Министерство образования и науки Российской Федерации

**Государственное образовательное учреждение
высшего профессионального образования
«Тамбовский государственный технический университет»**

**В.Е. Подольский, И.Л. Коробова, И.В. Милованов,
И.А. Дьяков, Н.В. Майстренко**

# МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ СИНТЕЗА ПРОЕКТНЫХ РЕШЕНИЙ

Рекомендовано Учёным советом университета
в качестве учебного пособия для студентов 5 курса дневного отделения
специальности 230104 и магистрантов направления 230100

Тамбов
♦Издательство ГОУ ВПО ТГТУ♦
2010

УДК 004.896(075)
ББК ←813я730
М545

Р е ц е н з е н т ы :

Заместитель директора ФГУ ГНИИ ИТТ «Информатика»,
доктор технических наук
*А.К. Скуратов*

Декан факультета электронной техники и приборостроения
ГОУ ВПО СГТУ доктор технических наук, профессор
*А.А. Большаков*

Доктор технических наук, профессор ГОУ ВПО ТГТУ
*С.В. Карпушкин*

М545 Методы искусственного интеллекта для синтеза проектных решений : учебное пособие / В.Е. Подольский, И.Л. Коробова, И.В. Милованов, И.А. Дьяков, Н.В. Майстренко. – Тамбов : Изд-во ГОУ ВПО ТГТУ, 2010. – 80 с. – 100 экз.
ISBN 978-5-8265-0954-8

Рассматриваются общие сведения по структуре и составу экспертных систем, различные методы принятия решения в экспертных системах. Приводятся описание языка Пролог, примеры использования методов искусственного интеллекта в подсистемах САПР.

Предназначено для студентов 5 курса дневного отделения специальности 230104 «Системы автоматизированного проектирования» и магистрантов направления 230100 программы 23010014 «Методы анализа и синтеза проектных решений», изучающих дисциплины «Интеллектуальные подсистемы в САПР», «Системы искусственного интеллекта», «Нейросетевые технологии в САПР».

УДК 004.896(075)

ББК ←813я730

**ISBN 978-5-8265-0954-8**

Учебное издание

ПОДОЛЬСКИЙ Владимир Ефимович,
КОРОБОВА Ирина Львовна,
МИЛОВАНОВ Игорь Викторович,
ДЬЯКОВ Игорь Алексеевич,
МАЙСТРЕНКО Наталья Владимировна

**МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА
ДЛЯ СИНТЕЗА ПРОЕКТНЫХ РЕШЕНИЙ**

Учебное пособие

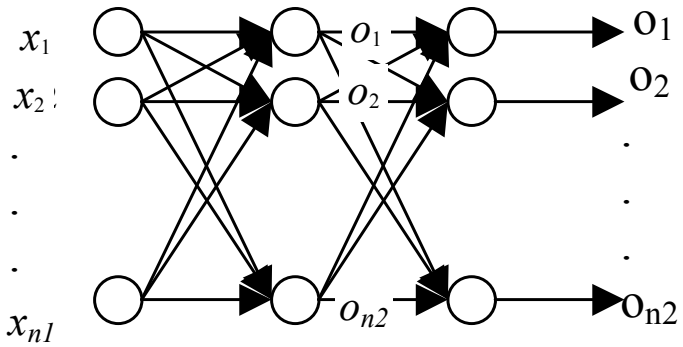
Редактор З.Г. Чернова
Инженер по компьютерному макетированию М.С. Анурьева

Подписано в печать 21.11.2010
Формат 60 × 84 /16. 4,65 усл. печ. л. Тираж 100 экз. Заказ № 515.

Издательско-полиграфический центр ГОУ ВПО ТГТУ
392000, Тамбов, Советская, 106, к. 14

**В.Е. Подольский, И.Л. Коробова, И.В. Милованов,
И.А. Дьяков, Н.В. Майстренко**

# МЕТОДЫ ИСКУССТВЕННОГОИНТЕЛЛЕКТА ДЛЯ СИНТЕЗА ПРОЕКТНЫХ РЕШЕНИЙ




**Тамбов**
**♦Издательство ГОУ ВПО ТГТУ♦**
**2010**

# ВВЕДЕНИЕ

Система искусственного интеллекта (ИИ) – это программная система, имитирующая на компьютере мышление человека. Для создания такой системы необходимо изучить процесс мышления человека, решающего определённые задачи или принимающего решение в конкретной области; выделить основные шаги этого процесса и разработать программные средства, воспроизводящие их на компьютере.

Существует несколько причин интеллектуализации систем автоматизированного проектирования (САПР):

- необходимость использования накопленного опыта специалистов конкретной предметной области;
- решение трудноформализуемых задач, т.е. задач, которые характеризуются сложностью или невозможностью полного и точного математического описания с последующим применением известных методов решения или большой размерностью, не позволяющей получить решение за приемлемое время.
- необходимость взаимодействия пользователя и САПР на языке, близком к естественному.

В САПР методы искусственного интеллекта используются для:

- разработки удобного интерфейса пользователя и системы (в том числе при обучении пользователя, самообучении САПР, ведения интеллектуального диалога);
- создания экспертных систем, позволяющих принимать решения в задачах структурного и параметрического синтеза;
- при создании подсистем переработки визуальной информации;
- при решении задач защиты информации.

Наибольшее внимание в пособии уделяется созданию экспертных систем, а также различным способам представления знаний.

Предлагаемое пособие в рамках учебных дисциплин “Интеллектуальные подсистемы САПР”, “Системы искусственного интеллекта”, “Нейросетевые технологии в САПР” для студентов специальности 230104 и магистрантов направления 230100 программы 23010014 “Методы анализа и синтеза проектных решений” предназначено для изучения методов представления знаний в системах искусственного интеллекта и дальнейшего их использования при принятии решений в процессе автоматизированного проектирования.

## 1. ОБЩИЕ СВЕДЕНИЯ О СИСТЕМАХ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Любая система искусственного интеллекта опирается на знания о процессе человеческого мышления [1 – 3].

### 1.1. ОСНОВНЫЕ СОСТАВЛЯЮЩИЕ ПРОЦЕССА МЫШЛЕНИЯ

***Цели.*** В основе человеческой деятельности лежит мышление. Когда утром звонит будильник, мозг человека даёт команду руке выключить звонок. Это не автоматическая реакция – решение конкретной задачи требует определённого ответа мозга. Целью называется конечный результат, на который направлены мыслительные процессы человека. Как только цель (выключить звонок) достигнута, перед человеком сразу встают новые цели, например, почистить зубы, одеться, позавтракать, выйти на автобусную остановку. Осуществление всех этих целей приводит к достижению главной цели – вовремя попасть на работу. Мысли, ведущие к конечному результату, не случайны, а строго обоснованы. Каждый шаг на пути к главной цели имеет свою локальную цель. Мозг всегда сосредоточен на цели независимо от того, выполняет ли человек простую физическую работу или решает сложную интеллектуальную задачу.

При проектировании экспертной системы всегда следует помнить о цели, для достижения которой она предназначена [2].

***Факты и правила***. Человек хранит большое количество знаний. В общем случае, интеллект можно представить как совокупность фактов и правил их использования. Отчасти цели достигаются с помощью правил использования всех известных фактов.

***Пример***.

Факт 1: *Температура кипения воды 100°С.*

Правило 1: *ЕСЛИ температура процесса меньше 100°С, ТО для нагрева можно использовать воду.*

Правило 2: *ЕСЛИ температура процесса больше 100°С, ТО для нагрева используют электронагреватель.*

Заметим, что в приведённом примере все правила выражены условным отношением вида *ЕСЛИ…ТО…*, т.е. если выполняется некоторое условие, то следует некоторое действие [2].

***Упрощение.*** Когда человеческий мозг приступает к решению даже самой простой задачи, для выбора нужных действий в его распоряжении имеется огромный объём информации. Например, переходя улицу, человек анализирует скорость и объём движения, расстояние до противоположного тротуара, сигнал светофора. Одновременно мозг обрабатывает впечатления, не имеющие прямого отношения к переходу улицы (например, цвет проезжающих машин, вид деревьев и окружающих зданий, одежду проходящих мимо людей и пр.). Если бы человек, прежде чем шагнуть на проезжую часть, анализировал все факты, имеющие хоть какое-нибудь отношение к цели, он простоял бы на тротуаре несколько лет. Но мышление человека включает сложную систему, руководящую выбором правильной реакции на конкретную ситуацию. Такой выбор называется упрощением. Механизм упрощения блокирует факты и правила, не имеющие прямого отношения к решаемой в данный момент задаче [2]. Схематично работа механизма упрощения представлена на рис. 1.

