Сразу надо сказать, что я не записывал ничего, связанного с практикой пролога или примерами кода на прологе. Надеюсь, этого в тесте не будет

Искусственный интеллект – комплекс технологических решений, позволяющий имитировать когнитивные функции человека (включая самообучение и поиск решений без заранее заданного алгоритма) и получать при выполнении конкретных задач результаты, сопоставимые, как минимум, с результатами интеллектуальной деятельности человека.

Интеллектуальная система – это техническая или программная система, способная решать задачи, традиционно считающиеся творческими, принадлежащие конкретной предметной области, знания о которой хранятся в памяти системы. Интеллектуальная система включает три основных блока: базу знаний, решатель и интеллектуальный интерфейс

Два подхода к моделированию мышления:

1) Символический (логический) подход. Исторически был доминирующим подходом к моделированию интеллекта. Согласно этому подходу, все интеллектуальные действия сводятся к оперированию символами или понятиями. Он включает построение формальных моделей и соответствующих им механизмов рассуждений.

2) Нейро-сетевой или нейрокибернетический подход. Является противоположностью символического подхода к исследованию и моделированию интеллекта. Основная идея этого направления: единственный объект, способный мыслить, — это человеческий мозг. Поэтому любое мыслящее устройство должно каким-то образом воспроизводить его структуру. Подход ориентирован на программно-аппаратное моделирование структур, подобных структуре мозга. Делается упор на создание элементов, аналогичных нейронам, и их объединение в функционирующие системы, т.е. в нейронные сети (НС).

Машинное обучение – это исследование компьютерных алгоритмов, которые могут автоматически улучшаться через опыт и использование данных. Алгоритмы машинного обучения создают модель на основе примеров данных, известных как данные обучения для того, чтобы делать предсказания или находить решения, не будучи явно запрограммированными на это. Машинное обучение рассматривается как раздел искусственного интеллекта.

Объяснимый искусственный интеллект – набор процессов и методов, позволяющих пользователям понять, почему алгоритмы машинного обучения пришли к тем или иным результатам или выводам. Объяснимый ИИ помогает охарактеризовать точность, достоверность и прозрачность модели, предназначенной для принятия решений с помощью ИИ. Объяснимый ИИ играет важнейшую роль для повышения достоверности и надежности производственных моделей ИИ. Кроме того, объяснимость ИИ помогает организациям с большей ответственностью подходить к разработке ИИ

Основные модели представления знаний: 1) Логическая модель 2) Сетевая модель 3) Продукционная модель

Логическая модель

Формальна логика - один из достаточно известных средств представления знаний

Формальная система - совокупность абстрактных объектов, в которой представлены правила оперирования множеством символов в чисто синтаксической трактовке без учета их семантики (или смыслового содержания).

Формальная система (ФС) определена, если:

1) задан конечный алфавит α (или конечное множество символов);

2) определена процедура построения правильных формул β;

3) выделено некоторое множество формул A, называемых аксиомами;

4) задано конечное множество правил вывода P, которые позволяют получать из некоторого конечного множества формул другое множество формул

Логика высказываний (ЛВ) или **исчисление высказываний** (ИВ) – это простейшая математическая логика.

Логика первого порядка — формальное исчисление, допускающее высказывания относительно переменных, фиксированных функций и предикатов. Расширяет логику высказываний

В логике первого порядка используются традиционные для любой логики: атомы (атомарные формулы), логические связки (И, ИЛИ, НЕ, ИМПЛИКАЦИЯ, ТОЖДЕСТВО), а также термы, предикаты и кванторы.

Для **построения атомов** разрешается использовать следующие четыре типа символов:

(i) Индивидные символы или константы. Это обычно имена объектов такие, как Мэри, Джон и 3.

(ii) Символы предметных переменных. Это обычно строчные буквы x, y, z, ..., возможно, с индексами.

(iii) Функциональные символы. Это обычно строчные буквы f, g, h, ... или осмысленные слова из строчных букв такие, как отец и плюс.

(iv) Предикатные символы. Это обычно прописные буквы P, Q, R, . . . или осмысленные слова из прописных букв такие, как БОЛЬШЕ или ЛЮБИТ

Термы определяются рекурсивно следующим образом:

(i) Константа есть терм.

(ii) Переменная есть терм.

(iii) Если f есть n-местный функциональный символ и t_1 ..., t_n – термы, то f (t_1 ..., t_n) – терм.

(iv) Никаких термов, кроме порожденных применением указанных выше правил, нет.

Если P – n–местный предикатный символ и t_1, ..., t_n – термы, то P (t_1, ..., t_n) – **атом (или атомарная формула)**.

Правильно построенные формулы или формулы логики первого порядка рекурсивно определяются следующим образом:

(i) Атом есть формула.

(ii) Если F и G – формулы, то ¬ (F), (F ∨ G), (F ∧ G), (F → G) и (F↔G) – формулы.

(iii) Если F – формула, а x – свободная переменная в F, то (∀ x) F и (∃ x) F – формулы.

(iv) Формулы порождаются только конечным числом применений правил (i), (ii) и (iii).

Формула T называется теоремой, если существует доказательство, в котором она является последней, т.е. F_r=T.

Интерпретация представляет собой распространение исходных положений формальной системы на реальный мир.

Если некоторое множество формул не удовлетворяется ни при какой интерпретации, то оно **называется противоречивым (или невыполнимым)**

Было доказано, что **метод резолюции является исчерпывающим** методом доказательства теорем.

Это значит, что если формула противоречива, то с помощью метода резолюции всегда можно обнаружить это противоречие

Много примеров по резолюции и дискретке за 1 семак, всё есть в 1 лекции….

Основы Логического программирования (ПРО ПРОЛОГ)

Основными **областями применения языка Prolog** являются: 1) базы данных и базы знаний; 2) экспертные системы и исследования в области искусственного интеллекта; 3) общение с ЭВМ на естественном языке (естественно-языковые интерфейсы, машинный перевод, вопросно-ответные системы); 4) построение планов действий роботов; 5) составление расписаний; 6) быстрое прототипирование прикладных программ; 7) написание компиляторов, конверторов программ с одного языка в другой; 8) системы автоматизированного проектирования (САПРы)

Пролог-программа состоит из двух частей: 1) базы данных (соответствует множеству аксиом); 2) последовательности целевых утверждений (или запросов)

Переменная — это последовательность букв и/или цифр, начинающаяся либо с ПРОПИСНОЙ буквы, либо со знака «_» (подчеркивания).

Составной терм — это выражение вида f (t1, ..., tn), где f — имя функтора (атом), t1, ..., tn — термы.

