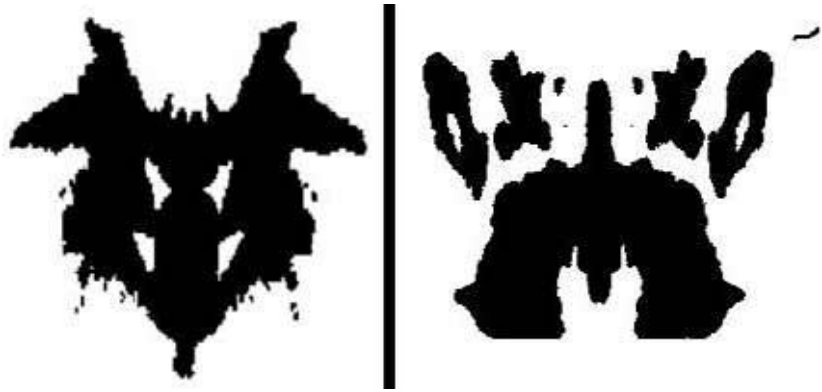
№ 5221

**С. И. Р о д з и н**
**О. Н. Р о д з и н а**

# МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

## Практикум по курсу

## СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА



Практикум представляет собой специальное учебное пособие, содержащее краткое изложение учебного материала; развернутое описание методов решения задач. В Практикум также включены вопросы для самопроверки и тесты. В пособии классифицированы образцы решения типовых задач представления и вывода знаний. Чтобы научиться решать задачи того или иного типа, рекомендуется сначала изучить план решения в общем виде (алгоритм), затем рассмотреть пример реализации плана в конкретном случае и, по аналогии с ним, решить не менее двух задач из числа предлагаемых для самостоятельного решения.

Практикум адресован бакалаврам и магистрам направлений «Программная инженерия», «Математическое обеспечение и администрирование информационных систем», «Информатика и ВТ», «Информационные системы и технологии», изучающих дисциплины «Системы искусственного интеллекта», «Интеллектуальные системы», а также студентам смежных направлений, интересующихся задачами представления и вывода знаний.

Может быть полезен аспирантам, занимающимся проблематикой системного анализа, теоретической информатики, искусственного интеллекта, автоматизации управления, моделирования, системами автоматизации проектирования.

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Южный федеральный университет»**

**С.И. Родзин**

**О.Н. Родзина**

# МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Практикум по курсу

Системы искусственного интеллекта

Учебное пособие

Таганрог 2014

УДК 519.6(075.8)

**Рецензенты:**

доктор технических наук, профессор, заведующий кафедрой компьютерных образовательных технологий Санкт-Петербургского государственного университета информационных технологий, механики и оптики **Лисицына Л.С**.;

доктор технических наук, профессор Ростовского государственного университета путей сообщения **Ковалев С.М**.

**Родзин С.И., Родзина О.Н.** Модели представления знаний. Практикум по курсу «Системы искусственного интеллекта»: Учебное пособие. – Таганрог: Изд-во ЮФУ, 2014. – 150 с.

Цель практикума в том, чтобы помочь студентам овладеть навыками практического представления знаний в интеллектуальных системах[1]. Пособие содержит материалы по разделам дисциплины «Системы искусственного интеллекта», которые рассматриваются на практических занятиях, и основано на образовательном контенте, используемом авторами в учебном процессе ЮФУ в ходе профессиональной подготовки бакалавров, магистров и аспирантов.

Практикум включает описание четырех практических занятий по темам: «Продукционные модели», «Модели семантических сетей и фреймов», «Нейросетевые модели и эволюционные вычисления», «Байесовские и нечеткие модели». Включает необходимые для решения задач искусственного интеллекта теоретические сведения, а также описание процесса их решения. В пособии приведены как оригинальные задачи, так и представленные в учебной литературе.

Предназначен для студентов направления «Программная инженерия», «Математическое обеспечение и администрирование информационных систем», «Информатика и вычислительная техника», изучающих дисциплину «Системы искусственного интеллекта», «Интеллектуальные системы», а также для студентов смежных направлений, интересующихся решением интеллектуальных задач.

Табл. 33. Ил. 36. Библиогр.: 13 назв.



---

[1] Практикум издается при частичной финансовой поддержке грантов РФФИ №№ 13-07-00204, 13-01-00475).

**Содержание**

# Введение

Для применения на практике современных технологий искусственного интеллекта, включающих множество программно-аппаратных средств, экспертных систем, необходимо обладать элементарными знаниями в области интеллектуальных систем. Интеллектуальная система – это система, оперирующая знаниями о проблемной области. Без базы знаний интеллектуальных систем не существует. Модели представления знаний – это одно из важнейших направлений исследований в области искусственного интеллекта. Проблема представления знаний заключается в несоответствии между сведениями о зависимостях данной предметной области, имеющимися у эксперта, методами, используемыми им при решении задач, и возможностями формального и однозначного представления такой информации в ЭВМ. Для формализации знаний разрабатываются специальные модели представления знаний и языки для описания знаний, выделяются различные типы знаний.

Одна из целей данного пособия – помочь будущим специалистам в IT-сфере овладеть теоретическими и практическими знаниями и навыками представления знаний в интеллектуальных системах. В настоящее время разработаны две группы моделей представления знаний: теоретически обоснованные и эмпирические.

Первая группа моделей представлена моделями, основанными на исчислении высказываний, исчислении предикатов, формальных грамматиках, тензорных и алгебраических моделях. До настоящего времени с помощью этих моделей удавалось решить только сравнительно простые задачи из узкой предметной области.

Вторая группа моделей, называемых эмпирическими, основана на изучении принципов организации человеческой памяти и моделировании механизмов решения задач человеком. Наиболее проработанными из этих моделей и получившими широкую известность являются продукционные модели, семантические сети, фреймы, нейросети, нечеткие модели. Особенностью моделей этой группы является широкое использование эвристик, что в каждом случае требует доказательства правильности получаемых решений.

В практикуме представлены задачи и методы, позволяющие сформировать базовые навыки при изучении продукционных, сетевых и фреймовых моделей представления знаний, обучения нейросетей, анализа НЕ-факторов знаний и применения нечетких вычислений.

Практикум рекомендуется использовать совместно с учебным пособием «Искусственный интеллект» [1] и литературой, указанной в библиографическом списке.

Предполагается, что читатели знакомы с курсами алгебры, дискретной математики, моделирования, теории принятия решений. А вообще, следуйте рецепту великого Даламбера: «Идите вперед, а понимание придет потом!».

Практикум подготовлен на основе лекций по курсу «Системы искусственного интеллекта», читаемых на кафедре математического обеспечения и применения ЭВМ. По каждой из четырех тем приводится краткое теоретическое описание, примеры задач, вопросы, тесты и упражнения для самостоятельной работы. Чтобы научиться решать задачи того или иного типа, рекомендуется сначала изучить план решения, затем рассмотреть пример реализации плана в конкретном случае и, по аналогии с ним, решить несколько задач из числа предлагаемых для самостоятельной работы.

# 1. ПРОДУКЦИОННЫЕ МОДЕЛИ

## Основные определения по теме

**Ассоциативная память** – тип памяти с адресацией по содержанию, которая способна «вспомнить» аналогичный образец при предъявлении некоторого объекта или восстановить полную ситуацию по частичной информации о ней.

**База знаний** – основной компонент интеллектуальной системы, содержащий экспертные знания о предметной области. Собрание правил, эвристик и процедур, организованных различными моделями представления знаний.

**Вывод** – получение новых информационных единиц из ранее известных. Частным случаем является логический вывод.

**Гипотеза Ньюэлла**. Необходимое и достаточное условие интеллектуальности системы – универсальность формальных манипуляций над конкретными символами. Иными словами, достаточно прочесть строку символов, разделить её на компоненты и переупорядочить, добавив или удалив какие-то символы без учёта смысла (семантики) логических и математических символов.

**Дерево решений** – иерархическая древовидная структура, представляющая собой совокупность решений. Деревья решений разбивают данные на группы на основе значений переменных, в результате чего возникает иерархия правил «ЕСЛИ (антецедент) ТО (консеквент)», которые классифицируют данные.

**Дизъюнкт Хорна** – предложение исчисления высказываний, записываемое в виде выражения «Q, если P1&P2&…&Pk – истина»; он имеет вид ¬P1 ¬P2 … ¬Pk Q и является основной конструкцией языка Пролог.

**Знак** – информационная единица, структура которой представляет собой треугольник Фреге.

**Знания** – 1) в узком смысле это обработанная информация, хранимая в базе знаний и отражающая убеждения специалистов (экспертов) в определённой предметной области, а также образующая целостное описание некоторой проблемы с доступной степенью детализации; 2) в широком смысле концепция знаний объединяет черты процедурной и декларативной информаций и трактуется как

обоснованное истинное убеждение, проверенный практикой результат познания действительности. Отличительные характеристики знаний: внутренняя интерпретируемость, структурированность, связность, семантическое пространство с метрикой, активность. Метазнания – знания о знаниях.

**Консеквент** – действие в правой части продукционного правила, которое должно быть совершено над базой данных в случае выполнения соответствующих условий в левой части правила.

**Логический вывод** – 1) формальный метод доказательства путём логических рассуждений; 2) в экспертных системах новое заключение, полученное по определенным правилам вывода из фактов, имеющихся в базе знаний. Различают абдуктивный, дедуктивный, индуктивный, нечеткий и др. выводы.

**Модель представления знаний** – это структуры предметных знаний в интеллектуальной системе с целью облегчения поиска решения задачи. Различают декларативные и процедурные модели знаний. Декларативные модели знаний, явно не содержащие описания выполняемых процедур, обычно подразделяются на продукционные, сетевые (семантические сети) и фреймы. В процедурных моделях семантика непосредственно заложена в описание базы знаний.

**Обработка знаний** – описание смыслового содержания задач в форме, которая гарантирует их правильную обработку формальными методами.

**Онтология** – база, знания в которой могут читаться, пониматься, отчуждаться от разработчика и/или физически разделяться пользователями. Формальной моделью онтологии является тройка вида O=<X, R, F>, где X – множество понятий предметной области, которую представляет онтология; R – множество отношений между понятиями; F – множество функций интерпретации, заданных на понятиях или отношениях онтологии.

**Поиск** – движение в структурированном пространстве от одних узлов этого пространства к другим. Если поиск является целенаправленным, то задаётся множество начальных узлов, с которых поиск может начинаться, и множество конечных узлов, при достижении которых поиск прекращается. Движение по структуре поискового пространства определяется стратегией поиска.

**Поиск по дереву решений** – совокупность различных методов организованного процесса поиска решений (в глубину, в ширину, с возвратом и т.д.) в пространстве состояний для эффективного нахождения приемлемого решения.

**Правило modus ponens** – если P – истина и из P следует Q (что равносильно ¬P Q), то Q тоже истина, т.е. если имеются две пары P и ¬P Q, то Q (резольвента) выводится путём удаления литер P и ¬P.

**Продукционная (каноническая) система Поста** – система, включающая алфавит, из символов которого формируются строки; множество строк, которые рассматриваются как аксиомы, а также множество грамматических правил манипулирования строками символов (правила порождений). Система должна обладать разрешимостью, непротиворечивостью и полнотой. В продукционной системе знания представляются совокупностью правил вывода вида «ЕСЛИ (условия) ТО (действия)». Различают продукционные системы с прямым и обратным логическими выводами. Прямой вывод означает, что рассуждения строят, отталкиваясь от выполненных условий (антецедент), к заключениям (консеквент), вытекающим из этих условий. Обратный вывод означает, что рассуждения строят, отталкиваясь от заданной цели, к условиям, при которых возможно ее достижение. Продукционные системы включают базу правил, состоящую из набора правил вывода (продукций), рабочей памяти (база данных, которая определяет текущее состояние задачи и содержит множество фактов, описание цели и промежуточные результаты) и интерпретатора правил, который осуществляет логический вывод на основании фактов и решает, когда надлежит применить каждое из правил. Одним из языков продукционного программирования является CLIPS.

**Треугольник Фреге** – базовая структура в прикладной семиотике, представляющая знак как триединство понятия объекта, его имени и знаний о нём.

**Формальная система** – модель, лежащая в основе многих математических теорий, состоящая из множества базовых элементов, синтаксических правил, аксиом и правил вывода.

**Экспертная система** – интеллектуальная система, основанная на знаниях и использующая логику эксперта для эффективного решения

задач в узкой предметной области. Такие системы представляют знания символически, исследуют и объясняют свои рассуждения. В динамических экспертных системах, в отличие от статических систем, информация изменяется в режиме реального времени решения задачи.

**Эпистемология** – наука о знании в рамках философии. Философы, занимающиеся данной проблематикой, решают вопросы, схожие с теми, которые решаются в искусственном интеллекте (ИИ): как лучше представлять и использовать знания и информацию.

**Язык представления знаний** – способ описания моделей представления знаний. На сегодняшний день известны языки для модели знаний в виде фреймов (LISP, FRL, KRL и др.), а также ряд продукционных языков.

**CLIPS** (от англ. *C Language Integrated Production System*) – программная среда для разработки экспертных систем. CLIPS является продукционной системой, написанной на языке Си. Версии CLIPS для Windows (clipswin.exe) не поддерживают кириллицу. Однако открытость исходных кодов CLIPS позволяет исправить эту ситуацию.

## Продукционные правила

В инженерии знаний принято различать «жесткие» и «мягкие» модели представления знаний. К «мягким» моделям относятся нейросети, эволюционные алгоритмы, нечеткая логика и их гибриды, к «жестким» моделям – продукционные правила, семантические сети, фреймы и различные логические модели (логика высказываний и предикатов).

В области ИИ и психологии утверждение, что разумное поведение направляется правилами, превратилось в аксиому. Разумное поведение, такое как правильная речь, представляет собой процесс, регламентируемый правилами, которые мы даже не можем точно сформулировать. В ИИ правила играют даже более явно выраженную роль. В теории автоматов, формальной грамматике, программировании правила используются как формализм для представления связей между данными и действиями, которые система должна предпринять в ответ: «условие – действия», «ситуация – действия». В экспертных системах (ЭС) правила указывают, что нужно сделать, чтобы перейти из текущего состояния проблемы к

представлению, более близкому к решению. В СИИ продукционные системы являются одними из наиболее оригинальных моделей, основанных на знаниях.

Для реализации правил логического вывода необходим специальный язык. Программная реализация вывода возможна на любом алгоязыке, однако это зачастую сложно. Программист должен думать о природе решаемой задачи, а не о деталях ее реализации, поэтому нужна универсальная абстрактная языковая форма для описания правил. В самом общем виде продукция представляется выражением вида

$$(W, U, P, A \rightarrow B, C), \qquad (1)$$

где A→B – ядро продукции, соответствующее правилу «если (условия A), то (действия B)»; W – сфера применения продукции в классе ситуаций некоторой предметной области; U – предусловие, содержащее информацию об истинности продукции (коэффициент уверенности), ее значимости и т.п.; P – внешнее, не входящее в A условие, разрешающее применять данную продукцию; C – постусловие, определяющее изменения, которые возможно надо внести в продукционную систему после выполнения данной продукции. Обязательной частью продукции является только ядро, трактовка которого может быть разной: если A истинно, то B также истинно; если A содержится в БЗ, то B следует включить в БЗ; если A истинно, то следует выполнить B и т.д.

Продукционное программирование (ПП в ИИ раньше считалось разновидностью эвристического программирования (метод ветвей и границ, динамическое программирование, методы лабиринтного поиска и т.д.), однако сейчас область его применения расширилась до промышленных ЭС, систем автоматического доказательства теорем и т.д. ПП привлекает разработчиков своей наглядностью, модульностью, легкостью внесения дополнений, простотой механизма логического вывода. Некоторыми недостатками ПП считается сложность проверки на непротиворечивость системы продукций, а также недетерминированность (неоднозначность выбора из фронта активизируемых продукций) и трудности с функционированием системы при большом числе (свыше 1000) продукций.

В БЗ, состоящих из множества продукционных правил вида (1), под условием понимается некоторое предложение-образец, по которому осуществляется поиск в БЗ, а под действием – действия, выполняемые при успешном исходе поиска. Программа, управляющая перебором правил, называется машиной вывода. Вывод бывает прямой (от данных к поиску цели) или обратный (от цели для ее подтверждения – к данным). Данные – это исходные факты, на основании которых запускается машина вывода – программа, перебирающая правила из БЗ.

***Пример***. Имеется фрагмент БЗ из двух правил:

П1: ЕСЛИ "отдых – летом" и "человек – активный", ТО "ехать в горы".

П2: ЕСЛИ "любит солнце", ТО "отдых летом".

Предположим, в систему поступили данные – "человек активный" и "любит солнце".

*Прямой вывод* – исходя из данных, получить ответ.

1-й проход.

Шаг 1. Пробуем П1, не работает (не хватает данных "отдых – летом").

Шаг 2. Пробуем П2, работает, в базу поступает факт "отдых – летом".

2-й проход.

Шаг 3. Пробуем П1, работает, активируется цель "ехать в горы", которая и выступает как совет, который дает ЭС.

*Обратный вывод* – подтвердить выбранную цель при помощи имеющихся правил и данных.

1-й проход.

Шаг 1. Цель – "ехать в горы". Пробуем П1 – данных "отдых – летом" нет, они становятся новой целью, и ищется правило, где она в правой части.

Шаг 2. Цель "отдых – летом": правило П2 подтверждает цель и активирует ее.

2–й проход.

Шаг 3. Пробуем П1, подтверждается искомая цель.

## Классификация продукционных правил

Ядра продукции (А→В) делятся на детерминированные и недетерминированные. В детерминированных ядрах при выполнении условия А действие В выполняется обязательно; в недетерминированных ядрах выполнение условий А необязательно приводит к актуализации правила и выполнению действия В. Иначе говоря, в детерминированных ядрах импликация реализуется с необходимостью, а в недетерминированных – с возможностью. Интерпретация ядра в этом случае может, например, выглядеть так:

ЕСЛИ А, ТО *возможно* В.

Например, если задана вероятность выполнения В при актуализации А, то продукция может быть такой:

ЕСЛИ А, ТО *с вероятностью р реализовать* В.

Оценка реализации ядра может быть лингвистической, связанной с понятием терм-множества лингвистической переменной, например:

ЕСЛИ А, ТО *с большей долей уверенности* В.

Возможны иные способы реализации ядра.

Детерминированные продукции могут быть однозначными и альтернативными. Во втором случае в правой части ядра указываются альтернативные возможности выбора, которые оцениваются специальными весами выбора. В качестве таких весов могут использоваться вероятностные оценки, лингвистические оценки, экспертные оценки и т.п.

Особым типом являются прогнозирующие продукции, в которых описываются последствия, ожидаемые при актуализации А, например:

ЕСЛИ А, ТО *с вероятностью р можно ожидать* В.

Классификацию ядер продукции можно провести, опираясь на типовую схему СИИ (рис. 1).

Если *x* и *y* обозначают любой из блоков рисунка (О, Д, З, Л), то ядро А*x*→В*y* означает, что информация об А берется из блока *x*, а результат срабатывания продукции В посылает в блок *y*.

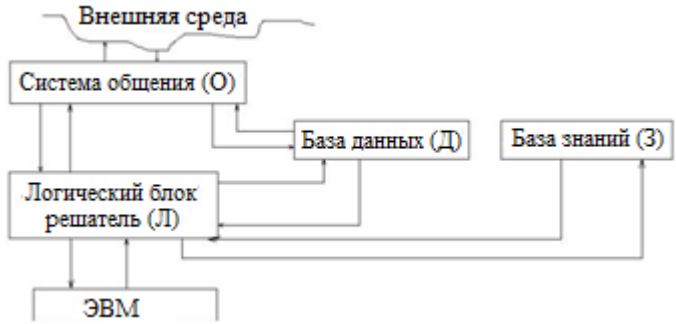



Рис. 1. Схема СИИ

Комбинации *x* и *y*, осмысленные с точки зрения СИИ, отмечены в табл. 1 знаком "+".

Таблица 1

| **В** / **А** | *О* | *Д* | *З* | *Л* |
|---|---|---|---|---|
| *О* | | + | | + |
| *Д* | + | + | + | + |
| *З* | + | | + | + |
| *Л* | + | | + | + |

Рассмотрим, например, часто встречающийся тип продукции А*З*→В*З*. В этом случае А*З* и В*З* представляют собой некоторые фрагменты информации, хранящейся в БЗ, в виде продукционных правил. Тогда смысл продукции А*З*→В*З* состоит в замене одного фрагмента БЗ другим. При поиске в базе знаний А играет роль образца, а процедура такого поиска называется поиском по образцу.

Продукция А*Д*→В*З* может соответствовать процедуре нахождения закономерностей по эмпирическим данным. Логический блок на основании просмотра и анализа данных выдвигает гипотезы о наличии закономерностей и, убедившись в их приемлемости и достаточной обоснованности, записывает их в БЗ. Аналогично можно интерпретировать другие типы продукций из табл. 1.

Представлению знаний присущ пассивный аспект: книга, таблица, заполненная информацией память. В ИИ подчеркивается активный аспект операции представления знаний, позволяющей не только запоминать, но и извлекать полученные знания путем построения

логических рассуждений и программирования на их основе языковыми средствами информатики. Для описания логических рассуждений используется символический язык математической логики, который одновременно близок к обычному языку и к языкам программирования. Поэтому математическая логика лежит в основе различных представлений знаний в ИИ. В частности, теоретической основой продукционного программирования является каноническая система Поста.

## Каноническая система Поста

В теории автоматов, формальной грамматике, программировании важную роль играет понятие «порождающих правил». Порождения – это грамматические правила манипулирования строками символов, поэтому их иногда называют правилами переписывания (*Rewrite*). Понятие «порождение» взято из теоретической лингвистики (там оно имеет вид: $S \rightarrow NP + VP$; означает один из способов сформировать предложение *S*: взять существительное *NP* и добавить к нему глагол *VP*).

Пост изучил свойства систем правил, базирующихся на порождениях. Эти правила он назвал каноническими системами. Каноническая система – это разновидность формальной системы (ФС = < Алфавит, Синтаксические правила, Аксиомы из общезначимых формул, Правила вывода новых формул >), которая должна обладать разрешимостью, непротиворечивостью и полнотой.

Канонические системы (КС) включают следующие компоненты:

1. Алфавит А, из символов которого формируются строки.

2. Множество строк, которые рассматриваются как аксиомы.

3. Множество порождений в форме $a_1\$_1...a_m\$_m \rightarrow b_1£_1...b_n£_n$, где $a_i$ и $b_j$ – фиксированные строки (часто нули), $ и £ являются переменными строками (могут быть нулевыми), причем при порождении каждое $\$_i$ заменяется определенным $£_j$.

Пост доказал следующие свойства КС.

1. Некоторое слово В выводимо из другого слова А, если его можно получить из А за конечное число применений правил порождения.

2. Слово B доказуемо в КС, если для него найдётся аксиома C, из которой это слово выводимо.

3. Множеством теорем КС называется множество слов в ее алфавите, доказуемых или порождённых ее аксиомами.

4. КС с продукциями вида $a_1\$a_2 \rightarrow a_1£a_2$ лежат в основе грамматики формальных языков, из которой согласно Н.Хомскому построены все языки программирования.

Поясним это на примере. Пусть A = {a, b, c} – алфавит системы, { a, b, c, aa, bb, cc } – аксиомы. Палиндром (перевертень) – это текст, одинаково читающийся от начала к концу и от конца к началу (А.Фет: «А роза упала на лапу Азора». Палиндромы применяются в экспериментальной поэзии В. Хлебникова и др.). Тогда следующие правила (порождения) сгенерируют все палиндромы в данном алфавите:

(P1) $\$ \rightarrow a\$a$, (P2) $\$ \rightarrow b\$b$, (P3) $\$ \rightarrow c\$c$.

Более того, можно проследить применение правил, которые должны привести к росту определенного палиндрома. Например, чтобы сгенерировать *bacab*, нужно применить P1 к аксиоме *c*, а затем P2 – к результату, т.е. из *c* можно вывести теорему *aca*, добавить ее к имеющимся аксиомам, а затем вывести новую теорему *bacab*. Обратите внимание, что эта последовательность порождений не обладает свойством коммутативности, т.е. если применять указанные правила в ином порядке, то получится другой результат.

На первый взгляд КС довольно тривиальны: все, что можно сделать – преобразовать одну строку символов в другую. Но если вдуматься, то любое математическое или логическое исчисление, в конце концов, сводится к набору правил манипулирования символами (Гипотеза Ньюэлла). Мы упускаем это из виду. Почему? – Потому что для нас важен смысл (семантика) логических и математических символов, чего не скажешь о строках типа *abcba*.

Отсюда следует, что любая формальная система может рассматриваться как КС путем тривиальной оговорки: такая система может нуждаться еще в дополнительном алфавите или буквах, используемых в качестве знаков пунктуации в сложных доказательствах.

Таким образом, чтобы выполнить любую эффективную процедуру, достаточно способности прочесть строку символов, разделить ее на компоненты и переупорядочить, добавив или удалив какие-то символы.

## Пример построения продукционной модели

Чтобы построить продукционную модель, необходимо выполнить следующие шаги:

1. Определить целевые действия задачи (они являются решениями).

2. Определить промежуточные действия между начальным и конечным состояниями.

3. Определить условия для каждого действия, при котором его целесообразно и возможно выполнить, а также порядок выполнения действий.

4. Добавить конкретики при необходимости, исходя из поставленной задачи.

5. Преобразовать полученный порядок действий и соответствующие им условия в продукции.

6. Проверить продукции на непротиворечивость, записав цепочки продукций и явно проследив связи между ними.

Двигаться при построении продукционной модели можно от результата к начальному состоянию либо от начального состояния к результату.

***Пример***. Построить продукционную модель представления знаний в предметной области «Посещение кафе».

*Решение***.** Для данного примера шаги построения продукционной модели можно сформулировать следующим образом.

1. Обязательное действие, выполняемое в кафе – поглощение пищи и ее оплата. Следовательно, есть уже два целевых действия «съесть пищу» и «оплатить», которые взаимосвязаны и следуют друг за другом.

2. Прежде чем что-либо съесть в кафе, туда нужно прийти, дождаться официанта и сделать заказ. Кроме того, нужно выбрать, в

какое именно кафе пойти. Значит, цепочка промежуточных действий: «выбор кафе и пути к нему», «сделать заказ официанту».

3. Прежде чем идти в кафе, необходимо убедиться, что есть необходимая сумма денег. Выбор кафе может обуславливаться многими причинами, мы выберем территориальный признак – к какому ближе, в то и идем. В разных кафе работают разные люди, поэтому в зависимости от выбора кафе, официанты будут разные. Кроме того, разные кафе специализируются на разных кухнях, поэтому заказанные блюда будут в разных кафе отличаться. Значит вначале идут действия, позволяющие выбрать кафе, затем характеризующие кафе, а уже после заказ, еда и оплата заказа.

4. Пусть в задаче будут рассматриваться два кафе: «2 фунта» и «Осака». Первый – паб и заказы приносят быстрее, чем во втором, второй – пиццерия. В первом работает официант Борис, а во втором официантка Юля. Антон – это клиент.

5. Преобразуем указанные действия и соответствующие им условия в продукции «Если, то»:

– Если субъект хочет кушать и у субъекта есть достаточная сумма денег, то субъект может пойти в кафе.

– Если субъект ближе к кафе «2 фунта», чем к кафе «Осака» и субъект может пойти в кафе, то субъект идет в кафе «2 фунта».

– Если субъект ближе к кафе «Осака», чем к кафе «2 фунта» и субъект может пойти в кафе, то субъект идет в кафе «Осака».

– Если субъект идет в кафе «Осака» и в нем работает официант Юля, то у субъекта принимает заказ Юля.

– Если субъект идет в кафе «2 фунта» и в нем работает официант Борис, то у субъекта принимает заказ Борис.

– Если субъект выбрал блюда и у субъекта принимает заказ Юля, то заказ принесут через 20 мин.

– Если субъект выбрал блюда и у субъекта принимает заказ Борис, то заказ принесут через 10 мин.

– Если заказ принесут через 20 мин или заказ принесут через 10 мин, то субъект может кушать.

– Если субъект может кушать, то после еды субъект должен оплатить заказ.

Введем обозначения для фактов (Ф), действий (Д) и продукций (П), тогда:

Субъект = Антон;
Ф1 = субъект хочет есть;
Ф2 = у субъекта есть достаточная сумма денег;
Ф3 = субъект ближе к кафе «2 фунта», чем к «Осака»;
Ф4 = в кафе «Осака» работает официант Юля;
Ф5 = в кафе «2 фунта» работает официант Борис;
Ф6 = субъект выбрал блюда;
Д1 = субъект может пойти в кафе;
Д2 = субъект идет в кафе «2 фунта»;
Д3 = субъект идет в кафе «Осака»;
Д4 = у субъекта принимает заказ Юля;
Д5 = у субъекта принимает заказ Борис;
Д6 = заказ принесут через 20 мин.
Д7 = заказ принесут через 10 мин.
Д8 = после еды субъект должен оплатить заказ.
П1: (Д6 или Д7) → Д8;
П2: (Д5) → Д7;
П3: (Д4) → Д6;
П4: (Д2 и Ф5) → Д5;
П5: (Д3 и Ф4) → Д4;
П6: (не Ф3 и Д1) → Д3;
П7: (Ф3 и Д1) → Д2;
П8: (Ф1 и Ф2) → Д1.

Для продукций можно установить следующий приоритет (чем выше приоритет, тем раньше проверяется правило): П8 – 1, П6 = П7 = 2, П4 = П5 = 3, П2 = П3 = 4, П1 = 5.

Отобразим взаимосвязи продукций на графе (рис. 2).

***Пример***. Построить продукционную модель представления знаний в предметной области «Автозаправка».

*Решение***.** Для данного примера шаги построения продукционной модели можно сформулировать следующим образом.

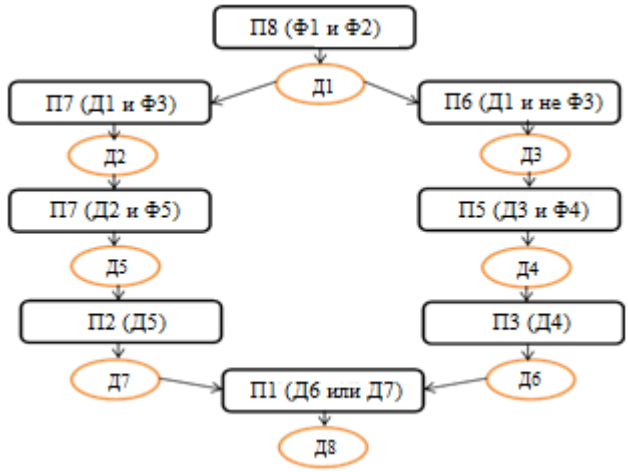



Рис. 2. Взаимосвязь продукций предметной области «Кафе»

1. Обязательное действие, выполняемое на автозаправке, – получение топлива и его оплата. Значит, есть уже два целевых действия «получить» и «оплатить», которые взаимосвязаны и следуют друг за другом.

2. Прежде чем получить топливо на автозаправке, пользователю необходимо выбрать топливо и заказать его заправщику. Кроме того, нужно выбрать автозаправку и приехать на нее. Значит, цепочка промежуточных действий: «выбор автозаправки и пути к ней», «выбор топлива и его заказ заправщику».

3. Прежде чем ехать на автозаправку, необходимо убедиться, что есть необходимая сумма денег. Выбор автозаправки может обуславливаться многими причинами, мы выбираем те автозаправки, владельцы которых не являются спонсорами противников любимой футбольной команды и к которым можно доехать на имеющемся топливе. Следовательно, вначале идут действия, позволяющие выбрать автозаправку, затем уточняются характеристики автозаправки, а уже после осуществляется заказ, получение топлива и его оплата.

4. Пусть в задаче рассматриваются две автозаправки: «Лукойл» и «Кобарт». В первой обслуживают быстрее, чем во второй, но владельцы первой – спонсоры футбольного клуба «Спартак», а клиент

Сергей является болельщиком футбольного клуба ЦСКА, главным противником которого является «Спартак». Владельцы «Кобарт» не являются ничьим спонсором. На автозаправке «Лукойл» работает Иван, на автозаправке «Кобарт» – Федор.

5. Преобразуем указанные действия и соответствующие им условия в продукции «Если, то»:

– Если субъекту необходимо топливо и у него есть достаточная сумма денег, то субъект может поехать на автозаправку.

– Если у субъекта достаточно топлива, чтобы доехать на автозаправку «Кобарт», и субъект может поехать на автозаправку, то субъект едет на автозаправку «Кобарт».

– Если у субъекта недостаточно топлива, чтобы доехать на автозаправку «Кобарт», и субъект может поехать на автозаправку, то субъект едет на автозаправку «Лукойл».

– Если субъект едет на автозаправку «Кобарт» и на автозаправке «Кобарт» работает Федор, то субъекта будет обслуживать Федор.

– Если субъект едет на автозаправку «Лукойл» и на автозаправке «Лукойл» работает Иван, то субъекта будет обслуживать Иван.

– Если субъект оплатил топливо и субъекта обслуживает Федор, то он обслужит его за 8 минут.

– Если субъект оплатил топливо и субъекта обслуживает Иван, то он обслужит его за 5 минут.

– Если субъекта обслужат за 8 минут или субъекта обслужат за 5 минут, то субъект получает топливо.