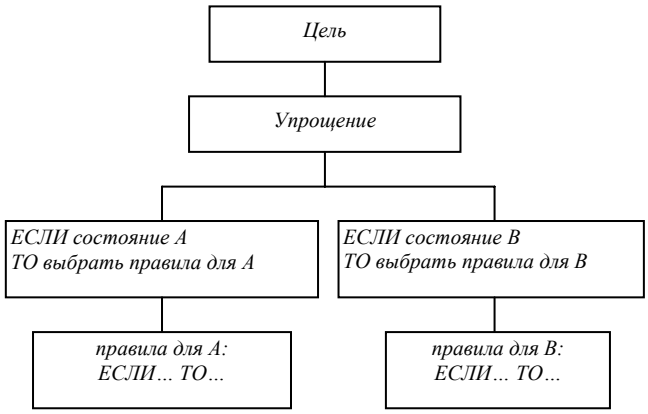



**Рис. 1**

***Механизм вывода.*** Достигая цели, человек не только приходит к решению поставленной перед ним задачи, но и одновременно приобретает новые знания. Часть интеллекта, которая помогает извлекать новые факты, называется механизмом вывода [2].

***Пример***.

Факт 1: *Мария и Петр – родители Вани.*

Факт 2: *Мария и Петр – родители Тани.*

Правило 1: *ЕСЛИ у мальчика и девочки одни и те же родители ТО дети – брат и сестра.*

Цель: *Определить степень родства Тани и Вани.*

С помощью имеющихся фактов и правил цель может быть достигнута сразу. Кроме того, в процессе достижения цели получен новый факт.

Новый факт: *Ваня и Таня – брат и сестра.*

## 1.2. СТРУКТУРА И СОСТАВ СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Любая система искусственного интеллекта должна содержать все элементы, составляющие процесс принятия решения человеком: цели, знания, механизмы вывода и упрощения [1 – 4].

Порядок работы таких систем (т.е. систем, основанных на правилах) представлен на рис. 2.

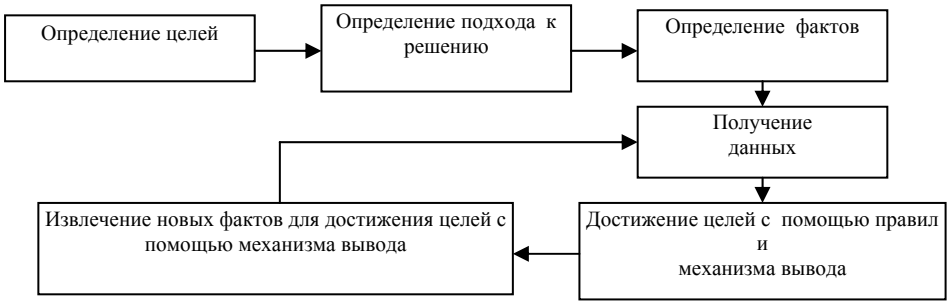



**Рис. 2**

Типовая система включает в себя следующие основные компоненты (рис. 3): 1) базу знаний (БЗ); 2) механизм вывода; 3) модуль извлечения знаний; 4) систему объяснений.

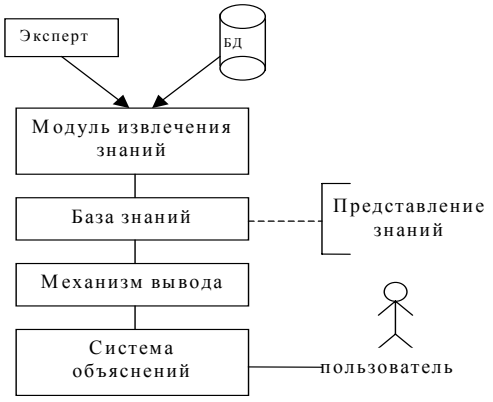



**Рис. 3**

### 1.2.1. База знаний

Важнейшей составляющей системы искусственного интеллекта является БЗ, содержащая факты и правила, по которым в зависимости от входной информации принимается то или иное решение.

Факты представляют собой краткосрочную информацию, которая может изменяться в процессе решения задачи. Правила представляют более долговременную информацию о том, как порождать новые факты и гипотезы из имеющихся данных.

Основное отличие БЗ от обычной методики использования базы данных (БД) состоит в больших творческих возможностях. Факты в БД обычно пассивны: они либо там есть, либо их там нет. База знаний, со своей стороны, активно пытается пополнить недостающую информацию.

Правила в формате *ЕСЛИ ... ТО ...* являются распространённым, но не единственным способом представления знаний. Для этой цели используются семантические сети, фреймы, нейронные сети и другие способы. Они будут рассмотрены далее. На некотором глубоком уровне все типы представления данных должны быть, очевидно, эквивалентны между собой. Выбор того или иного способа определяется видом задачи и спецификой предметной области.

Многие правила являются эвристическими, основанными на опыте экспертов в данной предметной области. Если алгоритмический метод гарантирует корректное или оптимальное решение, то эвристический метод даёт в большинстве случаев лишь приемлемое решение.

База знаний является входным потоком данных для механизма логического вывода.

### 1.2.2. Механизм логического вывода

Механизмом логического вывода называются общие знания о процессе нахождения решения. Он выполняет две основные функции: 1) дополнение, изменение БЗ на основе анализа БЗ и исходной информации; 2) управление порядком обработки правил в БЗ.

Если база знаний содержит высококачественные знания о предметной области, то механизм логического вывода содержит информацию о том, как эти знания эффективно использовать.

Механизм логического вывода функционирует циклически. В каждом цикле решаются следующие задачи:

1. Сопоставление – эта задача предполагает сравнение условных частей правил с исходными данными и имеющимися фактами в БЗ.
2. Выбор – в случае наличия множества правил с истинностью условных частей необходимо выбрать одно из них для срабатывания.
3. Действие – эта задача предполагает выполнение какого-либо действия, предусмотренного в случае срабатывания правила. Обычно это приводит к выполнению какого-либо физического действия и к модификации БЗ.

Таким образом, каждый цикл начинается с последовательного просмотра всех правил и сопоставления их условных частей с исходными данными и фактами в БЗ. Если правил, у которых условные части и факты совпадают, несколько, то возникает конфликтное множество правил. На основе каких-либо критериев выбирается одно правило, которое считается сработавшим, и выполняется действие.

Существует две основные стратегии логического вывода [2, 4].

1. Прямая цепочка рассуждений – основана на сопоставлении исходных данных с правилами и фактами БЗ с получением результата.
2. Обратная цепочка рассуждений – предполагает, что выдвигается некоторая гипотеза о предполагаемом решении задачи и путём анализа БЗ ищется подтверждение этой гипотезы путём сравнения результатов с исходными данными. Если гипотеза не подтверждается, то ищется новое решение. Наиболее ценными являются ЭС, которые реализуют прямую и обратную цепочки рассуждений.

### 1.2.3. Модуль извлечения знаний

Важной составной частью системы искусственного интеллекта является модуль извлечения знаний. Его основное назначение – предоставление экспертных знаний, их структурирование в виде, пригодном для использования в компьютерной системе. В задачу модуля входит приведение правила к виду, позволяющему применить это правило в процессе работы. В простейшем случае в качестве такого модуля может выступать обычный редактор, который просто заносит правила в файл.

В некоторых системах извлечение знаний осуществляется не одним, а несколькими способами, например, часть знаний извлекается с помощью программных средств, анализирующих грамматику описания знаний (эта грамматика задаёт форму представления знаний); другие знания могут быть представлены графически и потребуются специальные средства, которые позволят воспринимать графически изображения и проверять их на правильность (например графически могут быть представлены электрические схемы); наконец, возможны знания, которые самой системой не используются, а при необходимости могут вводиться в диалоговом режиме.

Модуль извлечения знаний является наиболее трудоёмким и дорогостоящим.

### 1.2.4. Система объяснений

Система объяснений предназначена для показа пользователю всего процесса рассуждений, в результате которого было найдено или не найдено решение.

Большинство специалистов-пользователей не смогут с доверием относиться к выведенному системой заключению, пока не будут знать, как оно было получено. Если врач установил у вас наличие некоторого заболевания, то вы, конечно, захотите знать, почему он пришёл к такому выводу. Вы, вероятно, попросите показать вам рентгеновский снимок, результаты анализов или что-то другое, на основе чего врач сделал свое заключение. К системе искусственного интеллекта предъявляются те же самые требования, т.е. необходимо получить не только само решение, но и всю цепочку вывода в форме, понятной пользователю.

## 2. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ

Наиболее общими методами представления знаний являются: правила, семантические сети, фреймы.

В данном разделе рассматривается представление знаний в виде нечётких правил, а также в виде нейронных сетей.

### 2.1. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В ВИДЕ ПРАВИЛ

Такой способ является наиболее понятным и популярным методом представления знаний [1 – 4]. Правила обеспечивают формальный способ представления рекомендаций, знаний или стратегий. Они чаще подходят в тех случаях, когда предметные знания возникают из эмпирических ассоциаций, накопленных за годы работы по решению задач в данной области.