При поиске решений в Прологе используется Бэктрекинг или откат. **Откат (Бэктрекинг)** – это механизм, который используется для нахождения дополнительных фактов и правил, необходимых для вычисления цели, если текущая попытка вычислить цель (или подцель) оказалась неуспешной.

Пролог-программ используются два встроенных предиката cut и fail (отсечение и отказ/неудача). 1) **fail** — тождественно ложный предикат (fail/0). Его исполнение вырабатывает значение «неудача». 2) **true** — тождественно истинный предикат (true/0). Его исполнение всегда успешно. 3) **cut** — выполнение предиката cut/0 (!) всегда завершается успешно, но сопровождается рядом побочных эффектов.

Базис рекурсии — это предложение, определяющее некую начальную ситуацию или ситуацию в момент завершения рекурсии

Шаг рекурсии — это правило, тело которого содержит в качестве подцели вызов определяемого предиката

Список – это упорядоченный набор объектов любого типа. Объекты списка называются его элементами.

Экспертные системы (ЭС) – это программные системы, которые при решении задач, трудных для человека-эксперта (т.е. квалифицированного специалиста в данной предметной области), получают результаты, не уступающие по качеству и эффективности решениям, получаемым самим экспертом

ЭС предназначены для решения неформализованных задач и обладают следующими особенностями:

1) Алгоритм решения задач не известен заранее, а строится самой ЭС с помощью символических рассуждений, базирующихся на эвристических приемах;

2) «Прозрачность» (ясность) полученных решений. Т.е. система «осознает» в терминах пользователя, как она получила решение, и может проанализировать и объяснить свои действия и знания;

3) Способность приобретения новых знаний от пользователя-эксперта и изменения в соответствии с ними своего поведения;

4) Обеспечение «дружественного» интерфейса.

Структура типовой экспертной системы:

1) **Рабочая память (РП)** предназначена для хранения исходных и промежуточных данных решаемой в текущий момент задачи
2) **База знаний (БЗ)** включает универсальные данные (факты), описывающие рассматриваемую область знаний, и множество правил, описывающих целесообразные преобразования данных в этой области.
3) **Решатель**, используя исходные данные из РП и знания из БЗ, формирует такую последовательность правил, которые, будучи примененными к исходным данным, приводят к решению задачи
4) **Подсистема приобретения знаний** автоматизирует процесс наполнения ЭС знаниями, осуществляемый, как правило, пользователем-экспертом
5) **Подсистема объяснений** объясняет, как система получила решение задачи (или почему она не получила решение) и какие знания она при этом использовала, что

облегчает эксперту тестирование системы и повышает доверие пользователя к полученному результату

6) **Интерфейс** обеспечивает дружественное общение с пользователем как в ходе решения задач, так и в процессе приобретения знаний и объяснения результатов работы.

В разработке ЭС участвуют: 1) эксперт – носитель знаний в проблемной области, задачи которой будет решать ЭС;

2) инженер знаний – специалист по представлению знаний; 3) программист – специалист по программированию.

Режимы работы ЭС:

1) В **режиме приобретения знаний** общение с ЭС осуществляет (через посредничество инженера знаний) эксперт. В этом режиме эксперт, используя компонент приобретения знаний, наполняет систему знаниями, которые позволяют ЭС в режиме решения решать задачи из проблемной области.

2) В **режиме решения задачи** общение с ЭС осуществляет конечный пользователь, которого интересует результат решения задачи и, как правило, способ его получения.

Способы организации логического вывода ЭС:

1) Первый метод (**прямой вывод**) использует рассуждения для нахождения различных предположений, обусловленных имеющимися фактами и правилами.

2) Второй метод **(обратный вывод)** основан на исследовании заключений, которые представляют интерес и могут быть (а могут и не быть) истинными.

Вторая модель представления знаний – Сетевая модель

В основе сетевых моделей лежит понятие сети. В этих моделях в явной форме выделены все отношения, составляющие каркас знаний описываемой области знаний, и учитывается их семантика.

Семантическая сеть (СС) – одни из средств представления знаний, это ориентированный граф, вершинам которого сопоставляются объекты (понятия, конкретные объекты, события, процессы, явления и т.п.), а дугам – отношения, существующие между объектами.

Зафиксируем конечное множество символов $A = \{A_1, \ldots, A_r\}$, которые будем называть атрибутами. **Схемой или интенсионалом некоторого отношения** Ri в атрибутивной форме будем называть набор пар

INT (Ri) = {..<Aj, DOM(A)>…}, где Ri – имя отношения местности ni , $Aj \in A$, j = 1...ni – атрибуты отношения Ri , DOM(Aj) – домен атрибута Aj (множество значений атрибута Aj отношения Ri).

Экстенсионалом отношения Ri будем называть множество EXT(Ri)={...Fk ...},k=1...pi , где pi – кардинал множества EXT(Ri), Fk – факты отношения Ri записываемые в виде: Fk = Mk : Ri (A1vi1k ... Aj vijk ...), где Mk – метка факта, $vijk \in DOM(Aj)$ – значение атрибута Aj.

Тогда семантическая сеть есть совокупность пар {...< INT(Ri) EXT(Ri) >...} (для i = 1...m), где m – число различных отношений Ri .

Типы семантических сетей:

1) По виду вершин: **простые** (сети, включающие вершины, не имеющие внутренней структуры,) и **иерархические** сети.
2) По типу дуг (отношений): **однородные** (если все отношения между вершинами сети одинаковы (одного типа), то такая сеть называется однородной, в противном случае - неоднородной) и **неоднородные сети**; **бинарные** (если в сети используются только бинарные отношения, то такая сеть называется бинарной.) и **небинарные** сети.

Классификация отношений:

1) **Отношения таксономии**: а) «класс-подкласс» (SUB), «множество-подмножество» – между понятиями; б) «элемент-класс» (ISA), «элемент-множество» – между экземплярами понятий и понятиями.
2) **Отношения партономии**: - «часть-целое» (Part_of) – отношение включения одного понятия (объекта) в другой.
3) **Атрибутивные отношения или отношения свойство-значение** («цвет», «вес», «рост», «объем» и т.п.).
4) **Логические отношения** («и», «или», «не», «следование»).
5) **Темпоральные отношения** («раньше», «позже», «одновременно» и т.п.).
6) **Пространственные отношения** («далеко от», «близко от», «под», «над» и т.п.).
7) **Глубинно-падежные семантические отношения Филмора**, служащие для выражения в предложении глубинных семантических отношений между группой существительного и действием («агент», «объект», «инструмент», «время действия», «место действия» и т.п.).

