МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования «Южный федеральный университет»

Институт компьютерных технологий и информационной безопасности
Кафедра математического обеспечения и применения ЭВМ

С.И. Родзин

Системы искусственного интеллекта

Руководство к практическим занятиям

Для студентов направлений
«Программная инженерия»
«Математическое обеспечение и администрирование информационных систем»
«Информатика и вычислительная техника»

Таганрог 2015-16

УДК 007. 52: 611.81 (075.8) + 519.7 (075.8)

Рецензенты:

Кафедра систем автоматизированного проектирования Южного федерального университета, доктор технических наук, профессор *Лебедев Б.К.*

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

Родзин С.И. **Системы искусственного интеллекта. Руководство к практическим занятиям**. Учебное пособие. Таганрог: ЮФУ, 2015. 95 с.

Данное руководство включает описание четырех практических занятий по темам: «Продукционные модели представления знаний», «Семантические сети и фреймы для представления знаний», «Нейросети» и «НЕ-факторы знаний». Включает необходимые для решения задач искусственного интеллекта теоретические сведения, а также описание процесса их решения. В практикуме приведены как оригинальные задачи, так и заимствованные из специальной и учебной литературы.

Для студентов, изучающих дисциплины «Системы искусственного интеллекта», «Интеллектуальные системы», а также специализирующихся в области интеллектуальных информационных технологий.

Библиогр.: 12 назв.

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.

Одна из целей данного руководства – помочь будущим специалистам в IT-сфере овладеть теоретическими и практическими знаниями и навыками использования средств и методов искусственного интеллекта.

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

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

Практическое занятие №1. Продукционные модели представления знаний

Целью занятия является изучение одной из наиболее распространенных, особенно в области экспертных систем, продукционной модели представления знаний, а также решение практической задачи на построение продукционной модели.

Рекомендуемый объем занятия по теме – 4 часа.

1.1. Продукционная модель представления знаний

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 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→B, C), (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 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.

1.2. Классификация продукционных правил

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

ЕСЛИ A, ТО *возможно* B.

Например, если задана вероятность выполнения B при актуализации A, то продукция может быть такой:

ЕСЛИ A, ТО *с вероятностью p реализовать* B.

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

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

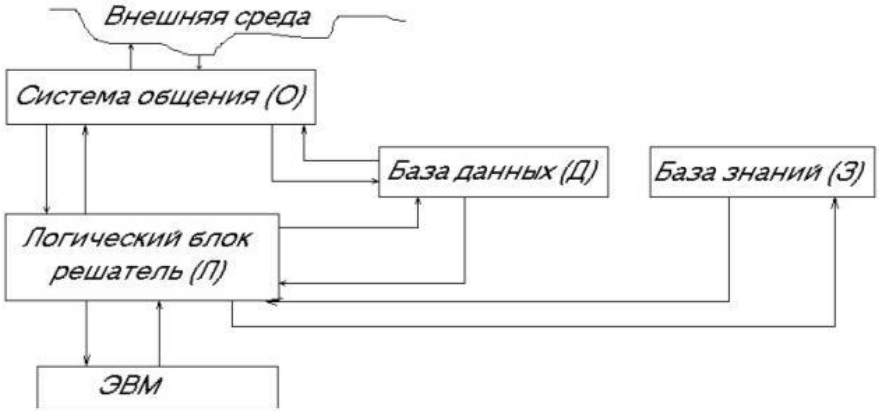
Возможны иные способы реализации ядра.

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:



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

Комбинации *x* и *y*, осмысленные с точки зрения СИИ, отмечены в таблице знаком "+":

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 таблицы 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.

1.3. 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→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 **A**, из символов которого формируются строки.

2. Множество строк, которые рассматриваются как *аксиомы*.

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

Пост доказал следующие свойства КС.

1. Некоторое слово **B** *выводимо* из другого слова **A**, если его можно получить из **A** 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 $\mathbf{a_1\$a_2 \rightarrow a_1£a_2}$ 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». П. применяется в экспериментальной поэзии В. Хлебникова и др.). Тогда следующие правила (порождения) сгенерируют все палиндромы в данном алфавите:

(P1) $ → a$a, **(P2)** $ → b$b, **(P3)** $ → 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: необходимое и достаточное условие интеллектуальности системы – универсальность формальных манипуляций над конкретными символами*). 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.

1.4. Пример построения продукционной модели

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

Описание процесса решения. 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.

Решение.

1) Обязательное действие, выполняемое в кафе – поглощение пищи и ее оплата. Значит, 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 путь туда*», «*сделать заказ официанту*».

3) Прежде чем идти в кафе, необходимо убедиться, что есть необходимая сумма денег. Выбор кафе может обуславливаться многими причинами, выберем территориальный признак – к какому ближе в тот 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 есть 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 есть.

- Если субъект может есть, 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): П1 – 1, П2 = П3 = 2, П4 = П5 = 3, П6 = П7 = 4, П8 = 5.

Отобразим взаимосвязи продукций на графе:

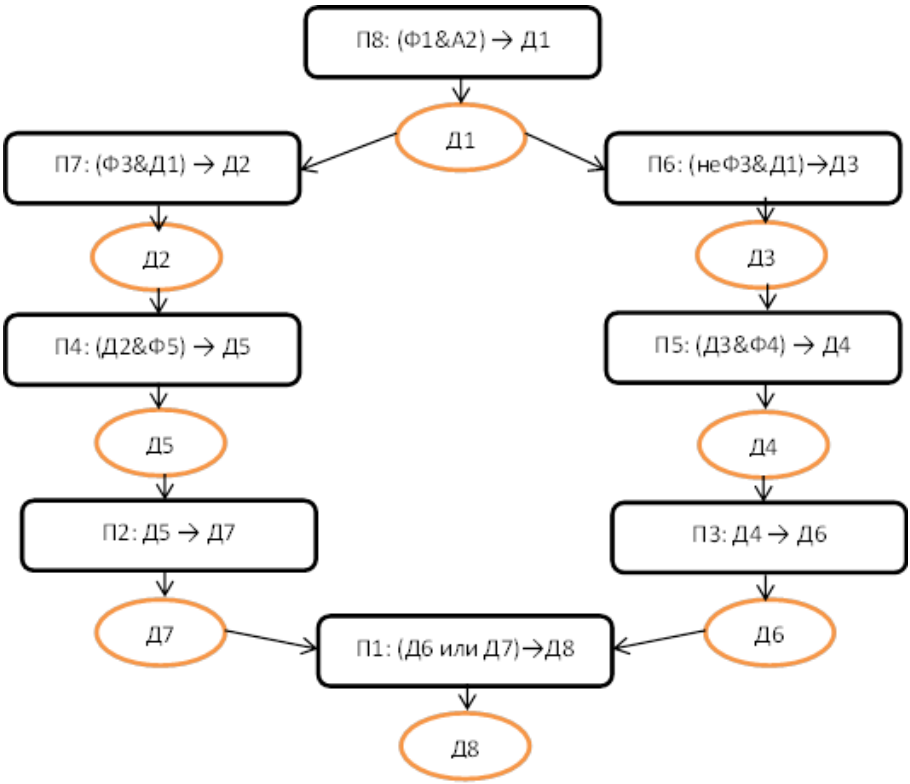


Рис. Взаимосвязь продукций предметной области «Кафе»

1.5. Представление правил в продукционном программировании

В продукционном программировании СИИ 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 ситуа¬ции. Разработчик ЭС определяет совместную совокупность правил для решения проблемы. Правило состо¬ит из двух частей: слева записываются предпосылки (антицедент или условия), которые обычно представляются в виде триады «объект – атрибут – значение атрибута», а справа – действия (заключе¬ния, консеквент):

```
( defrule <имя правила>
   <условие1>
   ….
   <условиеm>
   =>
   <действие1>
   ….
   <действиеn>
)
```

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. Наконец, если какое-то одно правило из списка будет выполнено, действия задают изменения, которые должны быть внесены в состояние рабочей памяти.

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);
3. 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, т.к. 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, т.к. 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:

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.