В системах, основанных на правилах, предметные знания представляются набором правил, которые проверяются на группе фактов и знаний о текущей ситуации (входной информации). Когда часть правила *ЕСЛИ* удовлетворяет фактам, то действия, указанные в части *ТО*, выполняются. Когда это происходит, то говорят, что правило срабатывает. Интерпретатор правил сопоставляет части правил *ЕСЛИ* с фактами и выполняет то правило, часть *ЕСЛИ* которого сходится с фактами, т.е. интерпретатор правил работает в цикле “сопоставить – выполнить”, формируя последовательность действий (рис. 4).

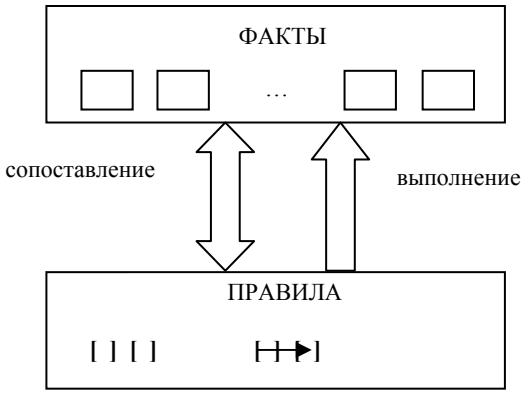



**Рис. 4**

Действия правил могут состоять:

– в модификации набора фактов в базе знаний, например добавление нового факта, который сам может быть использован для сопоставления с частями *ЕСЛИ*;

– во взаимодействии с внешней средой (например “Вызвать пожарную команду”).

***Пример.*** Рассмотрим базу знаний, которая включает три факта и девять правил.

Факты:

1) кислота = агрессивная жидкость;
2) азотная кислота = кислота;
3) сернистая кислота = кислота.

Правила:

1. *ЕСЛИ* среда = кислота *И* концентрация >70%
   *ТО* среда = концентрированная кислота.
2. *ЕСЛИ* среда = кислота *И* концентрация <70%
   *ТО* среда = разбавленная кислота.
3. *ЕСЛИ* среда = концентрированная кислота
   *ТО* материал ванны = хромоникелевая сталь.
4. *ЕСЛИ* среда = разбавленная кислота
   *ТО* материал ванны = углеродистая сталь.
5. *ЕСЛИ* среда = агрессивная жидкость *ТО* футеровка = есть.
6. *ЕСЛИ* футеровка = есть *И* среда = сернистая кислота
   *ТО* материал футеровки = свинец.
7. *ЕСЛИ* футеровка = есть *И* среда = азотная кислота
   *ТО* материал футеровки = винипласт.

8. *ЕСЛИ* температура = меньше 100 °С
   *ТО* тип обогрева = пароводяная рубашка.
9. *ЕСЛИ* температура = больше 100 °С
   *ТО* тип обогрева = электронагреватель.

Процесс сопоставления с фактами частей *ЕСЛИ* порождает цепочку выводов. Эта цепочка выводов показывает, как система, используя правила, выводит заключение. Цепочки выводов могут быть предъявлены пользователю, что помогает понять, как система достигает свои заключения.

Правила, по сравнению с другими способами представления знания, имеют следующие преимущества [2, 4]:

1) модульность;
2) единообразие структуры;
3) естественность (вывод заключения в такой системе аналогичен процессу рассуждения эксперта);
4) гибкость иерархии понятий, которая поддерживается только как связи между правилами (изменив правило, вы можете изменить иерархию).

Однако, такие системы не свободны от недостатков:

1) процесс вывода менее эффективен, чем при других способах представления, так как большая часть времени затрачивается на непроизводительную проверку применимости правил;
2) этот процесс трудно поддаётся управлению;
3) сложно представить иерархию понятий.

Представление знаний в виде правил иногда называют плоским (по аналогии с реляционными базами данных), так как в них отсутствуют средства для установления иерархии правил. Объём базы знаний растёт линейно по мере включения в неё новых фрагментов знаний. Большинство существующих коммерческих экспертных систем основаны на правилах. При этом правила могут быть представлены в одном из двух видов [5].

1. Если в зависимости от возможных чётких значений входных параметров делается вывод о значениях выходного параметра, то такая система называется системой $L^{(1)}$-типа. Данная система представляется в виде

$$L^{(1)} = \begin{cases} L_1^{(1)} : \langle ЕСЛИ\ A_1\ ТО\ B_1 \rangle \\ L_2^{(1)} : \langle ЕСЛИ\ A_2\ ТО\ B_2 \rangle \\ \cdots \\ L_m^{(1)} : \langle ЕСЛИ\ A_m\ ТО\ B_m \rangle \end{cases} \quad (1)$$

где $m$ – число экспертных высказываний; $Aj$ – чёткое значение входного параметра; $Bj$ – чёткое значение выходного параметра или некоторое конкретное действие процесса проектирования.

2. В случаях, когда в зависимости от возможных значений выходной ситуации ($Bj$) экспертом делается предположение о возможной входной ситуации ($Aj$), система экспертных высказываний называется системой $L^{(2)}$-типа и представляется в виде

$$L^{(2)} = \begin{cases} L_1^{(2)} : \langle ЕСЛИ\ B_1\ ТО\ A_1 \rangle \\ L_2^{(2)} : \langle ЕСЛИ\ B_2\ ТО\ A_2 \rangle \\ \cdots \\ L_m^{(2)} : \langle ЕСЛИ\ B_m\ ТО\ A_m \rangle \end{cases} \quad (2)$$

## 2.2. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ С ИСПОЛЬЗОВАНИЕМ ФРЕЙМОВ

Системы, базы знаний которых насчитывают сотни правил, отнюдь не считаются чем-то необычным [1]. При такой сложности системы для инженера знаний процесс обновления состава правил и контроль связей между ними становится весьма затруднительным, поскольку добавляемые правила могут дублировать имеющиеся знания или вступать с ними в противоречие. Для выявления подобных фактов можно использовать программные средства, но включение их в работу системы приводит к ещё более тяжёлым последствиям – потере работоспособности, так как в этом случае инженер знаний теряет представление о том, как взаимодействуют правила. Так как возрастает количество связей между понятиями, инженеру знаний трудно их контролировать.

Представление знаний, основанных на фреймах [1 – 3], является альтернативным по отношению к системам, основанным на правилах: оно даёт возможность хранить иерархию понятий в базе знаний в явной форме.

Фреймом называется структура для описания стереотипной ситуации, состоящая из характеристик этой ситуации и их значений.

Характеристики называются слотами, а значения – заполнителями слотов. Слот может содержать не только конкретное значение, но и имя процедуры, позволяющей вычислить его по заданному алгоритму, а также одно или несколько правил, с помощью которых это значение можно найти.

В слот может входить не одно, а несколько значений. Иногда слот включает компонент, называемый фасетом, который задаёт диапазон или перечень его возможных значений.

Как уже отмечалось, помимо конкретного значения в слоте могут храниться процедуры и правила, которые вызываются при необходимости вычисления этого значения. Если, например фрейм, описывающий человека, включает слоты “*Дата рождения*” и “*Возраст*”, и в первом из них находится некоторое значение, то во втором слоте может стоять процедура, вычисляющая возраст по дате рождения и текущей дате.

Процедуры, располагающиеся в слоте, называются связанными процедурами. В предыдущем примере связанная процедура будет активизироваться при каждом изменении текущей даты.

Чаще всего используются процедуры трех видов:

1) процедура “*ЕСЛИ* – добавлено” – выполняется, когда новая информация помещается в слот;
2) процедура ”*ЕСЛИ* – удалено” – выполняется, когда информация удаляется из слота;
3) процедура “*ЕСЛИ* – нужно” – выполняется, когда запрашивается информация из слота, а он пустой.

Эти процедуры могут проверять, что при изменении значения производятся соответствующие действия.

Совокупность фреймов, моделирующая какую-нибудь предметную область, представляет собой иерархическую структуру, в которую соединяются фреймы. На верхнем уровне иерархии находится фрейм, содержащий наиболее общую информацию, истинную для всех остальных фреймов. Фреймы обладают способностью наследовать значения характеристик своих родителей, находящихся на более высоком уровне иерархии. Значения характеристик фреймов могут передаваться по умолчанию фреймам, находящимся ниже них в иерархии, но, если последние содержат собственные значения данных характеристик, то в качестве истинных данных принимаются именно они. Это обстоятельство позволяет легко учитывать во фреймовых системах различного рода исключения.

Различают статические и динамические системы фреймов. В системах статических фреймы не могут быть изменены в процессе решения задачи, в динамических системах это допустимо.

Наиболее ярко достоинства фреймовых систем представления знаний проявляется в том случае, если связи между объектами изменяются нечасто и предметная область насчитывает немного исключений. Значения слотов представляются в системе в единственном экземпляре, поскольку включается только в один фрейм, описывающий наиболее общее понятие из всех тех, которые содержат слот с данным именем. Такое свойство систем фреймов даёт возможность уменьшить объём памяти, необходимый для их размещения в компьютере. Однако основное достоинство состоит не в экономии памяти, а в представлении в базе знаний связей, существующих между понятиями предметной области.