Введем обозначения для фактов (Ф), действий (Д) и продукций (П), тогда:

Субъект = Сергей;

Ф1 = субъекту необходимо топливо;

Ф2 = у субъекта есть достаточная сумма денег;

Ф3 = у субъекта достаточно топлива, чтобы доехать на автозаправку «Кобарт»;

Ф4 = на автозаправке «Кобарт» работает Федор;

Ф5 = на автозаправке «Лукойл» работает Иван;

Д1 = субъект может поехать на автозаправку;

Д2 = субъект едет на автозаправку «Кобарт»;

Д3 = субъект едет на автозаправку «Лукойл»;

Д4 = субъекта обслуживает Федор;
Д5 = субъекта обслуживает Иван;
Д6 = субъекта обслужат за 8 минут;
Д7 = субъекта обслужат за 5 минут;
Д8 = после получения топлива субъект должен оплатить заказ;
П1: (Ф1 и Ф2) → Д1;
П2: (Д1 и Ф3) → Д2;
П3: (Ф3 и не Ф3) → Д3;
П4: (Д2 и Ф4) → Д4;
П5: (Д3 и Ф5) → Д5;
П6: (Д4) → Д6;
П7: (Д5) → Д7;
П8: (Д6 или Д7) → Д8.

Для продукций можно установить следующий приоритет (чем выше приоритет, тем раньше проверяется правило): П1 – 5, П2 = П3 = 4, П4 = П5 = 3, П6 = П7 = 2, П8 = 1.

Отобразим взаимосвязи продукций на графе (рис. 3).

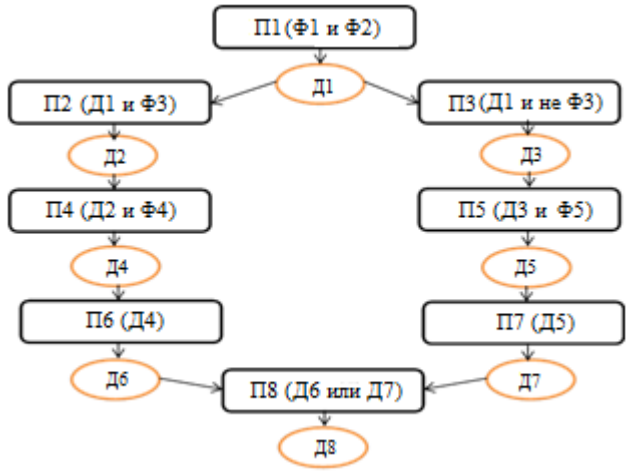



Рис. 3. Взаимосвязь продукций предметной области «Автозаправка»

***Пример***. Построить продукционную модель представления знаний в предметной области «Администрирование информационных систем».

*Решение***.** Для данного примера шаги построения продукционной модели можно сформулировать следующим образом.

1. Обязательное действие, выполняемое администратором ИС, – выявление и устранение ошибок.

2. Прежде чем выявить проблему и устранить ошибку, необходимо провести различные тесты.

3. Прежде чем устранять проблему, необходимо убедиться, что таковая имеется. После этого следует провести действия по проведению тестов для классификации ошибки, выявить проблему и устранить ее. Значит, вначале идут действия, позволяющие убедиться в наличии проблемы, затем проводятся тесты, а уже после выявляются и устраняются ошибки.

4. Пусть в задаче будут рассматриваться две проблемы: ошибки в программном обеспечении (ПО) и ошибки в аппаратном обеспечении (АО). В зависимости от этого будут проводиться различные тесты.

5. Преобразуем указанные действия и соответствующие им условия в продукции «Если, то»:

– Если поступила жалоба на работу системы, то субъект начинает поиск проблем.

– Если субъект обнаруживает проблему, то субъект начинает диагностику системы.

– Если субъект начинает диагностику и система запускается, то субъект проводит тест ПО.

– Если субъект начинает диагностику и система не запускается, то субъект проводит тест АО.

– Если субъект проблему не обнаружил, то субъект проводит инструктаж с персоналом.

– Если у субъекта есть возможность блочного теста, то он проводит блочный тест.

– Если у субъекта нет возможности блочного теста, то он отправляет аппаратное обеспечение системы на гарантийный ремонт.

– Если субъект выявил проблему в операционной системе, то субъект переустанавливает ОС.

– Если субъект не выявил проблему в ОС, то субъект проводит тест прикладного программного обеспечения.

– Если субъект выявил проблему в оперативной памяти, то субъект заменяет оперативную память.

– Если субъект не выявил проблему в оперативной памяти, то субъект проводит тест материнской платы.

– Если субъект выявил проблему в материнской плате, то субъект выполняет блочную замену.

– Если субъект не выявил проблему в материнской плате, то субъект тестирует устройства ввода/вывода.

– Если субъект выявил проблему в устройствах ввода/вывода, то субъект выполняет замену устройств.

– Если субъект не выявил проблему в устройствах ввода/вывода, то субъект проблему на месте не обнаружил. Требуется гарантийный ремонт аппаратного обеспечения.

Введем обозначения для фактов (Ф), действий (Д) и продукций (П), тогда:

Субъект = администратор Николай;
Ф1 = поступила жалоба на работу системы;
Ф2 = есть неполадки;
Ф3 = проблема с АО;
Ф4 = проблема с ПО;
Ф5 = возможность блочного ремонта;
Ф6 = проблема с оперативной памятью;
Ф7 = проблема в материнской плате;
Ф8 = проблема в устройстве ввода/вывода;
Ф9 = проблема с операционной системой;
Ф10 = проблема с прикладным программным обеспечением;
Д1 = субъект проводит проверку работоспособности системы;
Д2 = субъект проводит тест АО;
Д3 = субъект проводит тест ПО;
Д4 = субъект проверяет возможность блочного теста;
Д5 = субъект проводит инструктаж персонала;
Д6 = субъект проводит тест операционной системы;
Д7 = субъект проводит тест прикладных программ;
Д8 = субъект переустанавливает прикладное ПО;
Д9 = субъект переустанавливает ОС;
Д10 = субъект тестирует оперативную память;

Д11 = субъект отправляет систему на гарантийный ремонт;
Д12 = субъект меняет оперативную память;
Д13 = субъект тестирует материнскую плату;
Д14 = субъект меняет материнскую плату;
Д15 = субъект тестирует устройства ввода/вывода;
Д16 = субъект меняет устройства ввода/вывода;
П1: (Ф1) → Д1;
П2: (Д1 и Ф2) → Д2;
П3: (Д4 и не Ф5) → Д11.
П4: (Д2 и Ф3) → Д4;
П5: (Д2 и не Ф3) → Д3;
П6: (Д3 и Ф4) → Д6;
П7: (Д3 и не Ф4) → Д5;
П8: (Д6 и не Ф9) → Д9;
П9: (Д7 и не Ф10) → Д8;
П10: (Д6 и Ф9) → Д9;
П11: (Д4 и Ф5) → Д10;
П12: (Д10 и Ф5) → Д12;
П13: (Д10 и не Ф5) → Д13;
П14: (Д13 и Ф7) → Д14;
П15: (Д13 и не Ф7) → Д15;
П16: (Д15 и Ф8) → Д16;
П17: (Д15 и не Ф8) → Д11.
Отобразим взаимосвязи продукций на графе (рис. 4).

***Пример 3***. Построить продукционную модель представления знаний в предметной области «Больница» (диагностики заболеваний с синдромом острого живота).

*Решение***.** Для данного примера шаги построения продукционной модели можно сформулировать следующим образом.

1. Обязательным действием при диагностике заболеваний, входящих в состав синдрома острого живота, является опрос больного. Значит, есть уже два целевых действия – «опросить больного» и «поставить диагноз», которые взаимосвязаны и следуют друг за другом.

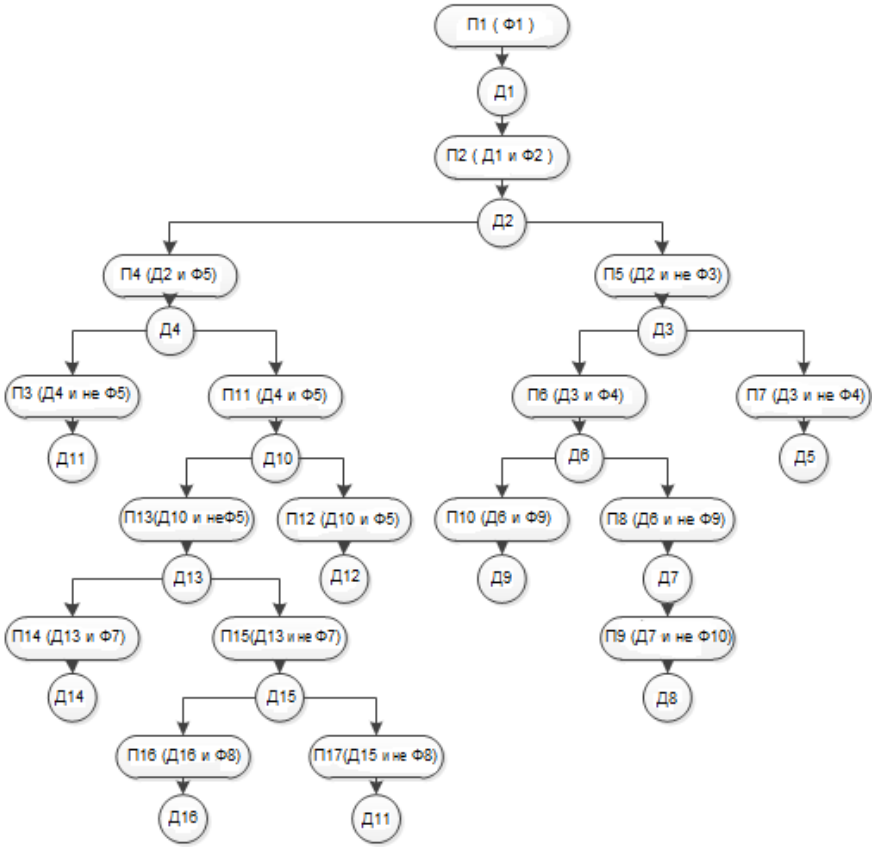



Рис. 4. Взаимосвязь продукций предметной области «Администрирование информационных систем»

2. Выбор заболевания для диагноза обуславливается наличием определенных симптомов. Следовательно, необходимо определить симптомы для выявления диагноза.

3. Прежде чем поставить диагноз, необходимо определить список болезней, которые входят в состав синдрома острого живота.

4. Для данной экспертной системы были выбраны наиболее часто встречающиеся заболевания: острый аппендицит, острый холецистит, острый панкреатит, перфорация язвы при язвенной болезни желудка, острая кишечная непроходимость, тромбоз мезентериальных сосудов, острый аднексит, апоплексия яичника.

5. Преобразуем указанные действия и соответствующие им условия в продукции «Если, то»:

– Если у больного не болит живот, то работа ЭС на этом заканчивается.

– Если у больного болит живот и есть тошнота и рвота, пульс тахикардия, температура высокая и есть напряжение мышц брюшной стенки и постоянная боль, то диагноз острый аппендицит.

– Если у больного болит живот и есть тошнота и рвота, пульс тахикардия, температура высокая и есть напряжение мышц брюшной стенки и постоянная боль, иррадиирущая в правую лопатку, в область правого плеча, возникшая после приема жирной жареной пищи, то диагноз острый холецистит.

– Если у больного болит живот и есть тошнота, пульс тахикардия, температура нормальная или низкая и сильная опоясывающая постоянная боль, возникшая после приема жирной жареной пищи, то диагноз острый панкреатит.

– Если у больного болит живот и есть тошнота и рвота, пульс брадикардия, температура нормальная или низкая и есть напряжение мышц брюшной стенки и резкая кинжальная постоянная боль, то диагноз перфорация язвы при язвенной болезни желудка.

– Если у больного болит живот и есть тошнота и рвота, пульс тахикардия, температура высокая или низкая и есть напряжение мышц брюшной стенки и схваткообразная боль, то диагноз острая кишечная непроходимость.

– Если у больной болит живот, пульс тахикардия, температура нормальная или высокая и постоянная боль, то диагноз острый аднексит.

– Если у больной болит живот, пульс тахикардия, температура нормальная или низкая и есть напряжение мышц брюшной стенки и резкая схваткообразная боль, то диагноз апоплексия.

– Если у больного болит живот и есть тошнота и рвота, пульс тахикардия, температура нормальная или высокая, есть напряжение мышц брюшной стенки и сильная постоянная боль, возникшая внезапно, то диагноз тромбоз мезентериальных сосудов.

Введем обозначения для фактов (Ф), действий-диагнозов (Д) и продукций (П), тогда:

Субъект = больной;
Ф1 = поступила жалоба на боли в животе;
Ф2 = боль в животе;
Ф3 = пульс тахикардия;
Ф4 = пульс брадикардия;
Ф5 = температура тела нормальная;
Ф6 = температура тела высокая;
Ф7 = температура тела низкая;
Ф8 = тошнота;
Ф9 = боль схваткообразная;
Ф10 = боль постоянная;
Ф11 = рвота;
Ф12 = пол женский;
Ф13 = напряжение мышц брюшной стенки;
Ф14 = боль резкая;
Ф15 = боль сильная;
Ф16 = боль после приема жирной, жареной пищи;
Ф17 = боль кинжальная;
Ф18 = боль опоясывающая;
Ф19 = боль внезапная;
Ф20 = боль иррадиирует в правую лопатку, в область правого плеча;
Д1 = врач опрашивает больного для выявления симптомов заболевания;
Д2 = острый аппендицит;
Д3 = острый холецистит;
Д4 = острый панкреатит;
Д5 = перфорация язвы при язвенной болезни желудка;
Д6 = острая кишечная непроходимость;
Д7 = острый аднексит;
Д8 = апоплексия яичника;
Д9 = тромбоз мезентериальных сосудов;
Д1 = диагноз не ясен, нужно дополнительное обследование;
П1:(Ф1) → Д1;
П2: (Д1 и Ф1 и Ф2 и Ф5 и Ф7 и Ф9 и Ф10 и Ф12) → Д2;

П3: (Д1 и Ф1 и Ф2 и Ф5 и Ф7 и Ф9 и Ф10 и Ф12 и Ф15 и Ф19) → Д3;

П4: (Д1 и Ф1 и Ф2 и (Ф4 или Ф7) и Ф7 и Ф9 и Ф14 и Ф15 и Ф17) → Д4;

П5: (Д1 и Ф1 и Ф3 и (Ф4 или Ф6) и Ф7 и Ф9 и Ф10 и Ф12 и Ф13 и Ф16) → Д5;

П6: (Д1 и Ф1 и Ф2 и (Ф5 или Ф6) и Ф7 и Ф8 и Ф10 и Ф12) → Д6;

П7: (Д1 и Ф1 и Ф2 и (Ф4 или Ф5) и Ф9 и Ф11) → Д7;

П8: (Д1 и Ф1 и Ф2 и (Ф4 или Ф6) и Ф8 и Ф11 и Ф12 и Ф13) → Д8;

П9: (Д1 и Ф1 и Ф2 и (Ф4 или Ф5) и Ф7 и Ф9 и Ф10 и Ф12 и Ф14 и Ф18) → Д9.

## Представление правил в продукционном программировании

В продукционном программировании порождающие правила реализуются в виде конструкций, манипулирующих структурами типа векторов, а не строк символов. Это влияние языков типа LISP, CLIPS и тех структур данных, которые они поддерживают.

В результате алфавит КС заменяется словарем символов или атомов и довольно простой грамматикой формирования этих структур. В частности, словарь, как правило, состоит из трех подмножеств: имен объектов ПО, имен свойств (атрибутов) объектов и допустимых значений атрибутов. Поэтому используемая грамматика, как правило, имеет вид вектора–триады «объект – атрибут – значение атрибута», например в CLIPS.

Имея словарь и грамматику, можно в машинном виде представить исходную задачу и решать ее, применяя имеющиеся правила. В программе условная часть продукции соответствует допустимым структурам, а действия или заключения – содержат операторы манипулирования такими структурами.

В целом в продукционном программировании система включает:

- продукционный набор правил;
- рабочую память, определяющую текущее состояние задачи и содержащую данные, описание цели и промежуточные результаты;

• интерпретатор правил, который решает, когда надлежит применить каждое из них.

Рассмотрим детали этого механизма на примере языка CLIPS.

*Продукционный набор правил*. Правила в CLIPS-программе используются для выполнения ряда действий, которые необходимо выполнить в определенной ситуации. Разработчик ЭС определяет совместную совокупность правил для решения проблемы. Правило состоит из двух частей: слева записываются предпосылки (антецедент или условия), которые обычно представляются в виде триады «объект – атрибут – значение атрибута», а справа – действия (заключения, консеквент):

```
( defrule <имя правила>
<условие1>
….
<условие m>
=>
<действие1>
….
<действие n>
)
```

Здесь для определения правил используется defrule.

Чтобы правило было выполнимым, необходимо (но не достаточно!), чтобы выполнялись все его условия. Если условная часть правила пуста, то для его активации необходимо наличие в списке фактов исходного факта (initial-fact). Условие выполняется, если соответствующий ему факт присутствует в списке фактов рабочей памяти. После определения, какие из правил являются выполнимыми, они помещаются в список активированных правил. Далее из этого списка выбирается одно правило, которое выполняется. Действия, предписанные данным правилом, вносят изменения в рабочую память.

*Рабочая память*. Ее функция – хранить данные (факты) в формате векторов «объект – атрибут – значение атрибута». Эти данные используются интерпретатором, который активизирует те правила, условия которых на имеющихся данных выполняются.

*Интерпретатор правил*. Процесс его работы описывается в терминах цикла «распознавание – действие»:

1. Сопоставить условия правил и элементы рабочей памяти.

2. Если окажется, что можно активизировать более одного правила, то выбрать одно из них (разрешить конфликт). Применить выбранное правило. Результатом, скорее всего, будет добавление нового элемента данных в рабочую память и/или удаление каких-либо данных из рабочей памяти. Затем перейти к п.1.

Перед началом этого процесса в рабочую память вводится initial-fact. Цикл останавливается, если обнаруживается, что ни одно правило не активизируется или если правило содержит команду прекращения работы (halt). Остается вопрос, связанный с разрешением конфликтных ситуаций. Рассмотрим его подробнее, так как он специфичен для каждой системы. Конечно, можно пытаться сформулировать такой набор правил, что в любой ситуации будет удовлетворяться только одно из них. Однако в ЭС обычно используются недетерминированные наборы правил, так как в реальной жизни ситуации позволяют использовать более одного правила.

**Разрешение конфликтов.** Выбранные интерпретатором несколько правил называют конфликтующим множеством (в CLIPS – agenda – список заявок). Цель процедуры разрешения конфликтов – выбрать из списка одно правило. Производительность ЭС зависит от ее чувствительности к изменению рабочей памяти и стабильности как степени консерватизма системы.

При всем разнообразии свойства механизмов разрешения конфликтов можно разделить на три группы:

1. Разнообразие. Не следует применять к одним и тем же данным правило, которое уже применялось ранее. Самое простое – удалять из списка примененное ранее правило. Но если надо его повторить, то программист использует функцию refresh.

2. Новизна. В CLIPS элементы рабочей памяти снабжаются атрибутом времени их порождения и приоритет отдается правилам, «реагирующим» на более «свежие» данные.

3. Специфика. Чем больше условий содержит правило, тем оно более специфично.

В интерпретаторе CLIPS использованы все три группы механизмов разрешения конфликтов, а также свойство выпуклости

(salience). Приведем используемые CLIPS стратегии разрешения конфликтов и описание свойства выпуклости.

Стратегия глубины. Правила на основе данных, недавно включенных в рабочую память, располагаются в списке заявок с высшим приоритетом (эта стратегия реализована по умолчанию).

Стратегия ширины. Правила на основе данных, давно включенных в рабочую память, располагаются в списке заявок с высшим приоритетом (эта стратегия реализована по умолчанию).

Стратегия простоты. Определяется сложность правила по числу операций проверки условий данного правила. Приоритет отдается более простым правилам.

Стратегия сложности. Определяется сложность правила по числу операций проверки условий данного правила. Приоритет отдается более сложным правилам.

LEX-стратегия. Из списка удаляются ранее использованные правила, остальные сортируются по «новизне» данных.

Свойство выпуклости (salience). Предпочтение отдается тому правилу, которое характеризуется большим значением выпуклости. По умолчанию любое правило имеет нулевое значение выпуклости. Поясним salience на примере. Пусть дан простой таксономический граф, не учитывающий исключений из правил: Животные, способные летать = Летающие рыбы + Птицы (Воробьи,…, Пингвины) + Летучие мыши. Есть два правила:

```
(defrule
(птица (тип ?X))
=>
(assert (да))
) /* содержит утверждение, что случайно выбранная птица способна летать*/
(defrule
(птица (пингвин))
=>
(assert (нет))
) /* содержит утверждение, что если птица – пингвин, то она не способна летать*/
```

Нужно так организовать систему правил, чтобы правило, касающееся пингвинов, имело более высокий приоритет перед более

общим правилом, относящимся к птицам. Для этого в CLIPS-программе правилу для пингвинов назначается большая выпуклость: после строки (птица (пингвин)) вставляется строка вида (declare (salience 100)). Свойству выпуклости можно придавать и отрицательное значение (правило насильно отправляется в хвост списка). Правда, теория не рекомендует приписывать правилам жесткие приоритеты, хотя в отдельных случаях такой подход дает неплохие результаты.

**Прямая и обратная цепочки рассуждений.** С точки зрения последовательности применения правил в продукционном программировании можно выделить две стратегии решения задачи: применять правила в прямом и обратном порядке.

Прямой порядок означает, что рассуждения строятся, отталкиваясь от условий, о которых известно, что они удовлетворяются, к действиям (заключениям), вытекающим из этих условий.

Обратная цепочка означает, что рассуждения строятся, отталкиваясь от заданной цели, к условиям, при которых возможно ее достижение.

В CLIPS строится прямая цепь рассуждений, а, например, порождающие правила в ЭС MYCIN используют обратную цепочку. В CLIPS всегда сопоставляется состояние рабочей памяти и левые (условные) части правил, а затем выполняются действия, предусмотренные правой частью правил. В MYCIN ведущей является правая часть правила.

Отличия прямой и обратной цепочки проще представить в терминах грамматических правил. Ранее рассмотренный для палиндромов набор из трех правил

(P1) $ → a$a, (P2) $ → b$b, (P3) $ → c$c

можно использовать двумя способами.

Во-первых, для формирования палиндромов. Так, применение правил P1, P1, P3, P2, P3 к исходному символу с приведет к формированию строк: aca, aacaa, caacaac, bcaacaacb, cbcaacaacbc. Это пример прямой цепочки, поскольку с и каждая последующая строка сопоставляется с левой частью правил.

Во-вторых, этот же набор правил можно использовать для распознавания палиндромов. Например, пусть имеется строка bacab. Проследим, в какой последовательности применялись правила при ее построении? Эта строка соответствует правой части правила P2. Левая часть этого правила – это строка aca, которая соответствует правой части правила P1, а его левая часть – аксиома c. Процесс распознавания успешно завершился, мы доказали, что bacab – палиндром. Если взять как исходную строку acbcb, то в имеющемся наборе правил не удастся найти такое, правая часть которого «допускала» бы эту строку, т.е. исходная строка – не палиндром.

В целом, при построении прямой цепочки может оказаться, что данные в рабочей памяти удовлетворяют условиям нескольких правил. При построении обратной цепочки – одна цель достигается при выполнении нескольких правил.

Пространство поиска правил можно также представить И/ИЛИ-деревом, вершины которого – состояния рабочей памяти, а ребра – правила. Если считать, что корень дерева – цель, то листья – данные или возможные варианты решения задачи. И-вершины соответствуют применению нескольких правил, а ИЛИ-вершины – наличию альтернативы при выборе правил. Продукционное программирование, основанное на правилах (логическое программирование), не снимает проблему комбинаторного взрыва, так как для многих задач И/ИЛИ-дерево может ветвиться по экспоненциальному закону.

Не существует способа, который бы позволял одному правилу вызвать другое (как в процедурном программировании). Правило P может облегчить активизацию правила P*, но единственный способ его активизации – изменить состояние рабочей памяти. Иногда, чтобы решить, какое правило активизировать, желательно использовать конкретные знания, а не следовать стратегии разрешения конфликтов. С этой целью в некоторые интерпретаторы включены метаправила, которые определяют правила применения правил. В CLIPS этого нет, но есть свойство выпуклости.

**Программные среды для проектирования продукционных БЗ ЭС.** В последние годы в связи со стремительным развитием IT возникла потребность в средствах для поддержки инженерии знаний:

оболочках и инструментальных пакетах для создания экспертных систем.

Среди инструментальных пакетов специалисты выделяют ART (1984), KEE (1987) и Knowledge Craft (1987). В середине 90-х годов в класс самых мощных и развитых систем вошла среда G2. Эти пакеты являются многофункциональными и достаточно дорогими системами.

Доступные российским разработчикам оболочки для создания ЭС бывают свободно распространяемыми и коммерческими. Среди свободно распространяемых оболочек можно выделить: CLIPS (http://www.jsc.nasa.gov/~clips/CLIPS.htm) и связанные системы – DYNACLIPS, FuzzyCLIPS, wxCLIPS, а также SOAR, OPS5, RT–EXPERT, MIKE, BABYLON, WindExS, ES. Эти оболочки достаточно популярны, хотя не все из них имеют нормальный графический интерфейс, а их коммерческое использование невозможно в связи с отсутствием технической поддержки.

В качестве примеров коммерческих оболочек можно привести: ACQUIRE, Easy Reasoner, ECLIPSE, EXSYS Professional. Они достаточно дороги. Среди российских разработок в этой области следует упомянуть о SIMER+MIR, а также инструментальный комплекс АТ-ТЕХНОЛОГИЯ.

## Задачи

1. Построить продукционную модель представления знаний в предметной области «Операционные системы» (функционирование).
2. Построить продукционную модель представления знаний в предметной области «Программное обеспечение» (виды и функционирование).
3. Построить продукционную модель представления знаний в предметной области «Предприятие по разработке программного обеспечения» (структура и функционирование).
4. Построить продукционную модель представления знаний в предметной области «Компьютерная безопасность» (средства и способы ее обеспечения).
5. Построить продукционную модель представления знаний в предметной области «Человеко-машинный интерфейс».

6. Построить продукционную модель представления знаний в предметной области «Корпоративное программное обеспечение».

7. Построить продукционную модель представления знаний в предметной области «Программное обеспечение систем цифровой обработки сигналов».

8. Построить продукционную модель представления знаний в предметной области «Администрирование СУБД».

9. Построить продукционную модель представления знаний в предметной области «Железная дорога» (продажа билетов).

10. Построить продукционную модель представления знаний в предметной области «Университет» (учебный процесс).

11. Построить продукционную модель представления знаний в предметной области «Интернет-кафе» (организация и обслуживание).

12. Построить продукционную модель представления знаний в предметной области «Туристическое агентство» (работа с клиентами).

13. Построить продукционную модель представления знаний в предметной области «Кухня» (приготовление пищи).

14. Построить продукционную модель представления знаний в предметной области «Больница» (прием больных).

15. Построить продукционную модель представления знаний в предметной области «Студенческая конференция».

16. Построить продукционную модель представления знаний в предметной области «Фильтрация спама».

17. Построить продукционную модель представления знаний в предметной области «Приложения дельта-преобразований 2-го порядка».

18. Построить продукционную модель представления знаний в предметной области «Нейросети».

19. Построить продукционную модель представления знаний в предметной области «Многоагентные системы».

20. Построить продукционную модель представления знаний в предметной области «Интеллектуальные сенсорные системы».

21. Построить продукционную модель представления знаний в предметной области «Онтологические системы».

22. Построить продукционную модель представления знаний в предметной области «Проект Semantic Web».

23. Построить продукционную модель представления знаний в предметной области «Базы знаний».

24. Построить продукционную модель представления знаний в предметной области «Интеллектуальная робототехника».

25. Построить продукционную модель представления знаний в предметной области «Игры и машинное творчество».

26. Построить продукционную модель представления знаний в предметной области «Эволюционные алгоритмы».

27. Построить продукционную модель представления знаний в предметной области «Проект 20Q».

28. Построить продукционную модель представления знаний в предметной области «Распознавание образов».

## Контрольные вопросы (тесты)

1. Необходимым и достаточным условием для осуществления интеллектуальных действий в символьных системах является универсальность манипуляций над символами, а основным инструментом символьных систем является логический вывод решения путем поиска по дереву решений, реализуя некоторую стратегию поиска. В ИИ эти утверждения являются: а) аксиомами; б) гипотезами; в) леммами; г) постулатами; д) теоремами.

2. Представление знаний – это: а) кодирование информации на каком-либо формальном языке; б) знания, представленные в программе на языке С++; в) знания, представленные в учебниках по математике; г) моделирование знаний специалистов-экспертов.

3. «Жёсткие» модели представления знаний: а) база данных; б) искусственная нейросеть; в) нечёткие множества; г) продукционные правила; д) семантическая сеть; е) теоремы; ж) фреймы.

4. Согласно Посту, каноническая система включает: а) алфавит; б) аксиомы; в) атрибуты; г) правила порождений; д) правила поведения.

5. Если в канонической системе некоторое слово Y можно получить из другого слова X за конечное число применений правил

порождения, то Y: а) доказуемо; б) выводимо; в) единственно; г) определено; д) разрешимо.

6. Если в канонической системе для некоторого слова X найдётся такая аксиома Y, из которой это слово выводимо, то слово: а) аксиоматизируемо; б) доказуемо; в) единственно; г) определено; д) разрешимо.

7. В основе грамматики формальных языков и большинства языков программирования лежат канонические системы с продукциями вида: а) a1a2$® a1a2£; б) $a1a2® £a1a2; в) a1$a2® a1£a2; г) a1$a1® a2£a2, где a1 и a2 – фиксированные строки, $ и £ – переменные строки.

8. Продукционная система включает: а) библиотеку; б) базу правил; в) рабочую память; г) память правил; д) компилятор правил; е) интерпретатор правил.

9. Необходимым условием выполнимости продукционных правил является: а) выполнение хотя бы одного из его условий; б) выполнение хотя бы одного из его заключений; в) выполнение всех его условий; г) выполнение всех его заключений.

10. Условное утверждение в левой части продукционного правила, которое должно выполняться в рабочей памяти для того, чтобы были выполнены соответствующие действия в правой части правила: а) антецедент; б) зависимость; в) консеквент; г) резолюция; д) силлогизм.

11. Заключение или действие в правой части продукционного правила, которое должно быть совершено над базой данных в случае выполнения соответствующих условий в левой части правила: а) антецедент; б) зависимость; в) консеквент; г) резолюция; д) силлогизм.

12. Максимальный размер базы знаний в продукционной модели не превышает: а) 10 записей; б) 100 записей; в) 1000 записей; г) 65 534 записей.

13. Укажите правильную последовательность работы интерпретатора правил CLIPS: а) разрешить конфликт; б) применить

выбранное правило; в) сопоставить условные части правил и элементы рабочей памяти.

14. Основные проблемы при обслуживании системы продукций: а) обеспечение корректности; б) программная поддержка; в) поддержание непротиворечивости; г) обеспечение эффективности вывода.

15. Укажите соответствие между используемыми в CLIPS механизмами разрешения конфликтов: а) стратегия сложности; б) стратегия глубины, в) MEA-стратегия и принципами их разрешения: 1) новизна; 2) разнообразие; 3) специфика.

16. CLIPS-программа может включать следующие элементы: а) аргументы; б) факты; в) шаблоны; г) шифры; д) комментарии; е) кванторы; ж) правила; з) порты.

17. Каждая из следующих последовательностей символов генерируется в соответствии с некоторым правилом. Опишите на CLIPS представление правила, необходимого для продолжения одной последовательности: а) 1, 2, 4, 8, 16,…; б) 1, 1, 2, 3, 5, 8,…; в) 1, a, 2, c, 3, f, 4,…

18. Укажите соответствие между понятиями: а) каноническая система; б) нечеткая логика; в) фрейм и их авторами: 1) Заде; 2) Минский; 3) Пост.

# 2. СЕМАНТИЧЕСКИЕ СЕТИ И ФРЕЙМЫ

## ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ ПО ТЕМЕ

**Ассоциативная память** – тип памяти с адресацией по содержанию, которая способна «вспомнить» аналогичный образец при предъявлении некоторого объекта или восстановить полную ситуацию по частичной информации о ней.

**Модель концептуальная** – модель предметной области из перечня всех понятий, используемых для описания этой области, вместе с их свойствами и характеристиками, классификаций этих понятий по типам, ситуациям, признакам в данной области и законами функционирования процессов, протекающих в ней. Модель концептуальная строится при погружении описания предметной области в базу знаний интеллектуальной системы.

**Модель сетевая** – модель представления знаний, в основе которой лежит семантическая сеть.

**Наследование** – свойство, используемое в базах знаний и заключающееся в том, что если две информационные единицы соединены между собой отношениями типа "род–вид" или "класс–элемент", то информация, общая для всех видов, входящих в род, или для всех элементов, входящих в класс, содержится в информационной единице более высокого уровня и при необходимости наследуется единицей более низкого уровня. Наследование позволяет ликвидировать дублирование в хранении информации в базах знаний.

**Оболочка** – инструментальное средство для проектирования и создания экспертных систем. В состав оболочки входят средства проектирования баз знаний с различными формами представления знаний и выбора режима вывода знаний. Для конкретной предметной области инженер по знаниям определяет нужное представление знаний и стратегии решения задач, а затем, вводя их в оболочку, создает конкретную экспертную систему.