Достоинства сетей:

1) высокая выразительность и гибкость в представлении информации;
2) универсальность (выбрав соответствующий набор отношений можно представить любую информацию);
3) наглядность представления знаний в виде графа;
4) соответствие современным представлениям об организации долговременной памяти человека.

Недостатки сетей:

1) при росте числа отношений сеть становится запутанной и трудной для понимания и управления человеком;
2) поиск решения в семантической сети сводится к задаче поиска фрагмента сети, соответствующего образцу (подсети), отражающему поставленный запрос, а это очень ресурсоемкая задача, зависящая от размера сети

Функциональная сеть – представляет собой двудольный ориентированный граф, включающий вершины двух типов – объекты (переменные) и операторы (функции).

Фрейм – это структура данных, представляющая стереотипную ситуацию. Похож на иерархическую сеть

Группы родственных фреймов объединяются в **систему фреймов.**

Системы фреймов, в свою очередь, связаны **информационно-поисковой сетью**.

Достоинства фрейма:

1) иерархичность описания понятий предметной области;
2) достаточно естественное объединение в рамках одного средства как декларативного, так и процедурного компонентов представления знаний.

Основной недостаток - слабая формализация, отсутствие общей теории.

Продукционная модель

Продукционная модель предполагает такой способ организации вычислительного процесса, при котором программа преобразования некоторой информационной структуры C задается в виде системы правил вида: Условие $\rightarrow$ Действие, где Условие специфицирует некоторые требования к текущему состоянию структуры C, а Действие содержит описание тех операций над C, которые надо выполнить, если C удовлетворяет этим требованиям

Системы подстановок служат для обработки слов, заданных в некотором алфавите.

К ним относятся: (1) **системы продукций Поста** (именно этим системам мы обязаны появлению термина "система продукций", который получил со временем более широкое употребление) и (2) **нормальные алгоритмы Маркова**.

Системы Поста определяются алфавитом S и набором правил-продукций вида: $a_iW \rightarrow Wb_i$ (i = 1, ..., m), (1) где ai и bi - некоторые слова в алфавите S.

Нормальные алгоритмы Маркова определяются алфавитом S и последовательностью подстановок вида: ai $\rightarrow$ bi
ai $\rightarrow$. bi, где ai и bi - некоторые слова в алфавите S

Формальные грамматики были введены Хомским, предложившим описывать их четверкой <V, T, P, Z>, где V - алфавит, T $\subseteq$ V - алфавит терминальных символов, P - конечный набор правил подстановки, Z - начальный символ.

Программные продукционные системы - системы, имеющие программную реализацию. Такие системы далее называются **системами продукций (СП)**.

Программная СП состоит из трех основных частей:

1) **База данных** представляет собой рабочую память (различной организации), над которой работает множество правил.
2) **Правила** могут иметь произвольную сложность, но структура у них прежняя: левая часть - условие применимости, а правая часть - действие, которое данное правило выполняет.

3) **Интерпретатор** - поисковый процесс, состоящий, по крайней мере, из двух фаз: выбора продукции и ее применения

Задача выбора продукции сводится к двум подзадачам: (1) максимально ограничить число продукций, условия применимости которых будут проверяться, (2) из полученного множества продукций выбрать одну – с истинным условием применимости.
Множество продукций, у которых условия применения истинны в данный момент, называется **конфликтным**. Процедура выбора продукции из такого множества называется **процедурой разрешения конфликта**

Основные способы выбора продукций из конфликтного множества:

1) **случайный выбор**;
2) **выбор по статическому критерию** (например, первой применяется продукция с самыми жесткими требованиями - продукция с самым длинным списком условий), либо на продукциях задан полный порядок или иерархия, при этом первой применяется самая "старшая" продукция и т.п.);
3) **выбор по динамическому критерию** (например, приписыванием правилам и/или компонентам базы данных динамически вычисляемых весов (приоритетов); в этом случае первой для исполнения выбирается продукция с наивысшим приоритетом, или продукция, условия (образец) которой удовлетворяется на данных, имеющих максимальный приоритет).

Выделяется **два класса управляющих стратегий применения продукций**:

1) При использовании **безвозвратной стратегии** на каждом шаге вычислений из конфликтного множества для выполнения выбирается одна из подходящих продукций, и в дальнейшем вернуться к этой точке вычислений и применить другую продукцию невозможно.
2) При **пробной стратегии (бэктрекинге)** обеспечивается возможность возврата к уже пройденной точке вычислений и применения другой (альтернативной) продукции из конфликтного множества

Прямой вывод предполагает использование правил для вывода новых фактов из имеющихся.
Прямой вывод применим в тех ситуациях, когда пространство возможных решений необозримо, в то время как количество исходных данных невелико

При **обратном способе вывода**, вывод начинается не с посылок правил, а сразу с интересующего нас заключения (будем называть его целевым или целью).
Обратный вывод применяется в тех задачах, где число возможных решений невелико, но присутствуют большие объемы исходных данных

Классификация СП по тому, как они решают проблему активации продукции:

1) В **простых СП** активными считаются все продукции. Такой способ активации правил применяется (1)в СП с небольшим количеством правил, (2) в СП, в которых структуризация множества продукций и их принудительная активация противоречат принципам, положенным в основу данной СП, или используемой ею стратегии выбора продукций

2) При этом подходе используются метапродукции, т.е. продукции, содержащие информацию о других продукциях и активирующих (дезактивирующих) на основе этих знаний и анализа текущего состояния БД другие продукции. СП могут включать несколько уровней метапродукций. Поэтому **такие СП называются иерархическими**.

3) **Последовательная** СП разбивается на несколько подмножеств, каждое из которых представляет автономный модуль обработки данных (продукционный модуль). Каждый модуль (СПi) соответствует определенному уровню знаний или этапу обработки данных.

4) В **параллельно-последовательных СП** множество правил разбито на непересекающиеся подмножества, каждое из которых имеет свою рабочую память.

Достоинства СП:

1) **Универсальность** СП, как метода описания широкого класса задач.

2) **Естественность спецификации знаний**. Для многих предметных областей естественно представлять знания в виде правил вида если А, то В (Условие $\rightarrow$ Действие).

3) **Высокая и естественная модульность СП**: (1) каждая продукция представляет собой автономное действие, снабженное индивидуальной функцией управления, самостоятельно определяющей момент выполнения действия; (2) все множество продукций может естественным образом структурироваться путем разбиения на подмножества, объединяющие продукции, которые относятся к одним и тем же компонентам знаний.

Недостатки СП:

1) Существенно более низкая эффективность вычислительного процесса по сравнению с программированием на традиционных языках.

2) Повышенная сложность контроля правильности СП-процесса.

3) Сложность отслеживания непротиворечивости множества правил.