Фрагмент базы знаний о свойствах горения нитей приведён на рис. 5.

Данный фрагмент основан на следующих знаниях:

1) анидные и капроновые нити являются нитями из полиамидного волокна;
2) нити из синтетических волокон включают полиамидные, лавсановые и акриловые нити;
3) нити из синтетических волокон горят;
4) продуктом сгорания большинства нитей из синтетических волокон является твёрдый шарик;
5) нити из полиамидного волокна горят медленно;
6) нити из лавсана и акрила горят быстро;
7) при сгорании акриловой нити образуется рыхлый шарик;
8) при горении анидной и акриловой нити запах отсутствует;
9) при горении капроновой нити чувствуется запах сургуча;
10) при горении лавсановой нити чувствуется сладковатый запах.

В результате с помощью базы знаний можно получить новые знания, например: капроновая нить горит медленно, при горении чувствуется запах сургуча, после сгорания образуется твёрдый шарик.


**Рис. 5**

## 2.3. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ С ИСПОЛЬЗОВАНИЕМ СЕМАНТИЧЕСКИХ СЕТЕЙ

Термин “семантическая сеть” используется для описания метода представления знания, основанного на сетевой структуре [1 – 3]. Этот метод является одним из наиболее эффективных методов хранения знаний. Семантические сети состоят из: узлов, соответствующих объектам, понятиям и событиям; дуг, связывающих узлы и описывающих отношения между ними.

Иными словами, семантическая сеть отображает совокупность объектов предметной области и отношений между ними. При этом, объектам соответствуют вершины сети, а отношениям – соединяющие их дуги.

В семантическую сеть включаются только те объекты предметной области, которые необходимы для решения прикладных задач. В качестве объектов могут выступать события, действия, обобщённые понятия или свойства объектов.

Вершины сети соединяются дугой, если соответствующие объекты предметной области находятся в каком-либо отношении. Наиболее распространёнными являются следующие типы отношений: а) “является” – означает, что объект входит в состав данного класса; б) “имеет” – позволяет задавать свойства объектов.

Возможны также отношения вида: а) “является следствием” – отражает причинно-следственные связи; б) “имеет значение” – задаёт значение свойств объектов.

***Пример.*** Требуется составить семантическую сеть для отображения следующих знаний:

1) оборудование для выполнения основных операций электрохимических покрытий представляет собой основное технологическое оборудование гальванических комплексов;
2) ванны цинкования и меднения – это оборудование для выполнения операций электрохимических покрытий;
3) ванны цинкования имеют защитную футеровку и вытяжную вентиляцию;
4) в состав вытяжной вентиляции входят бортовые отсосы;
5) оборудование для выполнения основных операций электрохимических покрытий имеет аппаратуру управления и регулирования *р*-тока и теплообменные аппараты.

Из построенной семантической сети (рис. 6) вытекают, например, следующие дополнительные факты:

1) ванны цинкования являются основным технологическим оборудованием гальванических комплексов;
2) бортовые отсосы входят в состав ванны цинкования;
3) ванны меднения имеют теплообменную аппаратуру.

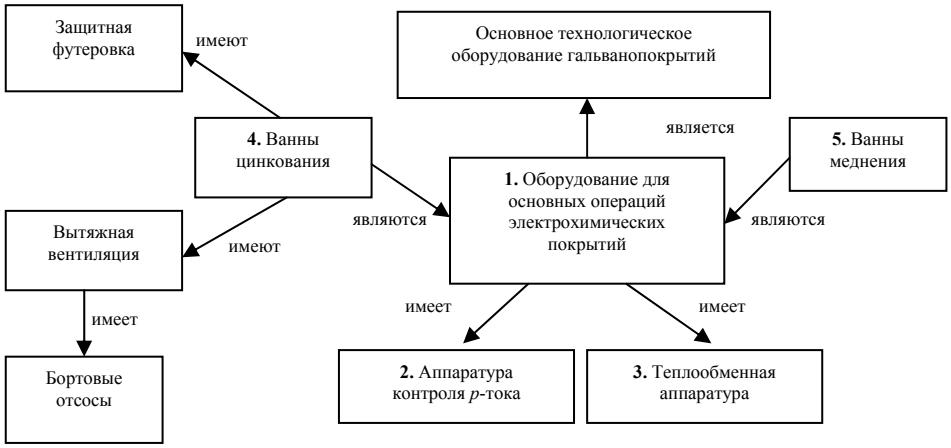



**Рис. 6**

Свойства семантических сетей наследовать узлами более высокого уровня свойств узлов более низкого уровня, принято называть иерархией наследования. Принцип иерархии наследования позволяет исключать дублирование информации в семантических сетях. Например, достаточно один раз связать узлы 1, 2 и 3 (рис. 6), чтобы не повторять информацию для узлов 4 и 5 и т.д.

Основной недостаток такого способа представления знаний – сложность обработки исключений.

## 2.4. НЕЙРОННЫЕ СЕТИ

Моделирование сложных систем требует большого числа знаний об объекте, в том числе экспериментальных и экспертных. Для их обработки в последнее время широко используются нейронные сети. В литературе встречаются несколько типов информационных моделей на основе нейронных сетей [6 – 15]:

- моделирование отклика системы на внешнее воздействие;
- классификация внутренних состояний системы;
- прогноз динамики изменения системы;
- оценка полноты описания системы и определение значимости параметров системы;
- оптимизация параметров системы по отношению к заданной целевой функции;
- управление системой.

В ряде случаев нейронные сети и физико-математические модели могут составлять единую модель, например, когда внешние условия описываются уравнениями кинетики, а отклик системы - нейронной сетью. Иногда используются гибридные нейронные модели, параметры которых являются нечёткими.

В основе теории нейронных сетей лежит желание воспроизвести функции мозга при решении конкретной задачи. Однако создающиеся системы не полностью воспроизводят функции мозга, а, скорее, представляют математическую модель, воспроизводящую отдельные возможности человеческого мозга, по аналогии с которым искусственные нейтронные сети характеризуются следующими свойствами [6, 15]:

– обучение (т.е. изменение поведения в зависимости от окружающей среды).

– обобщение (реакция сети после обучения будет, до известной степени, нечувствительна к малым изменениям входящих сигналов).

– абстрагирование (способность выявления различий во входных сигналах).

### 2.4.1. Описание биологического нейрона

Из нейробиологии известно, что человеческий мозг состоит из $10^{10} – 10^{11}$ нейронов. Один биологический нейрон схематично представлен на рис. 7. Он содержит клеточное тело и отростки (аксон и дендриты).

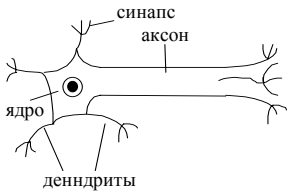



**Рис. 7**

*Клеточное тело* состоит из ядра и окружающей его цитоплазмы. На внешней поверхности содержится мембрана, включающая три слоя. Она отделяет клеточное тело от окружающих его крайних окончаний аксона.

*Аксон* (выход) – отросток нейрона, который служит для передачи нервных импульсов к другим нейронам или эффекторным органам (мускульным волокнам, клеткам желез).

*Дендриты* (входы) – отростки, которые связывают нейрон с другими нейронами. Связь осуществляется через специальные контакты, называемыми *синапсами*.

В упрощённом виде работу нейрона можно представить так. Клеточное тело принимает входной сигнал от других нейронов через синаптические связи дендритов, преобразует его и передает выходной сигнал через аксон другим нейронам. Скорость передачи зависит и от значений входных сигналов, и от силы синаптических связей. Несмотря на то, что функция нейрона – нелинейная, нейробиологи считают, что большинство нейронов производят линейную аппроксимацию, т.е. выходной сигнал нейрона пропорционален, в некоторой степени, линейной комбинации значений входных сигналов.

### 2.4.2. Искусственный нейрон

Отдельный обрабатываемый элемент искусственной нейронной сети называется искусственным нейроном [8, 9]. Каждый нейрон производит относительно простую работу. На его вход поступает набор сигналов $X = [x_1, x_2, ..., x_n]$, каждый из которых может быть выходом от другого нейрона или другого источника. Каждый вход умножается на соответствующий угловой коэффициент $W = [w_1, w_2, ..., wn]$, который соответствует силе синапса биологического нейрона, и поступает на вход суммирующего блока, где все произведения $w_i x_i$ суммируются. По этой величине определяется общий вход нейрона:

$$h = \sum_i (w_i x_i + \theta_i), \tag{3}$$

где $\theta_i$ – пороговая величина $i$-го нейрона.

