}\left(X_t / X_{t+1}\right) = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}$$

- Матрица вероятностей наблюдения переменных в каждом состоянии:

$$\mathbf{E} = \mathbf{P}\left(e_t / X_t = i\right) = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}$$

Распознавание речи

- Распознавание речи состоит в том, чтобы, имея акустический сигнал, выявить последовательность слов.
- Понимание речи – идентификация смысла



- **Акустическая модель** слова характеризуется разнообразием произношения данного слова:

$$P(signal/words)$$

- **Языковая модель** характеризуется вероятностями использования отдельных слов или последовательности слов:

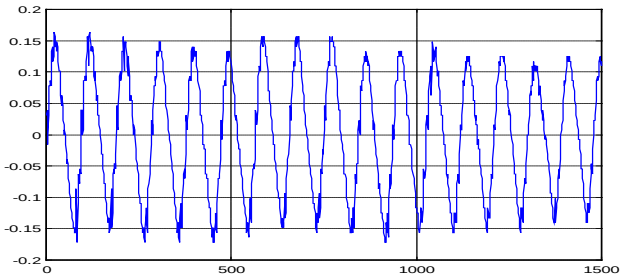
$$P(words)$$

Фонемы

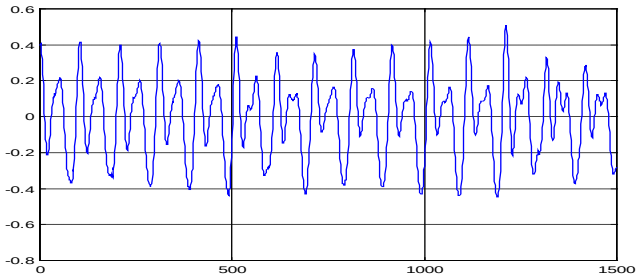
- Во всех языках используется ограниченный набор звуков (фонем)
- Фонема – это наименьший фрагмент звукового сигнала, который передает смысл. Число фонем от 40 до 50
- Акустическая модель характеризует вероятности измерения параметров **E** сигнала при условии произношения фонемы **X:**

$$P(E_t / X_t)$$

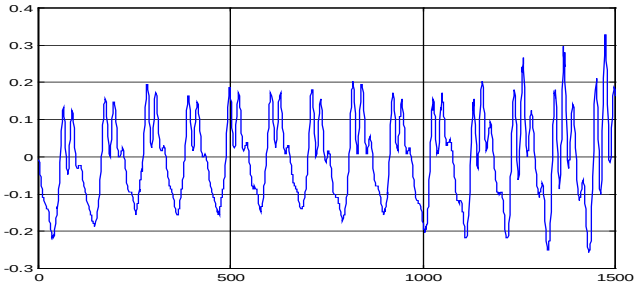
Гласные фонемы



а).Phoneme [i] f**ee**l

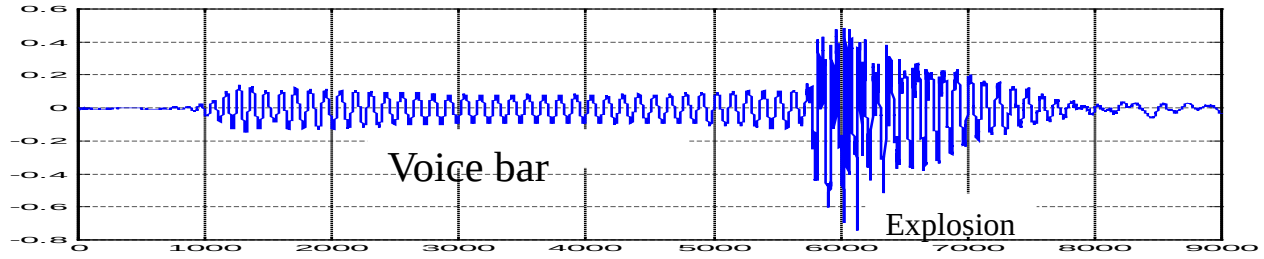


б).Phoneme [o] **o**nly

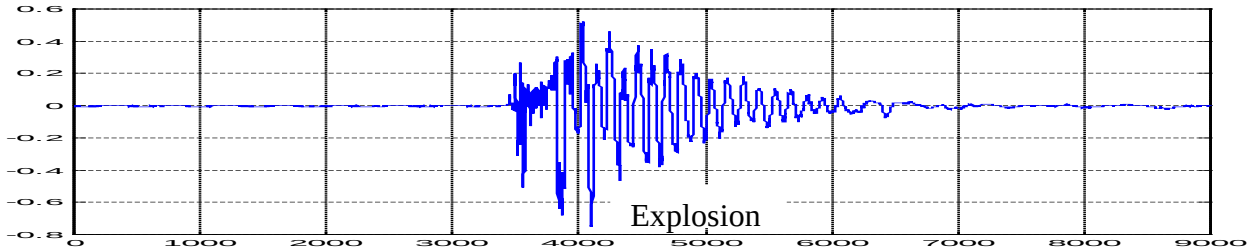


c).Phoneme [u] b**u**lldog

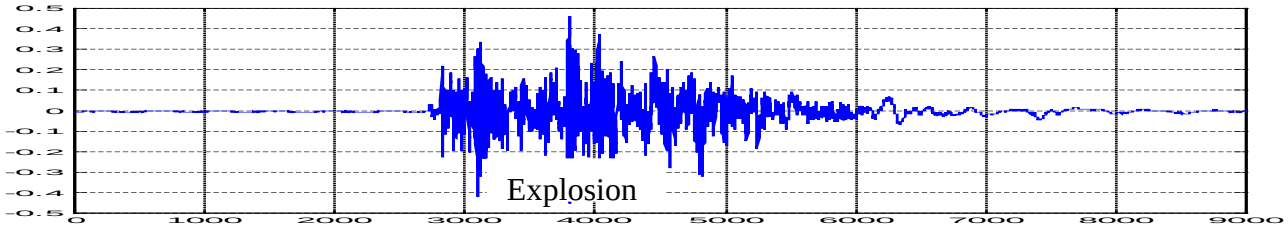
Взрывные согласные фонемы



a) Explosive phoneme [b]

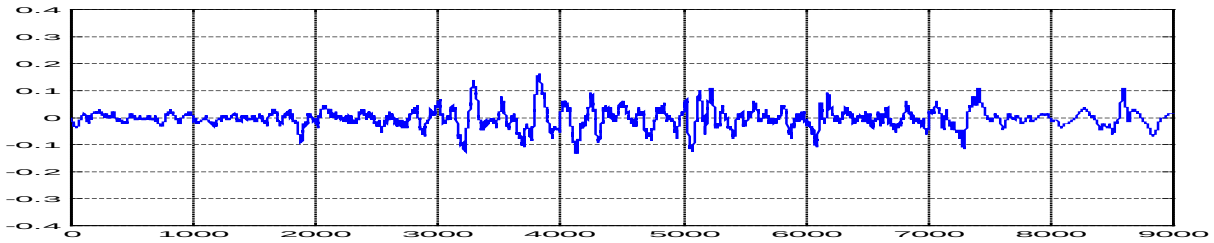


b) Consonant explosive phoneme [p]

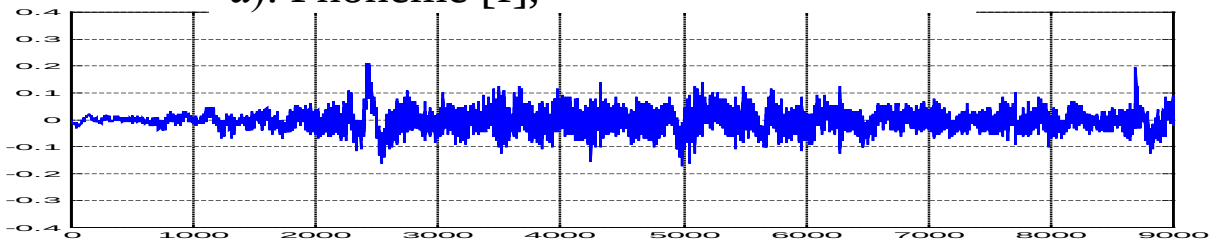


c) Consonant explosive phoneme [k]

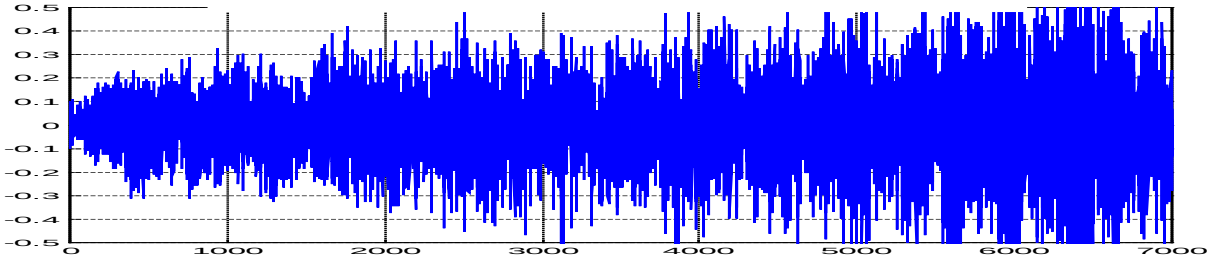
Фрикативные согласные фонемы



a). Phoneme [f],



b) Phoneme [s]