**Обработка естественного языка** – совокупность процессов анализа текстов на естественном языке, их понимания и синтеза текстов. В процессе анализа в наиболее развитых системах обработки естественно-языковых сообщений происходит морфологический, синтаксический и семантический анализ текста, в результате чего

выявляется глубинная структура текста, которая переводится во внутреннее представление, используемое в базе знаний интеллектуальной системы. Соотнесение этой структуры с теми знаниями, которые хранятся в системе, позволяет понять смысл исходного текста. При синтезе текстов сначала формируется семантическая структура текста, которая затем наполняется лингвистическими единицами с учетом синтаксиса и морфологии выбранного естественного языка. С обработкой естественного языка связано решение задач машинного перевода, автоматического реферирования, общения с пользователем на ограниченном профессиональном естественном языке и т. п.

**Обучение на примерах** – вид обучения, при котором индивиду или интеллектуальной системе предъявляется набор положительных и отрицательных примеров, связанных с какой-либо заранее неизвестной закономерностью. В интеллектуальных системах вырабатываются решающие правила, с помощью которых происходит разделение множества примеров на положительные и отрицательные. Качество разделения, как правило, проверяется экзаменационной выборкой примеров. Если качество разделения на экзаменационной выборке оказывается удовлетворительным, то выработанные решающие правила принимаются системой как окончательные. Если экзамен оказался неудовлетворительным, то экзаменационная выборка добавляется к обучающей и строятся новые решающие правила. После этого процесс экзамена повторяется.

**Объяснение** – одна из функций интеллектуальной системы. Объяснение предоставляет пользователю информацию о том, как интеллектуальная система получила выданное пользователю решение. В отличие от обоснования объяснение опирается лишь на тот маршрут, который сохранился в памяти системы от процесса поиска решения. Используя этот маршрут, интеллектуальная система формирует пользователю объяснение на профессиональном естественном языке, позволяющее ему представить все принципиальные шаги решения.

**Отношение** – задание на множестве декартова произведения. Пары, входящие в произведение, являются элементами отношения, а совокупность этих пар образует график или его экстенсионал.

Отношение может обладать рядом внутренних свойств (рефлексивностью, симметричностью и т. п.) и некоторой внешней семантикой, связанной с его именем. Вся эта информация образует семантику отношения или его интенсионал.

**Предметная область** – совокупность реальных или абстрактных объектов (сущностей), связей и отношений между этими объектами, а также процедур преобразования этих объектов для решения задач, возникающих в предметной области.

**Понятие** – имя, присваиваемое классу сущностей, объединяемых благодаря общности их признаков. В логике понятия являются строго определенными и неизменными образованиями. В искусственном интеллекте понятие понимается шире. В формировании понятия могут принимать участие не только признаки и свойства, но и результаты использования понятий в деятельности людей или при функционировании интеллектуальной системы. Именно в этом смысле используются понятия при рассуждениях о деятельности людей и о функционировании интеллектуальных систем. Для образования понятий в интеллектуальных системах используются различные приемы обобщения.

**Семантическая сеть** – декларативная модель представления знаний в виде графа, вершины которого соответствуют понятиям или объектам, а дуги – отношениям между объектами.

**Сеть ассоциативная** – семантическая сеть, в которой отношения указывают на ассоциативные связи между вершинами, характеризующими объекты, факты и ситуации для описываемой предметной области.

**Слот** – основная структурная единица фрейма. Слот представляет собой пару: (атрибут (имя слота) – значение). В качестве значения могут выступать константные факты, выражения, содержащие переменные, ссылки на другие слоты и т.п. Слот может иметь структуру, элементы которой сами являются слотами.

**Сценарий** – семантическая сеть, в которой в качестве отношений используются каузальные отношения или отношения типа "действие–результат", "действие–цель", "орудие–действие" и т.п.

**Фрейм** – декларативная модель представления и формализации знаний в виде структуры данных для представления стереотипных

ситуаций. Фрейм идентифицируется уникальным именем и включает в себя множество слотов, в которых описывается информация о свойствах и характеристиках фрейма. Отражение в иерархии фреймов родовидовых отношений обеспечивает возможность реализации операции наследования.

**Фрейм-образец** – фрейм, выступающий в качестве образца при поиске по образцу в базах знаний.

**Фрейм-прототип** – фрейм, у которого в части слотов (или во всех слотах) отсутствуют константные значения. Фрейм-прототип описывает знание о предметной области. При означивании всех слотов фрейма-прототипа константными значениями он превращается в фрейм-экземпляр.

**Язык фреймовый** – язык представления знаний и манипулирования знаниями, использующий в качестве модели знаний фреймовые представления. Наиболее известными являются языки FRL и KRL.

## Семантические и ассоциативные сети

Продукционные порождающие правила являются очень удобной моделью знаний для представления связей между состоянием проблемы и действиями, которые необходимо предпринять для ее решения. Иными словами, ПП очень подходят для ответа на вопрос «Что делать, если…?».

Иное дело, если необходимо представить знания о событиях, сложных объектах, связях между ними, их классификации или представить смысл выражений естественного языка человека. Здесь удобнее использовать структуры в виде семантической сети (СС). Это граф, вершины которого соответствуют понятиям или объектам, а дуги – отношениям между объектами.

По типам отношений СС могут быть однородными и неоднородными. Однородные сети обладают только одним типом отношений, а неоднородные сети представляют больший интерес для практических целей, но и большую сложность для исследования.

По арности типичными являются СС с бинарными отношениями. Бинарные отношения просты и удобно выглядят на графе в виде стрелки между двух концептов, они играют исключительную роль в

математике. На практике, однако, могут понадобиться N-арные отношения, связывающие более двух объектов. При этом возникает сложность, как изобразить подобную связь на графе, чтобы не запутаться. Для этого используют, например, концептуальные графы, которые представляют каждое отношение в виде отдельного узла.

Типы отношений в СС определяются её создателем, исходя из конкретных целей. В реальном мире число отношений стремится к бесконечности. Каждое отношение является, по сути, предикатом, простым или составным. Скорость работы с БЗ зависит от того, насколько эффективно сделаны программы обработки нужных отношений.

Наиболее часто используется отношение «объект–множество», обозначающее, что объект принадлежит некоторому множеству. Это отношение называется отношением классификации и обозначается ISA. Обозначение произошло от английского «IS A». Связь ISA предполагает, что свойства объекта наследуются от множества. Обратное к ISA отношение используется для обозначения примеров, поэтому так и называется – «Example», или, по-русски, «Например».

Отношение между надмножеством и подмножеством называется AKO – «A Kind Of» («разновидность»). Альтернативные названия – «SubsetOf» и «Подмножество». Это отношение определяет, что каждый элемент первого множества входит и во второе (выполняется ISA для каждого элемента), а также логическую связь между самими подмножествами: что первое не больше второго и свойства первого множества наследуются вторым.

Объект, как правило, состоит из нескольких частей, или элементов. Например, компьютер состоит из системного блока, монитора, клавиатуры, мыши и т. д. Важным отношением является HasPart, описывающее части/целые объекты: двигатель – это часть для автомобиля; автомобиль – это объект, который включает в себя двигатель.

Часто в семантических сетях требуется определить отношения синонимии, а также следующие отношения:

- функциональные связи (определяемые обычно глаголами «производит», «влияет»…);
- количественные (больше, меньше, равно…);

- пространственные (далеко от, близко от, за, под, над…);
- временные (раньше, позже, в течение…);
- атрибутивные (иметь свойство, иметь значение);
- логические (И, ИЛИ, НЕ);
- лингвистические.

Этот список может сколь угодно продолжаться: в реальном мире количество отношений огромно. Например, между понятиями может использоваться отношение «совершенно разные вещи» или: «не имеют отношения друг к другу (солнце, кухонный чайник).

## Особенности использования некоторых типов отношений

В СС в качестве понятий могут быть как экземпляры объектов, так и их множества. Использование одних и тех же отношений и для элементов, и для коллекций может привести к недоразумениям. Подобные ошибки в работе некоторых первых систем были описаны в статье Д. Макдермотта «Искусственный интеллект сталкивается с естественной глупостью».

Рассмотрим, например, четыре предложения:

1. У Павла есть отец по имени Алексей.
2. Для Павла найдётся отец из множества мужчин.
3. Найдется человек, для которого Алексей – отец.
4. У каждого человека есть отец из множества мужчин.

Для человека ясен смысл этих фраз и многие, не задумываясь, поставили бы во всех случаях отношение «*есть отец*». Однако это является ошибкой: в одном случае, действительно, описывается отношение между двумя экземплярами, но во втором и третьем – между экземпляром и множеством, а в четвёртом – отношение между представителями из двух множеств. Для предложений 1–4 соответственно в математической записи это выглядит так:

I. $\exists$ Павел & $\exists$ Алексей : отец (Алексей, Павел);

IIа. $\exists$ Павел $\rightarrow \exists$ x$\in$ мужчины: отец (x, Павел);

IIб. $\exists$ Алексей $\rightarrow \exists$ y$\in$ люди: отец (Алексей, y);

III. $\forall$ y $\epsilon$ люди $\rightarrow \exists$ x$\in$ мужчины: отец (x, y).

Мы видим, что случаи IIа и IIб различаются только порядком следования переменных в предикате, однако для правильности сети это может сыграть важную роль. В примере перечислены лишь 4 рода отношений, всего же для бинарной сети их существует девять. Они различаются кванторами ∃ и ∀, а также порядком переменных.

Наиболее часто встречающаяся путаница возникает насчёт отношения ISA. Поэтому во многих современных работах принимается, что ISA обозначает связь между экземпляром и множеством (вышеописанный случай IIб): *Мурка ISA кошка*. Использовать ISA для обозначения вхождения элементов одного множества в другое (случай III) не рекомендуется. Для обозначения подмножеств применяется отношение AKO. Различие между ISA и AKO заключается в том, что последнее отвечает ещё и за наследование свойств самих множеств, а не только элементов.

Ниже представлен фрагмент семантической сети (рис. 5), иллюстрирующей предложение «Петров на протяжении периода времени с $t_1$ по $t_2$ владел автомобилем марки "Святогор"».

Дуги s, e, владелец, объект, начало, конец на рисунке указывают на иерархическую связь понятий. Символ Q обозначает конкретную описываемую ситуацию.

СС, используемые для представления семантики естественного языка (ЕЯ), называют ассоциативными сетями (АС). Квиллиан обратил внимание на важность обобщенного абстрактного знания для понимания ЕЯ. Он первым предложил использовать в качестве модели памяти СС и предложил метод для извлечения информации из памяти. Этот метод напоминает организацию толковых словарей: каждое понятие в них определяется другими понятиями. Выяснилось, что предложенная модель и метод обладают важным свойством – когнитивной экономией (вычислительная машина – это конструкция из многих деталей, а персональный компьютер – это тоже вычислительная машина, тогда нет смысла в явном виде хранить информацию о составляющих его деталях, присоединяя эту информацию в сети к вершине PC). В современном программировании это принято называть свойством *наследования*.

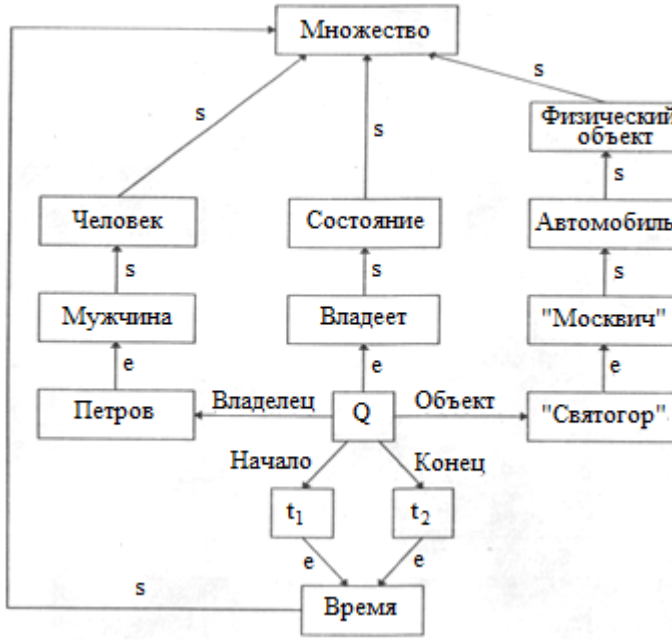



Рис. 5. Фрагмент семантической сети

***Пример***. Имеется фраза: «Автомобили Нива и Волга движутся навстречу друг к другу по направлению к городу Томску». Требуется построить фрагмент семантической сети для этой фразы.

*Решение*. Фрагмент семантической сети для заданной фразы представлен на рис. 6.

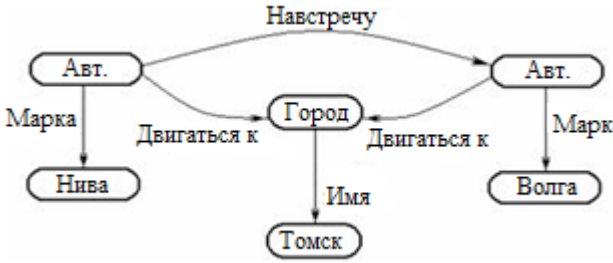



Рис. 6. Фрагмент семантической сети

## Построения семантической сети

Для построения СС необходимо выполнить следующие шаги:

1. Определить абстрактные объекты и понятия предметной области, необходимые для решения поставленной задачи. Оформить их в виде вершин.

2. Задать свойства для выделенных вершин, оформив их в виде вершин, связанных с исходными вершинами атрибутивными отношениями.

3. Задать связи между вершинами, используя функциональные, пространственные, количественные, логические, временные, атрибутивные отношения, а также отношения типа «являться наследником» (АКО) и «являться частью» (ISA).

4. Добавить конкретные объекты и понятия, описывающие решаемую задачу. Оформить их в виде вершин, связанных с уже существующими отношениями типа «являться экземпляром», «есть».

5. Проверить правильность установленных отношений (вершины и отношения при правильном построении образуют предложение).

***Пример***. Построить сетевую модель представления знаний в предметной области «Кафе» (посещение кафе).

*Решение*.

1. Ключевые понятия (объекты) данной предметной области – кафе, тот, кто посещает кафе (клиент) и те, кто его обслуживают (повара, метрдотели, официанты, для простоты ограничимся только официантами). У обслуживающего персонала и клиентов есть общие характеристики, поэтому целесообразно выделить общее абстрактное понятие – человек. Продукцией кафе являются блюда, которые заказывают клиенты.

Исходя из этого, вершины графа будут следующими: «Кафе», «Человек», «Официант», «Клиент», «Заказ» и «Блюдо».

2. У всех объектов есть определенные свойства и атрибуты. Например, кафе располагаются по определенным адресам, каждое блюдо из меню имеет свою цену. Поэтому добавим в граф вершины «Адрес» и «Цена».

3. Определим для имеющихся вершин отношения и их типы.

4. Добавим знание о конкретных фактах решаемой задачи. Пусть имеется два кафе: «Осака» и «2 фунта», в первом работает официантка Юля, а во втором официант Борис. Известны адреса этих кафе и их специфика. Антон решил пойти в кафе «2 фунта» и сделал заказ официанту из 2 блюд: картофель фри за 30 руб., бифштекс за 130 руб.

Исходя из конкретики, соответствующие вершины графа соединим функциональными отношениями и отношениями типа «например» или «наследник». Полученный в результате граф изображен на рис. 7.

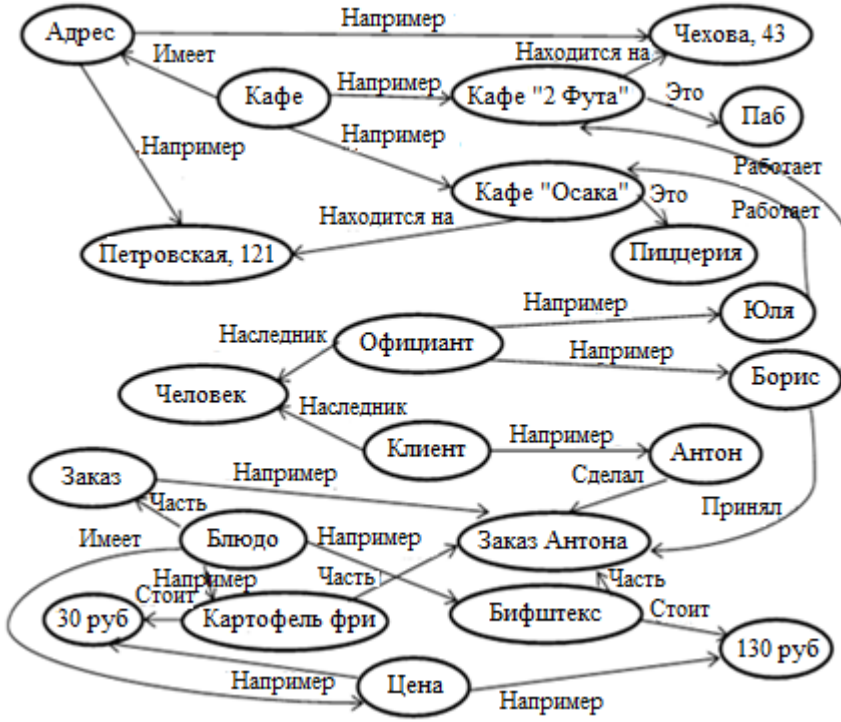



Рис. 7. Граф семантической сети представления знаний в предметной области «Посещение кафе»

5. Осуществим проверку установленных отношений. Например, возьмем вершину «Блюдо» и пройдем по установленным отношениям. Получаем следующую информацию: блюдо является частью заказа,

примерами блюд могут служить картофель фри (по цене 30 руб.) и бифштекс (по цене 130 руб.).

Для получения ответа на какой-либо вопрос по этой задаче, необходимо найти соответствующий участок сети и, используя отношения, получить результат.

Например, вопрос «Какова цена заказа Антона?» Из запроса понятно, что необходимо найти следующие вершины: «Цена», «Антон», «Заказ» и «Заказ Антона». Часть СС, находящаяся между этими вершинами, содержит ответ, а именно, частью заказа Антона являются картофель фри и бифштекс, которые стоят 30 и 130 руб. соответственно. Больше информации о заказе Петра в модели нет, поэтому делаем вывод – Петр заплатил 160 руб.

***Пример***. Построить сетевую модель представления знаний в предметной области «Автозаправка» (посещение автозаправки).

*Решение*.

1. Ключевые понятия данной предметной области – автозаправка, тот, кто посещает автозаправку (клиент), и те, кто его обслуживают (заправщики). У обслуживающего персонала и клиентов есть общие характеристики, поэтому целесообразно выделить общее абстрактное понятие – человек. Продукцией автозаправки является топливо.

Исходя из этого, вершинами графа будут: «Автозаправка», «Человек», «Заправщик», «Клиент», «Заказ» и «Топливо».

2. У этих объектов есть определенные свойства и атрибуты. Например, автозаправки располагаются по определенным адресам, топливо имеет цену. Поэтому добавим в граф вершины «Адрес» и «Цена».

3. Определим для имеющихся вершин графа отношения и их типы.

4. Добавим знание о конкретных фактах решаемой задачи. Пусть имеются две автозаправки: «Лукойл» и «Кобарт». На автозаправке «Лукойл» работает заправщик Иван, на автозаправке «Кобарт» – Федор. Известны адреса автозаправок. Сергей поехал на автозаправку «Кобарт» и заказал заправщику 15 литров бензина марки А-92 по 30 руб./л.

Исходя из конкретики, добавим в граф вершины и соединим их функциональными отношениями и отношениями типа «например» или «часть». Полученный в результате граф изображен на рис. 8.

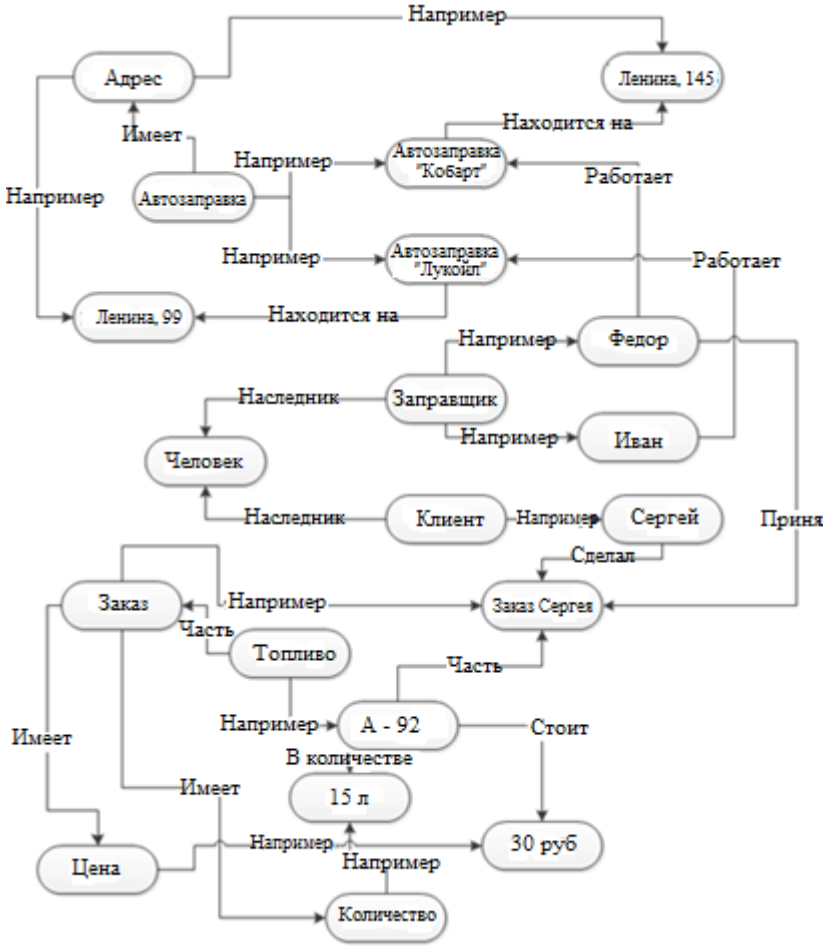



Рис. 8. Семантическая сеть предметной области «Автозаправка» (посещение автозаправки)

5. Осуществим проверку установленных отношений. Например, возьмем вершину «Топливо» и пройдем по установленным отношениям. Получаем следующую информацию: топливо является

частью заказа, примером топлива может служить марка бензина А-92 (по цене 30 руб.).

Для получения ответа на какой-либо вопрос по этой задаче, необходимо найти соответствующий участок сети и, используя отношения, получить результат.

Например, вопрос «Какова цена заказа Сергея?» Из запроса понятно, что необходимо найти следующие вершины: «Цена», «Сергей», «Заказ» и «Заказ Сергея». Часть СС, находящаяся между этими вершинами, содержит ответ, а именно частью заказа Сергея является бензин А-92, который стоит 30 руб. за литр. Больше информации о заказе Сергея в модели нет, поэтому делаем вывод – Сергей заплатил за топливо (15 л) 450 руб.

***Пример***. Построить семантическая модель представления знаний в предметной области «Администрирование информационных систем».

*Решение*.

1. Ключевые понятия данной предметной области – информационная система и те, кто ее обслуживает (администраторы). У обслуживающего персонала есть общие характеристики, поэтому целесообразно выделить общее абстрактное понятие – человек. Методы выявления проблем – проведение тестов.

Исходя из этого, вершины графа будут следующими: «Информационная система», «Человек», «Администратор», «Проблема», «Диагностика системы», «Тесты».

2. Информационная система имеет архитектуру. Поэтому в граф следует добавить вершину «Тип архитектуры».

3. Определим для имеющихся вершин отношения и их типы.

4. Пусть имеется информационная система «Безопасность жизнедеятельности завода». Администратору Николаю поступила жалоба. Николай обнаружил проблему и провел диагностику системы, после чего выявил проблему с ОС.

Исходя из этого, добавим соответствующие вершины в граф и соединим их функциональными отношениями и отношениями типа «например» или «являться экземпляром». Полученный в результате граф изображен на рис. 9.

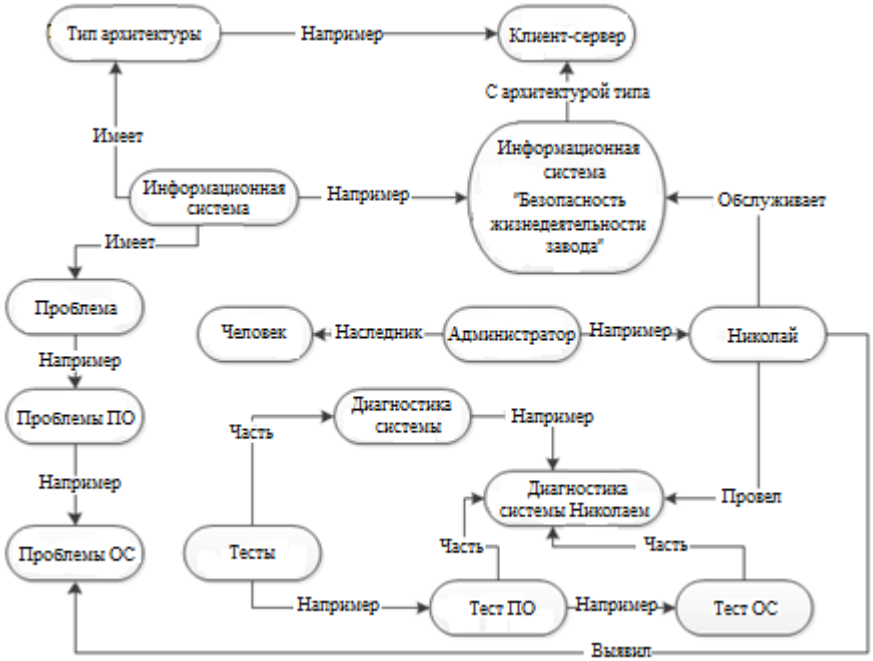



Рис. 9. Семантическая сеть предметной области «Администрирование информационных систем»

5. Осуществим проверку установленных отношений. Например, возьмем вершину «Тесты» и пройдем по установленным отношениям. Получаем следующую информацию: тесты являются частью диагностики системы, примером тестов может служить тест ПО.

***Пример***. Построить семантическую модель представления знаний в предметной области «Больница».

*Решение*.

1. Ключевые понятия данной предметной области – врач, пациенты, болезнь. У пациентов больницы есть общие характеристики, поэтому целесообразно выделить общее абстрактное понятие – человек. Метод постановки диагноза – анализ симптомов заболевания.

Исходя из этого, вершины графа будут следующими: «Врач», «Человек», «Симптом», «Болезнь», «Диагноз».

2. У этих объектов есть определенные свойства и атрибуты. Например, симптоматика включает ряд показателей. Поэтому в граф следует добавить, например, вершины «Пульс тахикардия», «Боль постоянная», «Тошнота», «Напряжение мышц брюшной стенки», «Рвота», «Боль в животе», «Температура 37,2 – 37,6». Одни и те же симптомы сопутствуют различным заболеваниям. Поэтому в граф добавим, например, вершины «Острый аппендицит», «Тромбоз сосудов», «Острая кишечная непроходимость», «Перфорация язвы желудка», «Острый холецистит», «Острый панкреатит», «Острый аднексит».

3. Определим для имеющихся вершин отношения и их типы.

4. Добавим знание о конкретных фактах решаемой задачи. Пусть на прием к врачу гастроэнтерологу пришел пациент с жалобой на боли в животе. Врач проводит осмотр и опрос пациента с целью выявления других симптомов его заболевания. Определив круг возможных заболеваний пациента, врач выдает пациенту рекомендации и, возможно, назначает дополнительные исследования для уточнения диагноза.

Исходя из этого, добавим в граф отношения «Задает вопрос», «Отвечает», «Выдает», «Получает». Итоговая семантическая сеть представлена на рис. 10.

5. Осуществим проверку установленных отношений. Например, возьмем вершину «Боль в животе» и пройдем по установленным отношениям. Получаем следующую информацию: боль в животе является частью симптомов, присущих болезни, которая и является диагнозом пациента.

Для получения ответа на вопрос «Пациент болен острым панкреатитом?», необходимо найти соответствующий участок сети и, используя отношения, получить результат. Из запроса понятно, что необходимо найти следующие вершины: «Пациент», «Врач», «Острый панкреатит». Часть СС, находящаяся между этими вершинами, содержит ответ, а именно: «Острый панкреатит» может стать диагнозом заболевания пациента, если выявятся соответствующие симптомы.

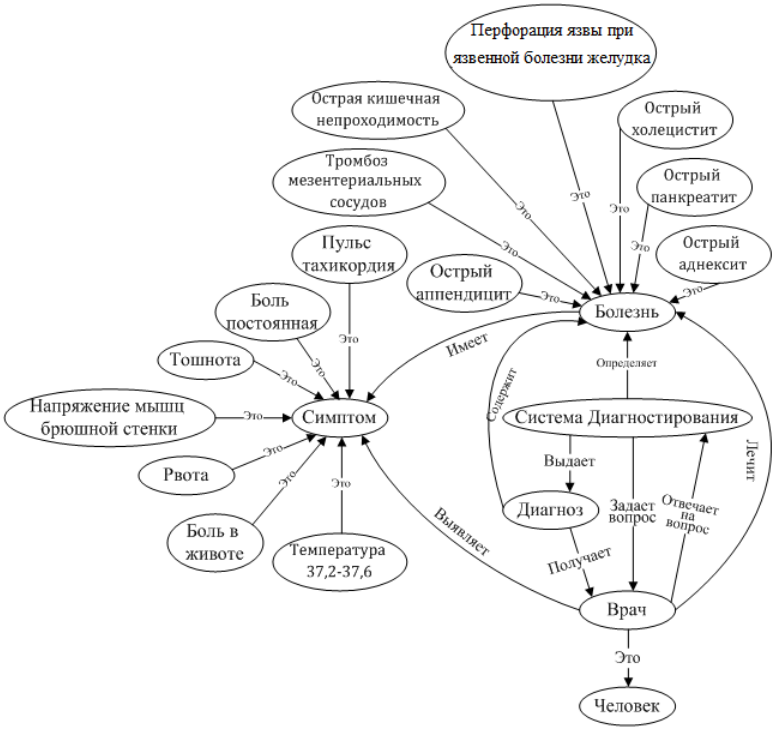



Рис. 10. Семантическая сеть предметной области "Больница"

## ФРЕЙМОВЫЕ МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Естественным желанием исследователей СИИ было объединить вместе представление знаний СС и ПП. Это привело к появлению теории фреймов (frame – остов, скелет, костяк, каркас). Используя идеи теории фреймов, в 70-х годах в MTI провели исследования, которые привели к разработке философии ООП и созданию таких языков, как Smalltalk, C++, Java.

Идея фреймов в том, что представление понятий в мозге базируется на довольно расплывчатых понятиях. Человек обращает внимание на свойства, которые у него ассоциируются с объектами-прототипами, наиболее ярко представляющими свой класс (птица – воробей, четырехугольник – прямоугольник и т.п.). Границы между разными классами всегда размыты, в каждом правиле или классе встречаются исключения. При использовании фреймов знания не

«размазываются» по программному коду приложения, как в ПП, но и не собираются воедино в виде метазнаний, как в СС или АС.

Марвин Минский определил фрейм как «структуру данных для представления стереотипных ситуаций». Реализованная им идея заключалась в том, чтобы сконцентрировать все данные (знания) о конкретном объекте или событии в единой структуре, а не распределять ее между множеством более мелких структур.

Фрейм имеет собственное название, а также список слотов и их значений (наполнителей). Значениями могут быть данные любого типа, а также название другого фрейма. Таким образом, фреймы образуют сеть. Кроме того, существует связь между фреймами типа АКО (a kind of), которая указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются список и значения слотов. При этом возможно множественное наследование – перенос свойств от нескольких прототипов.

Фрейм, как абстрактный образ, может быть представлен следующим образом:

(ИМЯ ФРЕЙМА:
(имя 1-го слота: значение 1-го слота),
(имя 2-го слота: значение 2-го слота),
…….
(имя N-го слота: значение N-го слота)).

Слоты заполнены значениями разнообразных атрибутов, ассоциирующихся с сущностью.

Табличное представление структуры фрейма со слотами выглядит следующим образом:

| Имя фрейма | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| | | | |

Передать данные во фрейм, заполнив его слоты, можно по-разному: при конструировании фрейма, через вызов функции, указанной в слоте, через присоединенную к слоту процедуру, которая называется демоном, из диалога с пользователем, через наследование свойств от других фреймов, из базы данных. Способ получения значения определяет, как именно устанавливается значение конкретного слота, а выбор способа зависит от свойств самих данных.

В табл. 2 представлены основные способы получения значений слотов и их краткое описание.

Таблица 2

| Способ получения значений слотов | Описание способа |
|---|---|
| 1. По умолчанию от прототипа (родителя) | Слоту присваивается значение, определенное по умолчанию во фрейме-прототипе, некоторые стандартные значения |
| 2. Через наследование | Отличается от первого способа тем, что значение задано в специальном слоте родительского фрейма, соединенного с текущим слотом связью AKO |
| 3. По формуле | Слоту назначается формула, результат вычисления которой является значением слота |
| 4. Через присоединенную процедуру (демон) | Слоту назначается процедура, позволяющая получить значение слота алгоритмически |
| 5. Из внешних источников данных | В интеллектуальных системах данные, являющиеся значениями слотов, могут поступать из баз данных и знаний, от системы датчиков, от пользователя |

*Демоном* называется процедура, автоматически запускаемая при выполнении некоторого условия (события) при обращении к соответствующему слоту. Демонов может быть несколько. Наиболее похож механизм присоединенных процедур к триггерам в реляционных базах данных. В табл. 3 представлены наиболее распространенные демоны.