Использование продукционной модели

1) Построение компиляторов

2) Автоматическая обработка текстов

3) Распознавание и синтез речи

4) Экспертные системы

Нечеткие модели

Нечеткость связана с отсутствием точных границ области определений и свойственна большинству понятий

Лингвистическая переменная (ЛП) – это переменная, значениями которой являются слова или выражения естественного (иногда искусственного) языка.
Пример: Переменную Возраст можно рассматривать как лингвистическую переменную,

если она принимает не числовые значения (например, от 0 до 100), а лингвистические значения, такие как молодой, старый, очень молодой, очень старый и т.п.

Если понимать истинность как лингвистическую переменную со значениями истинно, почти истинно, не очень истинно и т.п., то мы **переходим к так называемой нечеткой логике**, на которую могут опираться приближенные рассуждения.

Нечеткое множество (НМ) $A = \{ (x, \mu_A (x)) \}$ определяется как совокупность упорядоченных пар, составленных из элементов x универсального множества X и соответствующих степеней принадлежности $\mu_A (x)$, или непосредственно в виде функции принадлежности $\mu_A (x)$: $X \rightarrow [0,1]$.

Универсальным множеством (УМ) X нечеткого множества A называется область определения функции принадлежности μ_A .

Носителем НМ A называется множество таких точек в X, для которых $\mu_A (x)>0$.

Высотой НМ A называется величина $\sup \mu_A (x)$. X

Точкой перехода НМ A называется такой элемент множества X, степень принадлежности которого множеству A равна 0.5.

Операции над нечеткими множествами: 1)Дополнение 2) Объединение 3) Пересечение 4) Произведение

Слабым местом в нечеткой логике является функция принадлежности, вернее ее выбор

Существует свыше десятка **типовых форм кривых для задания функций принадлежности**. Наибольшее распространение получили: • треугольная, • трапецеидальная и • гауссова функции принадлежности

Шортлифф разработал схему, основанную на так называемых коэффициентах уверенности, которые он ввел для измерения степени доверия к любому данному заключению, являющемуся результатом полученных к этому моменту свидетельств

Коэффициент уверенности – это разность между двумя мерами:

КУ [h : e] = МД [h : e] - МНД [h : e], (1)

где

КУ [h : e] – уверенность в гипотезе *h* с учетом свидетельств *e,*

МД [h : e] – мера доверия гипотезе *h* при заданных свидетельствах *e,*

МНД [h : e] – мера недоверия гипотезе *h* при свидетельствах *e.*

КУ может изменяться от *-1 (абсолютная ложь)* до *+1 (абсолютная истина),* причем *0* означает *полное незнание.* 27

Формула уточнения позволяет непосредственно сочетать новую информации со старыми результатами. Она применяется и мерам доверия и недоверия, связанным с каждым предположением

Формула для МД выглядит следующим образом:

МД [h : e1, e2] = МД [h : e1] + МД [h : e2]*(1 - МД [h : e1]) ,

где запятая между свидетельствами e1 и e2 означает, что e2 следует за e1. Аналогичным образом уточняются значения МНД.
Смысл формулы состоит в том, что эффект второго свидетельства e2 на гипотезу h при заданном свидетельстве e1 сказывается в смещении МД в сторону полной определенности на расстояние, зависящее от второго свидетельства.

Формула МД имеет два важных свойства: 1. Она симметрична в том смысле, что порядок e1 и e2 не существенен. 2. По мере накопления подкрепляющих свидетельств МД (или МНД) движется к определенности.

Схема Шортлиффа допускает также возможность того, что правила, как и данные, могут быть **ненадежными**. Это позволяет описывать более широкий класс ситуаций. Каждое правило снабжается «**коэффициентом ослабления**» (числом от 0 до 1), показывающим надежность правила.

Часто вводят так называемый **порог уверенности (ПУ)** – число от 0 до 1. Если КУ некоторого заключения меньше этого числа (ПУ), то таким заключением можно пренебречь

Выводы: хотя схема Шортлиффа не достаточно теоретически обоснована, но она хорошо зарекомендовала себя в практических приложениях, в частности в экспертной системе MYCIN и последовавшими за ней другими системами.

Генетические алгоритмы

Генетические алгоритмы (ГА) есть поисковые алгоритмы, основанные на механизмах натуральной селекции и натуральной генетики. **Основой для возникновения генетических алгоритмов** считается модель биологической эволюции и методы случайного поиска.

Эволюционный поиск — это последовательное преобразование одного конечного множества промежуточных решений в другое. Само преобразование можно назвать алгоритмом поиска или алгоритмом эволюции.

Для сравнения решений (хромосом, особей) между собой вводится **—функция приспособленности или целевая функция**, которая показывает, насколько хорошо данное решение соответствуют поставленной задаче.

Выделяют **три особенности алгоритма эволюции**: 1) каждая новая популяция состоит только из "жизнеспособных" хромосом; 2) каждая новая популяция "лучше" (в смысле целевой функции) предыдущей; 3) в процессе эволюции последующая популяция зависит только от предыдущей.

Генетический алгоритм — это алгоритм, который позволяет найти удовлетворительное решение к аналитически неразрешимым проблемам через последовательный подбор и комбинирование искомых параметров с использованием механизмов, напоминающих биологическую эволюцию.

Простой ГА состоит из трех операторов:

1) **Репродукция** – процесс, в котором хромосомы копируются согласно их целевой функции (ЦФ) f i (x). Копирование хромосом с лучшим значением ЦФ имеет большую вероятность для их попадания в следующую генерацию.
2) **Оператор кроссинговера (ОК)** выполняется в 3 шага. На первом шаге члены нового репродуцированного множества хромосом выбираются сначала. Далее каждая пара хромосом (стрингов) пересекается по следующему правилу: целая позиция k вдоль стринга выбирается случайно между 1 и длиной хромосомы L, уменьшенной на единицу, т.е. в интервале (1, L-1). Следуя традициям генетики, хромосомы 1 и 2 часто называют родителями, а хромосомы 1', 2' — их потомками (детьми). Число k, выбранное случайно, называется точкой ОК или точкой разрыва ОК, или точкой пересечения ОК.
3) Оператор **мутации**, необходим для "выбивания" популяции из локального экстремума и способствует защите от преждевременной сходимости.

Критерием останова ГА может служить:

1) заданное количество поколений,
2) достижение определенного значения целевой функции (у отдельного стринга)
3) **схождение популяций** - состояние популяции, когда все строки находятся в области некоторого экстремума и почти одинаковы.