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

Специфика. Правила считаются более специфичными, принимающими во внимание больше информации, если включают большее число условий и, поэтому труднее удовлетворяются.

В интерпретаторе CLIPS использованы все три стратегических механизма разрешения конфликтов, а также свойство выпуклости (*salience*):

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 он не способен 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 *c* 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, т.к. 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 [Осипов], а также инструментальный комплекс АТ-ТЕХНОЛОГИЯ [Рыбина]

1.6. Задачи

1. Построить продукционную модель представления знаний в предметной области «**Аэропорт**» (диспетчерская).

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

3. Построить продукционную модель представления знаний в предметной области «**Автозаправка**» (обслуживание клиентов).

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

5. Построить продукционную модель представления знаний в предметной области «**Компьютерная безопасность**» (средства и способы ее обеспечения).

6. Построить продукционную модель представления знаний в предметной области «**Компьютерная безопасность**» (угрозы)

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

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

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

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

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

12. Построить продукционную модель представления знаний в предметной области «**Операционные системы**» (функционирование).

13. Построить продукционную модель представления знаний в предметной области «**Программное обеспечение**» (виды и функционирование).

14. Построить продукционную модель представления знаний в предметной области «**Предприятие по разработке программного обеспечения**» (структура и функционирование).

Контрольные вопросы

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. «Жёсткие» модели представления знаний: а) база данных, б) искусственная нейросеть, в) нечёткие множества, г) продукционные правила, д) семантическая сеть, е) теоремы, ж) фреймы.
3. Согласно Посту каноническая система включает: а) алфавит, б) аксиомы, в) атрибуты, г) правила порождений, д) правила поведения.
4. 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.
5. 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.
6. В основе грамматики формальных языков и большинства языков программирования лежат канонические системы с продукциями вида: а) $a_1a_2\$\rightarrow a_1a_2£$, б) $\$a_1a_2\rightarrow £a_1a_2$, в) $a_1\$a_2\rightarrow a_1£a_2$, г) $a_1\$a_1\rightarrow a_2£a_2$, где a_1 и a_2 – фиксированные строки, $\$$ и $£$ – переменные строки.
7. Продукционная система включает: а) библиотеку, б) базу правил, в) рабочую память, г) память правил, д) компилятор правил, е) интерпретатор правил.
8. 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.
9. 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.

10. 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.

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

12. Укажите правильную последовательность работы интерпретатора правил CLIPS: а) разрешить конфликт, б) применить выбранное правило, в) сопоставить условные части правил и элементы рабочей памяти.

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

14. 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.

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

16. 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,…

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

Практическое занятие №2. Семантические сети и фреймы для представления знаний

Целью занятия является изучение семантических сетей и фреймов в качестве популярных моделей представления знаний, а также решение практических задач их построения.

Рекомендуемый объем занятия по теме – 6 часов.

2.1. Семантические и ассоциативные сети

Продукционные порождающие правила являются очень удобной моделью знаний для представления 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.

Классификация семантических сетей.

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 отношения, связывающие более двух объектов — **N-арные**. 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.

Классификация семантических отношений

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 используются отношения «объект-множество», обозначающие, 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:

- 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 Дрю 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. В математической записи это выглядит так, соответственно для предложений 1—4:

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);

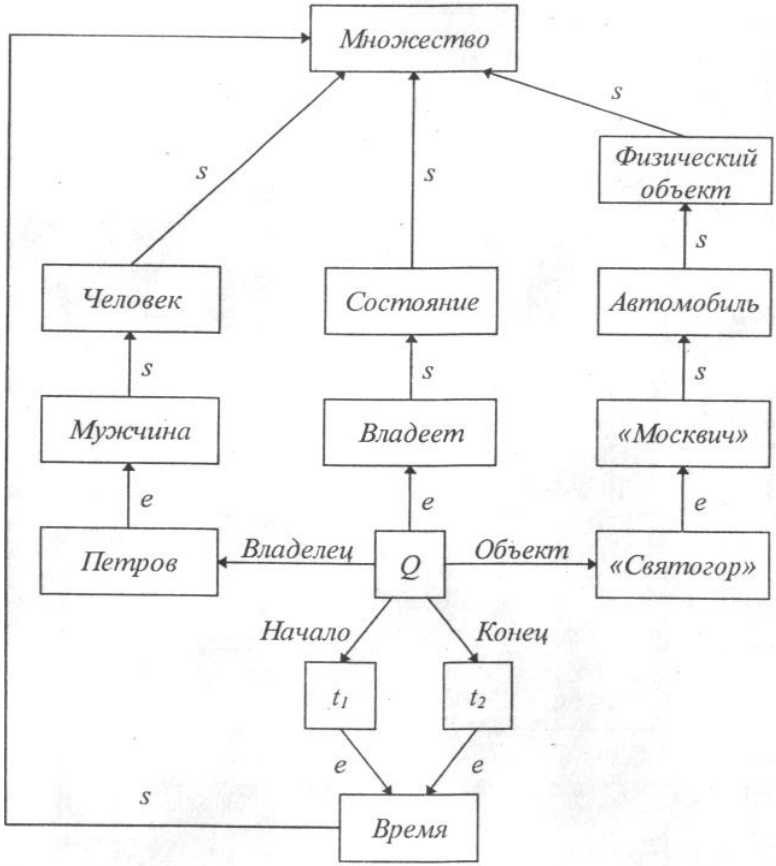
III. $\forall\, y \in$ люди $\rightarrow \exists\, x \in$ мужчины : отец(x, y);

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 $\exists$ и $\forall$, 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.

Пример. Ниже представлен фрагмент семантической сети, иллюстрирующей предложение **"Петров на протяжении периода времени с t1 по t2 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 ЕЯ, называют ***ассоциативными сетями*** (АС). Квиллиан не первым обратил

внимание на важность обобщенного абстрактного знания для понимания ЕЯ. Но 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 их многих деталей, а РС – это тоже вычислительная машина, тогда нет смысла в явном виде хранить эту информацию, присоединяя ее в сети к вершине РС). В современном программировании это принято называть свойством наследования.

Пример. Автомобили Нива и Волга движутся навстречу друг к другу по направлению к городу Томску. Фрагмент семантической сети для этой фразы на естественном языке имеет вид:

2.2. Пример построения семантической сети

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

Описание процесса решения задачи. 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 этими 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.

Решение.

1) Ключевые понятия 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.

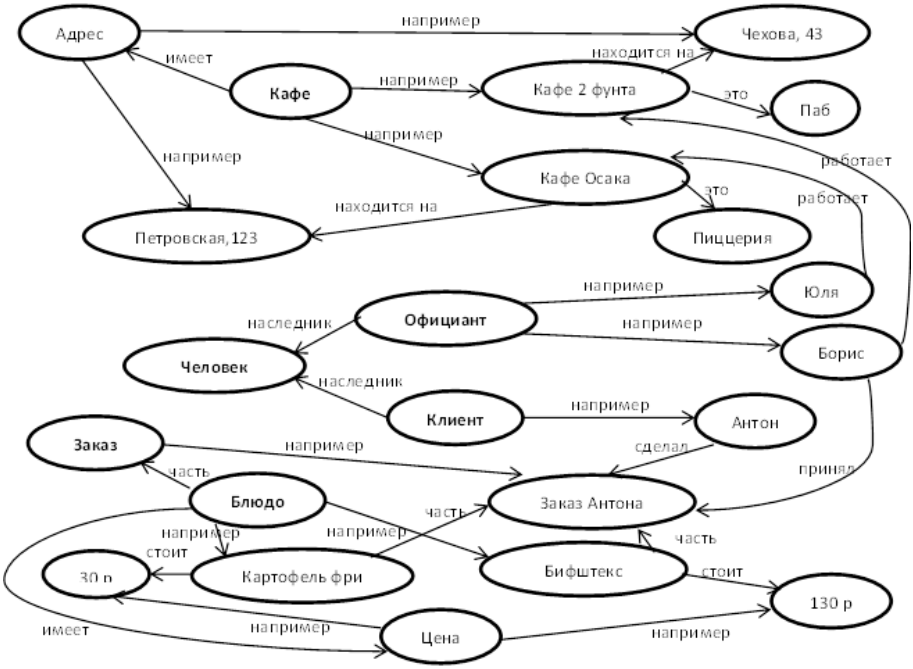
Исходя из этого, вершины графа будут следующими: «**Кафе**», «**Человек**», «**Официант**», «**Клиент**», «**Заказ**» и «**Блюдо**».