Таблица 3

| Демон | Событие | Описание |
|---|---|---|
| IF-REMOVED | Если удалено | Выполняется, когда информация удаляется из слота |
| IF-ADDED | Если добавлено | Выполняется, когда новая информация записывается в слот |
| IF-NEEDED | По требованию | Выполняется, когда запрашивается информация из пустого слота |
| IF-DEFAULT | По умолчанию | Выполняется, когда устанавливается значение по умолчанию |

Существует несколько видов фреймов, которые позволяют описать предметную область и решаемую задачу. В табл. 4 представлены наиболее распространенные типы фреймов, указаны

типы знаний, которые они отображают, а также примеры фреймов данного типа из различных предметных областей.

Таблица 4

| Тип фрейма | Тип знания | Описание | Пример |
|---|---|---|---|
| *По познавательному значению* | | | |
| Фреймы-прототипы (шаблоны, образцы) | Интенсиональные | Отражают знания об абстрактных стереотипных понятиях, которые являются классами каких-то конкретных объектов | Человек, автомобиль |
| Фреймы-экземпляры (примеры) | Экстенсиональные | Отражают знания о конкретных фактах предметной области | Петров П.И. Хонда Fit |
| *По функциональному значению* | | | |
| Фреймы-структуры (объекты) | Декларативные | Отображают абстрактные и конкретные объекты и понятия предметной области (содержат характеристики объекта и понятия) | Человек, лекция, экзамен, кафедра |
| Фреймы-операции | Процедурные | Отображают различные процессы использования объектов предметной области (содержат характеристики процесса) | Проектирование программы процедура выполнения курсовой работы |
| Фреймы-ситуации | Прагматические | Отображают типичные ситуации, в которых могут находиться фреймы-объекты и фреймы-роли (содержат характеристики, идентифицирующие ситуацию) | Сессия, рабочий режим компьютера, авария |
| Фреймы-сценарии | Технологические | Отображают динамику развития ситуации, типовую структуру для некоторого действия, события (содержат характеристики, обеспечивающие развитие системы по данному сценарию) | Сдача экзамена, празднование именин, защита дипломной работы |
| Фреймы-роли | Функциональные | Отображают типичную роль, выполняемую фреймом-объектом в определенной ситуации (содержат характеристики роли) | Программист, администратор, студент, преподаватель |

**Проблема множественного наследования**. Наследование позволяет описывать свойства и поведение класса в терминах других классов. Объектно-ориентированный язык CLIPS поддерживает множественное наследование классов. Пользовательские классы могут быть конкретными и абстрактными. Абстрактные классы играют ту же роль, что и виртуальные классы в C++, т.е. они используются только для порождения производных классов (например, абстрактный класс PERSON, используя механизм наследования, может создать производные классы WOMAN и MAN). Такая организация связей между фреймами не влечет проблем, пока информация от разных источников наследования не является противоречивой.

Рассмотрим классический пример («Алмаз Никсона») конфликта при множественном наследовании свойств (рис. 11).

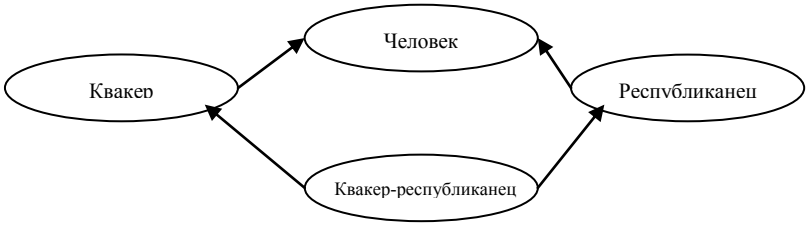



Рис. 11. «Алмаз Никсона» – пример конфликта при множественном наследовании

Квакеры – пацифисты, члены религиозной общины, проповедующие благотворительность (отвергают священников и церковное таинство). В слоте «пацифизм» фрейма «Квакер» хранится значение true. Республиканцы в США пацифистами не являются, поэтому значение слота «пацифизм» во фрейме «Республиканец» равно false. Что тогда можно сказать о квакере, который являлся сторонником республиканской партии США, за чьи голоса боролся президент Р. Никсон? Пацифисты они или нет? «Скептические» СИИ в таких случаях отказываются давать ответ. Другие, обнаружив конфликт, выносят заключение наудачу. Но программисту лучше заранее подумать о том, как избежать конфликта (например, подключив демон по требованию, использующий какие-то дополнительные знания: в год выборов квакеры пацифистами не

являются и т.п.). Конфликты в сети фреймов устанавливаются путем анализа ее топологии (отсюда – алмаз).

Таким образом, фреймы расширяют возможности СС, позволяя организовать иерархию знаний, процедурные вложения, связывающие программный код с сущностями фреймового представления. Например, с помощью фреймов в БЗ можно генерировать графические образы, фреймы можно использовать для моделирования рассуждений и семантики ЕЯ (концептуальные графы Sowa, 1984) и т.д.

## Пример построения фреймовой модели

Для построения фреймовой модели представления знаний необходимо выполнить следующие шаги:

1. Определить абстрактные объекты и понятия предметной области, необходимые для решения поставленной задачи. Оформить их в виде фреймов-прототипов (фреймов-объектов, фреймов-ролей).

2. Задать конкретные объекты предметной области. Оформить их в виде фреймов-экземпляров (фреймов-объектов, фреймов-ролей).

3. Определить набор возможных ситуаций. Оформить их в виде фреймов-ситуаций (прототипы). Если существуют прецеденты по ситуациям в предметной области, добавить фреймы-экземпляры и/или фреймы-ситуации.

4. Описать динамику развития ситуаций через набор сцен. Оформить их в виде фреймов-сценариев.

5. Добавить фреймы-объекты сценариев и сцен, которые отражают данные конкретной задачи.

***Пример.*** Построить фреймовую модель представления знаний в предметной области «Кафе» (посещение кафе).

*Решение.*

1. Ключевые понятия данной предметной области – кафе, клиент и те, кто его обслуживают (повара, метрдотели, официанты, для простоты ограничимся только официантами). У официантов и клиентов есть общие характеристики, поэтому целесообразно выделить общее абстрактное понятие – человек. Тогда фреймы «Кафе» и «Человек» являются прототипами-образцами, а фреймы «Официант» и «Клиент» – прототипами-ролями. Также нужно определить

основные слоты фреймов – характеристики, имеющие значения для решаемой задачи. Опишем указанные фреймы в виде табл. 5–8.

Таблица 5

| ЧЕЛОВЕК | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Пол | М или Ж | Из внешних источников | |
| Возраст | От 0 до 150 лет | Из внешних источников | |

Таблица 6

| КАФЕ | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Название | | Из внешних источников | |
| Адрес | | Из внешних источников | |
| Часы работы | | Из внешних источников | |
| Специализация | | Из внешних источников | |
| Класс | Средний/высший | Из внешних источников | |

Таблица 7

| ОФИЦИАНТ (АКО ЧЕЛОВЕК) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Возраст | От 18 до 55 лет | Из внешних источников | |
| Стаж работы | | Из внешних источников | |
| Зарплата | | Из внешних источников | |
| График работы | | Из внешних источников | |
| Место работы | Фрейм-объект | Из внешних источников | |

Таблица 8

| КЛИЕНТ (АКО ЧЕЛОВЕК) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Вид оплаты | Кэш или карточка | По умолчанию (кэш) | |
| Статус | Обычный или vip | По умолчанию (обычный) | |
| Форма заказа | Заказ есть/нет | По умолчанию (заказа нет) | |
| Чаевые | | Из внешних источников | |

2. Фреймы-образцы описывают конкретную ситуацию: какие кафе имеются в городе, как именно организовывается посещение, кто является посетителем, кто работает в выбранном кафе и т.д. Поэтому определим следующие фреймы-образцы, являющиеся наследниками фреймов-прототипов (табл. 9–13).

Таблица 9

| КАФЕ «ОСАКА» (АКО КАФЕ) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Название | ОСАКА | Из внешних источников | |
| Адрес | Таганрог, Петровская, 23 | Из внешних источников | |
| Часы работы | 10:00–23:00 | Из внешних источников | |
| Специализация | Пиццерия | Из внешних источников | |
| Класс | Средний | Из внешних источников | |

Таблица 10

| КАФЕ «2 фунта» (АКО КАФЕ) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Название | 2 фунта | Из внешних источников | |
| Адрес | Таганрог, Чехова, 43 | Из внешних источников | |
| Часы работы | 11:00–00:00 | Из внешних источников | |
| Специализация | Паб | Из внешних источников | |
| Класс | Средний | Из внешних источников | |

Таблица 11

| БОРИС (АКО ОФИЦИАНТ) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Возраст | 30 лет | Из внешних источников | |
| Пол | М | Из внешних источников | |
| Стаж работы | 7 | Из внешних источников | |
| Зарплата | 12000 | Из внешних источников | |
| График работы | Через день с 18:00 | Из внешних источников | |
| Место работы | Кафе «2 ФУНТА» | из внешних источников | |

Таблица 12

| ЮЛЯ (АКО ОФИЦИАНТ) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Возраст | От 20 лет | Из внешних источников | |
| Пол | Ж | Из внешних источников | |
| Стаж работы | 1 год | Из внешних источников | |
| Зарплата | 8000 | Из внешних источников | |
| График работы | Каждый день С 12:00 до 18:00 | Из внешних источников | |
| Место работы | Кафе «ОСАКА» | Из внешних источников | |

Таблица 13

| АНТОН (АКО КЛИЕНТ) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Пол | М | Из внешних источников | |
| Возраст | 25 | Из внешних источников | |
| Вид оплаты | Кэш | По умолчанию (кэш) | |
| Статус | Обычный | По умолчанию (обычный) | |
| Форма заказа | Заказа нет | По умолчанию (заказа нет) | |
| Чаевые | 7 % от суммы заказа | Из внешних источников | |

3. **Фреймы-ситуации** описывают возможные ситуации. В кафе клиент попадает в несколько типичных ситуаций: заказ и оплата. Конечно, возможны и другие не типичные ситуации: клиент подавился, у клиента нет наличности для оплаты счета и т.д. Рассмотрим несколько типичных ситуаций (табл. 14–15).

Таблица 14

| ЗАКАЗ | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Перечень блюд | | Из внешних источников | IF-ADDED (изменяет слот «Перечень цен») |
| Перечень цен | | Присоединенная процедура | IF-ADDED (изменяет слот «Сумма заказа») |
| Сумма заказа | | Присоединенная процедура | |
| Принял заказ | Фрейм-образец | Из внешних источников | |
| Сделал заказ | Фрейм-образец | Из внешних источников | |

Таблица 15

| ОПЛАТА | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Вид платежа | | Из внешних источников | IF-ADDED (изменяет слот «Чаевые») |
| Чаевые | | Присоединенная процедура | |
| Оплатил | Фрейм-образец | Присоединенная процедура | |
| Заказ | Фрейм-образец | Из внешних источников | IF-ADDED (изменяет слот «Оплатил») |

4. Ситуации возникают после наступления каких-то событий, выполнения условий и могут следовать одна за другой. Динамику

предметной области можно отобразить в фреймах-сценариях. Их может быть множество, опишем наиболее общий и типичный сценарий посещения кафе (табл. 16).

Таблица 16

| ПОСЕЩЕНИЕ КАФЕ | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Посетитель | Фрейм-объект | Из внешних источников | |
| Кафе | Фрейм-объект | Из внешних источников | IF-ADDED, IF-REMOVED (Изменяют слот «Официант») |
| Официант | Фрейм-объект | Присоединенная Процедура (определяется По выбранному кафе) | |
| Сцена 1 | Вход, выбор | Из внешних источников | |
| Сцена 2 | Заказ | Из внешних источников | IF-ADDED (изменяет слот «Оплатил») |
| Сцена 3 | Еда | Из внешних источников | |
| Сцена 4 | Оплата | Из внешних источников | |
| Сцена 5 | Выход | Из внешних источников | |

5. Пусть в рамках нашей задачи Антон посетил кафе «2 фунта». Тогда фреймы будут заполнены следующим образом (табл. 17 – 20).

Таблица 17

| ПОСЕЩЕНИЕ «2 фунта» (АКО ПОСЕЩЕНИЕ КАФЕ) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Посетитель | АНТОН | Из внешних источников | |
| Кафе | КАФЕ «2 ФУНТА» | Из внешних источников | IF-ADDED, IF-REMOVED (изменяют слот «Официант») |
| Официант | БОРИС | Присоединенная процедура (определяется по выбранному кафе) | |
| Сцена 1 | Вход, выбор | Из внешних источников | |
| Сцена 2 | ЗАКАЗ АНТОНА | Из внешних источников | |
| Сцена 3 | Еда | Из внешних источников | |
| Сцена 4 | ОПЛАТА АНТОНА | Из внешних источников | |
| Сцена 5 | Выход | Из внешних источников | |

Таблица 18

| ЗАКАЗ АНТОНА (АКО ЗАКАЗ) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Перечень блюд | Отбивная, темное пиво | Из внешних источников | IF-ADDED (изменяет слот «Перечень цен») |
| Перечень цен | 250; 75 | Присоединенная процедура | IF-ADDED (изменяет слот «Сумма заказа») |
| Сумма заказа | 325 | Присоединенная процедура | |
| Принял заказ | БОРИС | Из внешних источников | |
| Сделал заказ | АНТОН | Из внешних источников | |

Таблица 19

| ОПЛАТА АНТОНА (АКО ОПЛАТА) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Вид платежа | Наличные | Из внешних источников | IF-ADDED (изменяет слот «Чаевые») |
| Чаевые | 30 | Присоединенная процедура | |
| Оплатил | АНТОН | Присоединенная процедура | |
| Заказ | ЗАКАЗ АНТОНА | Пз внешних источников | IF-ADDED (изменяет слот «Оплатил») |

Таблица 20

| ЧЕЛОВЕК | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Сцена 1 | М или Ж | Из внешних источников | |
| Возраст | От 0 до 150 лет | Из внешних источников | |

На рис. 12 представлена граф-схема взаимосвязи фреймов в предметной области «Кафе».

Использование фреймовой модели аналогично семантической сети, только в процессе получения ответа кроме вершин учитываются и слоты. Например, получить ответ на вопрос «Кто работает официантом в кафе “2 фунта”?» можно следующим образом. Из запроса понятно, что необходимо найти фрейм «Кафе “2 фунта”» и

проследить связь с фреймом «Борис», являющимся наследником фрейма «Официант». Также можно найти слот «Место работы» и, проверив его значение во фреймах наследниках фрейма «Официант», определить, что официантом в кафе “2 фунта” работает Борис.

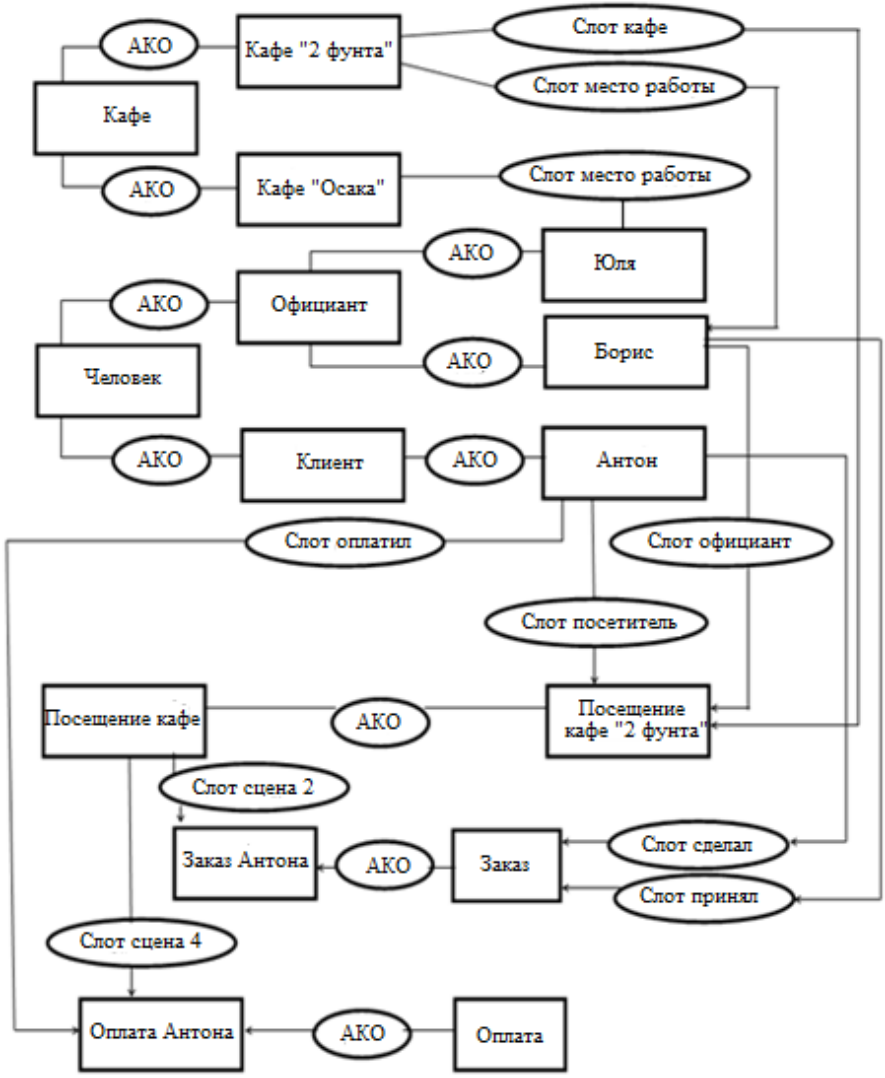



Рис. 12. Граф-схема взаимосвязи фреймов в предметной области «Посещение кафе»

***Пример.*** Построить фреймовую модель представления знаний в предметной области «Автозаправка» (посещение автозаправки).

*Решение.*

1. Ключевые понятия данной предметной области – автозаправка, клиент, обслуживающий персонал (заправщики). У клиентов и заправщиков есть общие характеристики, поэтому целесообразно выделить общее абстрактное понятие – человек. Фреймы «Автозаправка» и «Человек» являются прототипами-образцами, а фреймы «Заправщик» и «Клиент» – прототипами-ролями. Определим основные слоты фреймов – характеристики, имеющие значения для решаемой задачи в виде табл. 18 – 21.

Таблица 18

| ЧЕЛОВЕК | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Пол | М или Ж | Из внешних источников | |
| Возраст | От 0 до 150 лет | Из внешних источников | |

Таблица 19

| Автозаправка | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Название | | Из внешних источников | |
| Адрес | | Из внешних источников | |
| Часы работы | | Из внешних источников | |

Фреймы-наследники содержат все слоты своих родителей, они явно прописываются только в случае изменения какого-либо параметра

Таблица 20

| Заправщик (АКО ЧЕЛОВЕК) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Возраст | От 18 до 55 лет | Из внешних источников | |
| Стаж работы | | Из внешних источников | |
| Зарплата | | Из внешних источников | |
| График работы | | Из внешних источников | |
| Место работы | Фрейм-объект | Из внешних источников | |

Таблица 21

| Клиент (АКО Человек) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Вид оплаты | Наличные или карточка | По умолчанию (наличные) | |
| Статус | Обычный или Vip | По умолчанию (обычный) | |
| Форма заказа | Заказ есть или нет | По умолчанию (заказа нет) | |
| Чаевые | | Из внешних источников | |

2. Фреймы-образцы описывают конкретные ситуации: какие автозаправки имеются в городе, кто является клиентом, кто работает в выбранной автозаправке и т.д. Определим фреймы-образцы, являющиеся наследниками фреймов-прототипов (табл. 22 – 26).

Таблица 22

| Автозаправка «Кобарт» (АКО Автозаправка) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Название | Кобарт | Из внешних источников | |
| Адрес | ул. Дзержинского, 145 | Из внешних источников | |
| Часы работы | Круглосуточно | Из внешних источников | |

Таблица 23

| Автозаправка «Лукойл» (АКО Автозаправка) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Название | Лукойл | Из внешних источников | |
| Адрес | ул. Ленина, 100 | Из внешних источников | |
| Часы работы | Круглосуточно | Из внешних источников | |

Таблица 24

| Федор (АКО Заправщик) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Пол | Мужской | Из внешних источников | |
| Возраст | 22 | Из внешних источников | |
| Стаж работы | 1 | Из внешних источников | |
| Зарплата | 12 000 | Из внешних источников | |
| График работы | Каждый день с 8:00 до 17:00 | Из внешних источников | |
| Место работы | Автозаправка «Кобарт» | Из внешних источников | |

Таблица 25

| Иван (АКО Заправщик) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Пол | Мужской | Из внешних источников | |
| Возраст | 26 | Из внешних источников | |
| Стаж работы | 3 | Из внешних источников | |
| Зарплата | 16 000 | Из внешних источников | |
| График работы | Через день с 17:00 до 8:00 | Из внешних источников | |
| Место работы | Автозаправка «Лукойл» | Из внешних источников | |

Таблица 26

| Сергей (АКО Клиент) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Пол | Мужской | Из внешних источников | |
| Возраст | 23 | Из внешних источников | |
| Вид оплаты | Наличные | По умолчанию (наличные) | |
| Статус | Обычный | Из внешних источников | |
| Форма заказа | Заказа нет | По умолчанию (заказа нет) | |
| Чаевые | 5 % от суммы заказа | Из внешних источников | |

3. Определим фреймы-ситуации на автозаправке клиент: заказ и оплата. Рассмотрим несколько типичных ситуаций (табл. 27 – 28).

Таблица 27

| Заказ | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Марка топлива | | Из внешних источников | IF-ADDED (изменяет слот «Цена») |
| Цена | | Присоединенная процедура | IF-ADDED (изменяет слот «Сумма заказа») |
| Количество | | Присоединенная процедура | IF-ADDED (изменяет слот «Сумма заказа») |
| Сумма заказа | | Присоединенная процедура | |
| Принял заказ | Фрейм-образец | Из внешних источников | |
| Сделал заказ | Фрейм-образец | Из внешних источников | |

Таблица 28

| Оплата | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Вид платежа | | Из внешних источников | IF-ADDED (изменяет значение слота «Чаевые») |
| Чаевые | | Присоединенная процедура | |
| Оплатил | Фрейм-образец | Из внешних источников | |
| Заказ | Фрейм-образец | Из внешних источников | IF-ADDED (изменяет значение слота «Оплатил») |

4. Ситуации возникают после наступления каких-то событий, выполнения условий и могут следовать одна за другой. Фреймы-сценарии отображают динамику предметной области. Их может быть множество, опишем наиболее общий и типичный сценарий посещения автозаправки (табл. 29).

Таблица 29

| Посещение автозаправки | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Клиент | Фрейм-объект | Из внешних источников | |
| Автозаправка | Фрейм-объект | Из внешних источников | IF-ADDED, IF-REMOVED (изменяют слот «Заправщик») |
| Заправщик | Фрейм-объект | Присоединенная процедура (определяет по выбранной автозаправке) | |
| Сцена 1 | Выбор топлива | Из внешних источников | |
| Сцена 2 | Заказ | Из внешних источников | |
| Сцена 3 | Оплата | Из внешних источников | |
| Сцена 4 | Получение топлива | Из внешних источников | |

5. Пусть в рамках нашей задачи Сергей посетил автозаправку «Кобарт». Тогда фреймы будут заполнены следующим образом (табл. 30 – 32).

Таблица 30

| Посещение «Кобарт» (АКО Посещение автозаправки) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Клиент | Сергей | Из внешних источников | |
| Автоза-правка | «Кобарт» | Из внешних источников | IF-ADDED, IF-REMOVED (изменяют слот «Заправщик») |
| Заправ-щик | Федор | Присоединенная проце-дура (опред. по выбран-ной автозаправке) | |
| Сцена 1 | Выбор топлива | Из внешних источников | |
| Сцена 2 | Заказ Сергея | Из внешних источников | |
| Сцена 3 | Оплата Сергея | Из внешних источников | |
| Сцена 4 | Получ. топлива | Из внешних источников | |

Таблица 31

| Заказ Сергея (АКО Заказ) | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Марка топлива | А-92 | Из внешних источников | IF-ADDED (изменяет слот «Цена») |
| Цена | 30 | Присоединенная процедура | IF-ADDED (изменяет слот «Сумма заказа») |
| Количество | 15 | Присоединенная процедура | IF-ADDED (изменяет слот «Сумма заказа») |
| Сумма заказа | 450 | Присоединенная процедура | |
| Принял заказ | Федор | Из внешних источников | |
| Сделал заказ | Сергей | Из внешних источников | |

Взаимосвязь различных видов фреймов представлена в виде графа на рис. 13.

Использование фреймовой модели аналогично семантической сети, только в процессе получения ответа кроме вершин учитываются и слоты.

Например, получить ответ на вопрос "Кто работает заправщиком на автозаправке «Лукойл»?" можно следующим образом. Из запроса понятно, что необходимо найти фрейм «Автозаправка «Лукойл» и

Таблица 32

| Оплата Сергея | | | |
|---|---|---|---|
| Имя слота | Значение слота | Способ получения значения | Демон |
| Вид платежа | Наличные | Из внешних источников | IF-ADDED (изменяет значение слота «Чаевые») |
| Чаевые | 22.5 | Присоединенная процедура | |
| Оплатил | Сергей | Из внешних источников | |
| Заказ | заказ Сергея | Из внешних источников | IF-ADDED (изменяет значение слота «Оплатил») |

проследить его связь с фреймом «Иван», являющимся наследником фрейма «Заправщик». Также можно найти слот «Место работы» и, проверив его значение во фреймах-наследниках фрейма «Заправщик», определить, что заправщиком на автозаправке «Лукойл» работает Иван.

## Задачи

1. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Операционные системы» (функционирование).
2. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Программное обеспечение» (виды и функционирование).
3. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Предприятие по разработке программного обеспечения» (структура и функционирование).
4. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Компьютерная безопасность» (средства и способы ее обеспечения).
5. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Человеко-машинный интерфейс».

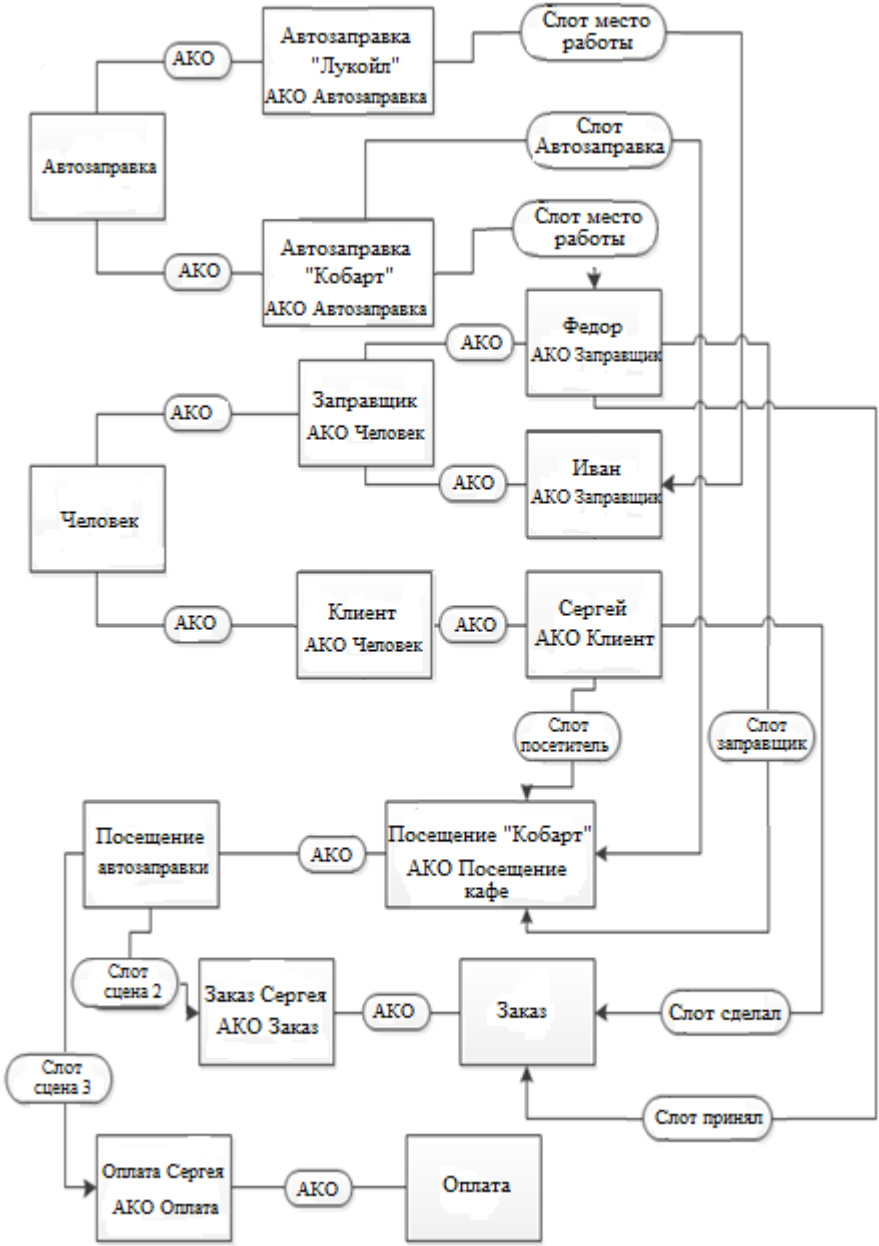



Рис. 13. Граф-схема взаимосвязи фреймов в предметной области «Автозаправка» (посещение автозаправки)

6. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Корпоративное программное обеспечение».

7. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Программное обеспечение систем цифровой обработки сигналов».

8. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Администрирование СУБД».

9. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Железная дорога» (продажа билетов).

10. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Университет» (учебный процесс).

11. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Интернет-кафе» (организация и обслуживание).

12. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Туристическое агентство» (работа с клиентами).

13. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Кухня» (приготовление пищи).

14. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Больница» (прием больных).

15. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Студенческая конференция».

16. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Фильтрация спама».

17. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Приложения дельта-преобразований 2-го порядка».

18. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Нейросети».

19. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Многоагентные системы».

20. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Интеллектуальные сенсорные системы».

21. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Онтологические системы».

22. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Проект Semantic Web».

23. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Базы знаний».

24. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Интеллектуальная робототехника».

25. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Игры и машинное творчество».

26. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Эволюционные алгоритмы».

27. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Проект 20Q».

28. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «Распознавание образов».

## Контрольные вопросы (тесты)

1. Семантическая сеть – это: а) орграф, в котором вершины являются отношениями, а ребра – понятиями; б) орграф, в котором вершины являются понятиями, а ребра – отношениями; в) иерархическая классификационная структура; г) несколько семантически связанных предложений в тексте.

2. Какой из основных типов отношений семантической сети, представленных ниже, может быть назван как АКО (A – Kind – Of): а) это; б) элемент класса; в) имеет частью; г) принадлежит, д) функциональная связь.

3. Фрейм (в инженерии знаний) – структура для представления знаний: а) об объектах без чёткой структуры; б) о стереотипных ситуациях; в) о вызываемых объектах; г) о схеме действий в реальной ситуации; д) об абстрактном образе с минимально возможным описанием сущности какого-либо объекта, явления, события, ситуации, процесса.

4. Характерной особенностью семантических сетей является наличие следующих типов отношений: а) ISa; б) транзитивность; в) АКО; г) строгий порядок; д) целое-часть (HasPart); е) пространственные; ж) логические; з) симметричность; и) временные.

5. Для фреймов характерно свойство наследования по АКО–связям: а) да; б) нет; в) зависит от контекста.

6. Рёбрами семантической сети обычно выступают: а) действия; б) понятия; б) абстрактные или конкретные объекты; в) отношения.

7. Чем отличаются семантические сети и фреймы: а) элемент модели состоит из множества незаполненных значений некоторых атрибутов, именуемых «слотами»; б) наследование по АКО–связям; в) элемент модели – структура, использующаяся для обозначения объектов и понятий.

8. Какие из выражений, представленных ниже, являются структурной частью фрейма: а) значение N–го слота; б) шаблон; в) примитивные типы данных.

# 3. НЕЙРОСЕТЕВЫЕ МОДЕЛИ И ЭВОЛЮЦИОННЫЕ ВЫЧИСЛЕНИЯ

## ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ ПО ТЕМЕ

**Адаптивный резонанс** – свойство ART-нейросетей (С. Гроссберг) обучаться запоминанию новых образов, не теряя при этом ранее накопленную информацию.

**Аксон** – выходное окончание биологического нейрона, простая длинная ветвь нейрона, которая передаёт выходной сигнал. В искусственных нейросетях аксон моделируется как выход системы.

**Алгоритм генетический** (ГА) – это мощный эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования с использованием механизмов, напоминающих биологическую эволюцию. Является разновидностью эволюционных алгоритмов, использует такие методы естественной эволюции, как наследование, мутации, отбор и скрещивание. Отличием ГА является акцент на использование оператора скрещивания, который производит операцию рекомбинации решений-кандидатов. Повторная оценка функции приспособленности (фитнесс-функции) иногда является фактором, ограничивающим использование ГА. ГА трудно масштабируемы под сложность решаемой проблемы, неясно условие остановки алгоритма для каждой проблемы, во многих задачах ГА имеют тенденцию сходиться к локальному оптимуму.

**Алгоритмы эволюционные** (ЭА) – общее название группы алгоритмов эвристического поиска решений, моделирующих процессы природной эволюции.

**Ген** – часть решения (несколько бит).

**Генетическое программирование** (ГП) – разновидность ЭА, ориентированная, в основном, на решение задач автоматического синтеза программ на основе обучающих данных путем индуктивного вывода. Хромосомами ГП являются математические выражения, представляющие компьютерные программы различной величины и сложности. Программы автоматически генерируются в соответствии с определенной функцией приспособленности для хромосом с помощью генетических операторов, имеют древовидную структуру, узлами которой являются функции, переменные и константы. В ГП целью

является поиск неизвестной программы по известным входным данным и выходным образцам.