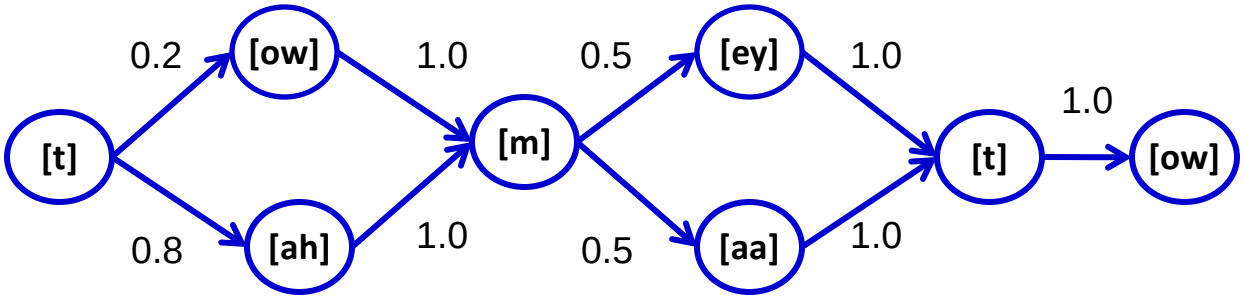
c) Phoneme [ʃ] (**sh**ift)

Слова

- **Модель произношения** задает последовательность фонем без привязки ко времени
- **Модель фонем** задает отображение фонем на фреймы сигнала

Модель произношения

- Произношение слова **«tomato» [t ow m ey t ow], [t ow m aa t ow], [t ah m ey t ow] [t ah m aa t ow]**

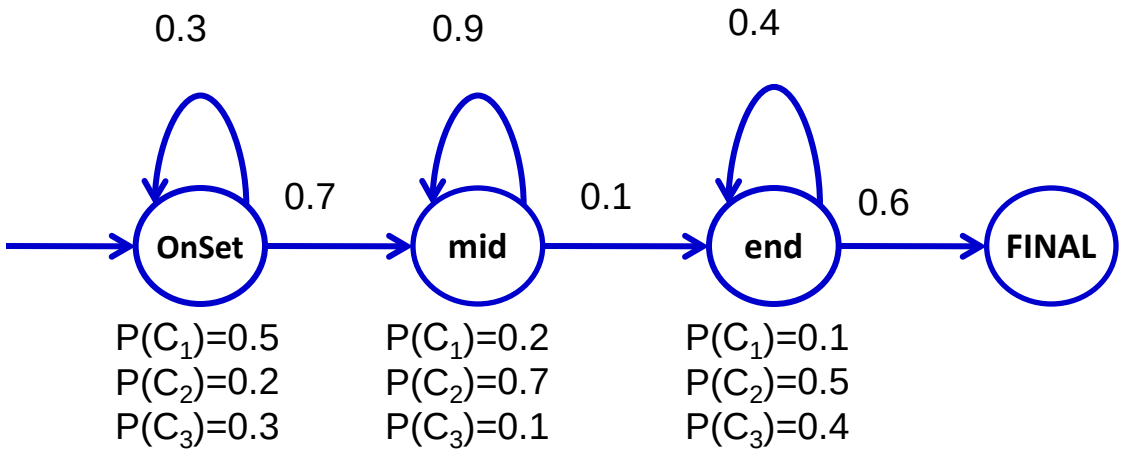


$$P([towmeytow]/'tomato') = P([towmaatow]/'tomato') = 0.2 \times 0.5 = 0.1$$

$$P([tahmeytow]/'tomato') = P([tahmaatow]/'tomato') = 0.8 \times 0.5 = 0.4$$

Модель фонем

Модель фонемы «m» с тремя состояниями: m_{onSet} , m_{mid}, m_{end}



Модель использует векторное квантование. Центр кластера C_1:

$$\mathbf{P(E_t=C_1 \mid X_t=[m]_{mid})}$$

Состояние [m_{mid}] сохраняется с вероятностью 0.9 поэтому ожидаемая продолжительность [m_{mid}] составляет10 фреймов

Для распознавания отдельных слов используется критерий:

$$\mathbf{P(word \mid e_1, e_2,\ldots,e_t)}$$

Предложения

- Распознавание непрерывной речи.
- Проблема 1: Решением является последовательность наиболее вероятных слов:

  «I have a gub» менее вероятна, чем «I have a gun»
- Проблема 2: нахождение границ слов
- **Модель языка** определяет вероятность последовательности слов.

$$P(w_1, w_2, \ldots, w_n) = P(w_1) \cdot P(w_2 / w_1) \cdot P(w_3 / w_1, w_2) \cdot \ldots \cdot P(w_n / w_1, \ldots, w_{n-1})$$

- Модель двухсловных сочетаний:

$$P(w_1, w_2, \ldots, w_n) = P(w_1) \cdot P(w_2 / w_1) \cdot P(w_3 / w_2) \cdot \ldots \cdot P(w_n / w_{n-1})$$

Основы теории полезности

Лекция 12

Ожидаемая полезность

- ожидаемая полезность:
- A – действие,
- Result(A) – результирующее состояние
- E – результаты измерения

Вероятность результата действия A при условии, что оно было выполнено и получено измерения E

$$P\left(Result_i(A)/Do(A), E\right)$$

Полезность достигнутого результата

$$U\left(Result_i(A)\right)$$

Ожидаемая полезность с учетом результатов измерения E:

$$EU(A/E) = \sum_i P\left(Result_i(A)/Do(A), E\right) \cdot U\left(Result_i(A)\right)$$

Принцип максимальной полезности

- Принцип максимальной полезности – агент выбирает действие, которое максимизирует ожидаемую полезность.
- Если агент **максимизирует функцию полезности,** правильно отражающую показатели производительности, по которым можно судить о его поведении , но этот агент достигает наибольших возможных значений показателей по всем вариантам среды, в которой находится агент.

- **Предпочтения агента**:

$A \succ B$ Вариант A предпочтительнее, чем B

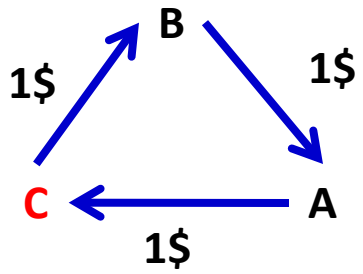
$A \approx B$ A и B одинаковы

$A \succ\approx B$ A либо предпочтительнее, чем B, либо безразличен

Ограничения на рациональные предпочтения

$$\left.\begin{array}{l} A \succ B \succ C \\ A \succ C \end{array}\right\}$$

Предпочтение является транзитивным



$A \succ B \succ C \succ A$ Не транзитивное предпочтение приводит к нерациональному поведению

$A \succ B \succ C \Rightarrow \exists p\left[p, A; (1-p), C\right] \approx B$ Существует некоторая вероятность p того, что агент будет безразличен к выбору

Принцип полезности

Принцип полезности:

$$U(A) > U(B) \Leftrightarrow A \succ B$$
$$U(A) = U(B) \Leftrightarrow A \approx B$$

Принцип максимальной ожидаемой полезности:

$$U\left(p_1, S_1, \ldots, p_n, S_n\right) = \sum_i p_i \cdot S_i$$

Полезность – это функция, отображающая состояния на действительные числа.