Для определения выхода нейрона $O$ (рис. 8) используется функция активации:

$$O = F(h) = F\left[\sum_i (w_i x_i + \theta_i)\right]. \tag{4}$$

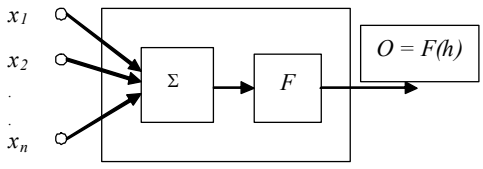



**Рис. 8**

Наиболее типичными функциями активации являются:

– экспоненциальная:

$$F(h) = \frac{2}{1+\exp(-\lambda h)} - 1, \quad \lambda > 0; \tag{5}$$

– функция знака:

$$F(h) = \operatorname{sgn}(h) = \begin{cases} +1, & h > 0; \\ -1, & h < 0. \end{cases} \tag{6}$$

Функции, записанные в таком виде, называются биполярными. Возможно использование униполярных функций:

$$F(h) = \frac{1}{1+\exp(-\lambda h)}, \quad \lambda > 0; \tag{7}$$

$$F(h) = \begin{cases} 1, & h > 0; \\ 0, & h < 0. \end{cases} \tag{8}$$

Следует отметить, что при $\lambda \to \infty$ экспоненциальная функция стремится к функции знака.

### 2.4.3. Многослойные нейронные сети

Для решения практических задач часто используются многослойные нейронные сети. Обычно в таких сетях все нейроны в слое связаны со всеми нейронами в предыдущем слое через однонаправленную связь [8, 9]. При решении задач аппроксимации чаще используется нейронная сеть с одним скрытым слоем (рис. 9).

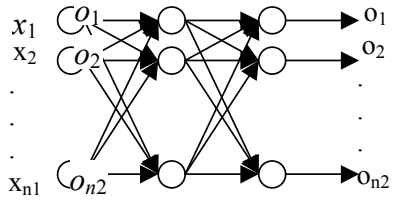



**Рис. 9**

Многослойная нейронная сеть имеет разное количество нейронов в слоях и разные весовые коэффициенты нейронов. Каждый нейрон характеризуется множеством входов и одним выходом. Связь вход-выход для сети, представленной на рис. 9, можно представить в матричной форме:

$$O = F(X) = f[W^2 f[W^1 X]], \tag{9}$$

где $X$ – вектор входных параметров; $O$ – вектор выходных параметров; $W^1$, $W^2$ – матрицы весовых коэффициентов для скрытого и выходного слоя, соответственно; $f$ – функция активации.

При решении задачи аппроксимации обычно используется экспоненциальная функция (уравнения (3), (5)).

## 3. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ В ВИДЕ НЕЧЁТКИХ ВЫСКАЗЫВАНИЙ

Методы построения математических моделей часто основаны хотя и не неточной, но в целом объективной информации об объекте. Однако возможны ситуации, когда при построении моделей решающее значение имеют сведения, полученные от эксперта, обычно качественного характера. Они отражают содержательные особенности изучаемого объекта и формулируются на естественном языке. Описание объекта в таком случае носит нечёткий характер. Например, экспертом предоставлена следующая информация:

- *ЕСЛИ* концентрация кислоты мала и температура раствора не высокая, *ТО* выбирается первый вариант конструкции аппарата.
- *ЕСЛИ* концентрация кислоты мала и температура раствора высокая, *ТО* выбирается второй вариант конструкции аппарата.
- *ЕСЛИ* концентрация кислоты большая и температура раствора высокая, *ТО* выбирается третий вариант конструкции аппарата.

Пусть ставится задача определить конструкцию аппарата, если концентрация кислоты – 25%, а температура раствора – 30 … 40 °С.

Каким образом работать с такой информацией?

Здесь существуют такие понятия, как “высокая”, “невысокая”, “большая”, “мала”. Применительно к понятию “концентрация” значение “мала” может быть интерпретирована, как любой из значений от 2 до 20%. Для понятия “температура” значение “высокая” может интерпретироваться в пределах 40 … 100 °С.

В данном случае, “высокая”, “мала”, “невысокая”, “большая” являются нечёткими переменными. Каждой нечёткой переменной соответствуют определённые значения в некотором интервале. Использование нечётких переменных для построения и анализа правил называют нечёткой логикой, в основе которой лежит понятие нечёткого множества.

### 3.1. НЕЧЁТКИЕ МНОЖЕСТВА И ОПЕРАЦИИ НАД НИМИ

Обозначим через $X = \{ x \}$ – универсальное множество.

Нечётким множеством $\widetilde{A}$ на множестве $X$ называется совокупность пар вида

$$\widetilde{A} = \{\langle \mu_A(x)/x \rangle\}, \tag{10}$$

где $\mu_A(x)$ -> [0,1] – отображение множества $X$ в единичный отрезок [0, 1]. Эта функция называется функцией принадлежности нечёткого множества $\widetilde{A}$.

Значение функции принадлежности $\mu_A(x)$ для конкретного элемента $x \in X$ называется степенью принадлежности.

Можно сказать, что степень принадлежности $\mu_A(x)$ является субъективной мерой того, насколько элемент $x \in X$ соответствует понятию, смысл которого формируется нечётким множеством $\widetilde{A}$.

Носителем нечёткого множества $\widetilde{A}$ называется множество

$$S_A = \{x \mid x \in X \,\&\, \mu_A(x) > 0\}, \tag{11}$$

т.е. носителем нечёткого множества $\widetilde{A}$ является подмножество $S_A$ универсального множества $X$, для элементов которого функция принадлежности $\mu_A(x)$ строго больше нуля.

Нечёткое множество $\widetilde{A}$ называется нормальным, если границы

$$\sup_{x \in X} \mu_A(x) = 1. \tag{12}$$

Мы будем рассматривать только нормальные нечёткие множества, так как если нечёткое множество ненормально, то его всегда можно превратить в нормальное, разделив все значения функции принадлежности на её максимальное значение.

Для нечётких множеств вводятся операции объединения, пересечения и дополнения.

Пусть $\widetilde{A}$ и $\widetilde{N}$ два нечётких множества, заданных на универсальном множестве $X$ с функциями принадлежности $\mu_A(x)$ и $\mu_N(x)$.

Рассмотрим основные операции над нечёткими множествами, которые будут использоваться в дальнейшем.

Объединением нечётких множеств $\widetilde{A}$ и $\widetilde{N}$ называется нечёткое множество

$$\widetilde{A} \cup \widetilde{N} = \{\langle \mu_{A \cup N}(x)/x \rangle\}, \tag{13}$$

где $(\forall x \in X)\ \mu_{A \cup N}(x) = \max\{\mu_A(x), \mu_N(x)\}$.

Пересечением нечётких множеств $\widetilde{A}$ и $\widetilde{N}$ называется нечёткое множество вида

$$\widetilde{A} \cap \widetilde{N} = \{\langle \mu_{A \cap N}(x)/x \rangle\}, \tag{14}$$

где $(\forall x \in X)\ \mu_{A \cap N}(x) = \min\{\mu_A(x), \mu_N(x)\}$.

Очевидно, что при выполнении операции пересечения над нечёткими множествами получается множество не всегда являющееся нормальным.

Дополнением нечёткого множества $\widetilde{A}$ называется нечёткое множество

$$\neg\widetilde{A} = \{\langle \mu_{\neg A}(x)/x \rangle\}, \tag{15}$$

где $(\forall x \in X)\ \mu_{\neg A}(x) = 1 - \mu_A(x)$.

Носителем нечёткого множества $\neg\widetilde{A}$ будет являться множество тех элементов $x \in X$, для которых функция принадлежности $\mu_A(x) \neq 1$.

Обозначим $\widetilde{A}i$ – нечёткое множество, определённое на $Xi$ ($i = 1, \ldots n$).

Декартовым произведением нечётких множеств $\widetilde{A}i$ называется множество:

$$\widetilde{A}1 \cdot \widetilde{A}2 \cdot \ldots \cdot \widetilde{A}n = \{\langle \mu_A(x1, x2, \ldots, xn)/(x1, x2, \ldots, xn) \rangle\}, \tag{16}$$

где

$$xi \in Xi,\ \mu_A(x1, \ldots, xn) = \min\{\mu_{A1}(x1), \ldots, \mu_{An}(xn)\}.$$

Степенью $e$ множества $\widetilde{A}$ называется нечёткое множество

$$\widetilde{A}^e = \left\{ \left\langle \mu_{A^e}(x)/x \right\rangle \right\}. \quad (17)$$

При $e = 2$ получается частный случай операции возведения в степень- операция концентрации, обозначаемая CON.

$$\mathrm{CON}(\widetilde{A}) = \widetilde{A}^2. \quad (18)$$

Операция CON снижает степень нечёткости описания.

При $e = 0{,}5$ получается операция растяжения DIL:

$$\mathrm{DIL}(\widetilde{A}) = \widetilde{A}^{0.5}. \quad (19)$$

Операция DIL повышает степень нечёткости описания.

Множеством $\alpha$-уровня нечёткого множества $\widetilde{A}$ называется множество:

$$S_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\}, \quad (20)$$

где $\alpha \in [0, 1]$.