**Генотип** – закодированное решение.

**Кроссинговер** (скрещивание, размножение, *crossover*) – эволюционный оператор скрещивания в ЭА, когда для производства потомка нужны несколько родителей (обычно два). Главное требование к кроссинговеру, чтобы потомок или потомки имели возможность унаследовать черты родителей, «смешав» их каким-либо способом. Обычно задается вероятность скрещивания. Надо следить за корректностью результатов кроссинговера.

**Мутация** – эволюционный оператор, который в отличие от оператора скрещивания затрагивает только одну хромосому. Мутация в ЭА имеет аналогию в природе, когда в ДНК происходит замена одного гена другим под воздействием, например, радиоактивности. Считается, что мутация – это причина эволюции, и благодаря ей появляются новые виды. Мутация способствует защите от преждевременной сходимости, реализуется выбором случайного гена или группы генов и их изменения по определенным правилам. Обычно задается вероятность мутации. Надо следить за корректностью результатов оператора мутации.

**Нейрон биологический** – нервная клетка мозга, состоящая из тела (сомы), дерева входных дендритов и выходного аксона. На соме и дендритах располагаются окончания (синапсы) других нервных клеток.

**Нейрон искусственный** – элементарный процессор, используемый в узлах нейросети. Описывается математической моделью в виде уравнения $y = f(g) = f(\sum_i a_i x_i + a_0)$, где y – выходной сигнал нейрона; f(g) – функция активации выхода нейрона; $a_i$ – вес i-го входа; $x_i$ – i-й входной сигнал; $a_0$ – начальное состояние (возбуждение) нейрона; $i = 1, 2, \ldots, n$ – номер входа нейрона; n – число входов. Функция активации может быть пороговой, сигмоидальной, степенной и др.

**Нейронная сеть** – динамическая система, представляющая совокупность связанных между собой искусственных нейронов, способная генерировать выходную информацию в ответ на входное воздействие.

**Персептрон** – физическая модель зрительного восприятия (Ф. Розенблатт), состоящая из искусственных нейронов. Доказано, что если обучающую последовательность из векторов X и Y предъявить персептрону достаточное число раз, то он в конце концов разделит её (если это в принципе возможно) на два класса.

**Фенотип** – раскодированное решение.

**Хромосома** – решение (код).

**Эволюционные вычисления (ЭВ)** – это математические преобразования, позволяющие трансформировать входной поток информации в выходной по правилам, основанным на имитации механизмов эволюционного синтеза, а также на статистическом подходе к исследованию ситуаций и итерационном приближении к искомому решению.

**Эволюционное программирование** (ЭП) – разновидность ЭА, ориентированная на компьютерный синтез автоматов и артефактов, способных инновационным образом реагировать на стимулы, поступающие из внешней среды. Популяция в ЭП отражает характер поведения, вид общения и не предусматривает рекомбинации; в алгоритмах ЭП отсутствует оператор кроссинговера, так же как и некоторые другие генетические операторы; мутация в ЭП является единственным оператором поиска альтернативных решений на уровне фенотипа, а не на уровне генотипа.

## Искусственный нейрон МакКаллока–Питса. Функции активации

На вход нейрона поступает некоторое множество сигналов $x_j$, j=1, 2... *n*, каждый из которых является выходом другого нейрона. Нейрон вычисляет взвешенную сумму V=XW входных сигналов $x_j$ и формирует на выходе сигнал величины 1, если эта сумма превышает определенный порог, и 0 – в противном случае (рис. 14).

Нейрон описывается математической моделью в виде уравнения

$$Y = F(V) = F\left( \sum_{j=1}^{n} x_j \cdot w_j \rhd 0 \right), \quad j = 1, 2, \ldots, n,$$

где $Y$ – выходной сигнал нейрона, $F$ – функция активации выхода нейрона, $w_j$ – «вес» входа $x_j$. Добавив постоянный единичный вход $\boldsymbol{x}_0$=1 (смещение) и положив $\boldsymbol{w}_0$= – $\boldsymbol{\theta}$, получим

$$Y = F(V) = F\left( \sum_{j=0}^{n} x_j \cdot w_j + x_0 \right), \quad j = 0,1,...,n.$$

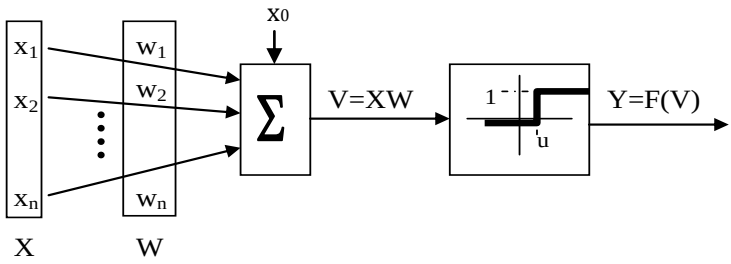



Рис. 14. Искусственный нейрон МакКаллока–Питса

Функции активации могут быть различными. Например, *пороговая функция* активации описывается формулой вида

$$Y(x) = \begin{cases} 1, \ \textit{если} \ x \geq \Theta, \\ 0, \ \textit{иначе}. \end{cases}$$

*Линейная функция* активации описывается формулой вида

$$Y(x) = \begin{cases} 0, \ \textit{если} \ x \leq -0{,}5, \\ 1, \ \textit{если} \ x \geq 0{,}5, \\ x, \ \textit{иначе}. \end{cases}$$

*Логистическая функция* описывается формулой вида

$$Y(x) = \frac{1}{1 + \exp(-ax)},$$

где $a$ – параметр функции, определяющий её крутизну. Когда $a$ стремится к бесконечности, функция вырождается в пороговую. При $a$ = 0 сигмоида вырождается в постоянную функцию со значением 0,5.

*Гиперболический тангенс* – функция активации, описываемая формулой

$$Y(x) = \frac{\exp(x/a) + \exp(-x/a)}{\exp(x/a) - \exp(-x/a)},$$

где $a$ – параметр функции, определяющий её крутизну.

Недостатками пороговой и линейной активационных функций является их недифференцируемость на всей оси абсцисс. Как следствие, нейроны с такими функциями нельзя использовать в сетях, обучающихся по алгоритмам, требующим дифференцируемости активационной функции. Использование сигмоидальных и тангенциальных функций, напротив, позволило перейти от бинарных выходов нейрона к аналоговым. Нейроны с такими функциями чаще всего используются в скрытых слоях нейросетей. Кроме того, у логистической функции простая производная:

$$\frac{dY(x)}{dx} = tY(x)(1-Y(x)).$$

## КЛАССИФИКАЦИЯ НЕЙРОСЕТЕЙ

Искусственная нейронная сеть – математическая модель, реализуемая программно или аппаратно, построенная по подобию естественных нейронных сетей (сетей нервных клеток живого организма), представляющая собой соединение простых взаимодействующих между собой процессоров – искусственных нейронов.

Существует множество нейронных сетей, которые классифицируются по нескольким признакам (табл. 33).

Наибольшее распространение получили слоистые сети прямого распространения.

Для решения конкретной задачи нужно выбрать подходящую нейросеть. При этом нужно учитывать не только перечисленные в таблице критерии, но и архитектуру сети. Выбор архитектуры подразумевает определение количества слоев и нейронов в этих слоях. Не существует формального алгоритма по определению нужной архитектуры, поэтому на практике выбирают или заведомо маленькую сеть и постепенно ее наращивают или заведомо большую и постепенно выявляют неиспользуемые связи и сокращают сеть.

Нейронная сеть, прежде чем использоваться на практике для решения какой-либо задачи, должна быть обучена. Обучение нейронной сети – это процесс настройки синаптических весов. Существует множество алгоритмов, ориентированных на определен-

ные типы сетей и на конкретные задачи, рассмотрим алгоритмы для однослойной и многослойной сетей.

Таблица 33

| Тип нейросети | Описание |
|---|---|
| *По топологии* | |
| Полносвязные | Каждый нейрон связан с другим нейроном в сети (из-за высокой сложности обучения не используется) |
| Слоистые | Нейроны располагаются слоями, каждый нейрон последующего слоя связан с нейронами предыдущего. Есть одно- и многослойные сети |
| *По типу связей* | |
| Прямого распространения | Все связи между нейронами идут от выходов нейронов предыдущего слоя к входам нейронов последующего |
| Рекуррентные | Допускаются связи выходов нейронов последующих слоев с входами нейронов предыдущих |
| *По организации обучения* | |
| С учителем | При обучении используются обучающие выборки, в которых определены требуемые от сети выходные значения, такие сети используют для решения задач классификации |
| Без учителя | Нейронная сеть сама в процессе работы выделяет классы объектов и относит объект к определенному классу, такие сети используют для задач кластеризации |
| *По типу сигнала* | |
| Бинарные | На вход нейронных сетей подают только нули или единицы |
| Аналоговые | Подаваемые на входы нейронов сигналы могут быть произвольными (вещественными числами) |
| *По типу структур* | |
| Однородная | Все нейроны в нейронной сети используют одну функцию активации |
| Неоднородная | Нейроны в нейронной сети имеют разные функции активации |

### Правила Хебба, обучение сети по дельта-правилу

Однослойная нейросеть (рис. 15) может обучаться по правилу Хебба.

Предъявляем на вход персептрона один пример. Если выходной сигнал персептрона совпадает с правильным ответом, то никаких

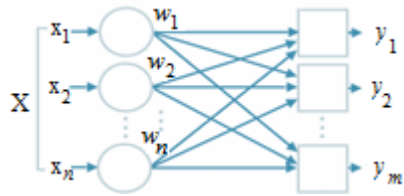


Рис. 15. Однослойная нейросеть

действий предпринимать не надо. В случае ошибки необходимо обучить персептрон правильно решать данный пример. Ошибки могут быть двух типов. Рассмотрим каждый из них.

Первый тип ошибки – на выходе персептрона 0, а правильный ответ – 1. Для того чтобы персептрон выдавал правильный ответ необходимо, чтобы сумма произведений XW стала больше порогового значения. Поскольку переменные $x_i$ принимают значения 0 или 1, увеличение суммы может быть достигнуто за счет увеличения весов $w_i$. Однако нет смысла увеличивать веса при переменных $x_i$, которые равны нулю. Таким образом, следует увеличить веса $w_i$ при тех переменных $x_i$, которые равны 1. Для закрепления единичных сигналов с $x_i$, следует провести ту же процедуру и на всех остальных слоях.

*Первое правило Хебба.* Если на выходе персептрона получен 0, а правильный ответ равен 1, то необходимо увеличить веса связей между одновременно активными нейронами. При этом выходной персептрон считается активным. Входные сигналы считаются нейронами.

Второй тип ошибки – на выходе персептрона 1, а правильный ответ равен нулю. Для обучения правильному решению данного примера следует уменьшить сумму произведений XW. Для этого необходимо уменьшить веса связей $w_i$ при тех переменных $x_i$, которые равны 1 (поскольку нет смысла уменьшать веса связей при равных нулю переменных $x_i$). Необходимо также провести эту процедуру для всех активных нейронов предыдущих слоев. В результате получаем второе правило Хебба.

*Второе правило Хебба.* Если на выходе персептрона получена 1, а правильный ответ равен 0, то необходимо уменьшить веса связей между одновременно активными нейронами.

Таким образом, процедура обучения сводится к последовательному перебору всех примеров обучающего множества с применением правил Хебба для обучения ошибочно решенных примеров. Если после очередного цикла предъявления всех примеров окажется, что все они решены правильно, то процедура обучения завершается.

Нерассмотренными остались два вопроса. Первый – насколько надо увеличивать (уменьшать) веса связей при применении правила Хебба. Второй – о сходимости процедуры обучения.

*Теорема о сходимости персептрона.* Если существует вектор параметров $\boldsymbol{\alpha}$, при котором персептрон правильно решает все примеры обучающей выборки, то при обучении персептрона по правилу Хебба решение будет найдено за конечное число шагов.

*Теорема о «зацикливании» персептрона.* Если не существует вектора параметров $\boldsymbol{\alpha}$, при котором персептрон правильно решает все примеры обучающей выборки, то при обучении персептрона по правилу Хебба через конечное число шагов вектор весов начнет повторяться.

*Алгоритм обучения однослойной нейросети по дельта-правилу*.

Ш а г 1. Инициализация матрицы весов (и порогов, в случае использования пороговой функции активации) случайным образом.

Ш а г 2. Предъявление нейросети образа (на вход подаются значения из обучающей выборки – вектор X) – берется соответствующий выход (вектор **D**).

Ш а г 3. Вычисление выходных значений нейронной сети (вектор Y).

Ш а г 4. Вычисление для каждого нейрона величины расхождения реального результата с желаемым:

$$\varepsilon_i = (d_i - y_i),$$

где $d_i$ – желаемое выходное значение на $i$-нейроне, $y_i$ – реальное значение на $i$-нейроне.

Ш а г 5. Изменение весов (и порогов при использовании пороговой функции) по формулам

$$w_{ij}(t+1) = w_{ij}(t) - \eta \cdot \varepsilon_i \cdot x_j,$$

$$\Theta_i(t+1) = \Theta_i(t) - \eta \cdot \varepsilon_i,$$

где $t$ – номер текущей итерации цикла обучения, $w_{ij}$ – вес связи $j$-входа с $i$-нейроном; $\eta$ – коэффициент обучения (от 0 до 1); $x_j$ – входное значение; $\Theta_i$ – пороговое значение $i$-нейрона.

Ш а г 6. Проверка условия продолжения обучения (вычисление значения ошибки и/или проверка заданного количества итераций). Если обучение не завершено, то шаг 2, иначе заканчиваем обучение.

## Пример обучения нейросети по дельта-правилу

Для обучения нейронной сети по Δ-правилу необходимо:

1. Графически отобразить структуру нейронной сети. Определить размерность матрицы синаптических весов.

2. Определить обучающую выборку, представив ее в табличном виде.

3. Выбрать входные данные, на которых будет рассматриваться итерация цикла обучения.

4. Следуя алгоритму обучения по Δ-правилу, просчитать одну итерацию цикла и представить новые синаптические веса в матричном виде.

***Пример.*** Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной неоднородной нейросети, состоящей из двух нейронов и имеющей функции активации: гиперболический тангенс (a=1) и пороговую функцию (Θ=0,7). Выход первого нейрона соответствует оператору эквивалентности, а второго – дизъюнкции. В качестве обучающей выборки использовать таблицу истинности для операций эквивалентности и дизъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

*Решение.*

1. По заданию нейронная сеть состоит из двух нейронов, значит, входов у однослойной нейронной сети будет 2 и выходов 2, а синаптических весов 4. Первый нейрон имеет пороговую функцию активации, второй – гиперболический тангенс (рис. 16).

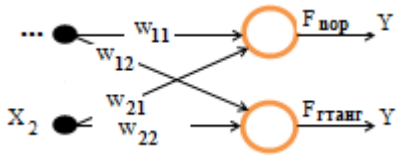



Рис. 16. Исходная нейросеть

2. По заданию нейронная сеть бинарная, поэтому на ее входы могут подаваться только нули и единицы, так как входа 2, то возможных комбинаций входных значений будет 4 (обучающая выборка будет состоять из 4-х векторов). Выход первого нейрона согласно заданию соответствует оператору эквивалентности, а второго – дизъюнкции.

Таблица с обучающей выборкой будет выглядеть следующим образом:

| $x_1$ | $x_2$ | $V_1$ | $V_2$ |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| ***1*** | ***0*** | ***0*** | ***1*** |
| 1 | 1 | 1 | 1 |

3. Пусть в качестве вектора обучения будет рассматриваться 3-я строка таблицы.

4. Следуя алгоритму обучения по Δ-правилу, выполним 6 шагов:

Ш а г 1. Зададим матрицу весов случайным образом из интервала [0,1]:

| $w_{ij}(1)$ | 1 | 2 |
|---|---|---|
| 1 | 0,7 | 1 |
| 2 | 0,5 | 0,2 |

Ш а г 2. Вектор $X = \{0, 1\}$, вектор $D = \{0, 1\}$.

Ш а г 3. Вычисление выходов нейросети Y:

$\Theta$=0,7;

$S_1 = x_1 w_{11} + x_2 w_{21} = 1 \cdot 0{,}7 + 0 \cdot 0{,}5 = 0{,}7;$

$$Y_1 = \begin{cases} 1 & \text{при } S_1 \geq \theta \\ 0 & \text{при } S_1 < \theta \end{cases} = \begin{cases} 1 \text{ при } 0{,}7 \geq 0{,}7 \\ 0 \text{ при } 0{,}7 < 0{,}7 \end{cases} = 1;$$

$a$=1;

$S_2 = x_1 \cdot w_{12} + x_2 \cdot w_{22} = 1 \cdot 0{,}9 + 0 \cdot 0{,}2 = 0{,}9;$

$$Y_2 = \frac{e^{0,9} + e^{-0,9}}{e^{0,9} - e^{-0,9}} \approx 1,39.$$

Ш а г 4. Вычисление величины расхождения реального результата с желаемым:

$\varepsilon_1 = (d_1 - y_1) = (0 - 1) = -1;$

$\varepsilon_2 = (d_2 - y_2) = (1 - 1,39) = -0,39.$

Ш а г 5. Зададим $\eta$ – коэффициент обучения от 0 до 1 и изменим веса:

$\eta = 0,8;$

$w_{11}(2) = w_{11}(1) - 0,8 \cdot \varepsilon_1 x_1 = 0,7 - 0,8(-1)\ 1 = 1,5;$

$w_{21}(2) = w_{21}(1) - 0,8 \cdot \varepsilon_1 x_2 = 0,5 - 0,8(-1)\ 0 = 0,5;$

$\Theta_1(2) = \Theta_1(1) - 0,8 \cdot \varepsilon_1 = 0,7 - 0,8(-1) = 1,5;$

$w_{12}(2) = w_{12}(1) - 0,8 \cdot \varepsilon_1 x_1 = 0,9 - 0,8(-0,39)\ 1 = 1,212;$

$w_{22}(2) = w_{22}(1) - 0,8 \cdot \varepsilon_1 x_2 = 0,2 - 0,8(-0,39)\ 0 = 0,2.$

Матрица весов будет иметь следующий вид:

| $w_{ij}(2)$ | 1 | 2 |
|---|---|---|
| 1 | 1,5 | 1,212 |
| 2 | 0,5 | 0,2 |

Ш а г 6. Вычислим среднеквадратичную ошибку:

$$\varepsilon = \sum_{i=1}^{N} (d_i - y_i)^2 = \sum_{i=1}^{2} \varepsilon_i^2 = \varepsilon_1^2 + \varepsilon_2^2 = (-1)^2 + (-0,39)^2 = 1,152.$$

Здесь $N$ – количество нейронов. Аналогично выполняются последующие итерации, пока среднеквадратичная ошибка не станет меньше заданной величины.

## Обучение сети по методу обратного распространения ошибки

Многослойная нейросеть (рис. 17) может содержать произвольное количество слоев (K), каждый слой состоит из нейронов, число которых $H_k$ также может быть произвольно, количество входов $n$, количество выходов $H = H_K$ равно числу нейронов в выходном слое.

Слои между первым (входы) и последним (выходы) слоями называются скрытыми. Веса в такой сети имеют три индекса: $i$ – номер нейрона следующего слоя, для которого связь входная; $j$ – номер

входа или нейрона текущего слоя, для которого связь выходная; $k$ – номер текущего слоя в нейронной сети (для входов вектора $X$ $k$=0).

Алгоритм обучения многослойной нейросети прямого распространения методом обратного распространения ошибки включает следующие шаги:

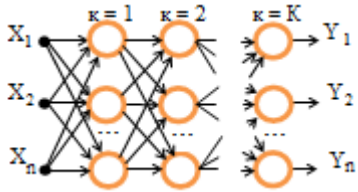



Рис. 17. Многослойная нейросеть

Ш а г 1. Инициализация матриц весов случайным образом (в циклах).

Ш а г 2. Предъявление нейронной сети образа (на вход подаются значения из обучающей выборки – вектор $X$) и берется соответствующий выход (вектор $D$).

Ш а г 3 (прямой проход). Вычисление в циклах выходов всех слоев и получение выходных значений нейронной сети (вектор $Y$):

$$y_i^k = f\left(\sum_{j=0}^{H_{k-1}} w_{ij}^k \cdot y_j^{k-1}\right), \quad y_j^0 = x_j, \quad y_0^{k-1} = 1, \quad x_0 = 1,$$

где $y_i^k$ – выход $i$-нейрона $k$-слоя; $f$ – функция активации; $w_{ij}^k$ – синаптическая связь между $j$-нейроном слоя $k$–1 и $i$-нейроном слоя $k$; $x_j$ – входное значение.

Ш а г 4 (обратный проход). Изменение весов в циклах по формулам

$$w_{ij}^k(t+1) = w_{ij}^k + \eta \cdot \delta_i^k \cdot y_j^{k-1},$$

$$\delta_i^k = (d_i - y_i) \cdot y_i \cdot (1 - y_i) \text{ – для выходного слоя,}$$

$$\delta_i^k = y_i \cdot (1 - y_i) \cdot \sum_{l=1}^{H_{k+1}} \delta_l^{k+1} \cdot w_l^{k+1} \text{ – для скрытых слоев,}$$

где $t$ – номер текущей итерации цикла обучения (номер эпохи); $\eta$ – коэффициент обучения (от 0 до 1); $y_i^k$ – выход $i$-нейрона $k$-слоя; $w_{ij}^k$ – синаптическая связь между $j$-нейроном $k$–1 слоя и $i$-нейроном

слоя $k$; $d_i$ – желаемое выходное значение на $i$-нейроне; $y_i$ – реальное значение на $i$-нейроне выходного слоя.

Ш а г 5. Проверка условия продолжения обучения (вычисление значения ошибки и/или проверка заданного количества итераций). Если обучение не завершено, то шаг 2, иначе заканчиваем обучение. Среднеквадратичная ошибка вычисляется следующим образом:

$$\varepsilon = \frac{1}{0Q} \sum_{q=1}^{Q} \sum_{i=1}^{H} (d_i - y_i)^2,$$

где $Q$ – общее число примеров; $H$ – количество нейронов в выходном слое; $d_i$ – желаемое выходное значение на $i$-нейроне; $y_i$ – реальное значение на $i$-нейроне выходного слоя.

## Пример обучения по методу обратного распространения ошибки

***Пример.*** Просчитать одну итерацию цикла обучения методом обратного распространения ошибки для нейросети, состоящей из двух слоёв, причем в первом слое находится 2 нейрона и используется сигмоидальная функция активации (a=0,9), а во втором – 1 с линейной (k=0,7) функцией активации. В качестве обучающей выборки использовать таблицу истинности для операции «штрих Шеффера» (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

*Решение.*

1. По заданию нейросеть состоит из трех нейронов – два входных, один выходной, значит, синаптических весов 6. Первый слой нейронов имеет сигмоидальную функцию активации, второй – линейную (рис. 18).

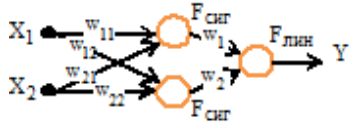



Рис. 18. Исходная многослойная нейросеть

2. По заданию нейросеть бинарная, поэтому на ее входы могут подаваться только 0 и 1, так как входа 2, то возможных комбинаций входных значений будет 4 (обучающая выборка будет состоять из 4-х

векторов). Выход нейронной сети согласно заданию соответствует оператору «штрих Шеффера». Поэтому матрица с обучающей выборкой будет выглядеть следующим образом:

| $x_1$ | $x_2$ | D |
|---|---|---|
| 0 | 0 | 1 |
| ***0*** | ***1*** | ***1*** |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

3. Пусть в качестве вектора обучения будет рассматриваться 2-я строка таблицы.

4. Следуя алгоритму обучения, выполним 5 шагов.

Ш а г 1. Зададим матрицу весов случайным образом из интервала [0,1]:

| $w_{ij}(1)$ | 1 | 2 |
|---|---|---|
| 1 | 0,6 | 0,9 |
| 2 | 0,1 | 0,5 |

| $W_g(1)$ | 1 | 2 |
|---|---|---|
| | 0,3 | 0,8 |

Ш а г 2. Вектор $X=\{0,1\}$, $D=\{1\}$.

Ш а г 3 (прямой проход). Вычисление в циклах выходов всех слоев и получение выходных значений нейронной сети (вектор $Y$).

$S_1=x_1w_{11}+x_2w_{21} = 0\cdot0,6 + 1\cdot0,1 = 0,1;$

$S_2=x_1w_{12}+x_2w_{22} = 0\cdot0,9 + 1\cdot0,5 = 0,5;$

$k = 0,9;$

$$Y_1 = \frac{1}{1+e^{-0,1\cdot0,9}} = 0,5224;$$

$$Y_2 = \frac{1}{1+e^{-0,5\cdot0,9}} = 0,61;$$

$S_3=Y_1w_1+Y_2w_2 = 0,5224\cdot0,3 + 0,61\cdot0,8 = 0,64472;$

$l = 0,7;$

$Y = l\cdot S = 0,4513.$

Ш а г 4 (обратный проход). Изменение весов:

$\eta = 0,7;$

$\delta^2 = (d - Y)Y(1 - Y) = (1 - 0,4513)\cdot0,4613(1 - 0,4513) = 0,3587;$

$w_1(2) = w_1(1) + \eta \cdot \delta^2 \cdot Y_1 = 0{,}3 + 0{,}7 \cdot 0{,}3587 \cdot 0{,}5224 = 0{,}431;$

$w_2(2) = w_2(1) + \eta \cdot \delta^2 \cdot Y_2 = 0{,}8 + 0{,}7 \cdot 0{,}3587 \cdot 0{,}61 = 0{,}953;$

$\delta_1^{\ 1} = Y_1(1 - Y_1) \sum_{l=1}^{H_{k+1}} \delta_l^{k+1} \cdot w_l^{k+1} = Y_1(1 - Y_1)\delta^2 \cdot w_1 =$

$= 0{,}5224(1 - 0{,}5224)0{,}3587 \cdot 0{,}3 = 0{,}0268;$

$\delta_2^{\ 1} = Y_2(1 - Y_2) \sum_{l=1}^{H_{k+1}} \delta_l^{k+1} \cdot w_l^{k+1} = Y_2(1 - Y_2)\delta^2 \cdot w_2 =$

$= 0{,}61(1 - 0{,}61)0{,}3587 \cdot 0{,}3 = 0{,}0682;$

$w_{11}(2) = w_{11}(1) + \eta \cdot \delta_1^{\ 1} \cdot x_1 = 0{,}6 + 0{,}7 \cdot 0{,}0682 \cdot 0 = 0{,}9;$

$w_{12}(2) = w_{12}(1) + \eta \cdot \delta_2^{\ 1} \cdot x_1 = 0{,}9 + 0{,}7 \cdot 0{,}3587 \cdot 0{,}61 = 0{,}9;$

$w_{21}(2) = w_{21}(1) + \eta \cdot \delta_1^{\ 1} \cdot x_2 = 0{,}1 + 0{,}7 \cdot 0{,}0268 \cdot 1 = 0{,}119;$

$w_{22}(2) = w_{22}(1) + \eta \cdot \delta_2^{\ 1} \cdot x_2 = 0{,}5 + 0{,}7 \cdot 0{,}0682 \cdot 1 = 0{,}548.$

Матрица весов примет следующий вид:

| $\boldsymbol{w}_{ij}(2)$ | 1 | 2 |
|---|---|---|
| 1 | 0,6 | 0,9 |
| 2 | 0,119 | 0,548 |

| $\boldsymbol{W}_g(1)$ | 1 | 2 |
|---|---|---|
| | 0,431 | 0,953 |

Ш а г 5. $\varepsilon = \sum_{i=1}^{H} (d_i - y_i)^2 = (1 - 0{,}4513)^2 = 0{,}237.$

Аналогично выполняются последующие итерации, пока среднеквадратичная ошибка не станет меньше заданной величины.

О сходимости необходимо сделать несколько дополнительных замечаний. Во-первых, практика показывает, что сходимость метода обратного распространения весьма медленная. Невысокий темп сходимости является “генетической болезнью” всех градиентных методов, так как локальное направление градиента отнюдь не совпадает с направлением к минимуму. Во-вторых, подстройка весов выполняется независимо для каждой пары образов обучающей выборки. При этом улучшение функционирования на некоторой заданной паре может, вообще говоря, приводить к ухудшению работы на предыдущих образах. В этом смысле нет достоверных (кроме весьма обширной практики применения метода) гарантий сходимости.

Исследования показывают, что для представления произвольного функционального отображения, задаваемого

обучающей выборкой, достаточно всего два слоя нейронов. Однако на практике, в случае сложных функций, использование более чем одного скрытого слоя может давать экономию полного числа нейронов.

## Эволюционные вычисления

Эволюционные алгоритмы (ЭА) – это математические преобразования, которые трансформируют входной поток информации в выходной согласно принятой модели. Модель основана на правилах имитации механизмов природной эволюции, а также на статистическом подходе к исследованию ситуаций и итерационном приближении к искомому решению.

ЭА изначально нацелены на недетерминированность. Случайное решение в среднем ничего интересного собой не представляет, и эффективность его крайне низка. Но стоит только множеству решений в процессе выполнения ЭА начать взаимодействовать между собой, как хорошее решение быстро появляется и прогрессирует. Плюсом ЭА является также применимость к решению прикладных задач, у которых фазовое пространство переменных не обязательно является метрическим. Кроме того, все ЭА обладают свойством массового параллелизма при обработке информации как на уровне организации работы алгоритма, так и на уровне его компьютерной реализации.

ЭА представляют собой общее название группы методов, которые моделируют базовые положения теории биологической эволюции – процессы отбора, мутации и воспроизводства. Согласно этой теории, поведение особей, множество которых принято называть популяцией, определяется окружающей средой, правилами отбора в соответствии с целевой функцией пригодности, причём размножаются, в основном, только наиболее пригодные виды, а рекомбинация и мутация позволяют особям изменяться и приспособляться к среде. К таким алгоритмам с адаптивным поисковым механизмом относятся генетические алгоритмы (ГА), генетическое программирование (ГП), эволюционные стратегии и эволюционное программирование (ЭП).

Несмотря на то, что каждый из этих алгоритмов возник независимо от других, они характеризуются рядом важных общих свойств. Для любого из них формируется исходная популяция особей, которая в последующем подвергается селекции и воздействию

различных генетических операторов (чаще всего скрещиванию и/или мутации), что позволяет находить более хорошие решения. Каждая особь представляет потенциальное решение задачи, которое может отображаться некоторой достаточно сложной структурой данных. Любое решение оценивается по приспособленности. Далее в процессе селекции на очередной итерации формируется новая популяция. Некоторые особи этой новой популяции трансформируются с помощью математических преобразований – «генетических операторов», что позволяет получать новые решения. От эволюционной программы ожидается, что после смены некоторого количества поколений наилучшая особь будет представлять решение, близкое к оптимальному.

В общем виде структура программы эволюционного алгоритма может быть представлена в виде псевдокода:

```
procedure <эволюционный алгоритм>
begin
t=0
<инициализация начальной популяции P(t)
оценивание приспособленности особей из P(t)>
while (not условие завершения) do
begin
t=t+1
<селекция особей из P(t–1) в P(t)
применение генетических операторов
оценивание приспособленности особей из P(t)>
end
end
```

По псевдокоду можно понять общую идею ЭА – наличие базового цикла, включающего следующую последовательность шагов: вычисление целевой функции, оценка качества решений, селективный отбор решений для репродукции и репродукция, т.е. создание новых решений.

Под начальной популяцией понимается некоторое количество получаемых, обычно случайным путем, объектов. В ГА такими объектами выступают векторы («генотипы») генов, где каждый ген может быть битом, числом или неким другим объектом. Алгоритмы эволюционных стратегий оперируют векторами действительных

чисел. В алгоритмах ГП и ЭП роль объектов играют программы, всё лучше и лучше (в соответствии с определенной функцией приспособленности) решающие поставленную оптимизационную или поисковую задачу.

Основными генетическими операторами являются скрещивание и мутация. Мутация – это случайное изменение генотипа. В ГА и эволюционных стратегиях оператор мутации может быть реализован простым добавлением нормально распределенной случайной величины к каждой компоненте вектора. В ГП и ЭП эта операция сильно зависит от способа кодирования выращиваемых программ. Например, при древовидном кодировании она может быть осуществлена случайным изменением информации в узле или добавлением, удалением узла или поддерева. Оператор скрещивания производит рекомбинацию решений-кандидатов, роль которых аналогична роли скрещивания в живой природе. Размножение в разных ЭА определяется по-разному – оно зависит от представления данных. Главное требование к размножению – чтобы потомки имели возможность унаследовать черты обоих родителей, «смешав» их каким-либо способом. На этапе отбора нужно из всей популяции выбрать определённую её долю, которая останется «в живых» на этом этапе эволюции. Есть разные способы проводить отбор. Вероятность выживания особи должна зависеть от значения функции приспособленности. Эта функция должна быть задана так, чтобы по ее значению на данном генотипе можно было судить о степени его «успешности». По итогам селекции из множества особей популяции должно остаться лишь определенное число особей, которые войдут в новую популяцию. Остальные особи погибают.

Практический интерес к моделям эволюционных вычислений объясняется тем, что эти методы позволяют найти достаточно хорошие (близкие к оптимальным) решения очень трудных задач оптимизации за меньшее время, чем другие, обычно применяемые в этих случаях, методы. ЭА оказались достаточно эффективными для решения ряда реальных задач программной инженерии, администрирования, планирования, инженерного проектирования, маршрутизации, управления портфелями ценных бумаг, поиска

оптимальных энергетических состояний химических и молекулярных структур, а также во многих других областях.