Полезность денег

- **Монотонное предпочтение** – агент всегда предпочитает больше денег, чем меньше.
- Гарантированный выигрыш или лотерея?

$$A \to \{p = 1, S = \$1000000; \quad B \to \begin{cases} p = 0.5, S = \$3000000 \\ p = 0.5, S = \$0 \end{cases}$$

- Ожидаемое денежное вознаграждение

$$U(Accept) = 0.5 \times \$3000000 + 0.5 \times 0.0 = \$1500000$$

$$U(Decline) = \$1000000$$

Большинство людей предпочитает вариант A

Влияние накопленной суммы денег

- Функция предпочтения агента зависят от количества **накопленных денег.**
- Гарантированный выигрыш или лотерея при имеющихся $500 000 000?

$$A \to \{p = 1, S = \$1000000; \quad B \to \begin{cases} p = 0.5, S = \$3000000 \\ p = 0.5, S = \$0 \end{cases}$$

- Ожидаемое денежное вознаграждение

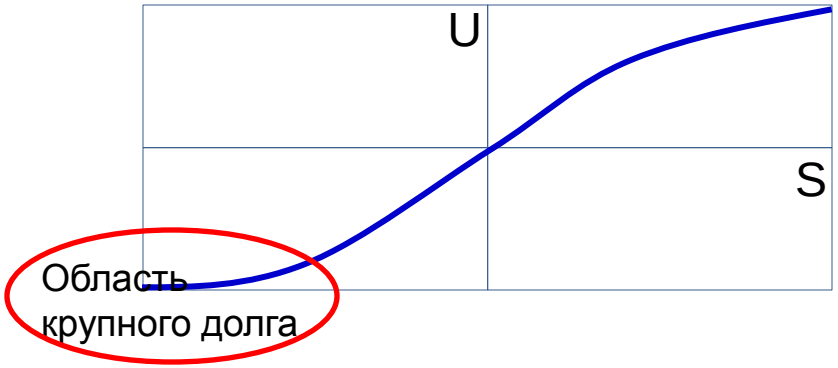
$$U(Accept) = \$500000000 + (0.5 \times \$3000000 + 0.5 \times \$0.0) = \$501500000$$

$$U(Decline) = \$500000000 + \$1000000 = \$501000000$$

Скорее всего агент безразличен $A \approx B$

Немонотонная полезность

- **Логарифмическая функция полезности** - предпочтения агента в отрицательной области. В области крупного долга.



$$A \rightarrow p = 0.5, S = \$10\,000\,000;\ p = 0.5, S = -\$20\,000\,000$$

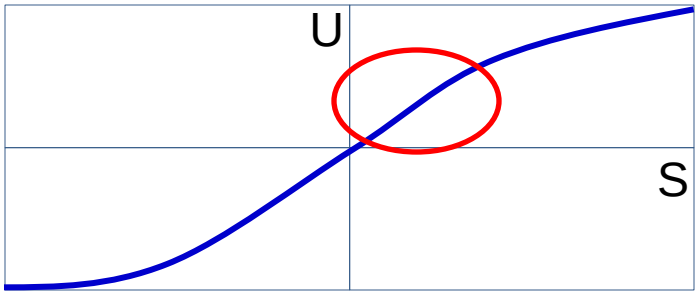
$$B \rightarrow p = 1.0, S = \$0.0$$

$$A \succ B$$

В области крупного долга – стремление к риску

Немонотонная полезность

- **Логарифмическая функция полезности** - предпочтения агента в линейной положительной области.



$A \to p = 1.0, S = \$400;$ $EU(A) = \$400;$

$B \to p = 0.5, S = \$1000;\ p = 0.5, S = \$0$ $EU(B) = \$500;$

$A \approx B$

- Эквивалент определенности лотереи $if\ (A \approx B)\ then\ EU(A) = \$400;$
- Страховая сумма $EU(B) - EU(A) = \$500 - \$400 = \$100;$

Люди предпочитают заплатить небольшую страховую сумму

Субъективные суждения, присущие человеку

Теория принятия решений является **нормативной** – как должен действовать агент.
Теория фактического принятия решений людьми является **описательной**.

Сожаление – эмоция, которая определяет выбор агента в пользу B:

$$A \rightarrow p = 0.8, S = \$4000$$
$$B \rightarrow p = 1.0, S = \$3000$$
$$B \succ A$$

При **малых вероятностях** отсутствует сожаление, агент выбирает A:

$$A \rightarrow p = 0.2, S = \$4000$$
$$B \rightarrow p = 0.25, S = \$3000$$
$$A \succ B$$

Шкалы полезности и оценка функций полезности

- **В детерминированной** среде агент руководствуется порядковой функцией полезности, обеспечивающей ранжирование состояний.
- **В стохастической** среде используется
  - Нормированное значение полезности в диапазоне [0,1]
  - Стандартная лотерея $[p,u_1;(1-p),u_2]$

Оценка

- В задачах транспорта, медицины, экологии и др. используется **цена человеческой жизни.**
- В среднем лицо может подвергать себя опасности за $20 000 000.
- Единицей измерения может быть
  - Микрошанс смерти оценивается как $20.
  - Год жизни в добром здравии

Многоатрибутные функции полезности

Выбор площадки для аэропорта

- Атрибуты функции полезности:
  - Вред окружающей среде,
  - Стоимость земельного участка,
  - Расстояние до города,
  - Уровень шума,
  - Безопасность.

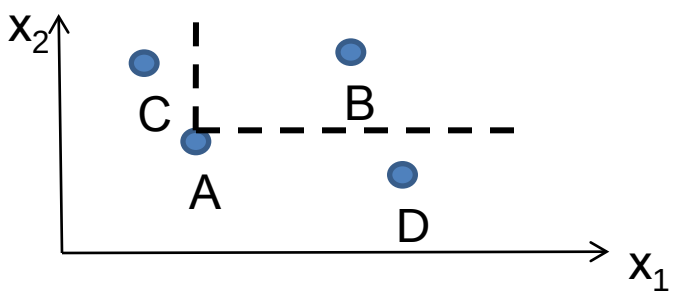
$$\mathbf{X} = X_1, X_2, \ldots, X_N$$

$$\mathbf{x} = \langle x_1, x_2, \ldots, x_N \rangle$$

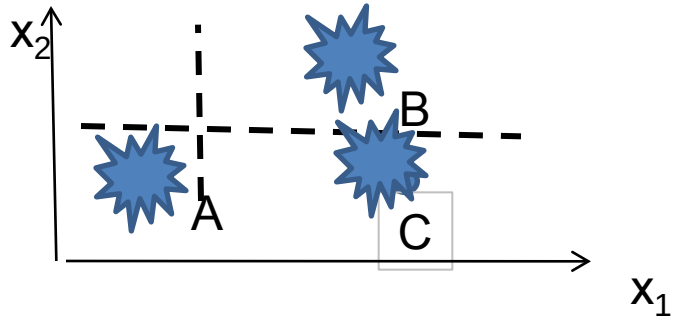
Доминирование

- **Строгое доминирование** – один вариант имеет меньшие значения всех атрибутов функции полезности

Детерминированный случай

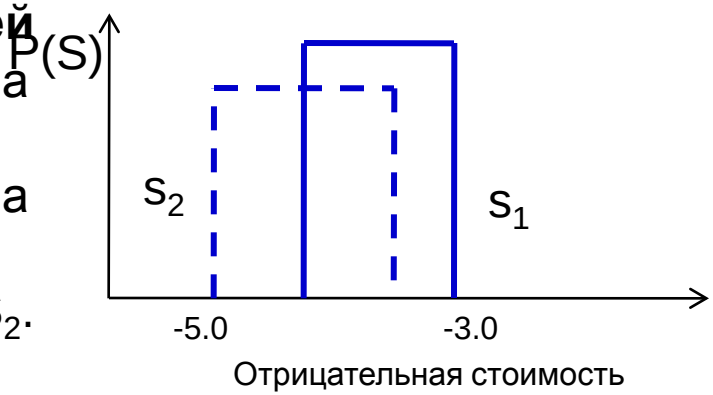


Неопределенный случай



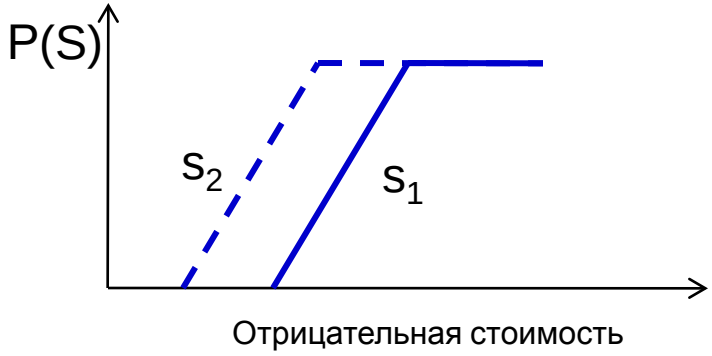
Стохастическое доминирование

- **Плотность распределения вероятностей**
  - стоимости строительства аэропорта на площадке S_1
  - стоимости строительства аэропорта на площадке S_2
- Вариант S_1 **стохастически** доминирует S_2.