### 3.2. НЕЧЁТКОЕ ВКЛЮЧЕНИЕ И РАВЕНСТВО МНОЖЕСТВ. НЕЧЁТКОЕ БИНАРНОЕ ОТНОШЕНИЕ

Понятие нечёткого включения и равенств множеств, нечёткого бинарного отношения являются одними из основных понятий процесса принятия решений при наличии экспертной информации.

Пусть $\widetilde{A}1$ и $\widetilde{A}2$ – нечёткие множества.

Степенью включения множества $\widetilde{A}1$ в $\widetilde{A}2$ называется величина

$$\eta(\widetilde{A}1, \widetilde{A}2) = \&_{x \in X} (\mu_{A1}(x) \to \mu_{A2}(x)). \quad (21)$$

Операция $\to$ есть импликация, определяемая как

$$\mu_{A1}(x) \to \mu_{A2}(x) = 1 \& (1 - \mu_{A1}(x) + \mu_{A2}(x)) = \min\{1, 1 - \mu_{A1}(x) + \mu_{A2}(x)\}.$$

Степенью равенства нечётких множеств $\widetilde{A}1$ и $\widetilde{A}2$ называется величина $\rho(\widetilde{A}1, \widetilde{A}2)$, определяемая как логическая сумма эквивалентностей:

$$\rho(\widetilde{A}1, \widetilde{A}2) = \&_{x \in X} (\mu_{A1}(x) \leftrightarrow \mu_{A2}(x)). \quad (22)$$

Здесь $\leftrightarrow$ операция эквивалентности

$$\mu_{A1}(x) \leftrightarrow \mu_{A2}(x) = (\mu_{A1}(x) \to \mu_{A2}(x)) \,\&\, (\mu_{A2}(x) \leftrightarrow \mu_{A1}(x)).$$

Очевидно, что $\rho(\widetilde{A}1, \widetilde{A}2) = \eta(\widetilde{A}1, \widetilde{A}2) \,\&\, \eta(\widetilde{A}2, \widetilde{A}1)$.

Степень включения и степень равенства могут принимать любые значения из отрезка [0, 1].

### 3.3. НЕЧЁТКАЯ И ЛИНГВИСТИЧЕСКАЯ ПЕРЕМЕННЫЕ

Понятие нечёткой и лингвистической переменной используется экспертом при описании сложных объектов и явлений, а также при формализации процессов принятия решений на трудно формализуемых этапах проектирования.

Нечёткой переменной называется тройка объектов вида

$$\langle \alpha, X, C_\alpha \rangle, \quad (23)$$

где $\alpha$ – наименование нечёткой переменной; $X = \{x\}$ – область её определения; $C_\alpha = \{\langle \mu_\alpha(x)/x \rangle\}$ – нечёткое множество на $X$, описывающее ограничения на возможные значения нечёткой переменной $\alpha$ (т.е. её семантику).

Лингвистической переменной называется пятёрка объектов:

$$\langle \beta, T, X, G, M \rangle, \quad (24)$$

где β – наименование лингвистической переменной; *T* – множество её значений (терм-множество), нечёткие переменные, областью определения каждой из которых является множество *X*; *G* – синтаксическая процедура, позволяющая оперировать элементами терм-множества *T*, в частности генерировать новые осмысленные термы (при традиционном подходе процедура *G* определяет новые значения лингвистической переменной, исходя из её базового терм-множества *T* и логических операций *И, ИЛИ, НЕ, ОЧЕНЬ, СЛЕГКА*); *M* – семантическая процедура, позволяющая превратить каждое новое значение лингвистической переменной, образуемое процедурой *G*, в нечёткую переменную путём формирования соответствующего нечёткого множества. Например, семантические процедуры могут иметь вид:

$$M(\tilde{C}1 \ \textit{ИЛИ} \ \tilde{C}2) = \tilde{C}1 \cup \tilde{C}2 \quad \text{– объединение нечётких множеств;}$$

$$M(\tilde{C}1 \ \textit{И} \ \tilde{C}2) = \tilde{C}1 \cap \tilde{C}2 \quad \text{– пересечение нечётких множеств;}$$

$$M(\textit{НЕ} \ \tilde{C}1) = \neg \tilde{C}1 \quad \text{– дополнение нечётких множеств;}$$

$$\mathrm{M}(\textit{ОЧЕНЬ} \ \tilde{C}1) = \mathrm{CON}(\tilde{C}1) \quad \text{– концентрация нечётких множеств;}$$

$$\mathrm{M}(\textit{СЛЕГКА} \ \tilde{C}1) = \mathrm{DIL}(\tilde{C}1) \quad \text{– растяжение нечётких множеств;}$$

где $\tilde{C}1$ И $\tilde{C}2$ – нечёткие множества, соответствующие нечётким переменным α1 и α2 рассматриваемой лингвистической переменной.

Лингвистическая переменная, у которой процедура образования нового значения *G* зависит от множества базовых терм-значений *T* называется синтаксически зависимой лингвистической переменной.

Наряду с рассмотренными выше синтаксически зависимыми лингвистическими переменными существуют переменные, у которых процедура образования новых значений зависит не от множества базовых терм – значений *T*, а от области определения *X* (универсальное множество), т.е. *G* = *G*(*X*). Например, значение лингвистической переменной “толщина изделия” может быть определено как “близкое к 20 мм” или “приблизительно к 75 мм”. Такие лингвистические переменные называются синтаксически независимыми.

Произвольные значения синтаксически независимой лингвистической переменной взаимно однозначно определяется некоторыми значениями *x* области определения *X*. Поэтому произвольное значение (нечёткую переменную α) синтаксически независимой лингвистической переменной задаётся в виде тройки объектов:

$$\alpha = \langle x, X, \mathrm{C}_\alpha \rangle. \qquad (25)$$

Например, нечёткая переменная α, определённая как “приблизительно 75 мм”, запишется в виде: $\alpha = \langle 75, [10, 80], \mathrm{C}_\alpha \rangle$.

### 3.4. МЕТОДЫ ПОСТРОЕНИЯ ТЕРМ-МНОЖЕСТВ

В основании теории из любой области естествознания лежит очень важное понятие элементарного объекта. Для теории нечётких множеств основополагающим понятием является понятие нечётного множества, которое характеризуется функцией принадлежности. Но основной трудностью мешающей интенсивному применению теории нечетного множества при решении практических задач, является то, что функция принадлежности должна быть задана вне самой теории и, следовательно, её адекватность не может быть проверена непосредственно средствами самой теории.

Будем считать, что функция принадлежности $\mu_A(x)$ элемента *x* к нечёткому множеству $\tilde{A}$ – это субъективная мера того, насколько $x \in X$ соответствует понятию, смысл которого формируется нечётким множеством $\tilde{A}$. Под субъективной мерой понимается определяемая опросом экспертов степень соответствия элемента *x* понятию, формализуемому нечётким множеством $\tilde{A}$. При этом степень соответствия – не условная вероятность наблюдения события $\tilde{A}$ при возникновении события *x*, а скорее возможность интерпретации понятия *x* понятием $\tilde{A}$.

Утверждается, что для практических задач достаточно наличия нечёткого языка с фиксированным конечным словарём. Это ограничение не слишком сильное с точки зрения практического использования. Лингвистическая переменная β на практике имеет базовое терм-множество $T = \{T_i\}$, состоящее из 2 – 10 нечётких переменных. Каждый терм описывается нечётким подмножеством множества значений и некоторой базовой переменной. Предполагается, что объединение всех элементов терм-множества покрывает всю область определения лингвистической переменной. Это гарантирует, что любой элемент $u \in U$ описывается некоторым $T_i \in T$. Все термы нумеруются $T_i \in T$ $T = \{T_j\}_1^n$ на множестве действительных чисел $u \in R$, так что имеющий левее расположенный носитель, имеет меньший номер. Правила для выбора терм-множества сведены в табл. 1.

**Таблица 1**

| Обозначение | Критерий | Типичные значения |
|---|---|---|
| $\lvert T\rvert$ | Выбирается в результате компромисса между сложностью и простотой | 2 … 10 |
| $U_{\max}$, $U_{\min}$ | Для измеримых переменных на основании априорных знаний определяют область значений базовых переменных | |
| $\omega_t$ | Должно быть достаточно широким, чтобы избежать чрезмерного влияния погрешностей при переходе от нечёткой переменной к лингвистической переменой | $\omega_t > 5\sigma$п |

Вводятся более строгие условия:

1) $\mu_{T1}(U_{\min}) = 1; \quad \mu_{T1}(U_{\max}) = 1;$

2) $\forall i, i+1 \le n$
$0 < \max\limits_{u \in U} \mu_{Ti \cap Ti+1}(U) < 1;$

3) $\forall i$ существует $u \in U$:
$\mu_{Ti}(U) = 1;$

4) $\forall i$ и $U \quad \sum \mu_{Ti}(U) > 1.$

### 3.4.1. Построение функций принадлежности на счётном множестве точек на основе экспертных оценок

Простейший способ построения функций принадлежности предполагает *опрос нескольких экспертов*.