## Примеры решения задач с помощью эволюционных алгоритмов

***Пример***. Дано диофантово уравнение a+2b+3c+4d=30 [http://algolist.manual.ru/ai/ga/dioph.php]. Найдите только целые решения (a, b, c, d) этого уравнения, используя ГА.

*Решение*. Для поиска решения в данном случае можно использовать метод грубой силы, подставив все возможные значения a, b, c, d с учетом очевидных ограничений ($1 \le a, b, c, d \le 30$). Однако ГА позволяет найти решение быстрее за счет более «осмысленного» перебора.

Вначале согласно ГА случайно, но с учетом ограничений, выберем начальное поколение хромосом из пяти решений:

| Хромосома | (a,b,c,d) |
|---|---|
| 1 | (1,28,15,3) |
| 2 | (14,9,2,4) |
| 3 | (13,5,7,3) |
| 4 | (23,8,16,19) |
| 5 | (9,13,5,2) |

Чтобы вычислить функцию приспособленности, подставим каждое решение в выражение a+2b+3c+4d. Расстояние от полученного значения до 30 и будет считать целевым значением:

| Хромосома | Функция приспособленности |
|---|---|
| 1 | $\|114 - 30\| = 84$ |
| 2 | $\|54 - 30\| = 24$ |
| 3 | $\|56 - 30\| = 26$ |
| 4 | $\|163 - 30\| = 133$ |
| 5 | $\|58 - 30\| = 28$ |

Меньшие значения функции являются более желательными. Чтобы хромосомы с более подходящими значениями имели большие шансы оказаться родителями, мы должны вычислить, с какой вероятностью (в %) может быть выбрана каждая из них для

скрещивания и производства потомства. Для этого можно, например, взять сумму обратных значений функции, и вычислить вероятность:

| Хромосома | Вероятность оказаться родительской хромосомой |
|---|---|
| 1 | (1/84)/0,135266 = 8,80 % |
| 2 | (1/24)/0,135266 = 30,8 % |
| 3 | (1/26)/0,135266 = 28,4 % |
| 4 | (1/133)/0,135266 = 5,56 % |
| 5 | (1/28)/0,135266 = 26,4 % |

Для выбора пяти пар родителей представим, что у нас есть 10 000-сторонняя игральная кость, на 880 сторонах которой отмечена хромосома 1, на 3080 – хромосома 2, на 2640 сторонах – хромосома 3, на 556 – хромосома 4 и на 2640 сторонах отмечена хромосома 5. Чтобы выбрать первую пару, кидаем кость два раза и выбираем выпавшие хромосомы. Таким же образом выбирая остальных, получаем:

| Моделирование выбора родителей | |
|---|---|
| Хромосома отца | Хромосома матери |
| 3 | 1 |
| 5 | 2 |
| 3 | 5 |
| 2 | 5 |
| 5 | 3 |

Каждый потомок должен содержать информацию о родительских генах. Вообще говоря, это можно обеспечить различными способами. В нашем примере используем простейший одноточечный кроссинговер. Пусть мать содержит следующий набор решений: $a_1,b_1,c_1,d_1$, а отец – $a_2,b_2,c_2,d_2$, тогда возможно 6 различных кроссинговеров (| = разделительная линия разрыва):

| Скрещивания между родителями | | |
|---|---|---|
| Хромосома–отец | Хромосома–мать | Хромосомы–потомки |
| $a_1 \mid b_1,c_1,d_1$ | $a_2 \mid b_2,c_2,d_2$ | $a_1,b_2,c_2,d_2$ и $a_2,b_1,c_1,d_1$ |
| $a_1,b_1 \mid c_1,d_1$ | $a_2,b_2 \mid c_2,d_2$ | $a_1,b_1,c_2,d_2$ и $a_2,b_2,c_1,d_1$ |
| $a_1,b_1,c_1 \mid d_1$ | $a_2,b_2,c_2 \mid d_2$ | $a_1,b_1,c_1,d_2$ и $a_2,b_2,c_2,d_1$ |

Есть достаточно много путей передачи информации потомку, и кросинговер – только один из них. Расположение разделителя может быть абсолютно произвольным, как и то, что отец или мать будут слева от черты. Теперь попробуем проделать это с нашими потомками:

| Моделирование скрещиваний хромосом родителей | | |
|---|---|---|
| Хромосома–отец | Хромосома–мать | Хромосома–потомок |
| (13 \| 5,7,3) | (1 \| 28,15,3) | (13,28,15,3) |
| (9,13 \| 5,2) | (14,9 \| 2,4) | (9,13,2,4) |
| (13,5,7 \| 3) | (9,13,5 \| 2) | (13,5,7,2) |
| (14 \| 9,2,4) | (9 \| 13,5,2) | (14,13,5,2) |
| (13,5 \| 7, 3) | (9,13 \| 5, 2) | (13,5,5,2) |

Вычислим функции приспособленности потомков:

| Хромосома–потомок | Функция приспособленности |
|---|---|
| (13,28,15,3) | \|126 – 30\| = 96 |
| (9,13,2,4) | \|57 – 30\| = 27 |
| (13,5,7,2) | \|57 – 30\| = 22 |
| (14,13,5,2) | \|63 – 30\| = 33 |
| (13,5,5,2) | \|46 – 30\| = 16 |

Среднее значение функции качества потомков оказалась 38,8 (у родителей – 59,4). Следующее поколение может мутировать. Например, мы можем заменить одно из значений какой-нибудь хромосомы на случайное целое от 1 до 30.

Если продолжить процесс таким образом далее, одна хромосома в конце концов достигнет значения функции качества, равного 0, т.е. станет решением диофантова уравнения. Отметим, что системы с большей популяцией (например, 50 вместо 5) сходятся к желаемому уровню более быстро и стабильно.

Программное обеспечение эволюционирует. Эволюция программного обеспечения имеет явную связь с дарвиновской эволюцией. Однако до конца прошлого века сравнительно мало исследований посвящалось применению ЭА поисковой оптимизации в области инженерии программного обеспечения: анализ требований, прогнозное моделирование, проектирование, тестирование, рефакторинг. Хотя еще А. Тьюринг писал о генерации программ (алгоритмов) посредством мутаций и естественного отбора.

***Пример.*** Задачей символьного регрессионного анализа является подбор математического выражения, наилучшим образом описывающего вычислительную модель для принятия решений. В качестве исходных данных в задаче выступает совокупность имеющихся экспериментальных данных или экспертных знаний об объекте, отражающих неизвестную зависимость определенного свойства объекта Y от другого свойства или параметра X со случайной погрешностью, распределенной, как правило, по нормальному закону. По совокупности этих данных требуется подобрать такую функцию (регрессию), которая отображает зависимость Y от X с минимальной погрешностью.

*Решение.* Задача символьной регрессии на практике связана с необходимостью по модели «черного ящика», в которой зафиксированы отдельные входные и выходные связи объекта со средой, но отсутствует информация о внутренней структуре объекта, установить аппроксимирующую функцию объекта в аналитическом виде и определить численные значения ее коэффициентов. Иногда этот процесс называют идентификацией объекта.

Применим ГП для решения задачи символьной регрессии. Пусть, например, регрессионная модель – квадратный полином вида $Y = X^2 – X + 2$, где X принимает значения в диапазоне [–1, 1]. Проблема состоит в том, чтобы автоматически создать компьютерную программу, реализующую данную регрессию.

Для программного вычисления регрессии будем использовать математические операции сложения (+), вычитания (–), умножения (*) и деления (%), множество переменных X, а также множество числовых констант из некоторого диапазона, например, от –2 до +2. Оценку программ различной величины и сложности, автоматически генерируемых с помощью генетических операторов, будем осуществлять в зависимости от того, насколько близки к целевому значению указанного полинома полученные программой результаты. Например, чем меньше среднеквадратичная ошибка, тем лучше программа. В качестве иллюстрации возьмем три случайных хромосомы, представленных на рис. 19 в виде древовидных структур.

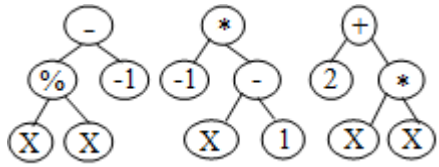



Рис. 19. Исходная популяция хромосом-программ

Структура, представленная на рис. 19 в виде дерева слева, была образована путем случайного выбора операции вычитания в качестве корневой вершины дерева. Затем случайно в качестве аргументов были выбраны операция деления (%) и константа (–1). Для операции деления случайно была выбрана одна переменная X. Программа реализует функцию $Y_1 = 2$. Аналогично были построены два других дерева, реализующих функции $Y_2 = 2–X$ и $Y_3 = X^2 + 2$ соответственно.

На рис. 20 представлены ошибки регрессии для каждой из трех случайно выбранных хромосом начальной популяции (область между кривыми на интервале [– 1, 1]).

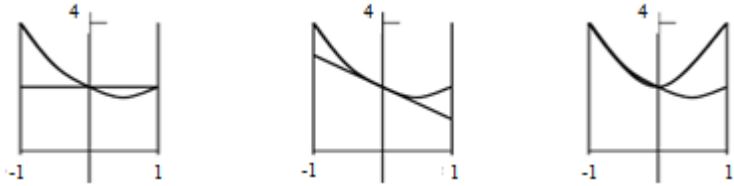



Рис. 20. Ошибка регрессии для трех программ

В частности, для программы, реализующей функцию $Y_1$, область на интервале [– 1, 1] между параболической кривой $Y=X^2 – X+2$ и прямой $Y_1$ графически иллюстрирует величину ошибки (величина интеграла абсолютной ошибки равна 1,0). Интеграл абсолютной ошибки для прямой $Y_2$ равен 1,33, а для параболы $Y_3$ – 1,0.

В качестве критерия завершения ГП выберем достижение ошибки не более 0,1. После того как установлено значение целевой функции пригодности каждой хромосомы, необходимо синтезировать программы-потомки. Произведем мутацию для $Y_1$ и кроссинговер между $Y_2$ и $Y_3$. Получаем деревья, представленные на рис. 21.

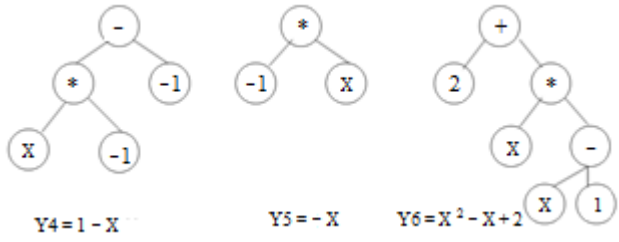



Рис. 21. Новая популяция хромосом-программ

Оператор мутации случайно заменяет в дереве, соответствующем $Y_1$, вершину «%» на вершину «*», а правую вершину «X» на вершину «–1». Перед выполнением одноточечного оператора кроссинговера случайным образом определяются точки кроссинговера в хромосомах, соответствующих $Y_2$ и $Y_3$. Пусть для родительской хромосомы $Y_2$ это будет вершина «–», а для $Y_3$ – правая вершина «X» на дереве, представленном на рис. 21.

Оператор кроссинговера образует два потомка (–X) и ($X^2$–X+2) путем обмена списками между двумя программами при сохранении синтаксической корректности вновь получаемых программ.

Видно, что один из них эквивалентен исходной регрессии, т.е. является алгебраически правильным решением задачи автоматического синтеза компьютерной программы, реализующей вычисление функции квадратного полинома вида $X^2$–X+2 в диапазоне значений X от –1 до +1.

***Пример.*** С помощью эволюционного алгоритма из случайно сгенерированного списка букв синтезировать фрагмент текста известной казачьей песни «*По Дону гуляет*». Для кодирования строки букв использовать кодовую таблицу символов ASCII (в операционной системе Windows), а критерием (целевой функцией) считать число правильно отгаданных букв.

*Решение.* Вначале оценим вероятность получения необходимой строки песенного текста случайным способом. Для каждой позиции текста возможны 32 различные буквы русского алфавита. Таких позиций в заданном выражении 12. Тогда искомая вероятность равна (1/32)**12 ≈ 1,27*10**(– 18).

В таблице ASCII строка «*По Дону гуляет*» преобразуется в следующую хромосому:

[207, 238, 196, 238, 237, 243, 227, 243, 235, 255, 229, 242].

Сгенерируем исходную популяцию, например, из 10 случайных фраз:

[232, 239, 225, 242, 227, 238, 255, 227, 186, 238, 236, 239] (ЦФ=0)
[228, 226, 244, 231, 231, 224, 226, 224, 237, 248, 243, 247] (ЦФ=0)
[252, 241, 243, 228, 228, 225, 246, 225, 234, 186, 230, 246] (ЦФ=0)
[249, 227, 252, 249, 244, 245, 236, 229, 248, 252, 224, 226] (ЦФ=0)
[230, 232, 234, 245, 231, 254, **227**, 245, 238, 247, 242, 232] (ЦФ=1)
[232, 228, 227, 245, 230, 226, 232, 179, 247, **255**, 238, 186] (ЦФ=1)
[232, 248, 231, 235, 248, 179, 235, 186, 224, **255**, 252, **242**] (ЦФ=2)
[239, 248, 236, 229, 179, 233, 232, 244, 249, 245, 252, 230] (ЦФ=0)
[249, 232, 236, 229, 240, 244, **227**, **243**, 230, 244, 254, **242**] (**ЦФ=3**)
[248, 230, 231, 252, 245, 239, 242, 254, 252, **255**, 240, 255] (ЦФ=1)

Если преобразовать эти хромосомы в строки символов, то получится 10 бессмысленных фраз:
ипбтгоягєомп, двфззаваншуч, ьсуддбцбкєжц, щгьщфхмешьав, жикхзюгхочти, идгхжви_чяоє, ишзлш_лєаяьт, пшме_йифщхьж, щимерфгужфют, шжзьхптюьяря.

Лучшее значение целевой функции, равное 3, в исходной популяции дает строка символов <щимерфгужфют>. Запустим ЭА. В таблице приведен список хромосом, которые были наилучшими на каждой из 32-х итераций ЭА.

| Популяция | Строка | ЦФ | Популяция | Строка | ЦФ |
|---|---|---|---|---|---|
| 1 | щимерфгужфют | 3 | 9 | пдшонагубгжт | 6 |
| 2 | ипбтгогуомпт | 3 | 10 | пдшоньгургжт | 6 |
| 3 | рикхгагуншйт | 3 | 11 | пддоньгургжт | 7 |
| 4 | пибхмогужшют | 4 | 12 | пддонигургжт | 7 |
| 5 | ппкомегудгжт | 5 | …… | …… | … |
| 6 | пдморагуягжт | 5 | 30 | подонугуляит | 11 |
| 7 | пемолюгуядот | 5 | 31 | подонугуляит | 11 |
| 8 | пдшонагувгжт | 6 | 32 | подонугуляет | 12 |

## Задачи

1. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной однородной нейронной сети, состоящей из двух нейронов и имеющей пороговую функцию активации (Θ=0,7). В

качестве обучающей выборки использовать таблицу истинности для операций дизъюнкции и импликации (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

2. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной однородной нейронной сети, состоящей из двух нейронов и имеющей линейную функцию активации ($k$ = 0,6). В качестве обучающей выборки использовать таблицу истинности для операций конъюнкции и дизъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

3. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной однородной нейронной сети, состоящей из двух нейронов и имеющей сигмоидальную функцию активации ($a$ = 1). В качестве обучающей выборки использовать таблицу истинности для операций импликации и конъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

4. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной однородной нейронной сети, состоящей из двух нейронов и имеющей функцию активации гиперболический тангенс ($a$ = 1). В качестве обучающей выборки использовать таблицу истинности для операций эквивалентности и импликации (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

5. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной неоднородной нейронной сети, состоящей из двух нейронов и имеющей функции активации: гиперболический тангенс ($a$ = 2) и пороговую функцию (Θ = 0,5). В качестве обучающей выборки использовать таблицу истинности для операций эквивалентности и конъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

6. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной неоднородной нейронной сети, состоящей из двух нейронов и имеющей функции активации: сигмоидальную ($a$ = 1) и линейную ($k$ = 0,6). В качестве обучающей выборки использовать таблицу истинности для операций импликации и конъюнкции (не

использовать первую строчку таблицы). Синаптические веса задать случайным образом.

7. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной неоднородной нейронной сети, состоящей из двух нейронов и имеющей функции активации: линейную ($k$ = 0,7) и пороговую ($\Theta$ = 0,75). В качестве обучающей выборки использовать таблицу истинности для операций конъюнкции и эквивалентности (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

8. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной неоднородной нейронной сети, состоящей из двух нейронов и имеющей функции активации: пороговую ($\Theta$ = 0,8) и сигмоидальную ($a$ = 1). В качестве обучающей выборки использовать таблицу истинности для операций конъюнкции и импликации (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

9. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной неоднородной нейронной сети, состоящей из двух нейронов и имеющей функции активации: гиперболический тангенс ($a$ = 2) и линейную ($k$ = 0,8). В качестве обучающей выборки использовать таблицу истинности для операций дизъюнкции и эквивалентности (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

10. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной неоднородной нейронной сети, состоящей из двух нейронов и имеющей функции активации: гиперболический тангенс ($a$ = 2) и сигмоидальную ($a$ = 0,9). В качестве обучающей выборки использовать таблицу истинности для операций импликации и дизъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

11. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной однородной нейронной сети, состоящей из трех нейронов и имеющей функцию активации гиперболический тангенс ($a$ = 3). В качестве обучающей выборки использовать таблицу истинности для $X_1\&X_2\rightarrow X_3$, $X_1\&X_2$ и $X_2\rightarrow X_3$ (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

12. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной однородной нейронной сети, состоящей из трех нейронов и имеющей сигмоидальную функцию активации ($a$ = 1). В качестве обучающей выборки использовать таблицу истинности для $X_1 \to X_2 \& X_3$, $X_1 \& X_2$ и $X_1 \& X_3$ (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

13. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной однородной нейронной сети, состоящей из трех нейронов и имеющей линейную функцию активации ($k$ = 0,9). В качестве обучающей выборки использовать таблицу истинности для $X_3 \to X_1 \& X_2$, $X_2 \& X_3$, $X_2 \to X_3$ (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

14. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной однородной нейронной сети, состоящей из трех нейронов и имеющей пороговую функцию активации ($\Theta$ = 0,4). В качестве обучающей выборки использовать таблицу истинности для $(X_2 \to X_1) \& X_3$ (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

15. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной аналоговой однородной нейронной сети, состоящей из трех нейронов и имеющей линейную функцию активации ($k$ = 0,9). Синаптические веса и обучающую выборку задать случайным образом (не нули).

16. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной аналоговой однородной нейронной сети, состоящей из трех нейронов и имеющей сигмоидальную функцию активации ($a$ = 0,8). Синаптические веса и обучающую выборку задать случайным образом (не нули).

17. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной аналоговой однородной нейронной сети, состоящей из трех нейронов и имеющей пороговую функцию активации ($\Theta$ = 0,8). Синаптические веса и обучающую выборку задать случайным образом (не нули).

18. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной аналоговой однородной нейронной сети, состоящей из трех нейронов и имеющей функцию активации – гиперболический

тангенс ($a$ = 1). Синаптические веса и обучающую выборку задать случайным образом (не нули).

19. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной аналоговой неоднородной нейронной сети, состоящей из трех нейронов и имеющей функции активации: сигмоидальную ($a$ = 1), линейную ($k$ = 0,8) и пороговую (Θ = 0,5). Синаптические веса и обучающую выборку задать случайным образом (не нули).

20. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной аналоговой неоднородной нейронной сети, состоящей из трех нейронов и имеющей функции активации: гиперболический тангенс ($a$ = 1), сигмоидальную ($a$ = 0,8) и пороговую (Θ = 0,6). Синаптические веса и обучающую выборку задать случайным образом (не нули).

21. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 2 нейрона, а во втором – 1. Функция активации нейронов сети – пороговая (Θ = 0,6) функция. В качестве обучающей выборки использовать таблицу истинности для операции «исключающее ИЛИ». Синаптические веса задать случайным образом.

22. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 2 нейрона, а во втором – 1. Функция активации нейронов сети – сигмоидальная ($a$ = 1) функция. В качестве обучающей выборки использовать таблицу истинности для операции импликации. Синаптические веса задать случайным образом.

23. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 2 нейрона, а во втором – 1. Функция активации нейронов сети – линейная ($k$ = 0,6) функция. В качестве обучающей выборки использовать таблицу истинности для операции «штрих Шеффера». Синаптические веса задать случайным образом.

24. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух

слоёв, причем в первом слое находится 2 нейрона, а во втором – 1. Функция активации нейронов сети – гиперболический тангенс ($a$ = 1). В качестве обучающей выборки использовать таблицу истинности для операции «стрелка Пирса». Синаптические веса задать случайным образом.

25. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 2 нейрона и используется сигмоидальная функция активации ($a$ = 0,9), а во втором – 1, пороговая ($\Theta$ = 0,7). В качестве обучающей выборки использовать таблицу истинности для операции «исключающее или». Синаптические веса задать случайным образом.

26. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 2 нейрона и используется линейная функция активации ($k$ = 0,5), а во втором – 1, сигмоидальная ($a$ = 0,7) функция. В качестве обучающей выборки использовать таблицу истинности для операции импликации. Синаптические веса задать случайным образом.

27. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 2 нейрона и используется пороговая функция активации ($\Theta$ = 0,4), а во втором – 1, линейная ($k$ = 0,6) функция. В качестве обучающей выборки использовать таблицу истинности для операции «штрих Шеффера» нейросети, состоящей из двух слоёв, причем в первом слое находится 2 нейрона и используется пороговая функция активации ($\Theta$ = 0,6), а во втором – 1, гиперболический тангенс ($a$ = 2). В качестве обучающей выборки использовать таблицу истинности для операции «стрелка Пирса». Синаптические веса задать случайным образом.

29. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 3 нейрона, а во втором – 2. Функция активации нейронов сети – линейная ($k$ = 0,6) функция. Синаптические веса и обучающую выборку задать случайным образом (не нули).

30. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 3 нейрона, а во втором – 2. Функция активации нейронов сети – сигмоидальная ($a = 1$) функция. Синаптические веса и обучающую выборку задать случайным образом (не нули).

31. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 3 нейрона, а во втором – 2. Функция активации нейронов сети – пороговая ($\Theta = 0{,}65$) функция. Синаптические веса и обучающую выборку задать случайным образом (не нули).

32. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 3 нейрона, а во втором – 2. Функция активации нейронов сети – гиперболический тангенс ($a = 3$). Синаптические веса и обучающую выборку задать случайным образом (не нули).

33. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 2 нейрона и используется сигмоидальная функция активации ($a = 0{,}9$), во втором – 2, пороговая ($\Theta = 0{,}7$). Синаптические веса и обучающую выборку задать случайным образом (не нули).

34. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 2 нейрона и используется линейная функция активации ($k = 0{,}5$), во втором – 2, сигмоидальная ($a = 0{,}7$) функция. Синаптические веса и обучающую выборку задать случайным образом (не нули).

35. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 2 нейрона и используется пороговая функция активации ($\Theta = 0{,}4$), во втором – 2, линейная ($k = 0{,}6$) функция. Синаптические веса и обучающую выборку задать случайным образом (не нули).

36. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, причем в первом слое находится 2 нейрона и используется пороговая функция активации ($\Theta = 0{,}6$), во втором – 1, гиперболический тангенс ($a = 2$). Синаптические веса и обучающую выборку задать случайным образом (не нули).

37. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из трех слоёв, использующей пороговую функцию активации ($\Theta = 0{,}5$), в первом слое 2 нейрона, во втором – 2, в третьем – 1. Синаптические веса и обучающую выборку задать случайным образом (не нули).

38. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, использующей пороговую функцию активации ($\Theta = 0{,}5$), в первом слое 3 нейрона, во втором – 1. В качестве обучающей выборки использовать таблицу истинности для $X_1 \rightarrow X_2 \& X_3$. Синаптические веса задать случайным образом.

39. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из двух слоёв, использующей сигмоидальную функцию активации ($a = 0{,}5$), в первом слое 3 нейрона, во втором – 1. В качестве обучающей выборки использовать таблицу истинности для $(X_1 \rightarrow X_2) \& X_3$. Синаптические веса задать случайным образом.

40. Представить в префиксной форме и в виде древовидной структуры для работы алгоритма генетического программирования следующую арифметическую формулу:

$2\pi + ((x+3) - y/(5+1))$.

41. Создать у напитка цветовую гамму, похожую на вишневое бренди. Ингридиенты: вода плюс красный, желтый и синий красители. Хромосома <*w, r, g, b*>. Объем напитка 30 мл. Селекция (1/8). Функцию качества определяют студенты, которые смешивают ингридиенты и сравнивают цвет напитка на степень его совпадения с целевым цветом. Критерий останова – студент доволен цветом созданного напитка.

42. Дана следующая функция Экли ($n = 30$):

$$f(x) = -20\exp[-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}] - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e, \quad -30 \lhd x_i \lhd 30.$$

(30, 200)-селекция. Алгоритм эволюционных стратегий. Останов после 200 000 оценок функции.

43. Решить с помощью эволюционного алгоритма задачу о размещении на шахматной доске восьми ферзей.

## КОНТРОЛЬНЫЕ ВОПРОСЫ (ТЕСТЫ)

1. Если существует множество значений весов, которые обеспечивают конкретное различение образов, то в конечном итоге алгоритм обучения персептрона приводит либо к этому множеству, либо к эквивалентному ему множеству, такому, что данное различение образов будет достигнуто: а) верно; б) неверно.

2. Классами нейронных сетей прямого распространения являются: а) многослойный персептрон; б) однослойная сеть Хопфилда; в) однослойный персептрон; г) сеть Кохонена; д) сеть радиальных базисных функций; е) соревновательные сети.

3. Базовый нелинейный элемент нейронной сети – это: а) синапс; б) нейрон; в) сумматор; г) персептрон.

4. Корректировка весов $w_j$ при обучении нейросети по правилам Хебба заключается в следующем: а) если выход неправильный и равен 0, то уменьшить веса для $x_j = 1$; б) если выход неправильный и равен 0, то увеличить веса для $x_j = 1$; в) если выход неправильный и равен 1, то увеличить веса для $x_j = 1$; г) если выход неправильный и равен 1, то уменьшить веса для $x_j = 1$.

5. Обучение ИНС – это: а) поиск экстремума функции, отображающей вход-выходное взаимодействие; б) ответы на вопросы и решение задач; в) двусторонний процесс передачи понятной информации.

6. Соответствие функций активации нейрона:

Название функции: а) линейная; б) логистическая; в) пороговая.

Функция: 1) $Y(x) = \begin{cases} 1, \textit{ если } x \geq \Theta, \\ 0, \textit{ иначе.} \end{cases}$ 2) $Y(x) = \begin{cases} 0, \textit{ если } x \leq -0{,}5, \\ 1, \textit{ если } x \geq 0{,}5, \\ x, \textit{ иначе.} \end{cases}$

3) $Y(x) = \dfrac{1}{1 + \exp(-ax)}$.

7. Математическая модель искусственного нейрона представляет собой: а) мультиплексор, который подключает один из нескольких входов к выходу; б) взвешенный сумматор с единственным выходом, который определяется через его входы и матрицу весов; в) функцию, отображающую значения, меньше заданного, в единицу, а значения, больше заданного, в нуль; г) аддитивную меру измерения количества информации; д) сумматор чисел, представленных входными сигналами.

8. Если существует множество значений весов, обеспечивающих распознавание образов, то в конечном итоге алгоритм обучения приведет к этому множеству. Это теорема: а) Байеса; б) Ковера о распознаваемости образов; в) о сходимости персептрона; г) выборок Котельникова; д) Гёделя о полноте.

9. Первый нейрокомпьютер разработал а) У. Маккалок; б) М. Минский; в) Ф. Розенблатт; г) нет правильного ответа.

10. Задачи, которые не решают нейронные сети: а) классификации; б) аппроксимации; в) памяти, адресуемой по содержанию; г) маршрутизации; д) управления; е) кодирования.

11. Функция, которую не может решить однослойная нейронная сеть: а) логическое «НЕ»; б) суммирование; в) логическое «Исключающее ИЛИ»; г) произведение; д) логическое «ИЛИ».

12. Что из нижеперечисленного относится к персептрону: а) однослойная нейронная сеть; б) нейронная сеть прямого распространения; в) многослойная нейронная сеть; г) нейронная сеть с обратными связями; д) создан Ф. Розенблаттом; е) создан У. МакКаллоком и В. Питтсом.

13. Книгу «Персептроны» написал: а) У. Маккалок и В. Питтс; б) М. Минский и С. Паперт; в) Ф. Розенблатт.

14. Какую нейронную сеть обучают с помощью дельта-правила: а) однослойную нейронную сеть; б) нейронную сеть прямого распространения; в) нейронную сеть с обратными связями; г) сеть Хопфилда, д) нет правильного ответа.

15. Нейросети, являющиеся рекуррентными: а) персептрон; б) сеть Хопфилда; в) сеть радиальных базисных функций; г) нет правильного ответа.

16. Какие определения не являются моделями представления знаний: а) продукционные модели; б) фреймы; в) имитационные модели; г) семантические сети; д) формально-логические модели.

17. Процесс математических преобразований, позволяющих трансформировать входной поток информации в выходной по правилам, основанным на имитации механизмов, инспирированных природными системами, статистическом подходе к исследованию ситуаций и итерационном приближении к искомому решению – это: а) адресация; б) компиляция; в) символьные вычисления; г) трансляция; д) эволюционные вычисления; е) другое.

18. Эволюционные алгоритмы отличаются от других оптимизационных и поисковых процедур тем, что: а) используют целевую функцию, а не ее приращения; б) дают точное решение задачи; в) используют вероятностные правила; г) имеют линейную временную сложность.

19. Разновидность эволюционных алгоритмов, направленная на решение задач автоматического синтеза программ на основе входных обучающих данных путем индуктивного вывода, целью является поиск неизвестной программы по известным входным данным и выходным образцам: а) генетический алгоритм; б) генетическое программирование; в) градиентный алгоритм; г) эволюционное программирование; д) эволюционные стратегии.

20. Записи вида +( / (a,+(b, c)), *(x, – (y, z))) в префиксной форме древовидной структуры в генетическом программировании соответствует: а) (((b + c) /a) + (x*(z – y))); б) (((b + c) /a) + (x*(y – z))); в) ((a / (b + c)) + (x * (y – z))); г) ((a / (b + c)) * (x + (y – z))); д) ((a * (b + c)) / (x * (y – z))).

21. Основные элементы эволюционного алгоритма: а) генотип; б) композиция операторов; в) критерий качества; г) начальная популяция; д) правило резолюции; е) селективный отбор; ж) сумматор; з) фаззификация; и) фенотип.

22. Правильная последовательность шагов цикла эволюционных вычислений: а) вычисление целевой функции; б) оценка качества решений; в) репродукция путем создания новых решений; г) селективный выбор для репродукции.

23. Процесс, посредством которого альтернативные решения с «лучшим» значением целевой функции имеют больше возможностей для воспроизводства потомков, называется: а) мутация; б) отбор; в) репродукция; г) селекция; д) эволюция.

24. Процедура, устанавливающая порядок обмена хромосом между популяциями называется: а) диффузия; б) миграция; в) мутация; г) сегрегация; д) транслокация.

25. Критериями остановки алгоритма эволюционных вычислений являются: а) комбинаторный взрыв; б) малая разница между лучшим и худшим значением целевой функции для текущей популяции; в) максимальное число шагов эволюции; г) минимальный средний риск; д) отсутствие прогресса в смысле заметного улучшения значений целевой функции.

# 4. БАЙЕСОВСКИЕ И НЕЧЕТКИЕ МОДЕЛИ

## Основные определения по теме

**Коэффициент уверенности** – оценка, предоставляемая экспертной системой, которая указывает на вероятность того, насколько заключение системы является правильным. Также степень уверенности эксперта в истинности заключения, если исходная предпосылка истинна. Используется в теории уверенности (Буханан, Шортлифф) для оценки уровня доверия к гипотезе.

**Лингвистическая переменная**. Её значениями являются нечёткие подмножества слов или предложений на естественном или искусственном языке. Задаётся набором {X, T(X), U, G, M}, где X – имя переменной; T(X) – терм-множество лингвистических значений переменной; U – универсальное множество; G – синтаксическое правило для порождения значений переменной X; M – семантическое правило, которое ставит в соответствие каждому значению переменной её смысл.

## НЕ–факторы знаний

Необходимым условием решения какой-либо задачи в рамках процедурного подхода является наличие четкого алгоритма. Поэтому автоматизация коснулась, прежде всего, так называемых формализованных задач, алгоритм решения которых хорошо известен.

Однако существует широкий класс неформализованных задач, характеризующихся одной или несколькими из следующих особенностей:

- алгоритм решения задачи неизвестен или не может быть использован из-за ограниченности ресурсов компьютера;
- задача не может быть определена в числовой форме;
- цели задачи не могут быть выражены в терминах точно определенной целевой функции.

Заметим, что по объему этот класс значительно превосходит класс формализованных задач.

Форма представления знаний оказывает существенное влияние на характеристики и свойства СИИ, поэтому представление знаний является одной из наиболее важных проблем, характерных для систем,

основанных на знаниях. Поскольку логический вывод и действия над знаниями производятся программным путем, знания не могут быть представлены непосредственно в том виде, в котором они используются человеком (например, в виде простого текста). В связи с этим для представления знаний разрабатываются формальные модели представления знаний.