Генетические алгоритмы применяются в основном там, где сложно или невозможно сформулировать задачу в виде, пригодном для более быстрых алгоритмов локальной оптимизации, либо если стоит задача оптимизации недифференцируемой функции или задача многоэкстремальной глобальной оптимизации.
С помощью генетических алгоритмов можно находить квазиоптимальные решения чуть ли не 99% задач принятия решения.

Применение ГА:

1) Оптимизация функций
2) Оптимизация запросов в базах данных
3) Разнообразные задачи на графах (задача коммивояжера, раскраска графа и т. п.) и комбинаторной оптимизации (например, задача о ранце)
4) Задачи компоновки (в САПР)
5) Составление расписаний
6) Игровые стратегии
7) Аппроксимация функций
8) Искусственная жизнь (моделирование эволюции)
9) Биоинформатика
10) Настройка и обучение искусственной нейронной сети

Нейронные сети

Нейронная сеть - совокупность нейронов, связанных, между собой определенным образом, с выделенными входными и выходными нейронами.

Искусственная нейронная сеть - совокупность связанных между собой формальных нейронов.

Бионейрон - клетка, имеющая ядро и два вида длинных отростков, связанных с другими нейронами.

Математически формальный нейрон – это пороговый элемент с единственным двоичным выходом, функция активации которого зависит от линейной комбинации всех входных сигналов

Формальный нейрон является примером нейрона с так называемой бинарной (или разрывной) функцией преобразования.

Существует **несколько основных типов соединения формальных нейронов** в сеть, а также комбинации и модификации:

1) **Последовательное соединение слоев**. Выбираются нейроны входного слоя, причем ровно столько, сколько сигналов собирается обрабатывать сеть. Далее определяется количество скрытых или внутренних слоев и количество нейронов в них. Эти числа зависят от конкретной задачи и выясняются уже в процессе обучения или формирования сети. После этого формируются нейроны выходного слоя - по количеству выходных сигналов.
2) **Сети с обратной связью, или так называемые полносвязные сети, или сети Хопфилда**. Здесь среди нейронов также выделяются входные и выходные, но сигнал передается не последовательно со слоя на слой, а последовательно, относительно времени.
3) **Два вида нейронных сетей комбинируют**, вводя обратную связь во внутренние слои последовательных сетей.

Основное свойство нейронных сетей - способность к обучению, или же накапливание опыта.

Пусть мы хотим, чтоб на вопрос In сеть выдала ответ Out. Для этого мы подаем In на вход сети и проводим прямое функционирование в результате, которого получаем ответ Out*. Теперь нужно оценить, на сколько полученный ответ Out* отличается от Out. Вводится функция оценки H=H(Out, Out*) со следующими свойствами: $H \geq 0$ и чем меньше H, тем меньше различий между Out и Out*
Считается, что сеть обучилась примеру, если функция оценки для него меньше заданного числа s. То есть нужно стремиться уменьшить H до заданного значения. Вспомним, что ответ Out* есть функция от In и от множества синаптических весов X. Теперь видно, что на самом деле H = H(In, Out, X), но In и Out - фиксированные величины, поэтому изменять и подстраивать можно только множество весовых коэффициентов X. **В этой подстройке и заключается суть обучения.**

Алгоритм обучения градиентными методами включает следующую последовательность действий:
1. Подать на вход вектор сигналов.

2. Провести нагруженное прямое функционирование.
3. Произвести оценку функционирования (для одного примера или же для страницы).
4. Подать на выход вектор ошибок.
5. Провести обратное функционирование и вычислить градиент.
6. Оптимизировать шаг обучения и изменить синаптические веса.

Обзор функциональных типов нейронных сетей и описание прикладных задач, решаемых ими:

1) Основная задача **классификатора** - попытаться определить принадлежность предложенного примера к одному или другому классу. (Например, в медицине это попытка определить принадлежность симптомов к классу того или иного заболевания.) Количество распознаваемых классов обычно влияет на число нейронов в последнем слое и, более того, обычно является важной характеристикой сети. Наибольшее применение нейросети этого типа нашли в экспертных системах и системах диагностики.

2) Второй по значимости тип нейросетей – **распознаватели**. Небольшое отличие от классификаторов заключается в уменьшении числа классов ответов до двух. То есть специализация распознавателей заключается в умении ответить на вопрос типа «свой - чужой». «Своими» считаются примеры, которым сеть предварительно обучили, а все оставшиеся автоматически считаются «чужими».

3) Третий тип – **предикторы или предсказатели**. Основной задачей сетей этого типа является предсказание поведения объектов на основе определенной известной о них информации. Если первые типы сетей работают в основном со статическими данными, то этот тип - с данными реального времени. Предикторы - наиболее «сложные» по своей структуре сети из-за огромной величины обучающей выборки и из-за необходимости постоянного обучения новым знаниям.

4) Четвертый тип – **ассоциативная память**. Здесь используется способность сетей к автоматическому выявлению закономерностей в представляемых выборках и возможность изменения сетью параметров представленного примера в сторону увеличения степени распознавания. На этом принципе работают, например, системы удаления шума в обработке изображений и сигналов.

5) **Сети нескольких типов интегрируют в одну систему**.

Перечисленные **функциональные типы нейронных сетей применимы к следующим задачам**:
1) восстановление и распознание изображений (ассоциативные массивы, классификаторы);
2) экспертные системы и системы диагностики (классификаторы, fuzzy нейроны);
3) системы управления и моделирования в реальном времени (классификаторы, предикторы, ассоциативная память);
4) защита информации, шифрация и средства безопасности (предикторы, распознаватели);
5) информационные и поисковые системы (предикторы, ассоциативная память, fuzzy нейроны). Построение так называемых интеллектуальных агентов для поиска нужной информации.

В настоящее время **существует несколько десятков топологий нейронных сетей**:
1) Глубокие сети доверия,

2) Сверточные нейронные сети,
3) Развертывающие нейронные сети,
4) Глубокие сверточные обратные глубинные сети,
5) Рекуррентные нейронные сети и др

Методы поиска

Два подхода к моделированию мышления:
1. **Символический подход**. Исторически был доминирующим подходом к моделированию интеллекта. Согласно этому подходу, все интеллектуальные действия сводятся к оперированию символами или понятиями. Он включает построение формальных моделей и соответствующих им механизмов рассуждений.
2. **Нейро-сетевой или нейрокибернетический подход**. Является противоположностью символического подхода к исследованию и моделированию интеллекта. Основная идея этого направления: единственный объект, способный мыслить, — это человеческий мозг. Поэтому любое мыслящее устройство должно каким-то образом воспроизводить его структуру.

Символическая система есть набор символов, образующих символические структуры, и набор процессов.
Символ есть первичное понятие. Примером символа является строка буквенно-цифровых знаков (список, изделие-35, 3.14 и т.п.).
Символическая структура представляет собой совокупность символов, соотнесенных определенным физическим способом (например, один символ следует за другим).