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

3) Определим для имеющихся вершин отношения и их типы.

4) Добавим знание о конкретных фактах решаемой задачи. Пусть имеется два ресторана: «Осака» и «2 фунта», в первом работает официантка Юля, а во втором официант Борис. Антон решил пойти в кафе «2 фунта» и сделал заказ официанту на 2 блюда: **картофель фри** за 30 р., **бифштекс** за 130 р. Также известны адреса этих ресторанов и их специфика.

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.:



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 р) и бифштекс (по цене 130 р)***.

Для получения ответа на какой-либо вопрос по этой задачи, 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 р. соответственно. Больше информации о заказе Петра в модели нет, поэтому делаем вывод – Петр заплатил 160 р.

2.3. 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 таблице представлены основные способы получения значений слотов и их краткое описание:

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 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	В интеллектуальных системах данные,

Способ получения значений слотов	Описание способа
источников данных	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. Ниже в таблице представлены наиболее распространенные демоны:

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. Ниже в таблице представлены наиболее распространенные типы фреймов, указаны типы знаний, которые они отображают, а также примеры фреймов данного типа из различных предметных областей:

Тип фрейма	Тип знания	Описание	Пример
По познавательному значению			
Фреймы-прототипы (шаблоны,	интенсиональные	отражают знания об абстрактных стереотипных понятиях,	человек, автомобиль

Тип фрейма	Тип знания	Описание	Пример
образцы)		которые являются классами каких-то конкретных объектов	
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	функциональ-	отображают типичную роль, выполняемую фреймом-	программист,

Тип фрейма	Тип знания	Описание	Пример
	ные	объектом в определенной ситуации (содержат характеристики роли)	администратор, студент, преподаватель

2.4. 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++, то есть 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 конфликта при множественном наследовании свойств («Алмаз Никсона»). 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.

2.5. Пример решения задачи

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

Описание процесса решения. 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.

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 таблиц:

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	

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	

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	

ОФИЦИАНТ (АКО **ЧЕЛОВЕК**)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
зарплата		из внешних источников	
график работы		из внешних источников	
место работы	фрейм-объект	из внешних источников	

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:

КАФЕ «ОСАКА» (АКО **КАФЕ**)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
название	ОСАКА	из внешних источников	
адрес	Таганрог, Петровская, 123	из внешних источников	
часы работы	10:00-23:00	из внешних источников	
специализация	пиццерия	из внешних источников	
класс	средний	из внешних	

КАФЕ «ОСАКА» (АКО **КАФЕ**)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
		источников	

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	

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	

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 12:00 до 18:00	из внешних источников	
место работы	**КАФЕ «ОСАКА»**	из внешних источников	

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:

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	

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:

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:

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»)

ПОСЕЩЕНИЕ «2 фунта» (АКО **ПОСЕЩЕНИЕ КАФЕ**)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
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	

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	

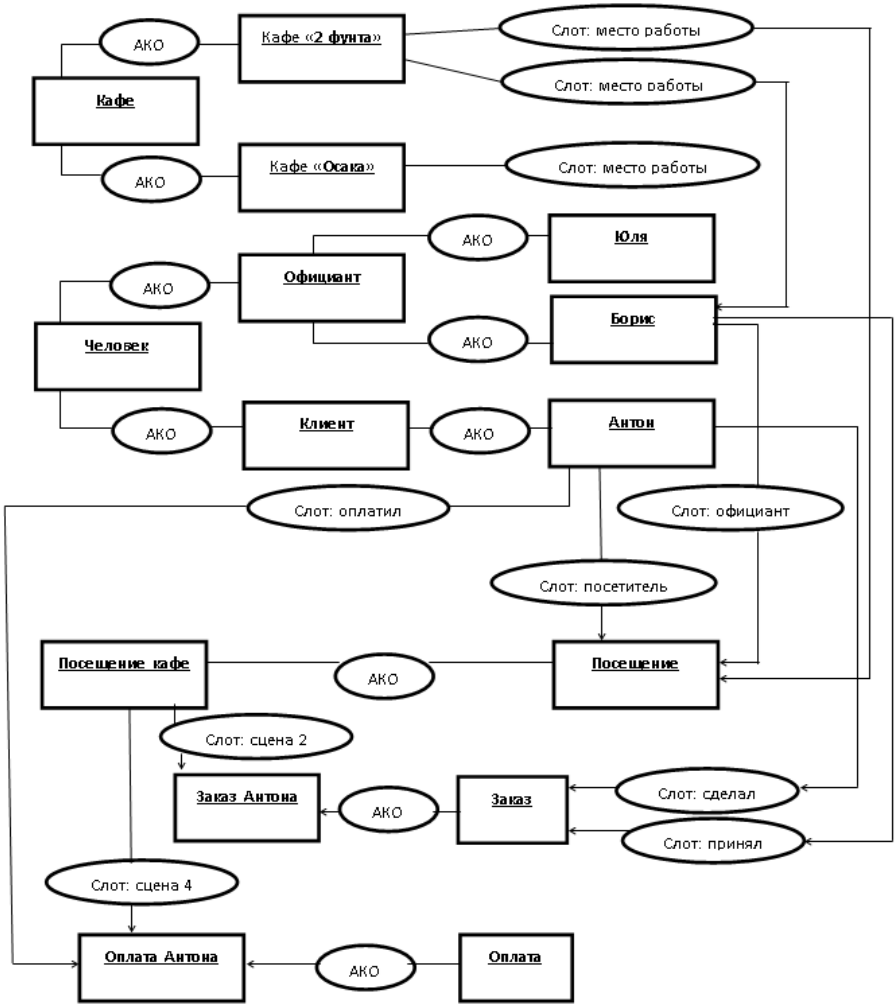
ОПЛАТА АНТОНА (АКО **ОПЛАТА**)			
Имя слота	**Значение слота**	**Способ получения значения**	**Демон**
вид платежа	наличные	из внешних источников	IF-ADDED (изменяет слот «чаевые»)
чаевые	30	присоединенная процедура	
оплатил	**АНТОН**	присоединенная процедура	
заказ	**ЗАКАЗ АНТОНА**	из внешних источников	IF-ADDED (изменяет слот «оплатил»)

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	

Ниже, на рисунке 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.



2.6. Задачи

1. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «**Аэропорт**» (диспетчерская).

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

3. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «**Автозаправка**» (обслуживание клиентов)

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

5. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «**Компьютерная безопасность**» (средства и способы ее обеспечения).

6. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «**Компьютерная безопасность**» (угрозы)

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

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

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

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

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

12. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «**Операционные системы**» (функционирование).

13. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «**Программное обеспечение**» (виды и функционирование).

14. Построить семантическую сеть и фреймовую модель представления знаний в предметной области «**Предприятие по**

разработке программного обеспечения» (структура и функционирование).