В реальных условиях знания, которыми располагает человек, всегда в какой-то степени неполны, приближенны, ненадежны. Тем не менее людям на основе таких знаний все же удается делать достаточно обоснованные выводы и принимать разумные решения. Следовательно, чтобы интеллектуальные системы были действительно полезны, они должны быть способны учитывать неполную определенность знаний и успешно действовать в таких условиях. Неопределенность (НЕ–факторы) может иметь различную природу.

Наиболее распространенный тип недостаточной определенности знаний обусловлен объективными причинами:

- действием случайных и неучтенных обстоятельств,
- неточностью измерительных приборов,
- ограниченными способностями органов чувств человека,
- отсутствием возможности получения необходимых свидетельств.

В таких случаях люди в оценках и рассуждениях прибегают к использованию вероятностей, допусков и шансов.

Другой тип неопределенности, можно сказать, обусловлен субъективными причинами:

- нечеткостью содержания используемых человеком понятий (например, "толпа"),
- неоднозначностью смысла слов и высказываний (например, "ключ" или знаменитое "казнить нельзя помиловать").

Неоднозначность смысла слов и высказываний часто удается устранить, приняв во внимание контекст, в котором они употребляются, но это тоже получается не всегда или не полностью.

Таким образом, неполная определенность и нечеткость имеющихся знаний – скорее типичная картина при анализе и оценке положения вещей, при построении выводов и рекомендаций, чем

исключение. В процессе исследований по искусственному интеллекту для решения этой проблемы выработано несколько подходов.

Самым первым можно считать использование эвристик в решении задач, в которых достаточно отдаленный прогноз развития событий невозможен (например, в шахматной игре).

Серьезное внимание этой проблеме стали уделять при создании экспертных систем и первым здесь был применен вероятностный подход (ЭС PROSPECTOR), поскольку теория вероятностей и математическая статистика в тот период были уже достаточно развиты и весьма популярны.

Однако проблемы, возникшие на этом пути, заставили обратиться к разработке особых подходов к учету неопределенности в знаниях непосредственно для ЭС (коэффициенты уверенности в системах MYCIN и EMYCIN).

В дальнейшем исследования в этой области привели к разработке нечеткой логики, основы которой были заложены Л. Заде.

В решении рассматриваемой проблемы применительно к ЭС, построенным на основе продукционных правил, выделяются четыре основных вопроса:

1. Как количественно выразить достоверность посылок?

2. Как выразить степень поддержки заключения конкретной посылкой?

3. Как учесть совместное влияние нескольких посылок на заключение?

4. Как строить цепочки умозаключений в условиях неопределенности?

На языке продукций эти вопросы приобретают следующий смысл. Будем обозначать ct(A) степень уверенности в A (от англ. *certainty* – уверенность).

Тогда первый вопрос заключается в том, как количественно выразить степень уверенности ct(A) в истинности посылки A.

Второй вопрос связан с тем, что истинность посылки A в продукции A→C может не всегда влечь за собой истинность заключения C. Степень поддержки заключения C посылкой A в продукции A→C обозначим через ct(A→C).

Третий вопрос обусловлен тем, что одно и то же заключение С может в различной степени поддерживаться несколькими посылками (например, заключение С может поддерживаться посылкой А посредством продукции А→С с уверенностью ct(A→C) и посылкой В посредством продукции В→С с уверенностью ct(B→C)). В этом случае возникает необходимость учета степени совместной поддержки заключения несколькими посылками.

Последний вопрос вызван необходимостью оценки степени достоверности вывода, полученного посредством цепочки умозаключений (например, вывода С, полученного из посылки А применением последовательности продукций А→В, В→С, обеспечивающих степени поддержки, соответственно, ct(A→B) и ct(B→C) ).

## БАЙЕСОВСКИЙ ПОДХОД

Вероятностный (байесовский) подход не является единственным подходом к построению выводов на основе использования вероятностей, но он представляется удобным в условиях, когда решение приходится принимать на основе части свидетельств и уточнять по мере поступления новых данных.

В сущности, Байес исходит из того, что любому предположению может быть приписана некая ненулевая априорная (от лат. a *priori* – из предшествующего) вероятность того, что оно истинно, чтобы затем путем привлечения новых свидетельств получить апостериорную (от лат. *a posteriori* – из последующего) вероятность истинности этого предположения. Если выдвинутая гипотеза действительно верна, новые свидетельства должны способствовать увеличению этой вероятности, в противном же случае должны ее уменьшать.

Примем для дальнейших рассуждений следующие обозначения:

P(H) – априорная вероятность истинности гипотезы H (от англ. *Hypothesis* – гипотеза);

P(H:E) – апостериорная вероятность истинности гипотезы H при условии, что получено свидетельство E (от англ. *Evidence* – свидетельство);

P(E:H) – вероятность получения свидетельства E при условии, что гипотеза H верна;

P(E:неH) – вероятность получения свидетельства E при условии, что гипотеза H неверна.

По определению условных вероятностей имеем:

$$P(H:E)=\frac{P(H\ и\ E)}{P(E)} \quad \text{и} \quad P(E:H)=\frac{P(E\ и\ H)}{P(H)}.$$

Учитывая, что P(H и E)=P(E и H), получаем теорему Байеса:

$$P(H:E)=\frac{P(E:H)P(H)}{P(E)}.$$

Учитывая, что P(E) = P(E:H)P(H) + P(E:неH)P(неH) и P(неH) = = 1– P(H), получаем формулу, позволяющую уточнять вероятность истинности проверяемой гипотезы H с учетом полученного свидетельства E:

$$P(H:E)=\frac{P(E:H)P(H)}{P(E:H)\,P(H)+P(E:не\ H)(1-P(H))},$$

$$P(H_i:E)=\frac{P(E:H_i)\times P(H_i)}{\sum_{k=1}^{m}P(E:H_k)\times P(H_k)},$$

$$P(H_i:E_1,E_2,...,E_n)=\frac{P(E_1:H_i)\times P(E_2:H_i)\times ...\times P(E_n:H_i)\times P(H_i)}{\sum_{k=1}^{m}P(E_1:H_k)\times P(E_2:H_k)\times ...\times P(E_n:H_k)\times P(H_k)}.$$

Здесь обнаруживаются достоинства байесовского метода. Первоначальная (априорная) оценка вероятности истинности гипотезы P(H) могла быть весьма приближенной, но она позволила путем учета свидетельства E получить более точную оценку P(H:E), которую можно теперь использовать в качестве обновленного значения P(H) для нового уточнения с привлечением нового свидетельства. Иначе говоря, процесс уточнения вероятности P(H) можно повторять снова и снова с привлечением все новых и новых свидетельств, каждый раз обращаясь к одной и той же формуле. В конечном счете, если свидетельств окажется достаточно, можно получить окончательный вывод об истинности (если окажется, что P(H) близка к 1) или ложности (если окажется, что P(H) близка к 0) гипотезы H.

***Пример***. Пусть некий тест на какую-нибудь болезнь имеет вероятность успеха 95 % (т.е. 5 % – вероятность как позитивной, так и негативной ошибки). Всего болезнь имеется у 1 % респондентов. Пусть некий человек получил позитивный результат теста (тест говорит, что он болен). С какой вероятностью он действительно болен?

*Решение.* Используем теорему Байеса:

$$p = \frac{0{,}95 * 0{,}01}{0{,}95 * 0{,}01 + 0{,}05 * 0{,}99} = 0{,}16.$$

*Ответ*. Вероятность $p$=0,16.

***Пример***. Есть две полных вазы конфет. В 1-й вазе 10 шоколадных и 30 простых конфет, в то время как во 2-й вазе по 20 конфет каждого сорта. Ваша подруга выбирает вазу наугад и затем выбирает конфету наугад. Выбранная конфета оказалась простой. Нет никакой причины полагать, что Ваша подруга предпочитает одну вазу другой. Это же можно сказать и о конфетах. Насколько вероятно, что Ваша подруга выбрала конфету из 1-й вазы?

*Решение*. Интуитивно, решение кажется ясным. Точный ответ дается теоремой Байеса. Пусть H1 – выбор 1-й вазы, а H2 – выбор 2-й вазы. Предполагается, что вазы идентичны и P(H1) = P(H2) = 0,5.

Событие – наблюдение простой конфеты. Из содержания ваз, мы знаем что P(E/H1)=30/40=0,75; P(E/H2)=20/40=0,5. По формула Байеса тогда находим

$$p\,(H_1|E) = \frac{0{,}75 \times 0{,}5}{0{,}75 \times 0{,}5 + 0{,}5 \times 0{,}5} = 0{,}6.$$

*Ответ*. До того как мы наблюдали конфету, вероятность, которую мы назначили для подруги, выбиравшую 1-ю вазу, была априорной вероятностью P(H1)=0,5. После наблюдения конфеты нужно пересмотреть вероятность P(H1/E), которая теперь равна 0,6.

Шансы и вероятности связаны между собой следующей формулой:

$$\mathrm{O}(H) = \frac{P(H)}{1 - P(H)}.$$

Если же перейти к логарифмам величин, а в базе знаний хранить логарифмы отношений P(E:H)/P(E:неH), то все вычисления сводятся просто к суммированию, поскольку

$$\ln |O(H:E)| = \ln |P(E:H)/P(E:неH)| + \ln |O(H)|.$$

Против использования шансов есть несколько возражений, главное из которых состоит в том, что крайние значения шансов равны "плюс" и "минус" бесконечности, тогда как для вероятностей это 0 и 1. Поэтому шансы использовать удобно в тех случаях, когда ни одна из гипотез не может быть ни заведомо достоверной, ни заведомо невозможной.

Как принцип байесовский подход выглядит прекрасно, но есть и несколько проблем, связанных с его применением.

Первое замечание касается возможности вычисления величины P(E). Эту величину легко определить, если есть возможность вычислить P(E:неH), а это не всегда можно сделать.

Одна из возможностей обойти это затруднение состоит в переходе к полной группе событий, однако это не спасает положение, если состав полной группы событий неизвестен. Можно пользоваться и грубыми оценками, если сохраняется точность диагноза. Кроме того, если P(H) уточняется в ходе работы, то P(E) можно тоже уточнять.

Второе замечание касается используемого в этом подходе предположения о независимости свидетельств. С теоретической точки зрения это замечание очень серьезно, но поскольку в конце процесса диагноза нас интересуют не столько точные значения вероятностей (это больше беспокоит статистиков), сколько соотношения вероятностей, то при одинаковом порядке ошибочности оценок вероятностей гипотез для практики более важной оказывается правильность общей картины, создаваемой экспертной системой.

Наиболее развитой и успешной оказалась схема, реализованная Шортлиффом в MYCIN, представляющая собой основанную на здравом смысле модификацию байесовского метода, позволяющая достаточно просто решить поставленные во вступлении к данному разделу четыре основных вопроса (Как количественно выразить достоверность посылок? Как выразить степень поддержки заключения конкретной посылкой? Как учесть совместное влияние нескольких

посылок на заключение? Как строить цепочки умозаключений в условиях неопределенности?).

*Коэффициенты уверенности для правила с одной посылкой.* В данном случае речь идет о вычислении коэффициентов уверенности для правил вида

E→C ("если E, то C").

Если иметь возможность присваивать коэффициент уверенности как посылке, так и импликации, то их можно использовать для оценки степени определенности заключения, выводимого по данному правилу. Шортлифф применяет в данном случае коэффициенты уверенности подобно вероятностям: коэффициент уверенности *ct(E)* в посылке подобен *p(E);* коэффициент уверенности *ct(E→C)* в импликации подобен *p(C:E).* Для определения коэффициента уверенности в заключении Шортлифф использует схему:

ct(посылки)·ct(импликации)=ct(заключения).

*Логические комбинации посылок в одном правиле*. Посылкой в правиле считается все, что находится между ЕСЛИ и ТО. В MYCIN избран принцип: делать все правила простыми. Тогда простейшими логическими комбинациями посылок являются их конъюнкция или дизъюнкция, т.е. правила вида

(E1&E2)→C или (E1v E2)→C.

*Коэффициент уверенности конъюнкции посылок* в MYCIN принято оценивать по наименее надежному свидетельству:

ct(E1&E2) = min {ct(E1),ct(E2)}.

Соответственно *коэффициент уверенности дизъюнкции* посылок в MYCIN принято оценивать по наиболее надежному свидетельству:

ct(E1v E2) = max {ct(E1),ct(E2)}.

*Поддержка одного заключения множеством правил*. В MYCIN применена схема, подобная схеме вычисления суммарной вероятности нескольких независимых событий.

Пусть имеются правила E1→C и E2→C, первое из которых обеспечивает поддержку заключения C с уверенностью ct1, а второе – с уверенностью ct2. Если обе посылки E1 и E2 верны, эти два правила совместно должны обеспечивать заключению C большую поддержку,

чем каждое из них в отдельности. В MYCIN это достигается вычислением степени совместной поддержки по следующей формуле:

$$ct = ct1 + ct2 – ct1 \bullet ct2.$$

При наличии более двух правил, поддерживающих одно и то же заключение, их совместное влияние может быть учтено последовательным применением этой схемы для объединения суммарной поддержки уже учтенных правил с поддержкой еще не учтенного правила.

Использованная Шортлиффом схема объединения поддержки одного заключения множеством правил позволяет учитывать коэффициенты уверенности поддерживающих правил в произвольном порядке и объединять их по мере поступления свидетельств.

*Биполярные схемы для коэффициентов уверенности.* Коэффициенты уверенности, примененные в MYCIN, представляют собой грубое подобие вероятностей. В усовершенствованной системе EMYCIN для выражения степени определенности использован интервал [–1,+1] в следующем смысле: **+1** – полное доверие посылке или заключению; **0** – отсутствие знаний о посылке или заключении; **–1** – полное недоверие посылке или заключению.

Промежуточные значения выражают степень доверия или недоверия к ситуации. Все описанные для однополярных коэффициентов процедуры здесь тоже имеют место, но при вычислении max и min учитываются знаки при величинах коэффициентов (например, что $+0{,}1 > – 0{,}2$). Биполярность коэффициентов повлияла и на вид используемых формул.

Так, для вычисления коэффициента уверенности отрицания посылки достаточно лишь поменять знак коэффициента, т.е. **ct(неE) = = –ct(E).**

Процедура расчета степени поддержки заключения несколькими правилами тоже претерпела соответствующие изменения:

а) если оба коэффициента положительны, то

$$ct = ct1 + ct2 – ct1 \bullet ct2;$$

б) если оба коэффициента отрицательны, то

$$ct = ct1 + ct2 + ct1 \bullet ct2;$$

в) если один из коэффициентов положителен, а другой отрицателен, то

$$ct = \frac{ct_1 + ct_2}{1 - \min(|ct_1| \cdot |ct_2|)},$$

при этом, если один из коэффициентов равен +1, а другой –1, то ct=0.

Работа с биполярными коэффициентами может привести к нереальным результатам, если правила сформулированы неточно.

Применение упрощенного Байесовского подхода может приводить к вполне удовлетворительным результатам. Примером являются антиспам-фильтры на основе формулы Байеса. Выявление спама – это задача выбора и принятия решения. Качество принятия решений характеризуется ошибками 1-го рода (пропущен спам) и 2-го рода (обычное письмо принято за спам). Для обучения фильтра успешно используют статистическую оценочную базу из двух наборов почты (спамовской и обычной). Например, в разработанном Грэмом прототипе фильтра каждому слову или тегу письма присваивается вероятность его наличия в спаме (высокая вероятность для слов вроде sexy, promotion или термом ff0000 – код ярко-красного цвета в языке HTML). Отсев спама основан на формуле Байеса. Оценка принадлежности спаму измеряется по шкале от 0 до 1. Математики называют данный поход «наивным», поскольку принимается не очень корректная гипотеза о независимости появления отдельных слов в письме. Для фильтрации не надо вычислять оценки всех слов, а только наиболее значимых (примерно 15 слов).

До сих пор речь шла о ситуациях, когда заключение отделялось от посылки одним шагом рассуждений. Более типична ситуация, когда вывод от посылок отделен рядом промежуточных шагов рассуждений.

Принцип последовательного учета свидетельств работает успешно лишь для случая монотонных рассуждений. Но часто встречаются ситуации, когда очередной факт опровергает ранее сделанные выводы. Другая трудность применения байесовского вывода – "незамкнутость мира": птицы летают, но не все; можно перечислить все предметы в комнате, но нельзя перечислить то, что вне ее. В таких случаях в ЭС принимается "гипотеза закрытого мира": все, что вне его, то – ложь.

## НЕЧЕТКИЙ ПОДХОД

Рассмотрим теперь другой, отличный от вероятностного, подход к представлению неопределенности знаний – нечеткую логику Л. Заде. Она уже с успехом применяется в самых разных приложениях:

- десятках промышленных изделий – от систем управления электропоездами и боевыми вертолетами до пылесосов и стиральных машин, автофокусирования видеокамер и фотоаппаратов;
- комплексном моделировании системы здравоохранения и социального обеспечения для оценки и оптимизации затрат на социальные нужды;
- экспертных системах поддержки принятия решений;
- банковской и финансовой сфере, в области политического и экономического анализа для создания моделей разных экономических, политических, биржевых ситуаций;
- более 100 пакетах с нечеткой логикой, трех десятках СУБД с функцией нечеткого поиска.

Нечеткая логика придумана для того, чтобы позволить программам работать в диапазоне (0, 1) различных степеней истины. Ее преимущества проявляются, когда система анализируется с помощью нечетких лингвистических переменных (низкий, высокий и т.п.). Нечеткая логика имеет свою аксиоматику и набор базовых операторов, действующих несколько иначе, чем аналогичные булевы операторы. Поскольку человеческая логика сама по себе является приблизительной, то и нечеткая логика имеет огромное значение при создании программного обеспечения, так как вместо четких знаний применяются экспертные знания, а не детерминированные алгоритмы.

Нечетким множеством Ã, по Заде, называется множество, определенное на произвольном непустом множестве X как множество пар вида

$$\tilde{A} = \{\mu_{\tilde{A}}(x)/x\}, \text{ где } x \in X, \mu_{\tilde{A}}(x) \in [0,1].$$

Множество $X$ называется базовым множеством или базовой шкалой, если множество $X$ линейно упорядочено.

Функция $\mu_{\tilde{A}}(x)$: $X \rightarrow [0, 1]$ называется *функцией принадлежности* (ФП) множества $\tilde{A}$, а величина $\mu_{\tilde{A}}(x)$ – степенью принадлежности элемента $x$ нечеткому множеству $\tilde{A}$.

*Носителем* нечеткого множества называется множество $\{x|x \in X, \mu_A(x)>0\}$.

*Ядром* нечеткого множества называется четкое подмножество базового множества $X$, элементы которого имеют степени принадлежности, равные единице: $\{x|x \in X, \mu_A(x) = 1\}$.

*α-сечением* нечеткого множества называется четкое подмножество базового множества $X$, элементы которого имеют степень принадлежности, большие или равные $\alpha$ :$\{x|x \in X, \mu_A(x) \geq \alpha, \alpha \in [0, 1]\}$.

*Нечетким высказыванием* $\tilde{A}$ называется высказывание, которое допускает, что $\tilde{A}$ может быть одновременно истинным и ложным (в обычной логике такая возможность исключается). Любое оценочное суждение, основанное на неполных или недостоверных данных, является нечетким и сопровождается обычно выражением степени уверенности (или сомнения) в его истинности. Например, утверждение: "Наверное, завтра похолодает" является нечетким высказыванием.

Мера истинности нечеткого высказывания $\tilde{A}$ определяется функцией принадлежности $\mu_{\tilde{A}}(x)$, $x \in X$, заданной на множестве X = {*false*, *true*}.

Аналогично четким множествам над нечеткими множествами можно производить ряд операций, которые могут определяться тремя способами:

| | |
|---|---|
| Максиминные | $\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$,<br>$\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$ |
| Алгебраические | $\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$,<br>$\mu_{A \cap B}(x) = \mu_A(x) \cdot \mu_B(x)$ |
| Ограниченные | $\mu_{A \cup B}(x) = \min\{1, \mu_A(x) + \mu_B(x)\}$,<br>$\mu_{A \cap B}(x) = \max\{0, \mu_A(x) + \mu_B(x) - 1\}$ |

Дополнение нечеткого множества определяется как

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x).$$

Чтобы сформировать нечеткие высказывания, экспертам необходимо прямо или косвенно задать функции принадлежности нечетких множеств. В качестве функций принадлежности обычно используются типовые функции (треугольник, трапеция, гауссова кривая, колокол, сигмоида и др.), вид которых определяется экспертами экспериментально (рис. 22).

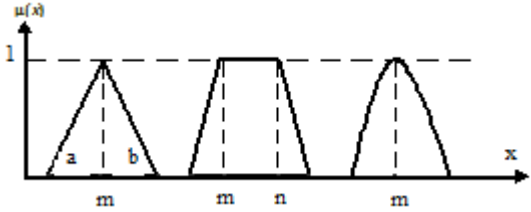



Рис. 22. Виды функций принадлежности нечетких множеств

При графическом определении функций принадлежности объединенного множества необходимо в каждой точке множества выбрать максимальное значение из двух (точку того графика, который выше) и объединить все полученные точки в график, который и будет отображением новой функции принадлежности.

Пересечение аналогично объединению, только выбирается минимальное значение в каждой точке.

При построении дополнения необходимо зеркально отобразить график от оси, параллельной оси абсцисс и проходящей через точку 0,5 оси ординат.

Чтобы построить функции принадлежности нового множества D, необходимо:

1. Определить последовательность выполнения операций в формуле.

2. Построить на отдельных графиках промежуточные множества, согласно определенной последовательности действий. Свести промежуточные множества на одном графике и определить итоговую функцию принадлежности.

3. Взять произвольный элемент, входящий в ядро итогового множества, и определить аналитически степень принадлежности элемента множеству.

4. Проверить аналитические вычисления по построенному графику функции принадлежности.

***Пример***. Даны 3 нечетких множества *A, B, C* (на рис. 23 графически заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества D = Ā∩(A∪C∪B) и определить степень принадлежности одного элемента множеству D.

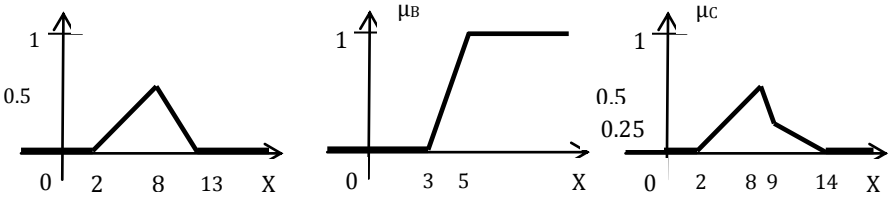



Рис. 23. Функции принадлежности множеств *A, B, C*

*Решение*. Таким образом, получаем следующее.

1. Множество D = Ā∩(A∪C∪B), поэтому последовательность операций будет следующей: Ā, A∪C∪B, Ā ∩ (A∪C∪B).

2. Построим согласно этой последовательности операций графики функций принадлежности (рис. 24).

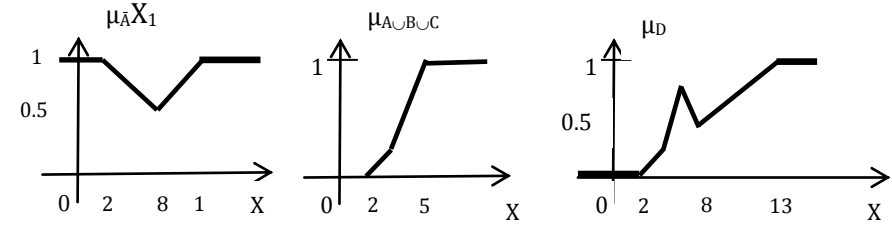



Рис. 24. Графики функций принадлежности последовательности операций для множества *D*

3. Ядро множества D состоит из элементов интервала (2,13). Выберем для аналитической проверки, например, элемент 8. Имеем

$\mu_A(8) = 0{,}5$; $\mu_B(8) = 1$; $\mu_C(8) = 0{,}5$;

$\mu_{\bar{A}}(8) = 1 - 0{,}5 = 0{,}5$;

$\mu_{A\cup C}(8) = \min\{1, \mu_A(8) + \mu_C(8)\} = \min\{1, (0{,}5 + 0{,}5)\} = 1$;

$\mu_{A\cup C\cup B}(8) = \min\{1, \mu_{A\cup C}(8) + \mu_B(8)\} = \min\{1, (1 + 1)\} = 1$;

$$\mu_{\bar{A}\cap(A\cup C\cup B)}(8) = \max\{0, \mu_{\bar{A}}(8) + \mu_{A\cup B\cup C}(8) - 1\} = \max\{0; 0{,}5\} = 0{,}5.$$

5. Проверяем аналитические вычисления по построенному графику функции принадлежности: $\mu_D(8) = 0{,}5$ (рис. 25).

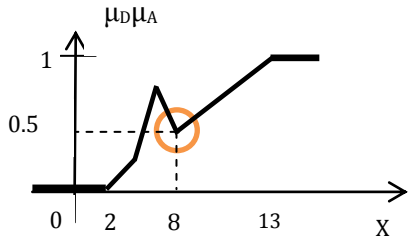



Рис. 25. Степень принадлежности элемента 8 множеству *D*

## МАШИНА НЕЧЕТКОГО ВЫВОДА

Архитектура интеллектуальной системы, основанной на использовании нечеткого вывода, имеет вид, представленный на рис. 26.

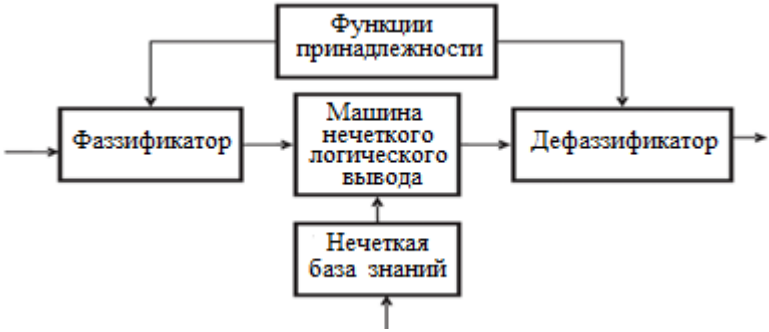



Рис. 26. Архитектура интеллектуальной системы, основанной на использовании нечеткого вывода

На этой схеме выполнение первого этапа нечеткого вывода – фаззификации – осуществляет фаззификатор. За процедуру непосредственно нечеткого вывода ответственна машина нечеткого логического вывода, которая производит второй этап процесса вывода на основании задаваемой нечеткой базы знаний (набора правил).

Дефаззификатор выполняет последний этап нечеткого вывода – дефаззификацию.

Основой для проведения нечеткого логического вывода является база знаний (правил), обычно содержащая нечеткие высказывания в форме "ЕСЛИ (условия) ТО (действия)" и функции принадлежности для соответствующих лингвистических переменных и термов.

При этом должны соблюдаться следующие условия:

- существует хотя бы одно правило для каждого лингвистического терма выходной переменной;
- для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве предпосылки.

В противном случае имеет место неполная база нечетких правил.

Существует множество различных алгоритмов нечеткого вывода. Они различаются главным образом видом используемых правил, логических операций и разновидностью дефазификации: машины нечеткого вывода Мамдани, Сугено, Ларсена, Цукамото и др.

Лингвистическая переменная – это переменная, значениями которой являются не числа, а слова или предложения естественного (или формального) языка. Каждому значению лингвистической переменной соответствует определенное нечеткое множество со своей функцией принадлежности. Например, лингвистическая переменная «Температура в аудитории Г – 418» (рис. 27).

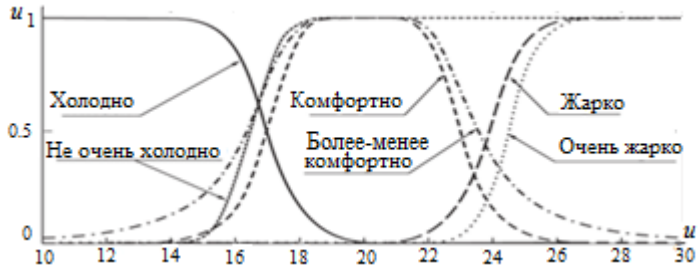



Рис. 27. Функции принадлежности лингвистических переменных, описывающих температуру в аудитории

Описание лингвистической переменной $\{x, T(x), X, G, M\}$ обычно включает ее наименование $x$; базовое терм-множество ее значений $T(x)$, представляющих собой наименования нечетких переменных,

областью определения каждой из которых является множество *X*; некоторое синтаксическое правило *G* для образования имен новых значений *x*; некоторую семантическую процедуру *M*, позволяющую превратить каждое новое имя в нечеткую переменную, т.е. сформировать соответствующее нечеткое множество, задав функции принадлежности.

Для реализации нечеткого логического вывода необходимо:

1. Сформулировать на естественном языке в виде предложений «ЕСЛИ .., ТО» закономерности предметной области.

2. Выделить из этих предложений лингвистические переменные, их значения (построить их функции принадлежности), высказывания различных видов, формализовать нечеткие правила.

3. Проверить полученную базу знаний на полноту.

4. Провести фаззификацию (входные данные выбираем случайным образом).

5. Провести аккумуляцию.

6. Провести дефаззификацию.

***Пример***. Построить нечеткую базу знаний, используя не менее трех лингвистических переменных, для определения затрат времени на решение студентом задач из данного пособия. Учитывать успеваемость студента и количество вариантов задач. Проверить БЗ на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

*Решение*.

1. Предложения, описывающие задачу следующие:

- ЕСЛИ (успеваемость студента высокая ИЛИ хорошая И он решает малое количество вариантов) ТО (ему требуется немного времени).
- ЕСЛИ (успеваемость студента высокая ИЛИ хорошая И он решает много вариантов) ТО (ему требуется достаточно большой промежуток времени).
- ЕСЛИ (успеваемость студента низкая И он решает много вариантов) ТО (ему требуется много времени).

- ЕСЛИ (успеваемость студента средняя И он решает достаточно большое количество вариантов) ТО (ему требуется достаточно большой промежуток времени).

Выделим из этих предложений лингвистические переменные:

1) $x$ = успеваемость студента, $T(x)$ = {«высокая», «средняя», «низкая»}, $X$ = [2, 5] (5-балльная система), $G$ = {«очень низкая», «высокая или средняя»}, $M$ – уменьшение на единицу степени принадлежности нечеткой переменной «высокая», операция объединения нечетких множеств;

2) $x$ – количество вариантов, $T(x)$ = {«мало», «достаточно», «много»}, X = [1, 20] (20 вариантов в каждой теме), $G$ = {«очень много», «достаточно или мало»}, $M$ – увеличение на единицу степени принадлежности нечеткой переменной «много», операция объединения нечетких множеств;

3) $x$ – количество времени, $T(x)$ = {«мало», «достаточно», «много»}, $X$ = [1, 7] (количество часов в неделю, уделенных СИИ), $G$ = {«очень много», «достаточно или мало»}, $M$ – увеличение на единицу степени принадлежности нечеткой переменной «много», операция объединения нечетких множеств.

Для полного задания лингвистической переменной необходимо определить нечеткие переменные, входящие в Т: успеваемость студента, количество вариантов и количество времени, задав их функции принадлежности (рис. 28).

С учетом выделенных лингвистических переменных, нечеткие правила будут иметь следующий вид:

1) ЕСЛИ (Успеваемость = «высокая» ИЛИ Успеваемость = = «средняя» И Количество вариантов = «мало») ТО (Количество времени = «мало»);

2) ЕСЛИ (Успеваемость = «высокая» ИЛИ Успеваемость = = «средняя» И Количество вариантов = «много») ТО (Количество времени = «достаточно»);

3) ЕСЛИ (Успеваемость = «низкая» И Количество вариантов = = «много») ТО (Количество времени = «много»);

4) ЕСЛИ (Успеваемость = «средняя» И Количество вариантов = = «достаточно») ТО (Количество времени = «достаточно»).

2. Проверим полученную базу на полноту:

- существует хотя бы одно правило для каждого лингвистического терма выходной переменной – выходная переменная «Количество времени» имеет 3 терма: «мало» используется в правиле 1, «достаточно» – в правилах 2 и 4, «много» – в правиле 3;

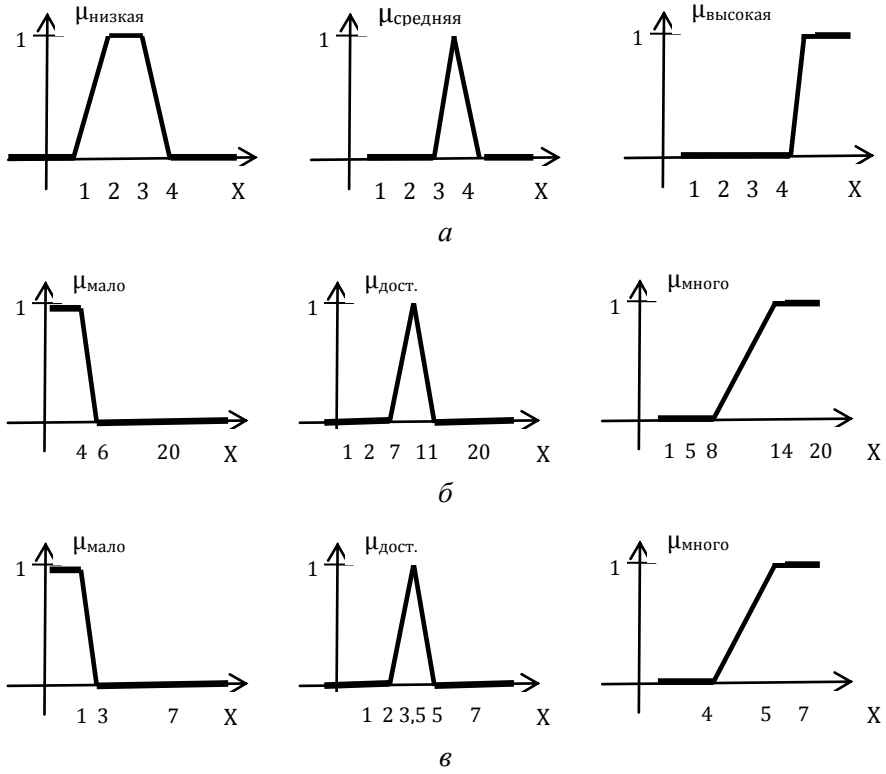



Рис. 28. Функции принадлежности нечетких переменных, входящих в множество Т: а – успеваемость студента; б – количество вариантов; в – количество времени

- для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве предпосылки – есть две входных переменных «Успеваемость» и «Количество вариантов» у каждой из них 3 терма: «высокая» используется в правилах 1 и 2, «средняя» – в правилах 1, 2 и 4, «низкая» – в правиле 3, «мало» – в правиле 1, «достаточно» – в правиле 4, «много» – в правилах 3 и 2.