Пусть имеется $m$ экспертов, часть которых на вопрос о принадлежности элемента $x \in X$ нечёткому множеству $A$ отвечает положительно. Обозначим их число через $n1$. Другая часть экспертов ($n2 = m - n1$) отвечает на вопрос отрицательно. Тогда функция принадлежности принимается $\mu_A(x) = n1/(n1+n2)$.

Необходимо отметить, что данная схема определения функции принадлежности самая простая, но и самая грубая.

Более точно функцию принадлежности можно построить на основе *количественного парного сравнения степеней принадлежности*. Такая схема допускает и одного эксперта.

Результатом опроса эксперта является матрица $M = \|m_{ij}\|$, $i, j = 1, \ldots n$, где $n$ – число точек, в которых сравниваются значения функции принадлежности. Число $m_{ij}$ показывает, во сколько раз, по мнению эксперта, степень принадлежности $\mu_A(x_i)$ больше $\mu_A(x_j)$. При этом количество вопросов, на которые надо ответить эксперту составляет не $n^2$, а лишь $(n^2 - n)/2$, так как по определению $m_{ii} = 1$ и $m_{ij} = 1/m_{ji}$.

При этом эксперт оперирует понятиями, представленными в табл. 2.

**Таблица 2**

| Смысл | $M_{ij}$ |
|---|---|
| $\mu(x_i)$ равна $\mu(x_j)$ | 1 |
| $\mu(x_i)$ немного больше $\mu(x_j)$ | 3 |
| $\mu(x_i)$ больше $\mu(x_j)$ | 5 |
| $\mu(x_i)$ заметно больше $\mu(x_j)$ | 7 |
| $\mu(x_i)$ намного больше $\mu(x_j)$ | 9 |
| Значения, промежуточные по степени между перечисленными | 2, 4, 6, 8 |

Далее, определить значение функции принадлежности $\mu_A$ в точках $x1, x2, \ldots xn$ можно, используя формулу

$$\mu_A(x_i) = \frac{m_{ij}}{\sum\limits_{i=1}^{n} m_{ij}}, \qquad (26)$$

где $j$ – произвольный столбец матрицы $M$.

### 3.4.2. Построение функции принадлежности на непрерывном множестве точек

Выбор вида функции принадлежности и их параметров определяется в большей степени опытом, интуицией и другими субъективными факторами лица, принимающего решение. Именно здесь возникают новые, связанные с неоднозначностью и другого рода нечёткостью, неопределённости, которые носят субъективный характер.

Задание функций степеней принадлежности является центральным вопросом формализации качественной информации. От корректности его выполнения в конечном итоге зависит степень достоверности результата решения задачи с использованием качественной информации.

Задание функции степеней принадлежности в нечётких подмножествах осуществляют несколькими способами:

- В ряде случаев исследователь может задать самостоятельно функцию, исходя из личного опыта. Например, проводя сопоставление результатов измерений, выполненных на различных технологических системах, исследователь наряду с количественными данными оперирует качественными факторами и описывает результаты сопоставления словесно.
- В более сложных и ответственных случаях задание функций принадлежности в нечётких подмножествах выполняется с привлечением группы экспертов с последующей обработкой их оценок. Так при оценке качества изделий, контроль которого осуществляется визуально, возникает задача выбора эталона. В этом случае к выбору и классификации эталонов целесообразно привлечь экспертов.

Рассмотрим процесс задания функции принадлежности. Пусть диапазон изменения величины $x \in X$ определяется отрезком $[x_n, x_k]$. Обычно на этом отрезке выделяют значение $x_0 \in X$, характеризующее понятие “норма”. Кроме этого на отрезке $[x_n, x_k]$ существуют противоположные по смысловому содержанию (с точки зрения нечёткого множества) термины. Иными словами, множество $[x_n, x_k]$ должно обладать симметрией относительно элемента $x_0$. Требуется, кроме того, выполнение следующих асимптотических свойств:

$$\lim_{x \to x_n} \mu(x) = a; \quad \lim_{x \to x_k} \mu(x) = b, \tag{27}$$

где $a$, $b$ – постоянные для данного термина. Например, на рис. 10 представлена функция принадлежности $\mu(x)$, формализующая понятие “высокий”.

На оси абсцисс отмечен опорный элемент $x_0$, соответствующий понятию “норма”. Обычно полагают $\mu(x) = 0{,}5$. Выбор $x_0$ подвержен субъективизму каждого исследователя и определяется уровнем знаний о конкретной системе. Выполнение условия $\lim_{x \to x_n} \mu(x) = 0$ отражает тот факт, что элементы $x < x_0$ в меньшей степени, чем $x_0$ относятся к понятию “высокий”.

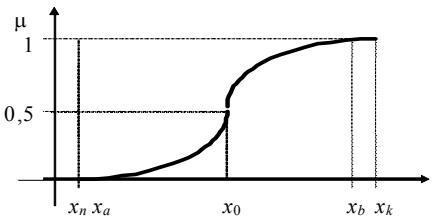



**Рис. 10**

Здесь важно заметить, что функции принадлежности должны быть сформированы с точностью до качественных различий первичных терминов (например, понятие “высокий” и “сверхвысокий”). Кроме того, заметим, что формируемое нечёткое множество предполагается нормальным, т.е. $\sup_{x \in X} \mu(x) = 1$.

Исходя из асимптотических свойств функции $\mu(x)$, исследователем могут быть установлены интервалы $[x_н, x_a]$ и $[x_b, x_k]$, на которых функция задается путём чёткой классификации:

$$\mu(x) = \begin{cases} 0, & x \in [x_н, x_a]; \\ 1, & x \in [x_b, x_к]. \end{cases} \tag{28}$$

Наиболее сложным является задание $\mu(x)$ при $x \in [x_a, x_b]$. Предполагается, что $\mu(x)$ является монотонной функцией.

Отмечается, что человек с достаточно хорошей точностью может запомнить в памяти и анализировать от пяти до семи признаков. Поэтому необходимо минимизировать психологическую нагрузку эксперта, который выполняет формализацию первичных терминов.

Например, проиллюстрируем способ задания функции принадлежности для формализации понятий “низкий”, “средний”, “высокий”.

Процедура задания функций принадлежности, которой должны придерживаться эксперты, заключается в следующем (рис. 11):

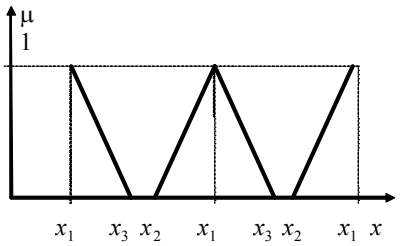



**Рис. 11**

1. Выделение точки $x_1 \in X$, которая с точки зрения эксперта точно соответствует нечёткому подмножеству. В этом случае $\mu(x) = 1$.

2. Нахождение точек слева и справа от $x_1$, которые с точки зрения эксперта не могут быть отнесены к рассматриваемому термину. Для них $\mu(x_2) = \mu(x_3) = 0$.

3. Графическое построение функций по выбранным точкам с использованием линейной аппроксимации.

4. Выделение подмножества $X_1 \in X$, на котором определена формализация термина, $X_1 \in [x_2, x_3]$. Следует отметить, что в ряде случаев точки $x_2$, $x_3$ могут быть отнесены в бесконечность.

Такой способ задания функций принадлежности обладает следующими особенностями:

- простотой выполнения экспертной оценки с точки зрения психологической нагрузки;
- компактность задания функций;
- простотой математических средств при переходе от одного термина к другому.

В ряде случаев функцию степеней принадлежности $\mu(x)$ нечёткого подмножества некоторого множества задают в виде функциональной зависимости, например экспоненциальной, полинома и т.п. с одним или несколькими неизвестными переменными.

Вообще, задание функций принадлежности требует знаний особенностей объекта исследований, принятой в данной отрасли терминологии и использование, по возможности, простых функциональных зависимостей. Для идентификации неизвестных параметров в функции принадлежности нечёткого подмножества могут быть использованы метод наименьших квадратов, симплекс-метод и другие.

***Пример.***

Параметр “расход сырья на установку” ($G$) определён на отрезке [70 – 100] и имеет три нечётких значения:

- малый (70 – 80) с функцией принадлежности
$$\mu_1(x) = \exp(-\frac{1}{5}\ln\frac{1}{2}|x-75|)$$
- средний (80 – 90) с функцией принадлежности
$$\mu_2(\text{›}) = \exp(-\frac{1}{5}\ln\frac{1}{2}|x-85|)$$
- большой (90 – 100) с функцией принадлежности
$$\mu_3(x) = \exp(-0{,}1\ln\frac{1}{2}|x-100|)$$

Здесь принятые термины описываются зависимостью вида

$$\mu(x) = \exp(-Q|x - a_p|),$$

где $Q$ – постоянная величина, которая находится при идентификации функции принадлежности: $a_p = (a_r + a_{r+1})/2$.