Процессы способны производить, разрушать и модифицировать символические структуры.

Символическая структура может рассматриваться как тип данных в некотором языке.
Символические структуры обладают двумя основными свойствами:
1) они могут обозначать (designate) объекты, процессы и другие символические структуры;
2) если они обозначают процессы, то они могут быть интерпретированы.

Символическая структура обозначает некоторую сущность (объект, процесс или символическую структуру), если символическая система может осуществлять поведение, определяемое данной сущностью, или может воздействовать на эту сущность. **Система может интерпретировать** символическую структуру, если структура обозначает некоторый процесс, и система может выполнить этот процесс.

Сложность решаемой задачи характеризуется следующим набором параметров:

1) **размер пространства**, в котором решается задача;
2) Параметр "**изменяемость области**" характеризует степень изменяемости области во времени и пространстве. По параметру "изменяемость" выделяются статические и динамические области.;
3) Параметр **"полнота модели"** характеризует адекватность модели, используемой для описания данной проблемной области. Обычно, если модель не полна, то для описания области используют несколько моделей, дополняющих друг друга за счет отражения различных свойств проблемной области (например, модель, описывающая электрическую схему двигателя, и модель, определяющая его механическую схему).

4) Параметр "**определенность данных**" характеризует степень точности/ошибочности и полноты/неполноты данных;
5) количество необходимых решений;
6) ограничения на результат и способ его получения.

"**Точность**" является показателем того, что проблемная область с точки зрения решаемых задач описана точными данными.
"**Ошибочность**" является показателем того, что данные о проблемной области не точны.
Под "**полнотой" (неполнотой) данных** понимается достаточность (недостаточность) входных данных для однозначного решения задачи. Обычно при неполноте данных для поиска решения необходимо использовать некоторые предположения или ограничения.

Требования пользователя к результату задачи, решаемой с помощью поиска, можно характеризовать параметрами:

1) "**количество решений**" может принимать следующие основные значения: одно решение, несколько решений, все решения
2) "**ограничения на результат и способ его получения**" определяют определенные требования к решению

Во введенных терминах **сложность задачи колеблется от простых задач** (малая область, статическая область, полная модель, определенные данные, какое-нибудь решение, отсутствие ограничений на результат и способ его получения) **до сложных задач** (большая область, динамическая область, неполнота одной модели, ошибочные и неполные данные, все решения, произвольные ограничения на результат и способ его получения).

Классификация методов решения задач на основе поиска:

1) **Методы поиска в одном пространстве** – методы, предназначенные для использования в следующих условиях: малые области, статические области, полнота модели, точные и полные данные.
2) **Методы поиска в иерархических пространствах** – методы, предназначенные для работы в больших статических областях.
3) **Методы поиска в динамической проблемной области** – методы, предназначенные для работы с областями, изменяемыми во времени и/или в пространстве.
4) **Методы поиска при неточных и неполных данных**.
5) **Методы поиска, использующие несколько моделей**, – методы, предназначенные для работы с областями, для адекватного описания которых одной модели недостаточно.

Решить задачу поиска в пространстве состояний – значит определить такую последовательность операторов, которая преобразует начальные состояния в конечные.
Процесс решения можно представить в виде графа G = (X, Y)

Дальше много терминов из теории графов, которые мы и так знаем (путь, стоимость пути и тп)

При поиске в глубину сначала раскрывается та вершина, которая была построена самой последней.

При практической реализации **поиск в глубину в некотором направлении завершается в следующих случаях:** 1) при достижении целевой вершины; 2) при достижении терминальной вершины; 3) при построении в ходе поиска вершины, глубина которой превышает некоторую граничную глубину.
При поиске в ширину **вершины раскрываются в том же порядке, в котором они порождаются**
Метод равных цен позволяет найти путь, стоимость которого минимальна.

Метод равных цен:

1) Поместить начальную вершину *s* в список ОТКРЫТ. Положить *g^(s) = 0*,
2) Если список ОТКРЫТ пуст, то НЕУДАЧА, иначе перейти к 3.
3) Взять из списка ОТКРЫТ ту вершину (*n)*, для которой величина *g^(n)* имеет наименьшее значение, и поместить ее в список ЗАКРЫТ. (При выборе целевая вершина имеет приоритет.)
4) Если *n* целевая вершина, то выдать решение, иначе переход к шагу 5.
5) Раскрыть вершину *n*, построив все ее дочерние вершины. Если дочерних вершин нет, то перейти к шагу 2. Для каждой дочерней вершины n_i вычислить стоимость $g^(n_i)$, положив $g^(n_i) = g^(n) + c(n, n_i)$. Поместить эти дочерние вершины вместе с вычисленными стоимостями в список ОТКРЫТ и построить указатели, идущие назад к *n*.
6) Перейти к шагу 2.

28

Стремление сократить время поиска привело к созданию **эвристических методов поиска**, т.е. методов, использующих некоторую (эвристическую) информацию о проблемной области для рассмотрения не всего пространства поиска, а таких путей в нем, которые с наибольшей вероятностью приводят к цепи

Таким образом, хорошим способом сокращения перебора является использование **эвристической информации** для определения на каждом шаге дальнейшего направления перебора. Для этого необходимо ввести меру "**перспективности**" вершины в виде некоторой **оценочной функции.**
Оценочная функция должна обеспечивать возможность ранжирования вершин – кандидатов на раскрытие – с тем, чтобы выделить ту вершину, которая с наибольшей вероятностью находится на лучшем пути к цели.

При **поиске методом редукции** решение задачи сводится к решению совокупности образующих ее подзадач. Этот процесс повторяется для каждой подзадачи до тех пор, пока каждая задача из полученного набора подзадач, образующих решение исходной задачи, не будет иметь очевидное решение. **Подзадача считается очевидной**, если ее решение общеизвестно или получено ранее.
Процесс решения задачи разбиением ее на подзадачи можно представить в виде специального **направленного графа G, называемого И/ИЛИ графом.**

Вершины И/ИЛИ графа:

1) **Конъюнктивные вершины** (вершины типа "И—) вместе со своими дочерними вершинами интерпретируются так: решение задачи сводится к решению всех ее подзадач, соответствующих дочерним вершинам конъюнктивной вершины.
2) **Дизъюнктивные вершины** (вершины типа "ИЛИ—) вместе со своими дочерними вершинами интерпретируются так: решение задачи сводится к решению любой из ее подзадач, соответствующих дочерним вершинам дизъюнктивной вершины.