Контрольные вопросы

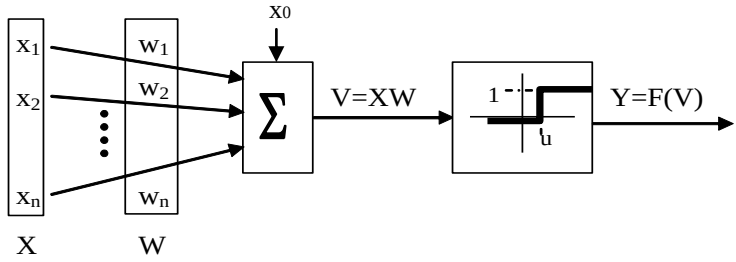
1. Семантическая сеть - это: а) орграф, в котором вершины являются отношениями, а ребра - понятиями, б) орграф, в котором вершины являются понятиями, а ребра - отношениями, б) иерархическая классификационная структура, в) несколько семантически связанных предложений в тексте.
2. Фрейм (в инженерии знаний) - 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.
3. Характерной особенностью семантических сетей является наличие следующих типов отношений: а) ISa, б) транзитивность, в) AKO, г) строгий порядок, д) целое-часть (HasPart), е) пространственные, ж) логические, з) симметричность, и) временные.
4. Для фреймов характерно свойство наследования по AKO-связям: а) да, б) нет, в) зависит от контекста.
5. Рёбрами семантической сети обычно выступают: а) действия б) понятия, б) абстрактные или конкретные объекты, в) отношения.

Практическое занятие № 3. Обучение нейросетей

Целью занятия является изучение моделей нейросетей и методов их обучения.

Рекомендуемый объем занятия по теме – 4 часа.

3.1. Искусственный нейрон МакКаллока-Питтса. Функции активации



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

Нейрон описывается математической моделью в виде уравнения

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

где $\boldsymbol{Y}$ – выходной сигнал нейрона, $\boldsymbol{F}$ - функция активации выхода нейрона, $\mathbf{w_j}$ – «вес» входа $\mathbf{x_j}$. Добавив постоянный единичный вход x_0=1 (смещение) и положив w_0= - **θ**, получим

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

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:

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

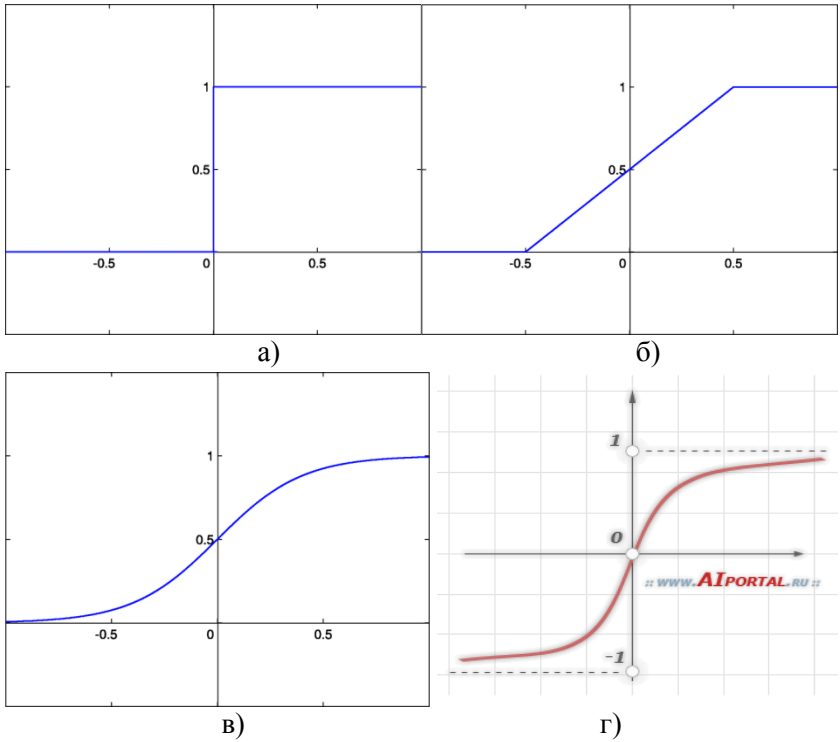


Рис. Функции активации нейрона: а) пороговая (Θ=0)*, б) линейная, в) логистическая (сигмоидальная), г) гиперболический тангенс*

3.2. 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:

Тип нейросети	**Описание**
По топологии	

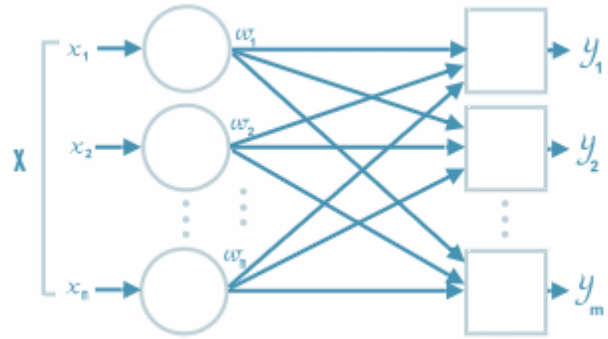
Тип нейросети	Описание
Полносвязные	Каждый нейрон связан с другим нейроном в сети (из-за высокой сложности обучения не используется).
Слоистые	Нейроны располагаются слоями, каждый нейрон последующего слоя связан с нейронами предыдущего. Есть одно- и многослойные сети.
По типу связей	
Прямого распространения	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

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

3.3. Правила Хебба, обучение сети по дельта-правилу



Однослойная нейросеть может обучаться по правилу Хебба. Предъявляем на вход персептрона один пример. Если выходной сигнал персептрона совпадает с правильным ответом, то никаких 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 (1) выдавал правильный ответ необходимо, чтобы сумма в правой части (1) стала больше порога. Поскольку переменные p_i принимают значения 0 или 1, увеличение суммы может быть достигнуто за счет увеличения весов α_i. Однако нет смысла увеличивать веса при переменных p_i, которые равны нулю. Таким образом, следует увеличить веса α_i при тех переменных p_i, которые равны 1. Для закрепления единичных сигналов с p_i, следует провести туже 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 в правой части (1). Для этого необходимо уменьшить веса связей α_i при тех переменных p_i, которые равны 1 (поскольку нет смысла уменьшать веса связей при равных нулю переменных p_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 осталось 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 α, 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, η - коэффициент обучения (от 0 до 1), x_j – входное значение, Θ_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.

3.4. Пример обучения нейросети по дельта-правилу

Постановка задачи.

Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной неоднородной нейросети, состоящей из 2 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.

Описание процесса решения.

Для обучения нейронной сети по Δ-правилу необходимо:

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

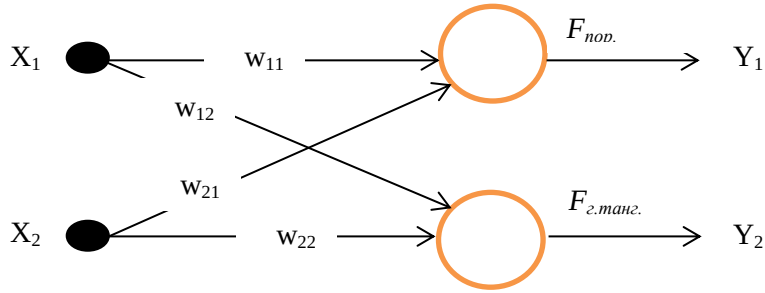
2) Определить обучающую выборку, представив ее в табличном виде.

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

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

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:



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 векторов). Выход первого нейрона согласно заданию соответствует оператору эквивалентности, а второго – дизъюнкции.

Поэтому 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-ая 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:

Θ=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:

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

$\varepsilon_2 = (d_2 - y_2) = (1\text{-}1.39) = -0.39$.

Ш а г 5: зададим η – коэффициент обучения от 0 до 1 и изменяем веса:

η =0.8;

$w_{11}(2) = w_{11}(1) - 0.8 \cdot \varepsilon_1 x_1 = \cdot 0.7 - 0.8(-1)\ 1 = 1.5$;

$w_{21}(2) = w_{21}(1) - 0.8 \cdot \varepsilon_1 x_2 = \cdot 0.5 - 0.8(-1)\ 0 = 0.5$;

$\Theta_1(2) = \Theta_1(1) - 0.8 \cdot \varepsilon_1 = \cdot 0.7 - 0.8(-1) = 1.5$;

$w_{12}(2) = w_{12}(1) - 0.8 \cdot \varepsilon_1 x_1 = \cdot 0.9 - 0.8(-0.39)\ 1 = 1.212$;

$w_{22}(2) = w_{22}(1) - 0.8 \cdot \varepsilon_1 x_2 = \cdot 0.2 - 0.8(-0.39)\ 0 = 0.2$.