- **Распределение вероятности** того, что стоимость меньше данной суммы
  - для строительства аэропорта на площадке S_1
  - для строительства аэропорта на площадке S_2

Доминирование S_1 над S_2 **более очевидно**



$$\forall x \int_{-\infty}^{x} p_1(x') \cdot dx' \le \int_{-\infty}^{x} p_2(x') \cdot dx'$$

Структура предпочтения

- Теоремы представления обосновывают функцию полезности, объединяющую множество атрибутов:

$$U(x_1, x_2, ..., x_N) = f(f_1(x_1), f_2(x_2), ... f_N(x_N))$$

- **Независимость предпочтений** при строительстве аэропорта можно предположить независимыми **<Cost, Noise, Deaths>**
- Если атрибуты являются **независимыми по предпочтению,** то поведение агента в отношении его предпочтений можно считать как максимизацию функции:

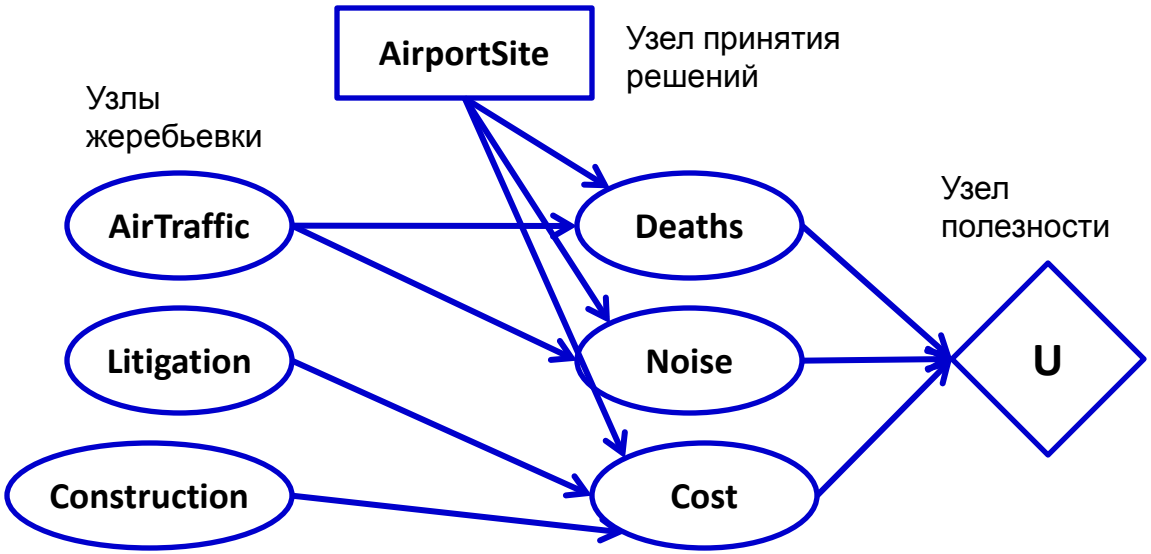
$$U(x_1, x_2, ..., x_N) = f_1(x_1) + f_2(x_2) + ... + f_N(x_N)$$

- Допустима такая функция полезности:

$$U(noise, cost, deaths) = -noise \times 10^4 - cost - deaths \times 10^{12}$$

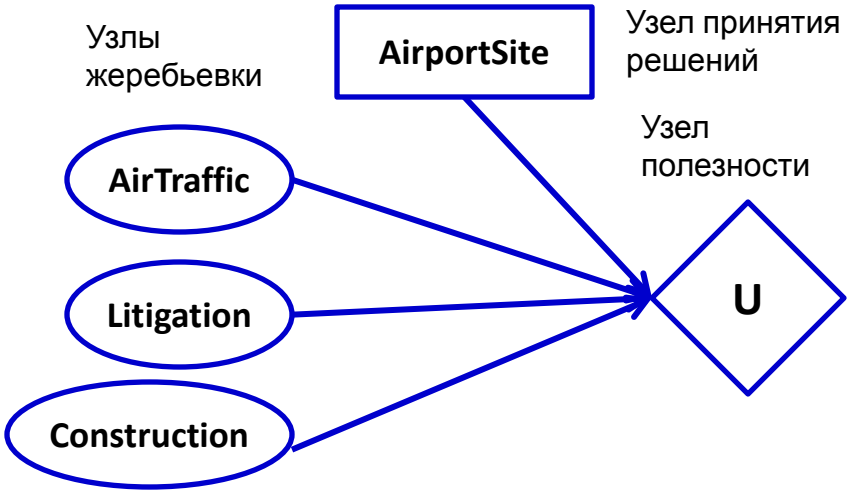
Сети принятия решения

- **Узлы жеребьевки** представляют случайные переменные, имеющие условное распределение вероятностей.
- **Узлы принятия решений** в которых агент выполняет выбор действий.
- **Узлы полезности** представляют функцию полезности.



Упрощение сети принятия решения

1. Определение значений переменных свидетельства для текущего состояния;
2. Для каждого значения узла принятия решений:
   1. Ввести значение в узел принятия решения .
   2. Вычислить апостериорные вероятности для родительских узлов узла полезности
   3. Вычислить результирующее значение полезности
3. Вычислить действие с максимальной полезностью



Теория стоимости информации

- Теория стоимости информации позволяет решить, какую информацию следует получить, какие тесты проводить.

Пример

- Покупка N равнозначных участков земли. Один из них имеет залежи нефти. Цена участка C/N.
- Сколько стоит информация о наличии залежей нефти?
- Вероятность купить участок с нефтью P=1/N без исследования.
- Затраты на покупку одного участка C/N.
- Прибыль при условии, что первый исследованный участок имеет нефть:

$$C - \frac{C}{N} = \frac{N-1}{N} \cdot C$$

- Ожидаемая прибыль после проведения исследования:

$$\frac{1}{N} \times \frac{(N-1)}{N} \cdot C + \frac{(N-1)}{N} \times \frac{C}{N(N-1)} = \frac{C}{N}$$

- Компания должна заплатить сейсмологу за информацию C/N.

Стоимость полной информации

- **Информация** – это точное свидетельство относительно случайной переменной.
- На основе полученной информации агент выполняет действие.
- Стоимость текущего наилучшего действия

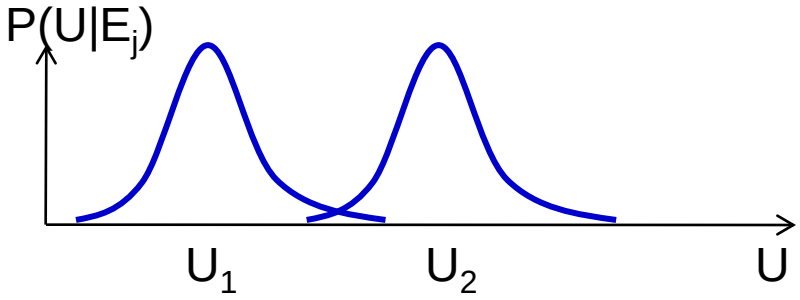
$$EU(\alpha/E) = \max_A \sum_i U(Result_i(A)) \cdot P(Result_i(A)/Do(A), E)$$

- Стоимость следующего наилучшего действия:

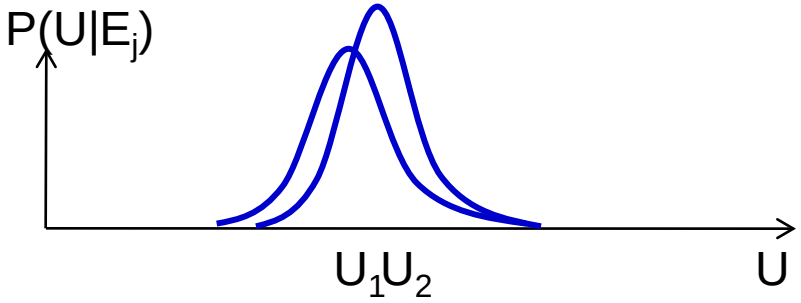
$$EU(\alpha/E, E_j) = \max_A \sum_i U(Result_i(A)) \cdot P(Result_i(A)/Do(A), E, E_j)$$

Стоимость информации определяется
- степенью, в которой она может вызвать изменение плана действий,
- насколько новый план окажется лучше

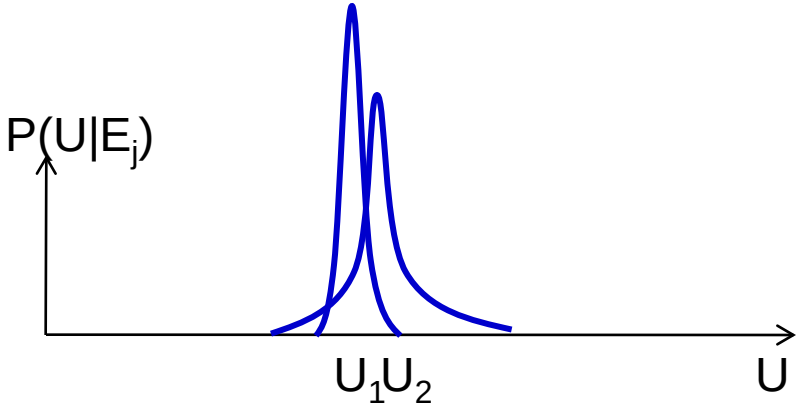
Стоимость полной информации о состоянии двух дорог



Действие А1 определенно лучше по сравнению с А2, дополнительная информация не нужна



Выбор не ясен, поэтому необходима дополнительная информация



Выбор не ясен, разница между полезностями действий невелика. Информация не имеет большой ценности

1. Не требуется оплачивать наблюдения со спутника за состоянием двух дорог.