Решение задачи при поиске методом редукции сводится к нахождению в И/ИЛИ графе решающего графа.
Решающий граф определяется как подграф из разрешимых вершин, показывающий, что начальная вершина разрешима.

Для некоторой подзадачи может быть **неизвестно ни ее решение, ни способ сведения ее к более простым подзадачам**. Такая подзадача называется **неразрешимой**. Определение неразрешимой вершины в И/ИЛИ графе можно сформулировать аналогично определению разрешимой

Говорят, что г**енератор является полным**, если он обеспечивает генерацию всех возможных решений.

Генератор является неизбыточным, если он генерирует каждое решение только один раз

При генерации текущего возможного решения (состояния или подзадачи) возникает **проблема распределения знаний между генератором и устройством проверки**
Выделяется важный вид метода «генерация — проверка»: **«иерархическая генерация — проверка»**

Каждое **частичное решение** описывает, например, не всё состояние, а только его некоторую часть, определяя таким образом класс возможных состояний

При большом размере пространства поиска имеет смысл разбить всё пространство на отдельные подпространства и осуществлять поиск сначала в каждом из них. Тогда можно сказать, что пространство поиска представлено иерархией пространств.

Абстрактные пространства - пространства, которые имеют описание только наиболее важных сущностей.

Классический **пример использования абстрактных пространств** − задача определения кратчайшего пути на карте.

Большей частью **эти методы опираются на абстрактные пространства**. При этом абстракция должна – подчеркнуть важные особенности рассматриваемой задачи, – позволить разбить задачу на более простые подзадачи и – определить последовательность подзадач (план решения), приводящую к решению основной задачи.

Если пространство поиска удается **факторизовать**, то поиск даже в очень большом пространстве можно организовать достаточно эффективно.

Пространство называется факторизованным, если оно разбивается на непересекающиеся подпространства (классы) частичными (неполными) решениями;

причем по виду частичного решения можно определить, что оно не приведет к успеху, т.е. что все полные решения, образованные из него, не приведут к целевым решениям.

Поиск в факторизованном пространстве осуществляется на основе метода иерархическая "генерация—проверка"

В простейшем случае **пространство поиска разбивается на фиксированную последовательность подзадач** (подпространств), с помощью которых можно решить любую входную задачу.

В ряде приложений не удается все решаемые задачи свести к фиксированному набору подзадач.
Примерами таких приложений являются **задачи планирования перемещений (объектов) в пространстве**.
План решения задачи в данном случае должен иметь переменную структуру и не может быть сведен к фиксированному набору подзадач.
Для решения подобных задач может быть использован **метод нисходящего уточнения**

Метод нисходящего уточнения базируется на следующих предположениях:
1) возможно осуществить частичное упорядочение понятий области, приемлемое для всех решаемых задач;
2) решения, принимаемые на верхних уровнях, нет необходимости отменять на более нижних.

Принципа наименьших свершений. В соответствии с данным принципом решение не строится сразу до конца на верхних уровнях абстракции. Частичное решение детализируется постепенно, по мере появления информации, подтверждающей возможность решения и вынуждающей принять решение.

Рассуждение, основанное на использовании принципа наименьших свершений, требует, **чтобы система была в состоянии**:

1) Определить, когда накопилось достаточно информации, для принятия решения;
2) Приостанавливать работу над некоторой подзадачей, когда для решения нет достаточной информации;
3) Переходить с одной подзадачи на другую, возобновляя выполнение приостановленной подзадачи при появлении недостающей информации;
4) Объединять информацию, полученную различными подзадачами.

Введение в теории игр

Теория игр изучает конфликтные ситуации, т.е. ситуации, при которых сталкиваются интересы двух или более сторон, преследующих различные цели. Стороны, участвующие в конфликте, называются **игроками**.

Следует отметить, что под **конфликтом будем понимать не только антагонистическое взаимодействие сторон**. Игроки могут и помогать друг другу, при условии совпадения (возможно, частичного) их целей. Главное — это то, что действия каждого игрока влияют на интересы других игроков и, наоборот, интересы каждого игрока зависят от действий остальных.

Игра всегда ведется по определенным правилам, известным каждому из игроков.
Интересы игроков определяются функциями выигрыша, или платежными функциями.
Игроки в ходе игры последовательно осуществляют некоторые действия, выбирают один из вариантов — делают ходы.
Последовательность ходов **составляет партию**.
Выигрыш игрока определяется не только его ходами, но и ходами других игроков, а также, возможно, и случайными ходами, предусмотренными правилами игры (например, расклад после сдачи в карточной игре либо бросание игральной кости).
Случайность в игровой ситуации может быть также следствием действия так называемой «**природы**», характеризуемой обстоятельствами, не зависящими от субъектов игровой ситуации.

Стратегия — это полный набор указаний, как нужно поступать во всех ситуациях в данной партии.

В теории игр имеется следующее ограничение: не предполагается наличие множества целей — **целевая функция только одна**.
Цель каждого игрока — максимизация своей функции выигрыша, но не минимизация выигрыша остальных игроков

Теория игр — это теория математических моделей принятия решений в условиях неопределенности, когда принимающий решение субъект («игрок») располагает информацией лишь о множестве возможных ситуаций, в одной из которых он в действительности находится, о множестве решений («стратегий»), которые он может принять, и о количественной мере того «выигрыша», который он мог бы получить, выбрав в данной ситуации данную стратегию.

Классификация игр

1) По возможности предварительных (до игры) договоренностей: **кооперативные и некооперативные игры**.
2) По количеству стратегий: **конечные и бесконечные игры**.
3) По наличию элементов случайности при выборе стратегий: **чистые и смешанные стратегии**. (Инструкции в чистых стратегиях не используют случайные числа.)
4) По свойствам функции выигрыша: в зависимости от вида функции — **матричные, биматричные, непрерывные, выпуклые и др**.; в зависимости от характера выигрышей: а) игры с нулевой суммой — игры, в которых сумма выигрышей игроков равна нулю; в случае двух игроков (антагонистические игры) это означает, что выигрыш одного из игроков равен проигрышу другого; б) игры с ненулевой суммой — игры, в которых игроки могут выигрывать (или проигрывать) одновременно.
5) По правилам осуществления ходов. **Статические игры** — игроки могут делать ходы одновременно. Примеры – игра в «орлянку» при одновременном выбрасывании монеты и игра в «чет-нечет» при одновременном выбрасывании пальцев.
Динамические игры — игроки выполняют ходы последовательно. Пример – шахматы.
6) По информации, которой располагают игроки: а) в играх с полной информацией каждый из игроков имеет полную информацию о платежных функциях каждого

игрока. б) в играх с неполной информацией некоторые из игроков не располагают полной информацией о платежных функциях всех других игроков. в) в играх с совершенной информацией каждый игрок знает всю предысторию развития игры, т.е. всю последовательность осуществляемых игроками ходов от начала игры до текущего момента. г) в играх с несовершенной информацией хотя бы один из игроков не располагает полным знанием всей предыстории развития игры.