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:

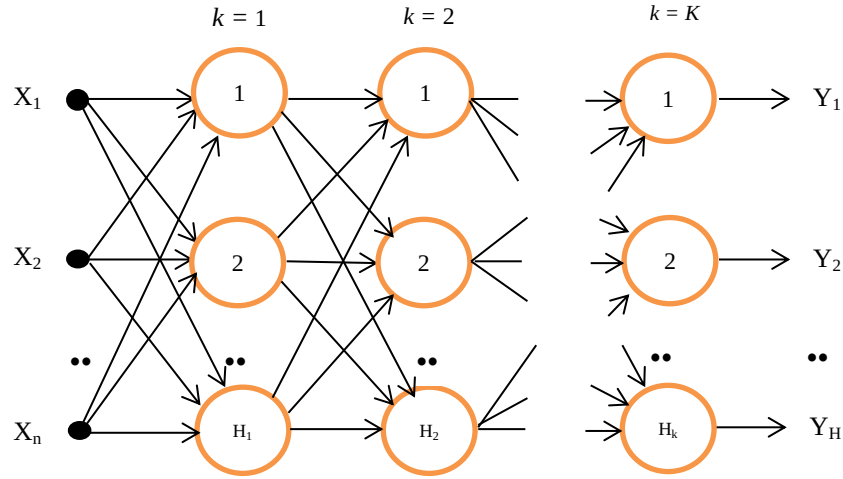
$$\varepsilon = \sum_{i=1}^{N} (d_i - y_i)^2$$

$$= \sum_{i=1}^{2} {\varepsilon_i}^2 = {\varepsilon_1}^2 + {\varepsilon_2}^2 = (-1)^2 + (-0.39)^2 = 1.152.$$

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

3.5. Обучение нейросети по методу обратного распространения ошибки

Многослойная нейросеть может содержать произвольное количество слоев (K), каждый слой состоит из нейронов, число которых H_k также может быть произвольно, количество входов *n*, количество выходов $H=H_K$ – 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:

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*):

$$y_i^k = f(\sum_{j=0}^{H_{k-1}} w_{ij}^k \cdot y_j^{k-1}),$$

$y_j^0 = x_j$,

$y_0^{k-1} = 1$,

$x_0 = 1$,

где y_i^k – выход *i*-нейрона *k*-слоя, *f*-функция активации, $\mathbf{w}_{ij}^k$ – 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:

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

$\delta_i^k = (d_i - y_i) \cdot y_i \cdot (1 - y_i)$ - для выходного слоя

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

где *t* - номер текущей итерации цикла обучения (номер эпохи), η - 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 4: 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:

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

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.

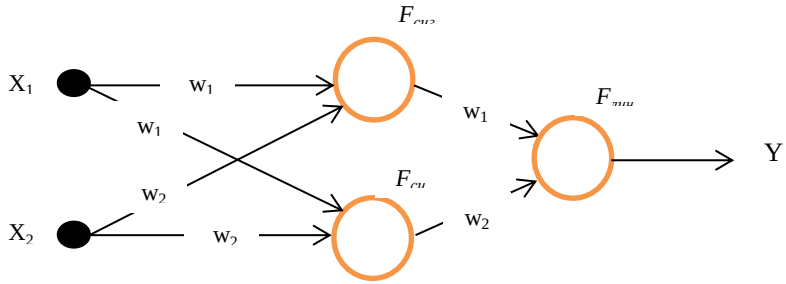
3.6. Пример обучения по методу обратного распространения ошибки

Постановка задачи.

Просчитать одну итерацию цикла обучения методом обратного распространения ошибки для нейросети, состоящей из 2 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 – линейная:



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 векторов). Выход нейронной сети согласно заданию соответствует

оператору «штрих Шеффера». Поэтому таблица 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-ая 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).

$S_1=x_1w_{11}+x_2w_{21} = 0\cdot0.6 + 1\cdot0.1 = 0.1$;

$S_2=x_1w_{12}+x_2w_{22} = 0\cdot0.9 + 1\cdot0.5 = 0.5$;

$k = 0.9$;

$$Y_1 = \frac{1}{1+e^{-0.1\cdot0.9}} = 0.5224,$$

$$Y_2 = \frac{1}{1+e^{-0.5\cdot0.9}} = 0.61,$$

$S_3=Y_1\cdot w_1+Y_2\cdot w_2 = 0.5224\cdot0.3 + 0.61\cdot0.8 = 0.64472$;

$l = 0.7$,

$Y = l\cdot S = 0.4513$.

Ш а г 4 (обратный проход) изменение весов:

$\eta = 0.7$,

$\delta^2 = (d\text{-}Y)\cdot Y\cdot(1\text{-}Y) = (1\text{-}0.4513)\cdot 0.4613\cdot(1\text{-}0.4513) = 0.3587;$
$w_1(2) = w_1(1) + \eta\cdot\delta^2\cdot Y_1 = 0.3 + 0.7\cdot 0.3587\cdot 0.5224 = 0.431,$
$w_2(2) = w_2(1) + \eta\cdot\delta^2\cdot Y_2 = 0.8 + 0.7\cdot 0.3587\cdot 0.61 = 0.953,$
$\delta_1{}^1 = Y_1(1-Y_1)\cdot\sum_{l=1}^{H_{k+1}}\delta_l^{k+1}\cdot w_l^{k+1} = Y_1\cdot(1\text{-}Y_1)\cdot\delta^2\cdot w_1 =$
$= 0.5224\cdot(1\text{-}0.5224)\cdot 0.3587\cdot 0.3 = 0.0268,$
$\delta_2{}^1 = Y_2(1-Y_2)\cdot\sum_{l=1}^{H_{k+1}}\delta_l^{k+1}\cdot w_l^{k+1} = Y_2\cdot(1\text{-}Y_2)\cdot\delta^2\cdot w_2 =$
$= 0.61\cdot(1\text{-}0.61)\cdot 0.3587\cdot 0.3 = 0.0682,$
$w_{11}(2) = w_{11}(1) + \eta\cdot\delta_1{}^1\cdot x_1 = 0.6 + 0.7\cdot 0.0682\cdot 0 = 0.9,$
$w_{12}(2) = w_{12}(1) + \eta\cdot\delta_2{}^1\cdot x_1 = 0.9 + 0.7\cdot 0.3587\cdot 0.61 = 0.9,$
$w_{21}(2) = w_{21}(1) + \eta\cdot\delta_1{}^1\cdot x_2 = 0.1 + 0.7\cdot 0.0268\cdot 1 = 0.119,$
$w_{22}(2) = w_{22}(1) + \eta\cdot\delta_2{}^1\cdot x_2 = 0.5 + 0.7\cdot 0.0682\cdot 1 = 0.548.$

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.

3.7. Задачи

1. Просчитать одну итерацию цикла обучения по Δ-правилу однослойной бинарной однородной нейронной сети, состоящей из 2 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 2 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 2 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 2 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 2 нейронов иимеющей 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 2 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 2 нейронов и имеющей функции активации: линейную (k=0,7) и пороговую (Θ =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 2 нейронов и имеющей функции активации: пороговую (Θ =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 2 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 2 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 3 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 3 нейронов и имеющей сигмоидальную функцию активации (a=1). В качестве обучающей выборки использовать таблицу истинности для $X_1\rightarrow X_2\&X_3$, $X_1\&X_2$ 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 3 нейронов и имеющей линейную функцию активации (k=0,9). В качестве обучающей выборки использовать таблицу истинности для $X_3\rightarrow X_1\&X_2$, $X_2\&X_3$, $X_2\rightarrow X_3$ (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 3 нейронов и имеющей пороговую функцию активации (Θ=0,4). В качестве обучающей выборки использовать таблицу истинности для $(X_2\rightarrow X_1)\&X_3$ (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 3 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 3 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 3 нейронов и имеющей пороговую функцию активации (Θ=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 3 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 3 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 3 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 2 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 2 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 2 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 2 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 2 слоёв, причем в первом слое находится 2 нейрона и используется сигмоидальная функция активации (a=0,9), а во втором – 1, пороговая (Θ=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 2 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 2 слоёв, причем в первом слое находится 2 нейрона и используется пороговая функция активации (Θ=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 2 слоёв, причем в первом слое находится 2 нейрона и используется пороговая функция активации (Θ=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 2 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 2 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 2 слоёв, причем в первом слое находится 3 нейрона, а во втором – 2. Функция активации нейронов сети - пороговая (Θ=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 2 слоёв, причем в первом слое находится 3 нейрона, а во втором – 2. Функция активации нейронов сети – гиперболический тангенс (a=3) функция. Синаптические веса и обучающую выборку задать случайным образом (не нули).