2. Распределения широкие, поэтому нужна дополнительная информация со спутника

3. Распределения узкие, полезности близки, дополнительная информация со спутника не поможет

Свойства показателей информации

- Стоимость полной информации является неотрицательной

$$I_E\left(E_j\right) \geq 0$$

- Неаддитивность информации:

$$I_E\left(E_i, E_j\right) \neq I_E\left(E_i\right) + I_E\left(E_j\right)$$

Экспертные системы

- **Лицо, принимающее решения**, определяет отношения предпочтения между результатами
- **Лицо, анализирующее решения,** составляет список возможных действий

Этапы создания экспертной системы

- **Создание причинной модели.** Определение симптомов, нарушений, способов лечения, результатов.
- **Упрощение причинной модели** до уровня качественной модели принятия решений.
- Присвоить значения **вероятностей**.
- Присвоение значений **полезностей**.

Принятие сложных решений

- **Стохастическая, полностью наблюдаемая** среда
- **Начальное состояние** S_0
- **Марковская модель** перехода P={0.8, 0.1, 0.1}
- Вероятность достижения состояния **S′** из состояния **S** при выполнении действия **a**

$$T(s, a, s')$$

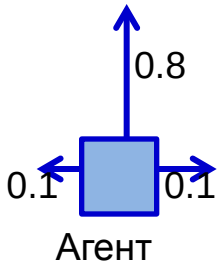
- **Функция полезности** для агента зависит от предыстории пребывания в среде.
- Агент получает вознаграждение:

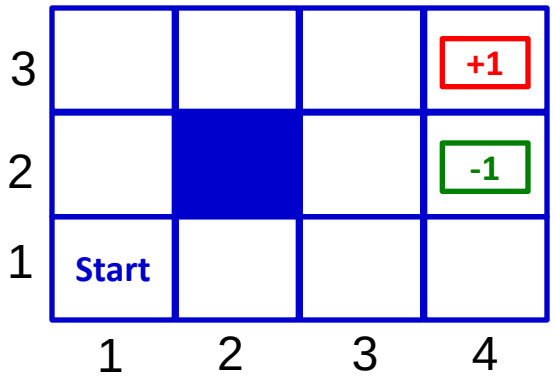
$$R(s)$$

- **Стратегия агента** – правило выполнения действия в каждом состоянии **S**

$$\pi(s)$$

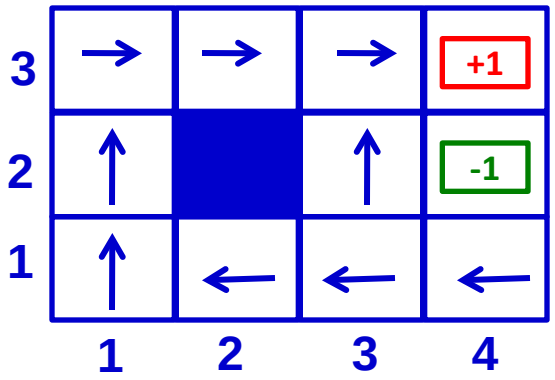
- **Оптимальной стратегией** называется стратегия, которая позволяет достичь максимальную ожидаемую полезность





Последовательное принятие решений

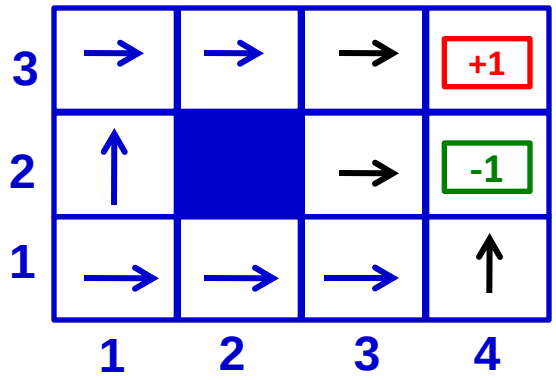
- Оптимальная стратегия при R(s)=-0.04
- Стратегия предполагает дальний обход препятствий.
- Агент стремится быстрее достичь состояния +1
- Если агенту не нравится в среде, то он стремится выйти из игры.



Оптимальные стратегии при низкой функции полезности

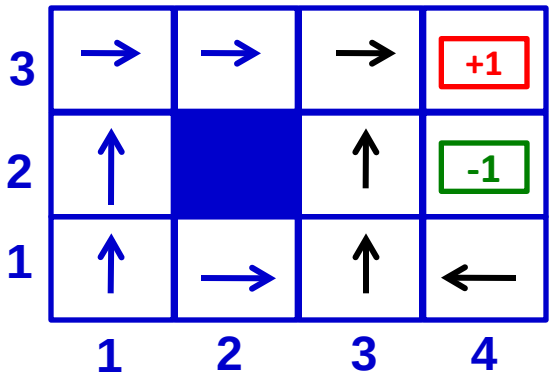
Оптимальная стратегия при **R(s)<-1.6284**

Жизнь агента просто мучительна, агент направляется к выходу, даже если стоимость **-1**



Оптимальная стратегия при **0.4278<R(s)<-1.6284**

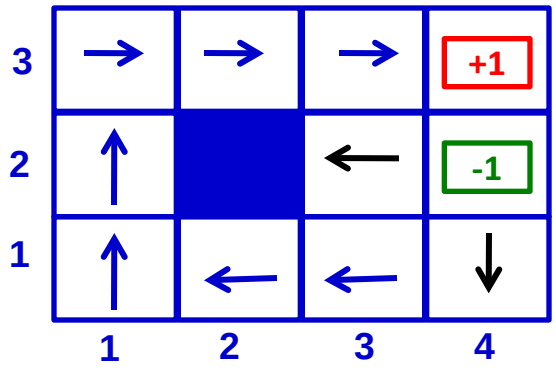
Жизнь агента дискомфортна, агент выбирает кратчайший путь к **+1**, старается избегать **-1**



Оптимальные стратегии при высокой функции полезности

Оптимальная стратегия при **-0.0221<R(s)<0**

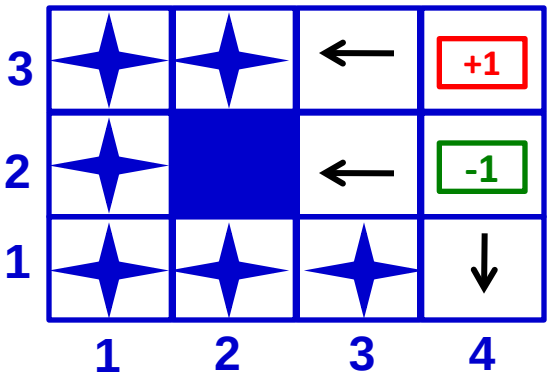
Жизнь агента неплохая, агент избегает какого-либо риска



Оптимальная стратегия при **R(s)>0**

Жизнь агента очень приятна, агент избегает любого выхода как **+1**, так и **-1.**

Любая стратегия в других квадратах является оптимальной



Последовательное принятие решений

- **Конечный горизонт** для принятия решений – окончание игры за N шагов.
  - При конечном горизонте оптимальное действие **изменяется** со временем;
  - Стратегия является **нестационарной**
- **Бесконечный горизонт** для принятия решений - окончание игры за большое число шагов.
  - Стратегия является стационарной {s0,s1,…,} и {s0',s1',…,} одинаковы
  - Конечный горизонт для принятия решений не назначается

- **Аддитивное вознаграждение**:

$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2), \ldots;$$

- **Обесцениваемое вознаграждение**:

$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma \cdot R(s_1) + \gamma^2 \cdot R(s_2), \ldots;$$

$$0 \le \gamma \le 1$$

Последовательное принятие решений

- При обесцениваемых вознаграждениях полезность бесконечной последовательности **конечна:**

$$U_n([s_0, s_1, s_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t \cdot R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t \cdot R_{\max} = R_{\max} / (1-\gamma)$$

$$\gamma < 1$$

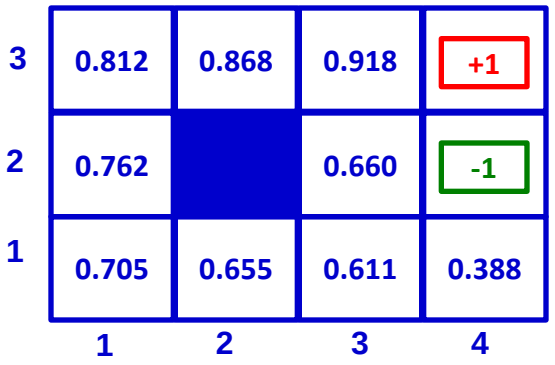
- **Правильная** стратегия

Полезности состояний

- Агент реализует стратегию **π,**
- Агент переходит из состояния s_0 в состояние s_t,
- Число шагов **t** реализации стратегии,
- Кратковременное вознаграждение **R(s),**
- Полезность реализуемой стратегии (долговременное вознаграждение) :

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t \cdot R(s_t) \middle/ \pi, s_0 = s\right]$$

Значение полезности по мере приближения состояний к выходу +1 становится выше, поскольку число шагов до цели становится меньше.



Полезности состояний

- Агент выбирает действие в соответствии с максимумом полезности.

$$\pi^*(s) = \arg\max \sum_{s'} T(s, a, s') \cdot U(s')$$

- Полезность состояния равна сумме
  - вознаграждения за **пребывания в состоянии**
  - и ожидаемой обесцениваемой полезности **следующего состояния**,
  - при условии, что агент выбирает оптимальное действие.