7) **Бескоалиционные игры** — это класс игр, в которых каждый игрок принимает решение независимо от других игроков, не участвуя ни в каких переговорах и соглашениях с другими игроками. К бескоалиционным играм относятся статистические игры (игры с «природой»), антагонистические игры (игры с противоположными интересами сторон), игры с непротивоположными интересами (в том числе биматричные игры) и др.

Определим **наилучшие стратегии игроков с позиций оценки результатов игры**

1) Ситуация равновесия игры (равновесие по Нэшу) — пара стратегий игроков, отклонение от которых в одиночку невыгодно ни одному из игроков. (Поиск стратегий, образующих ситуацию равновесия, выполняется на основе индивидуального рационального выбора.)

2) Ситуация (пара стратегий игроков) является оптимальной по Парето, если не существует другой ситуации, которая была бы предпочтительнее этой ситуации для всех игроков (т. е. увеличение выигрыша одного из игроков возможно только за счет уменьшения выигрыша другого).

Формы описания игр:

1) **Развернутая форма** описания игры указывает, какие ходы могут делать игроки, какой информацией они располагают, каковы размеры платежей в конце игры. Игра обычно описывается деревом игры; ветви дерева — ходы, которые могут делать игроки в сложившейся обстановке

2) **Нормальная форма игры**

В **бескоалиционной игре целью** каждого игрока является оптимизация индивидуального выигрыша (причем игроки не могут координировать совместно свои стратегии).

Равновесной стратегией игрока в бескоалиционной игре называется такая его стратегия, которая входит хотя бы в одну из равновесных ситуаций игры

Ситуация s называется **ситуацией равновесия по Нэшу** (или равновесной по Нэшу ситуацией), если она приемлема для всех игроков, т. е. для каждого $i \in I$ выполняется $H_i(s|s'_i) \le H_i(s)$ для любых $s \; ' \; i \in S_i$. Очевидно, что ни один из игроков не заинтересован в отклонении от своей стратегии, приводящей к ситуации равновесия, в одиночку

Определение. Стратегия $s_i \in S_i$ игрока i в игре $\Gamma = \{I, S, H\}$ *строго доминируема (строго доминируется)*, если существует другая стратегия $\bar{s}_i \in S_i$ такая, что

$$H_i(s_1, s_2, \ldots, s_{i-1}, \bar{s}_i, s_{i+1}, \ldots, s_n) > H_i(s_1, s_2, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_n) \qquad (2.1)$$

для всех $s_k \in S_k$, $k = 1, 2, \ldots, i-1, i+1, \ldots, n$.

В этом случае стратегия $\bar{s}_i \in S_i$ *строго доминирует* стратегию $s_i \in S_i$.

Свойство. Множество стратегий, выдерживающих такое исключение (оставшихся после удаления) строго доминируемых стратегий, не зависит от последовательности (порядка) исключений.

Замечание. Для слабо доминируемых стратегий данное свойство может не выполняться

Определение. Стратегия si ∈ Si называется доминирующей, если она доминирует (хотя бы слабо) все остальные стратегии игрока i.

Замечание. Наличие доминирующей стратегии у игрока приводит к тому, что он будет пользоваться только этой стратегией независимо от выбора других игроков. Тогда его можно исключить из рассмотрения и перейти к редуцированной игре с меньшим числом участников.

Определение. В бескоалиционной игре $\Gamma = \{I, S, H\}$ ситуация $s^0 = (s_1^0, s_2^0, \ldots, s_n^0)$, называется *оптимальной по Парето*, если не существует ситуации $s = (s_1, s_2, \ldots, s_n) \in S$, для которой имеет место неравенство $H_i(s) \geq H_i(s^0)$ для $\forall i \in I$, причем хотя бы для одного игрока неравенство строгое.

Множество всех ситуаций, оптимальных по Парето, будем обозначать через S^P.

В виду большого разнообразия бескоалиционных игр требуется объединение их в такие классы, внутри которых игры обладают одними и теми же свойствами.

Идея классификации состоит в том, чтобы все множество игр γ разбить на классы по определенному признаку эквивалентности (отношению эквивалентности), причем так, что получающиеся классы попарно не пересекаются и охватывают все элементы множества γ. Если элементы a ∈ γ, b ∈ γ, то запись a ~ b означает,
что элемент a эквивалентен элементу b.

Отношение эквивалентности обладает следующими свойствами: • рефлексивность: a ~ a; • симметричность: если a ~ b, то b ~ a; • транзитивность: если a ~ b и b ~ c, то a ~ c.

Пусть есть две бескоалиционные игры Γ' и Γ'' с одними и теми же множествами игроков и их стратегий, отличающиеся лишь функциями выигрыша: Γ' = {I, S, H'}, Γ'' = {I, S, H''}, и пусть существуют число k > 0 и для каждого игрока число ci , i ∈ I, такие, что в любой

ситуации s ∈ S Hi '(s) = k Hi "(s) + ci . Тогда игры Г' и Г" **называются стратегически эквивалентными**: Г' ~ Г"

Стратегически эквивалентные игры имеют одни и те же **ситуации равновесия**.

Бескоалиционная игра Г = {I, S, H} называется **игрой с постоянной суммой**, если

существует такая C = const, что [illegible] = C для любой ситуации s ∈ S.

Если C = 0, то такая игра называется **игрой с нулевой суммой**.
Всякая бескоалиционная игра с постоянной суммой стратегически эквивалентна некоторой игре с нулевой суммой.

Определение. *Наилучшим ответом* игрока *i* на стратегии остальных игроков s_{-i} в игре $\Gamma = \{I, S, H\}$ называется множество стратегий

$$BR_i(s_{-i}) = \{ \bar{s}_i \in S_i \mid \max_{s_i \in S_i} H_i(s_i, s_{-i}) = H_i(\bar{s_i}, s_{-i})\},$$

где $H_i(s_i, s_{-i}) \equiv H_i(s_1, s_2, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_n)$.

Если игрок *i* выбирает стратегию $s_i \in BR_i(s_{-i})$, то никакое отклонение от нее (при фиксированных стратегиях остальных игроков) не сможет дать ему больший выигрыш.

Ситуация $\bar{s}$ = ($\bar{s}$1 , …, $\bar{s}$n) такая, что $\bar{s}$i ∈ Bri ($\bar{s}$-i) , i = 1, 2, …, n, и **является равновесной по Нэшу**