33. Просчитать одну итерацию цикла обучения методом обратного распространения ошибки нейросети, состоящей из 2 слоёв, причем в первом слое находится 2 нейрона и используется сигмоидальная функция активации (a=0,9), во втором – 2, пороговая (Θ=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 2 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 2 слоёв, причем в первом слое находится 2 нейрона и используется пороговая функция активации (Θ=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 2 слоёв, причем в первом слое находится 2 нейрона и используется пороговая функция активации (Θ=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 3 слоёв, использующей пороговую функцию активации (Θ=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 2 слоёв, использующей пороговую функцию активации (Θ=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 2 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.

Контрольные вопросы

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: Функция:

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

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

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

Например, а1 б2 в3

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.

Практическое занятие № 4. НЕ-факторы знаний

Целью занятия является изучение НЕ-факторов знаний, вероятностного и нечеткого подхода, применяемого в интеллектуальных системах, а также решение практической задачи нечеткого вывода.

Рекомендуемый объем занятия по теме – 4 часа.

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).

Но самое 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 Лотфи Заде.

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

а) как количественно выразить достоверность, надежность посылок?

б) как выразить степень поддержки заключения конкретной посылкой?

в) как учесть совместное влияние нескольких посылок на заключение?

г) 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 C может в различной степени поддерживаться несколькими посылками (например, заключение C может поддерживаться посылкой A посредством продукции A→C с уверенностью ct(A→C) и посылкой B посредством продукции B→C 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 C, полученного из посылки A применением последовательности продукций A→B, B→C, обеспечивающих степени поддержки, соответственно, ct(A→B) и ct(B→C)).

4.1. Вероятностный 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:

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

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:

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

$$P(H_i:E) = \frac{P(E:H_i) \times P(H_i)}{\sum_{k=1}^{m} P(E:H_k) \times P(H_k)} \qquad (2)$$

$$P(H_i:E_1,E_2,...,E_n) = \frac{P(E_1:H_i) \times P(E_2:H_i) \times ... \times P(E_n:H_i) \times P(H_i)}{\sum_{k=1}^{m} P(E_1:H_k) \times P(E_2:H_k) \times ... \times P(E_n:H_k) \times P(H_k)} \qquad (4)$$

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. Шансы и вероятности связаны между собой следующей формулой:

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

В некоторых странах использование шансов более распространено, чем использование вероятностей. Кроме того, использование шансов вместо вероятностей может быть более удобным с точки зрения вычислений.

Переходя к шансам в рассмотренных нами формулах, получим:

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

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, позволяющую 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:

$$ct(\text{посылки}) \cdot ct(\text{импликации}) = ct(\text{заключения}).$$

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:

$$(E1 \& E2) \rightarrow C \text{ или } (E1 \vee E2) \rightarrow C.$$

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:

$$ct(E1 \vee E2) = \max \{ct(E1), ct(E2)\}.$$

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• ct2 ;

б) если оба коэффициента отрицательны, то

ct = ct1 + ct2 + ct1• ct2 ;

в) если один из коэффициентов положителен, а другой отрицателен, то

$$ct = \frac{ct_1 + ct_2}{1 - \min(|ct_1|, |ct_2|)}$$

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 термам 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.

4.2. Нечеткий подход

Рассмотрим теперь другой, отличный от вероятностного, подход к представлению неопределенности знаний - нечеткую логику Л. Заде. Она также уже обеспечила себе коммерческий успех в самых разных приложениях: разнообразные системы управления (от бытовой техники до автомобильных АБС-систем), автофокусирование видеокамер и фотоаппаратов и т.п. Она 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, где вместо четких применяются экспертные знания, а не детерминированные алгоритмы.

Нечетким множеством Ã, по Заде, называется множество, определенное на произвольном непустом множестве *X* как множество пар вида:

$$\tilde{A} = \{\mu_{\tilde{A}}(x)/x\},$$

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 $\{x|x \epsilon X, \mu_A(x)>0\}$.

Ядром нечеткого множества называется четкое подмножество базового множества *X,* элементы которого имеют степени принадлежности равные единице: $\{x|x \epsilon X, \mu_A(x) = 1\}$.

α-сечением нечеткого множества называется четкое подмножество базового множества *X,* элементы которого имеют степень принадлежности большие или равные α :$\{x|x \epsilon X, \mu_A(x) \geq \alpha, \ \alpha \in [0, 1]\}$.

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.

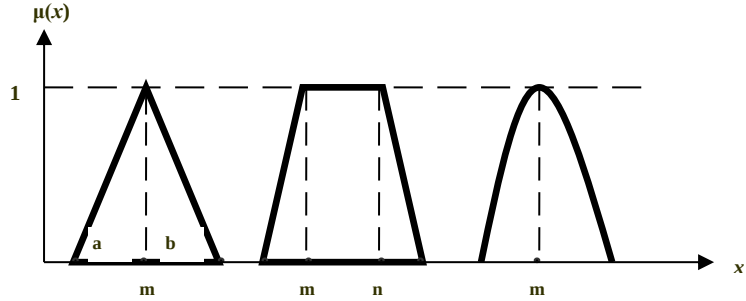
Мера истинности нечеткого высказывания $\tilde{A}$ определяется функцией принадлежности $\mu_{\tilde{A}}(x)$, $x \epsilon X$, заданной на множестве X= {*false*, *true*}.

Аналогично четким множествам над нечеткими множествами можно производить ряд операций, которые могут определяться 3 способами:

Максиминные	$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$, $\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$.
Алгебраические	$\mu_{A\cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\cdot\mu_B(x)$, $\mu_{A\cap B}(x) = \mu_A(x)\cdot\mu_B(x)$.
Ограниченные	$\mu_{A\cup B}(x) = \min\{1, \mu_A(x) + \mu_B(x)\}$, $\mu_{A\cap B}(x) = \max\{0, \mu_A(x) + \mu_B(x) - 1\}$.

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:

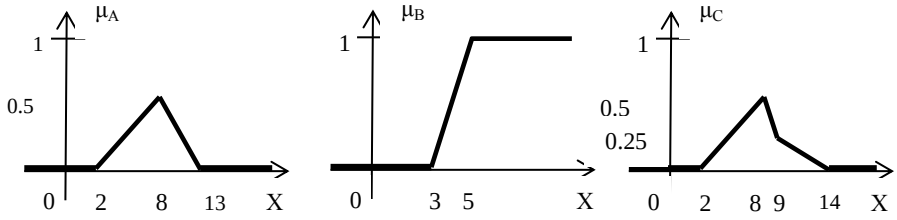


При графическом определении функций принадлежности объединенного множества необходимо в каждой точке множества выбрать максимальное значение из двух (точку того графика, который выше) и объединить все полученные точки в график, который и будет отображением новой функции принадлежности.

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.

Пример. Даны 3 нечетких множества *A, B, C* (графически заданы их функции принадлежности):



Построить функцию принадлежности нечеткого множества

D = Ā∩(A∪C∪B) и определить степень принадлежности одного элемента множеству D.

Описание процесса решения. 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.