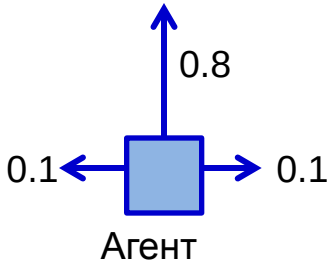
- Уравнение Беллмана

$$U(s) = R(s) + \gamma \cdot \max_a \sum_{s'} T(s, a, s') \cdot U(s')$$

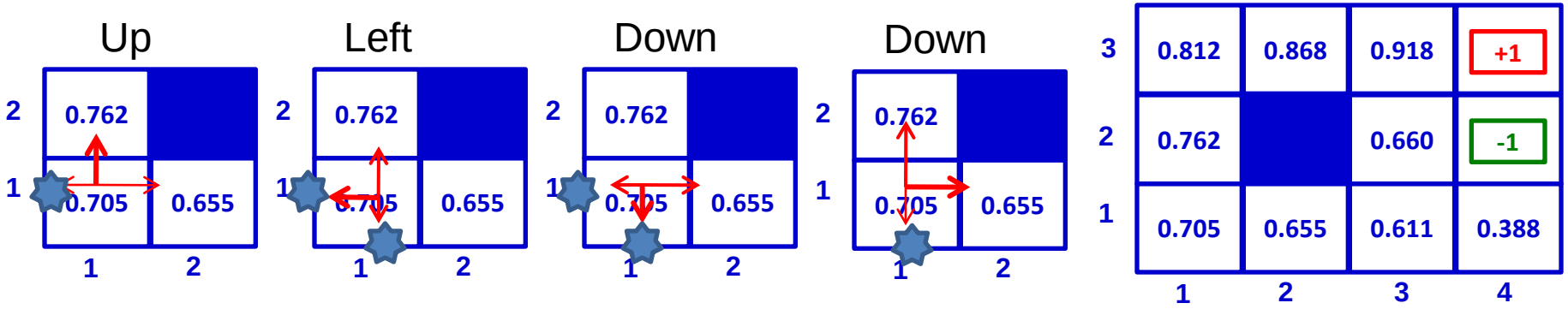
Полезности состояний

Уравнение Беллмана

$$U(s) = R(s) + \gamma \cdot \max_a \sum_{s'} T(s,a,s') \cdot U(s')$$



$$U(1,1) = -0.04 + \gamma \cdot \max \begin{pmatrix} 0.8 \cdot U(1,2) & +0.1 \cdot U(2,1) & +0.1 \cdot U(1,1) & Up \\ 0.9 \cdot U(1,1) & +0.1 \cdot U(1,2) & & Left \\ 0.9 \cdot U(1,1) & +0.1 \cdot U(2,1) & & Down \\ 0.8 \cdot U(1,2) & +0.1 \cdot U(2,1) & +0.1 \cdot U(1,1) & Right \end{pmatrix}$$



Последовательное принятие решений

Лекция 13

Последовательное принятие решений

- **Конечный горизонт** для принятия решений – окончание игры за N шагов.
  - При конечном горизонте оптимальное действие **изменяется** со временем;
  - Стратегия является **нестационарной**
- **Бесконечный горизонт** для принятия решений - окончание игры за большое число шагов.
  - Стратегия является стационарной {s0,s1,…,} и {s0',s1',…,} одинаковы
  - Конечный горизонт для принятия решений не назначается

- **Аддитивное вознаграждение**:

$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2), \ldots;$$

- **Обесцениваемое вознаграждение**:

$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma \cdot R(s_1) + \gamma^2 \cdot R(s_2), \ldots;$$

$$0 \le \gamma \le 1$$

Последовательное принятие решений

- При обесцениваемых вознаграждениях полезность бесконечной последовательности **конечна:**

$$U_n([s_0, s_1, s_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t \cdot R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t \cdot R_{\max} = R_{\max} / (1-\gamma)$$

$$\gamma < 1$$

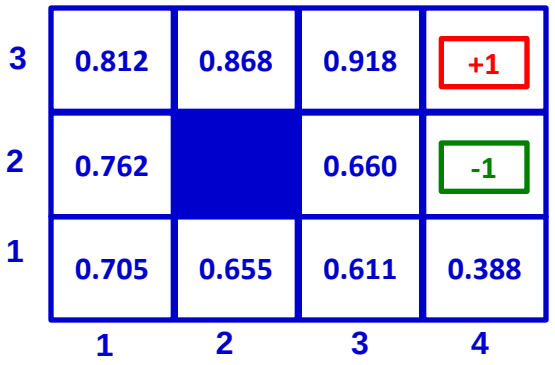
- **Правильная** стратегия

Полезности состояний

- Агент реализует стратегию **π,**
- Агент переходит из состояния s_0 в состояние s_t,
- Число шагов **t** реализации стратегии,
- Кратковременное вознаграждение **R(s),**
- Полезность реализуемой стратегии (долговременное вознаграждение) :

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t \cdot R(s_t) \middle/ \pi, s_0 = s\right]$$

Значение полезности по мере приближения состояний к выходу +1 становится выше, поскольку число шагов до цели становится меньше.



Полезности состояний

- Агент выбирает действие в соответствии с максимумом полезности.

$$\pi^*(s) = \arg\max \sum_{s'} T(s, a, s') \cdot U(s')$$

- Полезность состояния равна сумме
  - вознаграждения за **пребывания в состоянии**
  - и ожидаемой обесцениваемой полезности **следующего состояния**,
  - при условии, что агент выбирает оптимальное действие.

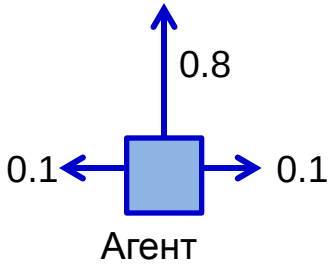
- Уравнение Беллмана

$$U(s) = R(s) + \gamma \cdot \max_a \sum_{s'} T(s, a, s') \cdot U(s')$$

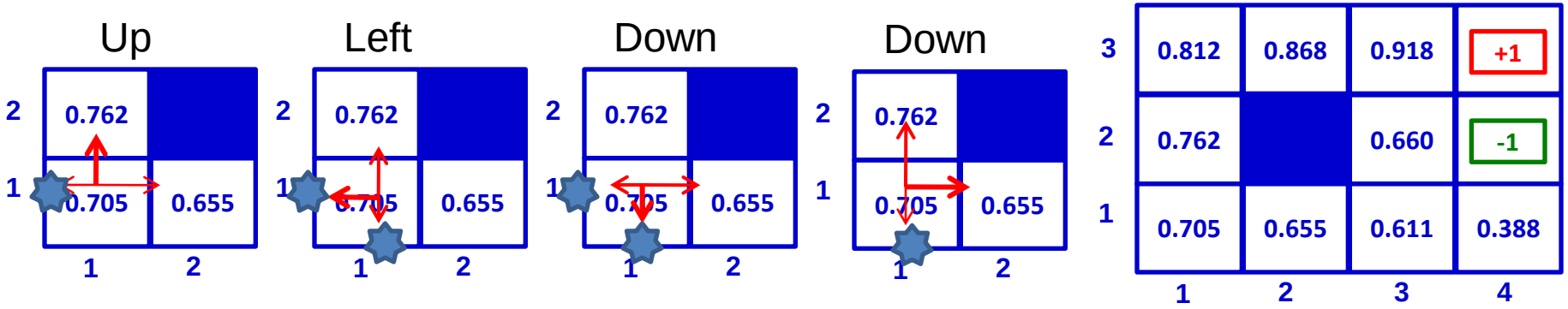
Полезности состояний

Уравнение Беллмана

$$U(s) = R(s) + \gamma \cdot \max_a \sum_{s'} T(s,a,s') \cdot U(s')$$



$$U(1,1) = -0.04 + \gamma \cdot \max \begin{pmatrix} 0.8 \cdot U(1,2) & +0.1 \cdot U(2,1) & +0.1 \cdot U(1,1) & Up \\ 0.9 \cdot U(1,1) & +0.1 \cdot U(1,2) & & Left \\ 0.9 \cdot U(1,1) & +0.1 \cdot U(2,1) & & Down \\ 0.8 \cdot U(1,2) & +0.1 \cdot U(2,1) & +0.1 \cdot U(1,1) & Right \end{pmatrix}$$



Итерации по значениям

- Применяем итерационные методы.
- $\mathbf{U_i(s)}$ – полезность для состояния s в *i*-ой итерации
- $\mathbf{U_{i+1}(s)}$ – полезность следующего состояния

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s,a,s') \cdot U_i(s')$$

Итерации по стратегиям

- Чередование двух этапов:
- **Оценка стратегии.** На основе стратегии π_i вычислить полезность состояния *i* $U_i=U^{\pi}$.
- **Усовершенствование стратегии.** Вычислить новую стратегию π_{i+1} с максимальной ожидаемой полезностью.