Значит, полученная база нечетких правил полная.

3. Пусть имеется студент Иванов В.В. Его средняя оценка 3,5. Он планирует решить 9 вариантов задач. Сколько ему понадобится времени?

Определим степени уверенности простейших высказываний:

*Успеваемость = «высокая» – 0.*

*Успеваемость = «средняя» – 0,5.*

*Успеваемость = «низкая» – 1.*

*Количество вариантов = «мало» – 0.*

*Количество вариантов = «достаточно» – 0,5.*

*Количество вариантов = «много» – 0,125.*

Определим степени уверенности посылок правил:

*Правило 1: min (max (0; 0,5), 0) = 0.*

*Правило 2: min (max (0; 0,5), 0.125) = 0,125.*

*Правило 3: min (1; 0,125) = 0,125.*

*Правило 4: min (0,5; 0,5) = 0,5.*

Построим новую выходную нечеткую переменную, используя полученные степени уверенности (рис. 29).

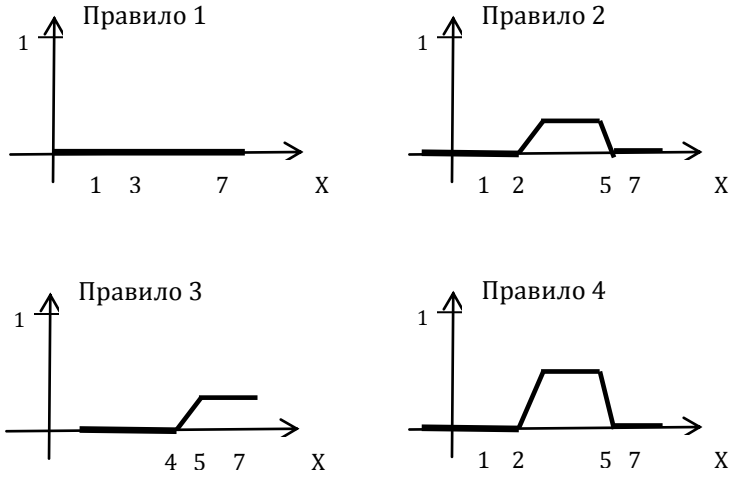



Рис. 29. Нечеткие переменные для правил 1 – 4

4. Аккумуляция (рис. 30).

5. Исходя из полученного графика степени принадлежности выходного терма, можно сказать, что Иванову В.В., имеющему

среднюю оценку 3,5, на решение 9-ти вариантов заданий понадобится не менее 2,75 часа (степень уверенности данного утверждения равна 0,5).

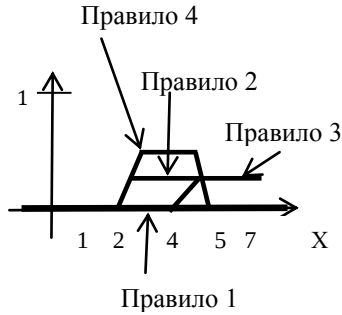



Рис. 30. Аккумуляция правил

***Пример***. Построить нечеткую базу знаний для определения удовлетворенности от автозаправки, которая выражается в размере чаевых. Проверить БЗ на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

*Решение.*

1. Предположим, что мы посетили автозаправку. Предположим также, что это была первая попавшаяся автозаправка, и мы не обладаем никакой информацией о ней, т.е. выбор данной автозаправки связан с неопределенностью. Далее предположим, что размер чаевых, которые определяют степень удовлетворенности от автозаправки, колеблются в интервале от 0 до 20 % от счета заказа. Будем полагать, что щедрость чаевых будет зависеть от двух факторов, которые обозначим как:

– стоимость топлива, которая будет оцениваться, как: высокая, средняя.

– качество обслуживания, оцениваемое по шкале: отличное, среднее, плохое.

Выделим из этих предложений лингвистические переменные:

1) β = стоимость топлива, T = {«высокая», «средняя»}, X = [27, 37], G = {«очень», «не очень»}, M = увеличение/уменьшение на единицу степени принадлежности нечеткой переменной;

2) β = уровень сервиса, T = {«высокий», «средний», «низкий»}, X = [1, 6], G = {«достаточно», «не достаточно»}, M = = увеличение/уменьшение на единицу степени принадлежности нечеткой переменной;

3) β = чаевые, T = {«большие», «средние», «маленькие»}, X = [1, 20], G = {«очень», «не очень»}, M = увеличение/уменьшение на единицу степени принадлежности нечеткой переменной.

Для полного задания лингвистической переменной необходимо определить нечеткие переменные, входящие в T: стоимость топлива, уровень сервиса и чаевые, задав их функции принадлежности (рис. 31).

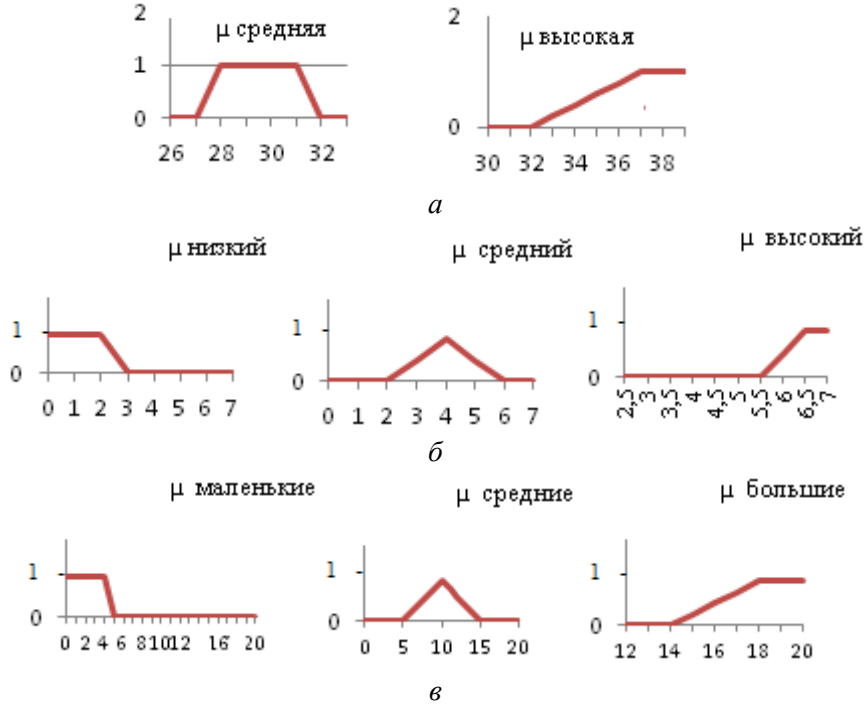



Рис. 31. Функции принадлежности нечетких переменных, входящих в множество T: а – стоимость топлива; б – уровень сервиса; в – чаевые

С учетом выделенных лингвистических переменных, нечеткие правила будут иметь следующий вид:

– Если Уровень сервиса = «низкий» или Стоимость топлива = = «высокая», то Чаевые = «маленькие».

– Если Уровень сервиса = «средний», то Чаевые = «средние».

– Если Уровень сервиса = «высокий» и Стоимость топлива = = «средняя», то Чаевые = «большие».

– Если Уровень сервиса = «высокий» и Стоимость топлива = «высокая», то Чаевые = «средние».

2. Проверим полученную базу на полноту.

- Для каждого лингвистического терма выходной переменной должно существовать хотя бы одно правило. Выходная переменная «Чаевые» имеет 3 терма: терм «маленькие» используется в правиле 1, терм «средние» – в правилах 2 и 4, терм «большие» – в правиле 3.
- Для любого лингвистического терма входной переменной должно существовать хотя бы одно правило, в котором этот терм используется в качестве предпосылки. У входной переменной «Стоимость топлива» 2 терма: терм «высокая» используется в правилах 1 и 4, «средняя» – в правиле 3. У входной переменной «Уровень сервиса» 3 терма: терм «высокий» используется в правилах 3 и 4, «средний» – в правиле 2, «низкий» – в правиле 1.

Следовательно, полученная база нечетких правил полная.

3. Фаззификация. Пусть стоимость топлива на заправке 29 руб./л, а уровень сервиса 6, нужно определить размер чаевых.

Определим степени уверенности простейших высказываний:

*Стоимость топлива = «высокая» – 0.*

*Стоимость топлива = «средняя» – 1.*

*Уровень сервиса = «высокий» – 0,5.*

*Уровень сервиса = «средний» – 0.*

*Уровень сервиса = «низкий» – 0.*

Определим степени уверенности посылок правил:

*Правило 1: max (0, 0) = 0.*

*Правило 2: 0 = 0.*

*Правило 3: min (0,5; 1) = 0,5.*

*Правило 4: min (0,5; 0) = 0.*

Построим новую выходную нечеткую переменную, используя полученные степени уверенности (рис. 32).

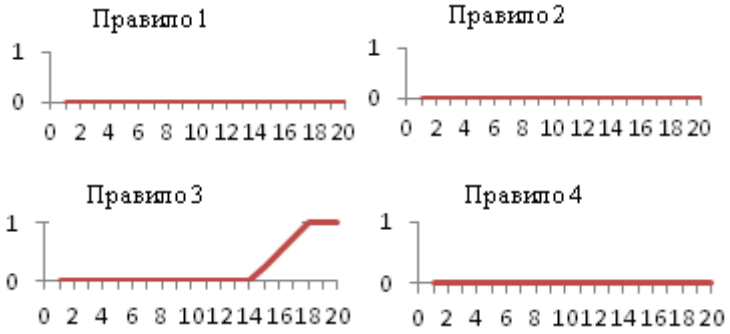



Рис. 32. Нечеткие переменные для правил 1 – 4

4. Аккумуляция правил и новый терм выходной переменной «Чаевые» представлены на (рис. 33).

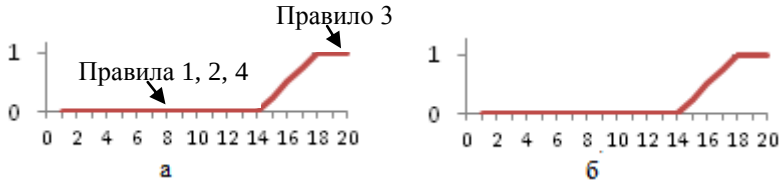



Рис. 33. Графики: а – аккумуляция правил задачи; б – терм выходной переменной «Чаевые»

5. Исходя из полученного графика степени принадлежности выходного терма, можно сказать, что размер чаевых будет не менее 18 % (степень уверенности равен 0,5).

***Пример***. На территории завода хранится хлор, на заводе внедрена система безопасности жизнедеятельности, контролирующая помещения, в которых он хранится. Надежность работы системы поддерживается администратором. Необходимо оценить риск использования данной информационной системы.

*Решение*.

1. В качестве входных параметров задачи рассмотрим надежность администрирования информационной системы и скорость нахождения возникающих ошибок.

Надежность администрирования информационной системы оценивается как высокая, средняя, низкая.

Скорость нахождения возникающих ошибок оценивается по шкале: высокая, средняя, низкая.

Предложения, описывающие задачу следующие:

- ЕСЛИ надежность администрирования низкая или скорость нахождения ошибок высокая, ТО риск высокий;
- ЕСЛИ надежность администрирования средняя, ТО риск умеренный;
- ЕСЛИ надежность администрирования высокая и скорость нахождения ошибок средняя, ТО риск будет умеренным;
- ЕСЛИ надежность администрирования высокая и скорость нахождения ошибок высокая, ТО риск будет низким.

Выделим из этих предложений лингвистические переменные: надежность администрирования информационной системы, которая оценивается как высокая, средняя, низкая, а скорость нахождения возникающих ошибок оценивается по шкале: высокая, средняя, низкая. Тогда:

1) α = надежность администрирования, T = {«высокая», «средняя», «низкая»}, X = [0, 10], G = {«очень», «не очень»}, M = увеличение/уменьшение на единицу степени принадлежности нечеткой переменной;

2) β = скорость нахождения ошибок, T = {«высокая», «средняя», «низкая»}, X = [1, 7], G = {«достаточно», «не достаточно»}, M = увеличение/уменьшение на единицу степени принадлежности нечеткой переменной;

3) γ = риск, T = {«высокий», «умеренный», «низкий»}, X = [0, 10], G = {«очень», «не очень»}, M = увеличение/уменьшение на единицу степени принадлежности нечеткой переменной.

С учетом выделенных лингвистических переменных, нечеткие правила будут иметь следующий вид:

– Если Надежность администрирования = «низкая» или Скорость нахождения ошибок = «низкая», то Риск = «высокий».

– Если Надежность администрирования = «средняя», то Риск = = «умеренный».

– Если Надежность администрирования = «высокая» и Скорость нахождения ошибок = «средняя», то Риск = «умеренный».

– Если Надежность администрирования = «высокая» и Скорость нахождения ошибок = «высокая», то Риск = «низкий».

Для полного задания лингвистической переменной необходимо определить нечеткие переменные, входящие в Т, задав их функции принадлежности (рис. 34).

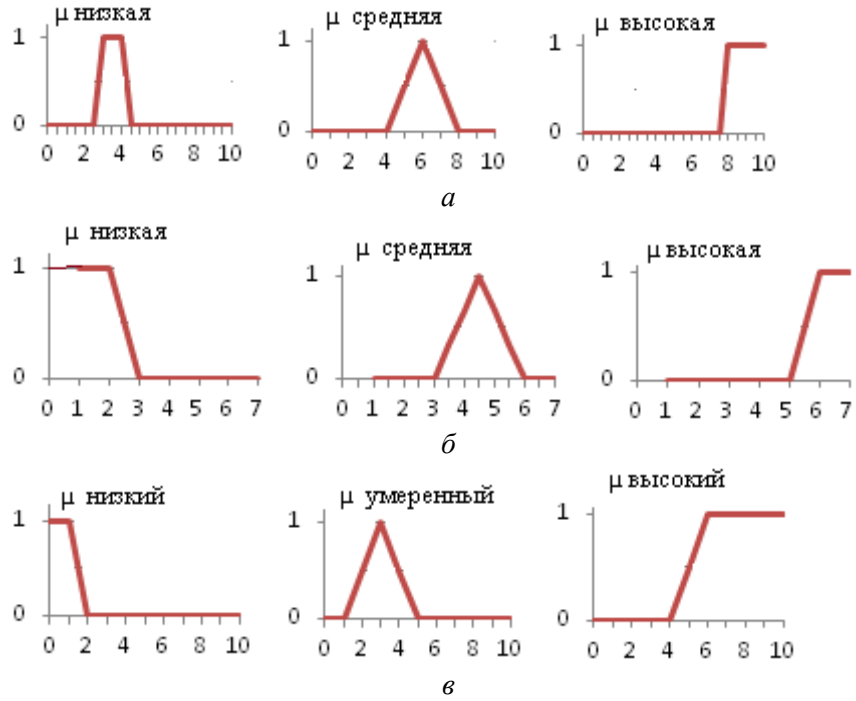



Рис. 34. Функции принадлежности нечетких переменных, входящих в множество Т: а – надежность администрирования; б – скорость нахождения ошибок; в – риск

2. Проверим полученную базу на полноту.

- Для каждого лингвистического терма выходной переменной должно существовать хотя бы одно правило. Выходная переменная «Риск» имеет 3 терма: терм «низкий» используется в правиле 4, «умеренный» – в правилах 2 и 3, «высокий» – в правиле 1.
- Для любого лингвистического терма входной переменной должно существовать хотя бы одно правило, в котором этот терм используется в качестве предпосылки. У входной переменной

«Надежность администрирования» 3 терма: терм «высокая» используется в правилах 3 и 4, «средняя» – в правиле 2, «низкая» – в правил е. У входной переменной «Скорость нахождения ошибок» 3 терма: терм «высокая» используется в правилах 1 и 4, «средняя» – в правиле 2, «низкая» – в правиле 1.

Признаки выполняются, следовательно, полученная база нечетких правил полная.

3. Фаззификация. Пусть надежность администрирования оценена в 7 баллов, а скорость нахождения ошибок – в 10.

Определим степени уверенности простейших высказываний:

*Надежность администрирования = «высокая»* – 0.

*Надежность администрирования = «средняя»* – 0,5.

*Надежность администрирования = «низкая»* – 0.

*Скорость нахождения ошибок = «высокая»* – 0,5.

*Скорость нахождения ошибок = «средняя»* – 0,33.

*Скорость нахождения ошибок = «низкая»* – 0.

Определим степени уверенности посылок правил:

*Правило 1: max* (0; 0) = 0.

*Правило 2:* 0,5 = 0,5.

*Правило 3: min* (0; 0,33) = 0.

*Правило 4: min* (0;0,5) = 0.

Построим новую выходную нечеткую переменную, используя полученные степени уверенности (рис. 35).

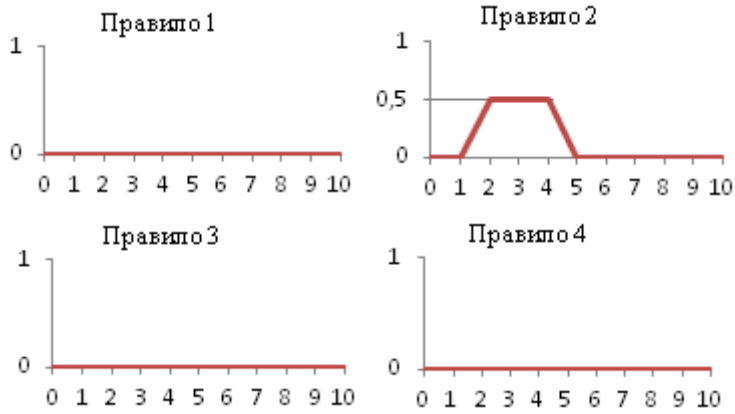



Рис. 35. Нечеткие переменные для правил 1 – 4

4. Аккумуляция правил и новый терм выходной переменной «Риск» представлены на рис. 36.

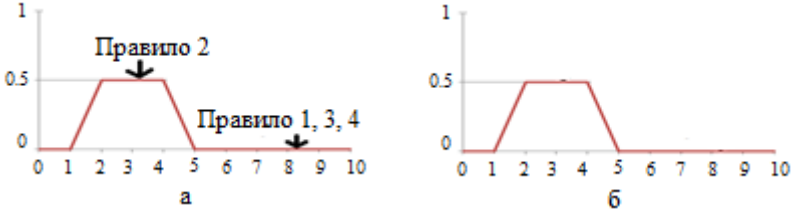



Рис. 36. Графики: а – аккумуляция правил задачи; б – терм выходной переменной «Риск»

5. Исходя из полученного графика степени принадлежности выходного терма, можно сказать, что риск будет не менее 1,5 баллов (степень уверенности равна 0,5).

## Задачи

1. Две урны наполнены шарами, причем в 1-й содержится 25 % белых и 75 % черных шаров, а во 2-й – 75 % белых и 25 % черных шаров. Если одна из этих урн выбрана случайно и случайным образом из нее извлечен один шар, то как вы оцените вероятность того, что этот шар белый?

2. Исследователь в связи с проведением эксперимента подсчитал, что если справедлива теория A, то можно наблюдать X с вероятностью около 0,9; если же справедлива теория B, то эта вероятность примерно равна 0,3. Он полагает, что теория A примерно вдвое более правдоподобна, чем теория B. Кроме теорий A и B других способов рационального объяснения наблюдаемых явлений нет. С какой вероятностью исследователь может ожидать появления X в ходе данного эксперимента?

3. Предположим, что новый прибор, «анализирующий» выдыхаемый воздух, позволяет с вероятностью 0,95 выявить превышение допустимого уровня содержания алкоголя в организме индивида и с вероятностью 0,95 установить, что этот уровень не превышен. Если в некоторый момент у 5 % обследуемых уровень алкоголя в организме выше допустимого, то какова вероятность, что при осмотре случайно выбранного из этой совокупности индивида

прибор зарегистрирует превышение уровня, и это будет соответствовать действительности?

4. Два аспиранта, Дмитрий и Стас, поставили эксперимент для проверки двух гипотез – $H_1$ и $H_2$. Априорную вероятность для гипотезы $H_1$ Дмитрий оценил, как 0,8, а Стас думал, что $H_2$ вдвое «вероятнее», чем $H_2$. Данные (D) были получены, и на их основе рассчитаны правдоподобия: $P(D|H_1) = 0{,}0084$, $P(D|H_2) = 0{,}0008$. Покажите, что апостериорные ожидания двух аспирантов сблизились по сравнению с их априорными ожиданиями.

5. Вы знаете, что в вашем университете 60 % студентов одного пола и 40 % другого, но забыли, кого больше, мужчин или женщин. Если первые два студента, которых вы встретили, женщины, то какова вероятность того, что справедлива гипотеза, в силу которой женщины составляют большинство? Что вы скажете, если следующие повстречавшиеся вам студенты – мужчины? Не означает ли это, что информация оказалась бесполезной?

6. Задать 3 нечетких множества A, B, C их функциями принадлежности. Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.

7. Задать 3 нечетких множества A, E, C их функциями принадлежности. Построить функцию принадлежности нечеткого множества $D = A \cup \bar{E} \cap C$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.

8. Задать 3 нечетких множества A, E, C их функциями принадлежности. Построить функцию принадлежности нечеткого множества $D = A \cup \bar{E} \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.

9. Задать 3 нечетких множества A, E, C их функциями принадлежности. Построить функцию принадлежности нечеткого множества $D = A \cup \bar{E} \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.

10. Задать 3 нечетких множества A, B, G их функциями принадлежности. Построить функцию принадлежности нечеткого

множества $D = \bar{A} \cup B \cap \bar{G}$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.

11. Задать 3 нечетких множества A, B, G их функциями принадлежности. Построить функцию принадлежности нечеткого множества $D = (\bar{A} \cup B) \cap \bar{G}$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.

12. Задать 3 нечетких множества A, B, G их функциями принадлежности. Построить функцию принадлежности нечеткого множества $D = (\bar{A} \cup B) \cap \bar{G}$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.

13. Задать 3 нечетких множества A, B, G их функциями принадлежности. Построить функцию принадлежности нечеткого множества $D = \bar{A} \cap (G \cup B) \cap \bar{G}$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.

14. Задать 3 нечетких множества A, B, G их функциями принадлежности. Построить функцию принадлежности нечеткого множества $D = \bar{A} \cap (G \cup B) \cap \bar{G}$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.

15. Задать 3 нечетких множества A, B, G их функциями принадлежности. Построить функцию принадлежности нечеткого множества $D = \bar{A} \cap (G \cup B) \cap \bar{G}$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.

16. Построить нечеткую базу знаний (использовать не менее трех лингвистических переменных) для задачи управления транспортным средством (регулировка скорости с учетом передачи, погодных условий, интенсивности потока и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

17. Построить нечеткую базу знаний (использовать не менее трех лингвистических переменных) для задачи подбора специй для блюда (соотношение количества и остроты специй, рецептуры, предпочтений едока, объема пищи и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

18. Построить нечеткую базу знаний (использовать не менее трех лингвистических переменных) для задачи подбора объема блюд (учитывать калорийность, вкусовые предпочтения, количество едоков и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

19. Построить нечеткую базу знаний (использовать не менее трех лингвистических переменных) для задачи подачи электроэнергии в условиях экономии (учет времени суток, типа помещений, количества людей, типа оборудования и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

20. Построить нечеткую базу знаний (использовать не менее трех лингвистических переменных) для задачи подбора интенсивности занятий (учитывать начальный уровень подготовки, объем учебного материала, количество человек в группе, необходимый уровень усвоения и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

21. Построить нечеткую базу знаний (использовать не менее трех лингвистических переменных) для задачи расчета потребления бензина (учитывать тип совершаемых маневров, уровень подготовки водителя, состояние автомобиля, тип автомобиля и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

22. Построить нечеткую базу знаний (использовать не менее трех лингвистических переменных) для задачи планирования объема производства продукции (с учетом возможной прибыли, необходимых ресурсов, платежеспособности населения, рынка сбыта и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

23. Построить нечеткую базу знаний (использовать не менее трех лингвистических переменных) для задачи регулирования кондиционера (учитывать его мощность, объем помещения, температуру окружающей среды, необходимую температуру в помещении и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

24. Построить нечеткую базу знаний (использовать не менее трех лингвистических переменных) для задачи распределения нагрузки между компьютерами при использовании их в кластерах (учитывать характеристики компьютеров, их количество, количество параллельного кода, характеристики сети и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

25. Построить нечеткую базу знаний (использовать не менее трех лингвистических переменных) для задачи выбора комплектующих для компьютера (учитывать цену, потребности пользователя, совместимость, сроки использования и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

## КОНТРОЛЬНЫЕ ВОПРОСЫ (ТЕСТЫ)

1. Известные вам НЕ-факторы знаний: а) незамкнутость; б) нелинейность; в) неопределённость; г) неполнота; д) неточность; е) неустойчивость; ж) нечёткость.

2. База знаний экспертной системы содержит правило вида $R \vee S \rightarrow T$. Коэффициенты уверенности посылок равны $ct(R) = 0{,}85$, $ct(\boldsymbol{S}) = 0{,}9$. Коэффициент уверенности правила $ct(R \vee S \rightarrow T) = 0{,}5$. Тогда коэффициент уверенности для логического вывода $ct(T)$ равен: а) 0; б) 0,15; в) 0,3; г) 0,45; д) 0,6; е) 0,75; ж) 0,85.

3. База знаний экспертной системы содержит правило вида $A\&B \rightarrow C$. Коэффициенты уверенности посылок равны $ct(A) = 0{,}3$, $ct(B) = 0{,}6$. Коэффициент уверенности правила $ct(A\&B \rightarrow C) = 0{,}5$. Тогда коэффициент уверенности для логического вывода $ct(C)$ равен: а) 0; б) 0,15; в) 0,3; г) 0,45; д) 0,6; е) 0,75; ж) 0,85.

4. Основные формы фазирования функции принадлежности нечетких множеств: а) круг; б) колокол; в) трапеция; г) треугольник; д) квадрат.

5. Функция принадлежности может принимать значения: а) $[0, \infty]$; б) $[-\infty, +\infty]$; в) [0 ,1]; г) нет правильного ответа.

6. Множество точек, для которых функция принадлежности равна 1, называется: а) носителем; б) ядром; в) α-сечением; г) нет правильного ответа.

7. Объединение нечетких множеств A и B определяется формулами:

а) $\min \{1, \mu_{A(x)} + \mu_{B(x)}\}$;

б) $\mu_{A(x)} + \mu_{B(x)} - \mu_{A(x)} \cdot \mu_{B(x)}$;

в) $\max \{0, \mu_{A(x)} + \mu_{B(x)} - 1\}$;

г) $\max \{\mu_{A(x)}, \mu_{B(x)}\}$.

8. В нечеткой логике степень истинности конъюнкции нескольких высказываний определяется: а) наиболее правдоподобным; б) наименее правдоподобным; в) средним значением.

# ЗАКЛЮЧЕНИЕ

Без знаний искусственный интеллект не может существовать в принципе. База знаний – ключевая компонента систем искусственного интеллекта. Поэтому при создании интеллектуальных систем особенное внимание уделяется моделям представления знаний. На сегодняшний день разработаны разнообразные модели знаний. Каждая из них обладает своими плюсами и минусами, и поэтому для каждой конкретной задачи необходимо выбрать именно свою модель. От этого может зависеть не столько эффективность выполнения поставленной задачи, сколько возможность ее решения вообще. При таком подходе не ставится вопрос об адекватности используемых в компьютере моделей представления знаний тем моделям, которыми пользуется в аналогичных ситуациях человек, а рассматривается лишь конечный результат решения конкретных задач. Проблема представления знаний заключается в несоответствии между сведениями о зависимостях данной предметной области, имеющимися у специалиста, методами, используемыми им при решении задач, и возможностями формального представления такой информации в ЭВМ. Часто проблема для эксперта осложняется трудностями по формулированию в явном виде имеющихся у него знаний.

Большинство исследователей искусственного интеллекта рассматривают задачу разработки моделей представления знаний как задачу программной реализации концепции баз знаний. Это означает, что модели представления знаний должны обладать всеми свойствами, присущими знаниям. Модели представления знаний различаются по идеям, лежащим в их основе, с точки зрения математической обоснованности. В данном практикуме рассмотрена только часть известных моделей – эмпирические модели, включающие продукционные правила, фреймы, семантические сети, а также нейросети и эволюционные вычисления, относящиеся к бионическому направлению в искусственном интеллекте.

Хочется выразить надежду, что после изучения практикума у слушателя сложились убеждения в полезности применения интеллектуальных систем, сформировались представления о моделях знаний.

Эволюция интеллектуальных систем и технологий протекает стремительно, базируется на результатах исследований в математической логике, дискретной математике, лингвистике, нейрофизиологии и др. В интеллектуальных системах используются достижения в технологии программирования, интернете, многоагентных системах и т.д. Однако основой искусственного интеллекта остаются опора на знания, модели их представления, вывода, обоснования и объяснения решений. Информацию по ним можно получить из литературы, указанной в библиографическом списке. Проблематика интеллектуальных систем и технологий представлена в [2, 3, 7, 8, 9, 13], продукционные правила, семантические сети и фреймы рассмотрены в [4, 7, 10], байесовский вывод, эволюционные вычисления очерчены в [1, 6, 9], нейросетевые и нечеткие модели – в [5, 7, 8, 12].

Авторы будут считать свою задачу выполненной, если после изучения практикума хотя бы один из будущих или действующих специалистов сочтет возможным и полезным использовать методы искусственного интеллекта на практике.

## Библиографический список

1. Родзин С.И. Искусственный интеллект: Учеб. пособие. – Таганрог: Изд-во ТТИ ЮФУ, 2009. – 200 с.

2. Башмаков А.И., Башмаков И.А. Интеллектуальные информационные технологии. – М.: Изд-во МГТУ им. Баумана, 2005. – 304 с.

3. Джексон П. Введение в экспертные системы. – М.: Изд. дом «Вильямс», 2001. – 624 с.

4. Джонс М.Т. Программирование искусственного интеллекта в приложениях. – М.: ДМК Пресс, 2006. – 312 с.

5. Зозуля Ю.И. Интеллектуальные нейросистемы. – М.: Радиотехника, 2003. – 144 с.

6. Курейчик В.В., Курейчик В.М., Родзин С.И. Теория эволюционных вычислений. – М.: Физматлит, 2012. – 260 с. http://www.rfbr.ru/rffi/ru/books/o_1780977.

7. Липатова С.В. Сборник задач по курсу «Интеллектуальные информационные системы». – Ульяновск: УлГУ, 2010. – 64 с.

8. Рассел С., Норвинг П. Искусственный интеллект: современный подход. – М: Изд. дом «Вильямс», 2006. – 1408 с.

9. Родзин С.И., Ковалев С.М. Информационные технологии: интеллектуализация обучения, моделирование эволюции, распознавание речи. – Ростов-на-Дону: Изд-во СКНЦ ВШ, 2002. – 224 с.

10. Частиков А.П. и др. Разработка экспертных систем. Среда CLIPS. – СПб.: БХВ-Петербург, 2003. – 608 с.

11. Штовба С.Д. Введение в теорию нечетких множеств и нечеткую логику. http://matlab.exponenta.ru/fuzzylogic/book1/ index.php.

12. Ярушкина Н.Г. Основы теории нечетких и гибридных систем: – М.: Финансы и статистика, 2004. – 320 с.

13. Ясницкий Л.Н. Искусственный интеллект: – М.: БИНОМ. Лаборатория знаний, 2011. – 240 с. http://lbz.ru/books/232/5563/.

*Учебное издание*

**Родзин Сергей Иванович**
**Родзина Ольга Николаевна**

# МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ
## Практикум по курсу
## Системы искусственного интеллекта

Ответственный за выпуск Родзина О.Н.

Редактор Проценко И.А.

Корректор Селезнева Н.И.

Подписано в печать __.__.2014 г.
Заказ №
Формат 60×84 $^{1}/_{16}$ .

Тираж 100 экз.
Усл. п.л. – 9,4.
Уч.-изд. л. – 9,2.

---

Издательство Южного федерального университета

344091, г. Ростов-на-Дону, пр. Стачки, 200/1. Тел. (8634)2478051

Отпечатано в Секторе обеспечения полиграфической продукцией кампуса в г. Таганроге отдела полиграфической, корпоративной и сувенирной продукции ИПК КИБИ МЕДИА ЦЕНТРА ЮФУ.

ГСП 17 А, Таганрог, 28, Энгельса, 1. Тел. (8634)371717, 371655