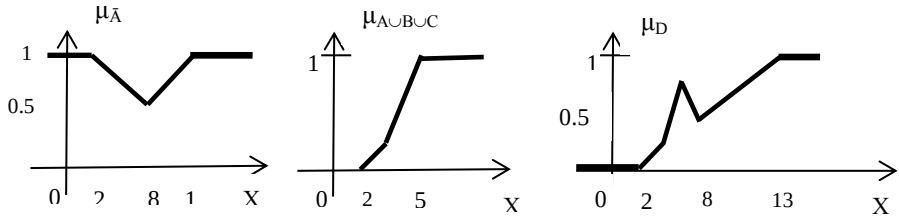
3) Взять произвольный элемент, входящий в ядро итогового множества, и определить аналитически степень принадлежности элемента множеству.

4) Проверить аналитические вычисления по построенному графику функции принадлежности.

Решение.

1) Множество D = Ā∩(A∪C∪B) , значит, последовательность операций будет следующей: Ā , A∪C∪B , Ā ∩ (A∪C∪B).

2) Построим согласно этой последовательности операций графики функций принадлежности:



3) Ядро множества D состоит из элементов интервала (2,13). Выберем для аналитической проверки, например, элемент 8. Имеем

$\mu_A(8) = 0.5$; $\mu_B(8) = 1$; $\mu_C(8) = 0.5$;

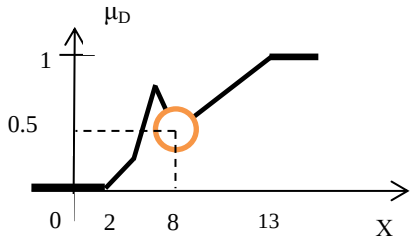
$\mu_{\bar{A}}(8) = 1 - 0.5 = 0.5$;

$\mu_{A\cup C}(8) = \min\{1, \mu_A(8) + \mu_C(8)\} = \min\{1, (0.5 + 0.5)\} = 1$;

$\mu_{A\cup C\cup B}(8) = \min\{1, \mu_{A\cup C}(8) + \mu_B(8)\} = \min\{1, (1 + 1)\} = 1$;

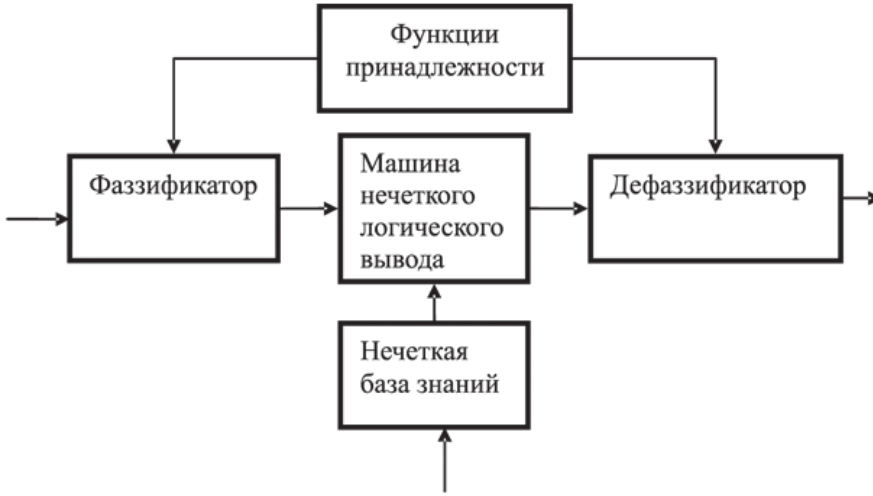
$$\mu_{\bar{A}\cap(A\cup C\cup B)}(8) = \\ = \max\{0, \mu_{\bar{A}}(8) + \mu_{A\cup B\cup C}(8) - 1\} \\ = \max\{0, (0.5 + 1 - 1)\} = 0.5.$$

4) $\mu_D(8) = 0.5$:



4.3. Машина нечеткого вывода

Архитектура интеллектуальной системы, основанной на использовании нечеткого вывода, имеет вид, представленный на рис.:



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:

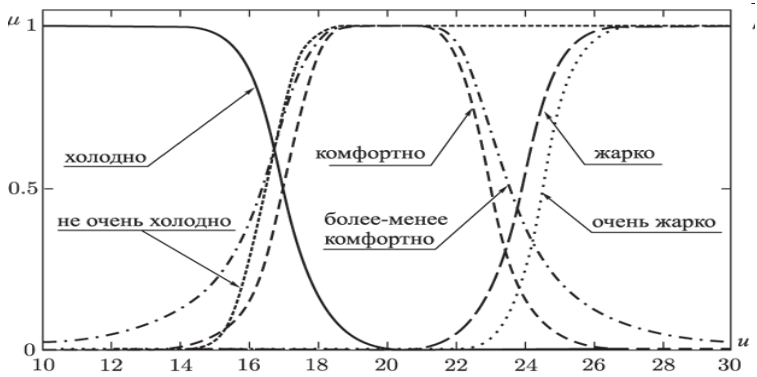
- существует хотя бы одно правило для каждого лингвистического терма выходной переменной;
- для любого терма входной переменной имеется хотя бы одно правило, в котором этот терм используется в качестве предпосылки.

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 ___»:



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, некоторая синтаксическое правило G для образования имен новых значений x, а также некоторая семантическая процедура M, позволяющая превратить каждое новое имя в нечеткую переменную, т.е. сформировать соответствующее нечеткое множество, задав функции принадлежности.

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).

Описание процесса решения. Для реализации нечеткого логического вывода необходимо выполнить следующее:

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 фазификацию (входные данные выбираем случайным образом).

5) Провести аккумуляцию.

6) Провести дефазификацию.

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:

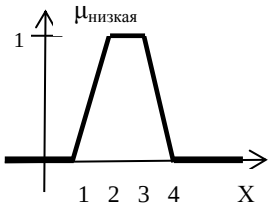
1. x = успеваемость студента, T(x) = («высокая», «средняя», «низкая»), X = [2, 5] (5-балльная система), G = («очень низкая», «высокая или средняя»), M – уменьшение на единицу степени принадлежность нечеткой переменной «высокая», операция объединения нечетких множеств;

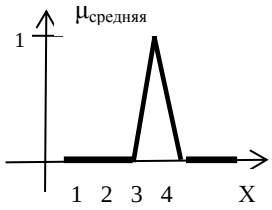
2. x - количество вариантов, T(x) = («мало», «достаточно», «много»), X = [1, 20] (20 вариантов в каждой теме), G = («очень много», «достаточно или мало»), M – увеличение на единицу степени принадлежность нечеткой переменной «много», операция объединения нечетких множеств;

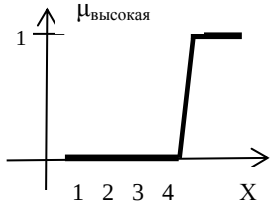
3. *x* - количество времени, T(*x*) = («мало», «достаточно», «много»), *X* = [1, 7] (количество часов в неделю, уделенных СИИ), *G* = («очень много», «достаточно или мало»), *M* – увеличение на единицу степени принадлежность нечеткой переменной «много», операция объединения нечетких множеств.