- **Полезность состояния s** и связанной стратегии π_i

$$U_{i+1}(s) = R(s) + \gamma \sum_{s'} T\left(s, \pi_i(s), s'\right) \cdot U_i(s')$$

$\pi_i(s)$ - это выбранная стратегия

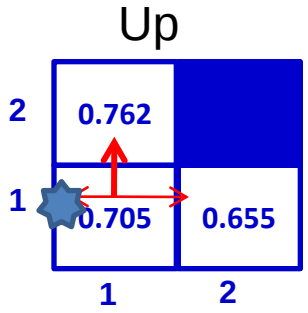
Итерации по стратегиям

Уравнения Беллмана

$$U(1,1) = -0.04 + \gamma \cdot \max \begin{pmatrix} 0.8 \cdot U(1,2) & +0.1 \cdot U(2,1) & +0.1 \cdot U(1,1) & Up \\ 0.9 \cdot U(1,1) & +0.1 \cdot U(1,2) & & Left \\ 0.9 \cdot U(1,1) & +0.1 \cdot U(2,1) & & Down \\ 0.8 \cdot U(1,2) & +0.1 \cdot U(2,1) & +0.1 \cdot U(1,1) & Right \end{pmatrix}$$

Стратегия $\pi_i(1,1) = U_p$

$$U_i(1,1) = -0.04 + 0.8 \cdot U_i(1,2) + 0.1 \cdot U_i(1,1) + 0.1 \cdot U_i(2,1)$$



Частично наблюдаемая среда

- Среда является частично наблюдаемой. Агент выполняет действие, рекомендуемое для этого состояния.
- Агент не имеет датчиков и не знает, где находится.

Первоначальное распределение вероятностей

	1	2	3	4
3	0.111	0.111	0.111	0.000
2	0.111		0.111	0.000
1	0.111	0.111	0.111	0.111

Распределение вероятностей после движения **Left** 5 раз

	1	2	3	4
3	0.300	0.010	0.008	0.000
2	0.221		0.059	0.012
1	0.371	0.012	0.008	0.000

Марковские процессы принятия решений в частично наблюдаемой среде

- Среда является частично наблюдаемой. Агент выполняет действие, рекомендуемое для этого состояния.
- Агент не имеет датчиков и не знает, где находится.

Распределение вероятностей после движения **Up** 5 раз

3	0.672	0.221	0.071	0.024
2	0.005		0.003	0.022
1	0.003	0.024	0.03	0.000
	1	2	3	4

Распределение вероятностей после движения **Right** 5 раз

3	0.005	0.007	0.019	0.775
2	0.034		0.007	0.105
1	0.005	0.006	0.008	0.000
	1	2	3	4

Модель наблюдения

Начальное доверительное состояние:

$$\left\langle \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, 0, 0 \right\rangle$$

Агент определяет текущее доверительное состояние *b(s)*:

$$b'(s') = \alpha \cdot O(s', o) \sum_{s} T(s, a, s') \cdot b(s)$$

α - нормализующая константа

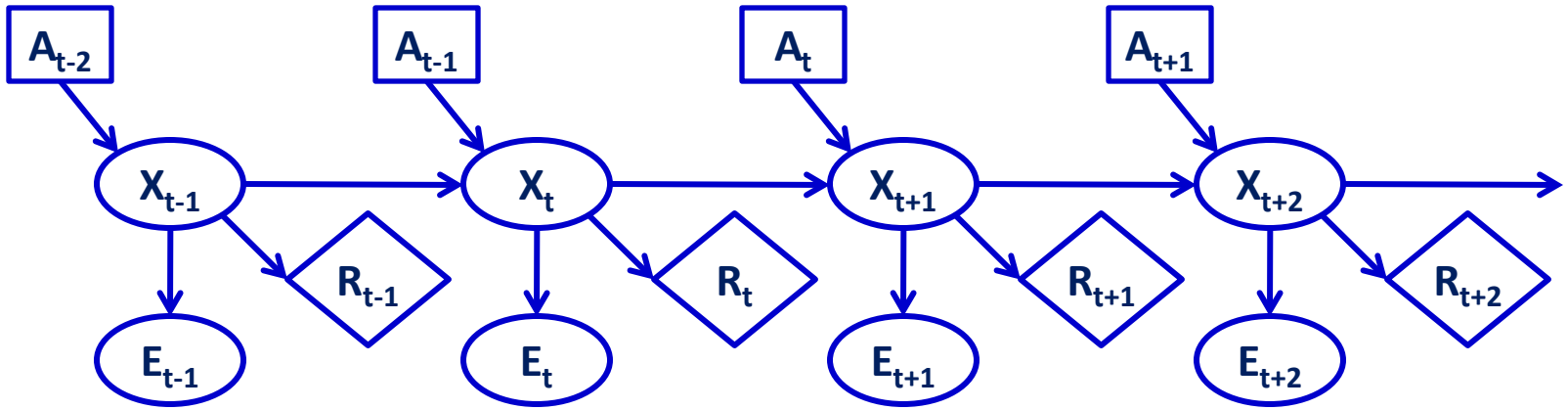
Оптимальное действие зависит от зависит только от текущего доверительного состояния агента.

Агенты, действующие на основе теории решений

- Модели перехода и наблюдения представлены в виде **динамических байесовских сетей**
- Динамическая байесовская сеть дополняется узлами **принятия решений** и узлами **полезности**.
- Для обновления доверительных состояний используется **алгоритм фильтрации**
- Решения принимаются путем выбора **наилучших действий**
- $\mathbf{X_t}$ – множество переменных состояния
- $\mathbf{E_t}$ – измеряемые переменные
- $\mathbf{P(E_t/X_t}$, A**)** – модель перехода

Агенты, действующие на основе теории решений

- $\mathbf{X_t}$ – множество переменных состояния
- $\mathbf{E_t}$ – измеряемые переменные
- $\mathbf{P(E_t/X_t}$,A**)** – модель перехода T(s,a,s')



Агент максимизирует сумму всех будущих вознаграждений
Значение полезности U доступно только в приближенной форме

Принятие решений при наличии нескольких агентов

- Теория игр применяется в серьезных приложениях:
  - Организация аукционов,
  - Распределение спектра радиочастот,
  - Назначение цен на продукцию
  - Национальная оборона
  - Защита информации
- Задачи:
- Проектирование агента. Вычислять стратегии каждого агента, определять полезность каждого решения агента (при условии их рациональности)
- Проектирование механизма. Определение правила игры, максимизировать общую полезность всех агентов. Создание мультиагентных систем.

Описание игры

- **Игроки** – агенты, принимающие решение
- **Действия**, которые выбирают агенты
- **Матрица вознаграждений** - определяет полезность для каждого игрока

Пример: Игра «чет\нечет»

- Игроки: Odd, Even
- Действия: выбросить число 1 или 2
- Матрица вознаграждений:

		Odd	
		one	**two**
Even	**one**	**E=2,O=-2**	**E=-3,O=3**
	two	**E=-3,O=3**	**E=4,O=-4**

Игра «чет-нечет»

- **Чистая стратегия** – это детерминированная стратегия, определяющая одно действие в каждой ситуации, возникающей в игре.
- **Смешанная стратегия** – это рандомизированная стратегия, предусматривающая вероятностный выбор действия в каждой ситуации.

  [p1,'one'; p2,'two'], p1+p2=1
- **Профиль стратегии** – вариант присваивания стратегии каждому игроку.
- **Результат игры** – числовое значение для каждого игрока.

		Odd	
		one	**two**
Even	**one**	**E=2,O=-2**	**E=-3,O=3**
	two	**E=-3,O=3**	**E=4,O=-4**

Дилемма заключенного

Игроки – грабители Alice и Bob пойманы с поличным
Действия – сознаться или не сознаться
Матрица вознаграждений

- за грабеж получат срок 5 (сознаться) или 10 (не сознаться)
- за владение краденым получат 1 год

		Alice	
		Сознаться	**Не сознаться**
Bob	**Сознаться**	**A=-5** **B=-5**	**A=-10** **B=0**
	Не сознаться	**A=0** **B=-10**	**A=-1** **B=-1**

Дилемма заключенного

- **Доминантная стратегия** – результат которой лучше для данного игрока при любом решении противоположного игрока.
- **Оптимальная по принципу Парето** стратегия – результат которой лучше для **всех игроков**.

- **Доминантная стратегия для Алисы:** свидетельствовать против Боба.
- **Доминантная стратегия для Боба:** свидетельствовать против Алисы.
- Имеет место **равновесие доминантных стратегий**
- **Равновесие** – ни одному из игроков не выгодно переключаться на другую стратегию
- **Равновесие** – локальный оптимум в пространстве стратегий

		Alice: Сознаться	Alice: Не сознаться
Bob: Сознаться		A=-5 B=-5	A=-10 B=0
Bob: Не сознаться		A=0 B=-10	A=-1 B=-1

Дилемма заключенного

- Результат равновесных доминантных стратегий хуже результата [-1,-1], выгодного для обоих грабителей
- Результат доминируется по принципу Парето результатом[-1,-1]

- **Равновесие Джона Нэша** – каждая игра имеет равновесие в смешанных стратегиях.
- Каждый игрок будет выбирать {свидетельствовать, свидетельствовать} вследствие наличия равновесия.

		Alice: Сознаться	Alice: Не сознаться
Bob: Сознаться		A=-5 B=-5	A=-10 B=0
Bob: Не сознаться		A=0 B=-10	A=-1 B=-1

Отсутствие доминантных стратегий

- **Игроки:** компании на рынке, работающие компьютерных игр:
  - A – производитель аппаратных средств,
  - B - производитель программных средств.
- **Действия:** производить игры
  - на CD,
  - на DVD.
- **Матрица вознаграждений**:

Если игроки примут несогласованные решения, то оба потерпят убытки

		A: DVD	A: CD
B	DVD	A=9,B=9	A=-4,B=-1
B	CD	A=-3,B=-1	A=5,B=5

Отсутствие доминантных стратегий

- Отсутствие доминантных стратегий
- Наличие двух равновесий Нэша.
- Имеется два приемлемых решения. Если каждый агент выберет другое решение, то оба агента получат убытки.
- Оба агента должны выбирать решение, оптимальное по Парето.
- Агенты должны сотрудничать.
- **Координационными** называются игры, в которых агенты должны взаимодействовать.

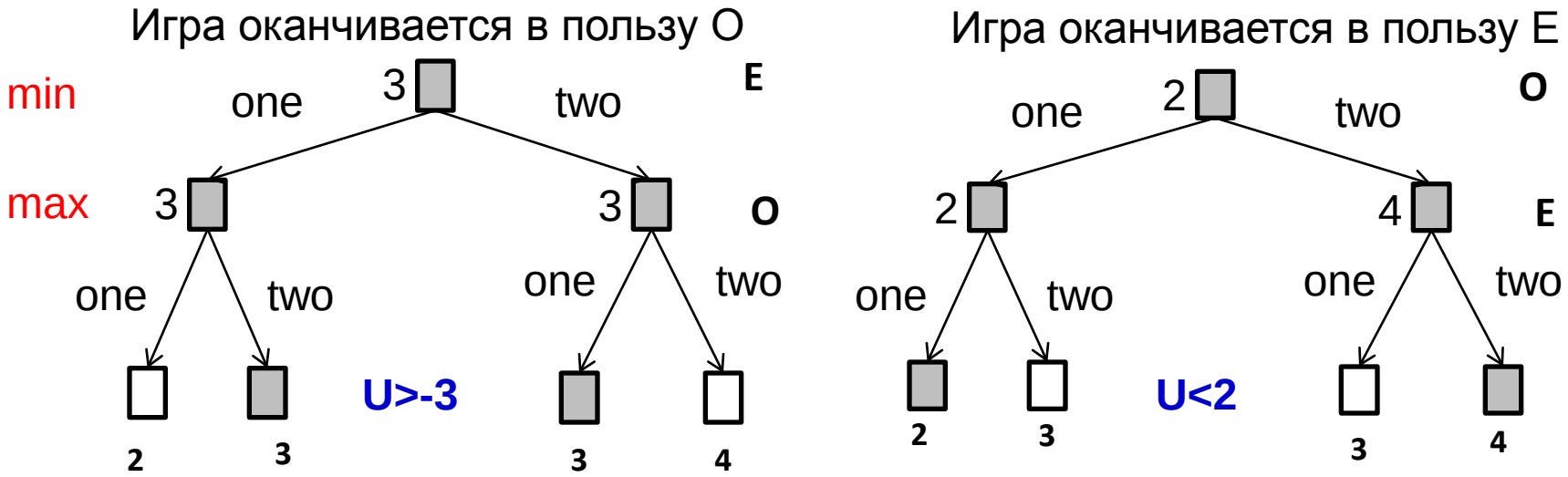
		A	
		DVD	CD
B	DVD	A=9,B=9	A=-4,B=-1
	CD	A=-3,B=-1	A=5,B=5

Игры с нулевой суммой

- Игры с нулевой суммой - имеют сумму вознаграждений в каждой клетке равна нулю.
- Метод поиска оптимальной смешанной стратегии разработал Фон Нейман.
- Пусть игрок Е первым вынужден раскрывать свою стратегию. Затем ходит игрок О. оптимальная стратегия **минимаксная.**

		Odd	
		one	**two**
Even	**one**	**E=2,O=-2**	**E=-3,O=3**
	two	**E=-3,O=3**	**E=4,O=-4**

Игры с нулевой суммой



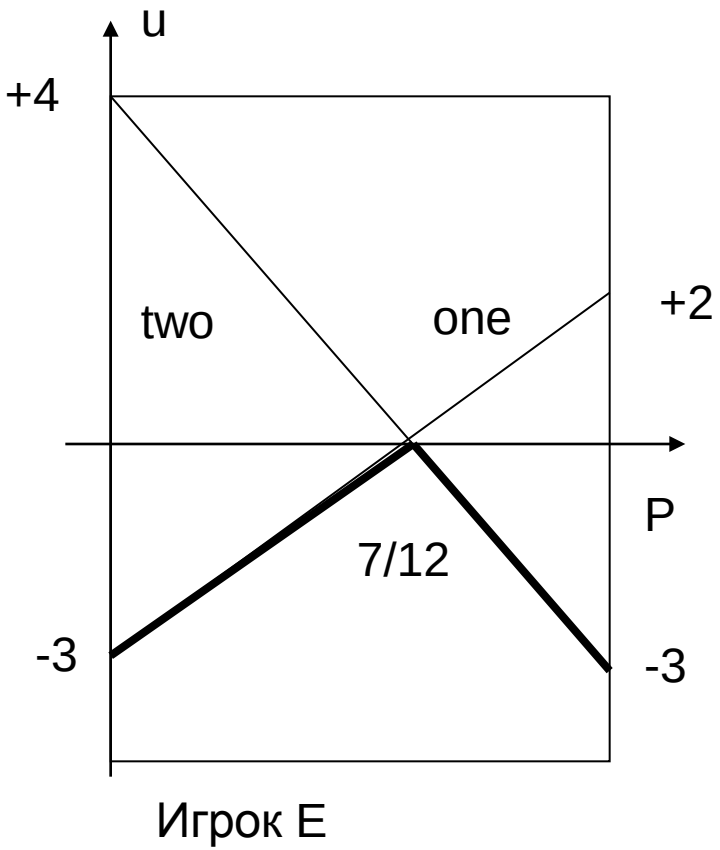
E может проиграть самое большее 3

E может выиграть самое большее 2

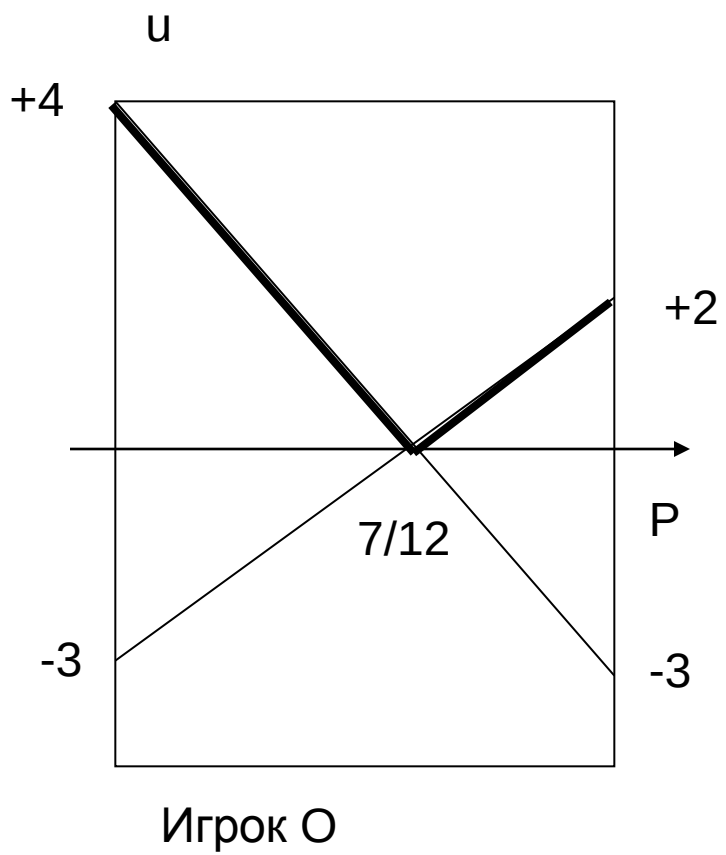
Выигрыш -3<U<2

Как только первый игрок раскрыл свою стратегию, второй не может проиграть согласно **чистой стратегии.**

Смешанные стратегии



Игрок E



Игрок O