Для полного задания лингвистической переменной необходимо определить нечеткие переменные, входящие в T (успеваемость студента, количество вариантов и количество времени), задав их функции принадлежности:

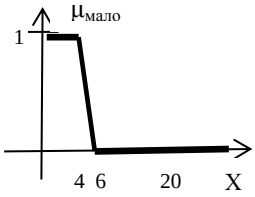
Успеваемость

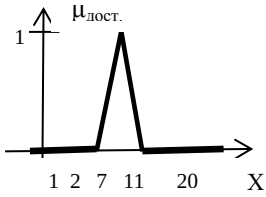


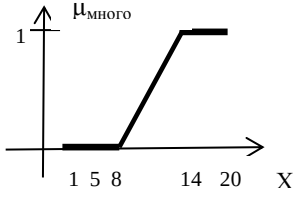




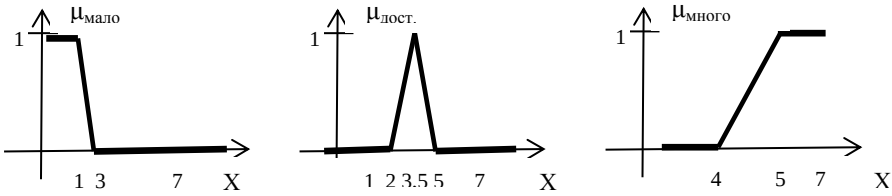
Количество вариантов







Количество времени



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 1 правиле, «достаточно» - в 2 и 4, «много» - в третьем;
- 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 1 и 2 правиле, «средняя» - 1,2 и 4, «низкая» - в 3, «мало» - в 1, «достаточно» - 4, «много» - 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;

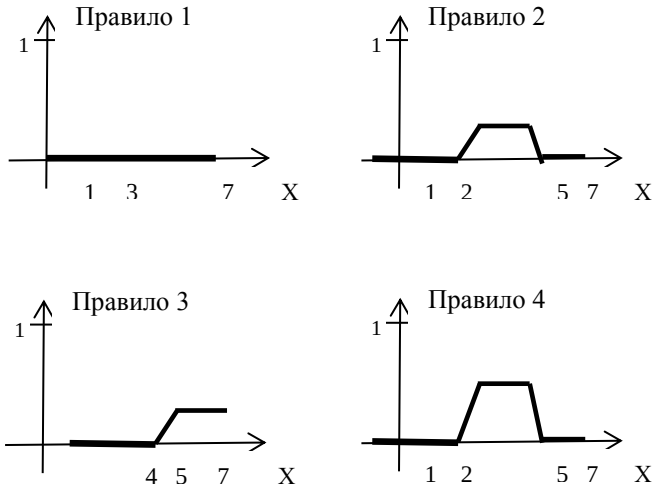
Успеваемость = «низкая» - 1;

Количество вариантов = «мало» - 0;
Количество вариантов = «достаточно» - 0.5;
Количество вариантов = «много» - 0.125.

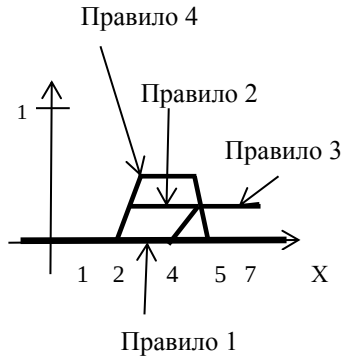
Определим степени уверенности посылок правил:

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.

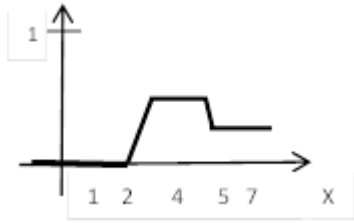
Построим новую выходную нечеткую переменную, используя полученные степени уверенности:



4) Аккумуляция:



Новый терм выходной переменной:
Количество часов



5) Исходя из полученного графика степени принадлежности выходного терма, можно сказать, что Иванову В.В., имеющему среднюю оценку 3.5, на решение 9 вариантов заданий понадобится не менее 2.75 часа (степень уверенности данного утверждения 0.5).

4.4. Задачи

1) 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.

2) 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.

3) Задать 3 нечетких множества A, E, C их функциями принадлежности. Построить функцию принадлежности нечеткого множества D = A∪Ē∩C и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.

4) Задать 3 нечетких множества A, E, C их функциями принадлежности. Построить функцию принадлежности нечеткого множества D = A∪Ē∩C и определить степень принадлежности одного элемента множеству D, используя метод ограничений.

5) Задать 3 нечетких множества A, B, G их функциями принадлежности. Построить функцию принадлежности нечеткого множества D = Ā∪B∩Ḡ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.

6) Задать 3 нечетких множества A, B, G их функциями принадлежности. Построить функцию принадлежности нечеткого множества D = (Ā∪B)∩Ḡ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.

7) Задать 3 нечетких множества A, B, G их функциями принадлежности. Построить функцию принадлежности нечеткого множества D = (Ā∪B)∩Ḡ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.

8) Задать 3 нечетких множества A, B, G их функциями принадлежности. Построить функцию принадлежности нечеткого множества D = Ā∩(G∪B)∩Ḡ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.

9) Задать 3 нечетких множества A, B, G их функциями принадлежности. Построить функцию принадлежности нечеткого множества D = Ā∩(G∪B)∩Ḡ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.

10) Задать 3 нечетких множества A, B, G их функциями принадлежности. Построить функцию принадлежности нечеткого множества D = Ā∩(G∪B)∩Ḡ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.

11) Построить нечеткую базу знаний (использовать не менее 3 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).

12) Построить нечеткую базу знаний (использовать не менее 3 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).

13) Построить нечеткую базу знаний (использовать не менее 3 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).

14) Построить нечеткую базу знаний (использовать не менее 3 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).

15) Построить нечеткую базу знаний (использовать не менее 3 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).

16) Построить нечеткую базу знаний (использовать не менее 3 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).

17) Построить нечеткую базу знаний (использовать не менее 3 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).

18) Построить нечеткую базу знаний (использовать не менее 3 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).

19) Построить нечеткую базу знаний (использовать не менее 3 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).

20) Построить нечеткую базу знаний (использовать не менее 3 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

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 $\boldsymbol{R \vee S \rightarrow T}$. Коэффициенты уверенности посылок равны ct($\boldsymbol{R}$) =0,85 , ct($\boldsymbol{S}$) =0,9 . Коэффициент уверенности правила ct($\boldsymbol{R \vee S \rightarrow T}$) = 0,5. Тогда коэффициент уверенности для логического вывода ct($\boldsymbol{T}$) 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 $\boldsymbol{A\&B \rightarrow C}$. Коэффициенты уверенности посылок равны ct($\boldsymbol{A}$) =0,3 , ct($\boldsymbol{B}$) =0,6 . Коэффициент уверенности правила ct($\boldsymbol{A\&B \rightarrow C}$) = 0,5. Тогда коэффициент уверенности для логического вывода ct($\boldsymbol{C}$) 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 , ∞], б) [-∞, +∞], 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:

$$\text{а) } \min\{1, \mu_A(x) + \mu_B(x)\},$$

$$\text{б) } \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x),$$

$$\text{в) } \max\{0, \mu_A(x) + \mu_B(x) - 1\},$$

$$\text{г) } \max\{\mu_A(x), \mu_B(x)\}.$$

8. В нечеткой логике степень истинности конъюнкции нескольких высказываний определяется: а) наиболее правдоподобным, б) наименее правдоподобным, в) средним значением.

Библиографический список

1. Родзин С.И. Искусственный интеллект: Уч. пособие. Таганрог: Изд-во ТТИ ЮФУ, 2009. 200 с.
2. Башмаков А.И., Башмаков И.А. Интеллектуальные информационные технологии. М.: Изд-во МГТУ им. Н.Э. Баумана, 2005. 304 с.
3. Джексон П. Введение в экспертные системы. М.: Издательский 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. Липатова С.В. Сборник задач по курсу «Интеллектуальные информационные системы»: Уч. пособие. Ульяновск: УлГУ, 2010. 64 с.

7. Родзин С.И., Ковалев С.М. Информационные технологии: интеллектуализация обучения, моделирование эволюции, распознавание речи. Ростов-на-Дону: Изд-во СКНЦ ВШ, 2002. 224 с.
8. Частиков А.П. и др. Разработка экспертных систем. Среда CLIPS. СПб.: БХВ-Петербург, 2003. 608 с.
9. Штовба С.Д. Введение в теорию нечетких множеств и нечеткую логику.–http://matlab.exponenta.ru/fuzzylogic/book1/index.php.
10. Ярушкина Н.Г. Основы теории нечетких и гибридных систем: Учеб. пособие. М.: Финансы и статистика, 2004. 320 с.
11. Рассел С., Норвинг П. Искусственный интеллект: современный подход. М: Издат. дом «Вильямс», 2006. 1408 с.
12. Ясницкий Л.Н. Искусственный интеллект: Уч. пособие. М.: БИНОМ. Лаборатория знаний, 2011. 240 с. http://lbz.ru/books/232/5563/.