## 3.5. НЕЧЁТКИЕ ВЫСКАЗЫВАНИЯ. ПРАВИЛА ПРЕОБРАЗОВАНИЯ НЕЧЁТКИХ ВЫСКАЗЫВАНИЙ

Нечёткими высказываниями называются высказывания следующего вида:

1. Высказывания вида ⟨β есть α⟩, где β – наименование лингвистической переменной, отражающей некоторый объект или параметр реальной действительности; α – наименование нечёткой переменной, которая является нечёткой оценкой β.

Например: ⟨давление большое⟩

⟨толщина равна 14⟩ (в этом случае значение α = 14 является чёткой оценкой лингвистической переменной β (толщина)).

2. Высказывания вида ⟨β есть $m$α ⟩;

⟨β есть Qα⟩;

⟨Qβ есть $m$α⟩;

⟨$m$β есть Qα⟩.

где $m$ – модификатор (ему соответствуют такие слова, как *ОЧЕНЬ*, *СРЕДНИЙ*, *БОЛЕЕ ИЛИ МЕНЕЕ*, *НЕЗНАЧИТЕЛЬНЫЙ* …); Q – квантификатор (ему соответствуют слова типа: *БОЛЬШИНСТВО*, *НЕСКОЛЬКО*, *МНОГО*, *НЕМНОГО*, *ОЧЕНЬ МНОГО* и др.)

Например: ⟨давление очень большое⟩

⟨большинство значений параметра очень мало⟩

3. Высказывания, образованные из высказываний 1-го и 2-го видов и союзов: *И, ИЛИ, ЕСЛИ ... ТО, ЕСЛИ ... ТО ... ИНАЧЕ ...*

Например: *ЕСЛИ* давление большое *ТО* толщина не мала.

Предположим, имеются некоторые нечёткие высказывания $\widetilde{C}$ и $\widetilde{D}$ относительно одной ситуации *A*. Эти высказывания имеют вид

$$\langle \beta \text{ есть } \alpha_C \rangle;$$
$$\langle \beta \text{ есть } \alpha_D \rangle,$$

где $\alpha_C$ и $\alpha_D$ – нечёткие переменные, определённые на универсальном множестве $X = \{x\}$.

Истинностью высказывания $\widetilde{D}$ относительно $\widetilde{C}$ называется значение функции $T(\widetilde{D}/\widetilde{C})$, определяемое степенью соответствия высказываний $\widetilde{D}$ и $\widetilde{C}$:

$$T(\widetilde{D}/\widetilde{C}) = \{\mu_T(\tau)/\tau\}, \qquad (29)$$

где $\tau = \mu_D(x) \ \forall \ x \in \underset{x \in X'}{X}; \ \mu_T(\tau) = \max \mu_C(x); \ X' = \{x \in X | \mu_D(x) = \tau\}$,

т.е. функция принадлежности значение истинности $\mu_T(\tau)$ для любого $0 \le \tau \le 1$ определяется как максимальное из $\mu_C(x)$ (функция принадлежности нечёткой переменной $\alpha_C$) для тех *x* у которых $\mu_D(x) = \tau$ ($\mu_D(x)$ – функция принадлежности нечёткой переменной $\alpha_D$).

Мы рассмотрели нахождение истинности высказываний вида: $\langle \beta$ есть $\alpha \rangle$. Чтобы определить истинность более сложных высказываний, необходимо привести эти высказывания к виду $\langle \beta$ есть $\alpha \rangle$. Такое приведение осуществляется по определённым правилам.

1. Правило преобразования конъюнктивной формы:

$$\langle \beta_x \text{есть } \alpha_{x1} \ И \ \beta_y \text{есть } \alpha_{y1} \rangle \rightarrow \langle (\beta_x, \beta_y) \text{ есть } \vec{\alpha}_{x1} \cap \vec{\alpha}_{y1} \rangle . \qquad (30)$$

Здесь $\vec{\alpha}_{x1} \cap \vec{\alpha}_{y1}$ – это значение лингвистической переменной $(\beta_x, \beta_y)$ с нечётким множеством $C_\cap = \vec{C}_{x1} \cap \vec{C}_{y1}$, где $\vec{C}_{x1}, \vec{C}_{y1}$ – цилиндрические продолжения нечётких множеств $C_x$ и $C_y$: $\vec{C}_{x1} = \{\langle \vec{\mu}_{x1}(x,y)/(x,y) \rangle\}$; $\vec{C}_{y1} = \{\langle \vec{\mu}_{y1}(x,y)/(x,y) \rangle\}$.

Причём $(x,y) \in XY \ (\forall x \in X), (\forall y \in Y)$, $\vec{\mu}_{x1}(x,y) = \mu_{x1}(x)$, $\vec{\mu}_{y1}(x,y) = \mu_{y1}(y)$.

2. Правило преобразования дизъюнктивной формы:

$$\langle \beta_x \text{есть } \alpha_{x1} \ ИЛИ \ \beta_y \text{есть } \alpha_{y1} \rangle \rightarrow \langle (\beta_x, \beta_y) \text{ есть } \vec{\alpha}_{x1} \cup \vec{\alpha}_{y1} \rangle . \qquad (31)$$

Здесь $\vec{\alpha}_{x1} \cup \vec{\alpha}_{y1}$ – это значение лингвистической переменной $(\beta_x, \beta_y)$ с нечётким множеством $C_\cup = \vec{C}_{x1} \cup \vec{C}_{н1}$ (объединение цилиндрических продолжений).

3. Правило преобразования высказываний импликативной формы:

$$\langle ЕСЛИ \ \beta_x \text{есть } \alpha_{x1} \ ТО \ \beta_y \text{есть } \alpha_{y1} \rangle \rightarrow \langle (\beta_x, \beta_y) \text{ есть } \vec{\alpha}_{x1} \Diamond \vec{\alpha}_{y1} \rangle . \qquad (32)$$

Знак $\Diamond$ означает пороговую сумму, определяемую как

$$(\forall x \in X)(\forall y \in Y) \ \mu_\Diamond(x, y) = 1 \wedge (1 - \mu_{\vec{\alpha}_{X1}}(x, y) + \mu_{\vec{\alpha}_{Y1}}(x, y)),$$

где $\mu_{\vec{\alpha}_{X1}}(x, y)$, $\mu_{\vec{\alpha}_{Y1}}(x, y)$ – функции принадлежности, соответствующие нечётким множествам $\vec{C}_{x1}$, $\vec{C}_{y1}$.

### 3.6. ПРЕДСТАВЛЕНИЕ ЭКСПЕРТНОЙ ИНФОРМАЦИИ В ВИДЕ СИСТЕМ НЕЧЁТКИХ ВЫСКАЗЫВАНИЙ

Обозначим через $X$, $Y$, $Z$ … – множество значений входных параметров процесса проектирования, существенно влияющих на выбор выходного параметра $V$. Введём лингвистические переменные: $\langle \beta_x, T_x, X_x, G_x, M_x \rangle$, $\langle \beta_y, T_y, X_y, G_y, M_y \rangle$, $\langle \beta_z, T_z, X_z, G_z, M_z \rangle$, и $\langle \beta_v, T_v, X_v, G_v, M_v \rangle$, определённые на множествах $X$, $Y$, $Z$ … и $V$.

Системы логических высказываний, отражающие опыт эксперта в типовых ситуациях, представим в виде

$$\widetilde{L}^{(1)} = \begin{cases} \widetilde{L}_1^{(1)} : \langle ЕСЛИ\ \widetilde{E}_{11} ИЛИ \ldots ИЛИ\ \widetilde{E}_{1n_1}\ ТО\ \beta_v \text{ есть } \mu\alpha_{v1} \rangle \\ \ldots \\ \widetilde{L}_m^{(1)} : \langle ЕСЛИ\ \widetilde{E}_{m1} ИЛИ \ldots ИЛИ\ \widetilde{E}_{mn}\ ТО \beta_v \text{ есть } \mu\alpha_{vm} \rangle \end{cases} \quad (33)$$

или в виде

$$\widetilde{L}^{(1)} = \begin{cases} \widetilde{L}_1^{(1)} : \langle ЕСЛИ\ \beta_v \text{ есть } \mu\alpha_{v1}\ ТО\ \widetilde{E}_{11} ИЛИ \ldots ИЛИ\ \widetilde{E}_{1n_1} \rangle \\ \ldots \\ \widetilde{L}_m^{(1)} : \langle ЕСЛИ\ \beta_v \text{ есть } \mu\alpha_{vm} ТО\ mo \widetilde{E}_{m1} ИЛИ \ldots ИЛИ\ \widetilde{E}_{mn_m} \rangle \end{cases} \quad (34)$$

где $m$ – число базовых значений лингвистической переменной $\beta_v$; $E_{ji}$ $(i = 1..n, j = 1..m)$ – высказывания вида:

$$\langle \beta_x \text{ есть } \mu\alpha_{x_{ji}}\ И\ \beta_y \text{ есть } \mu\alpha_{y_{ji}}\ И\ \beta_z \text{ есть } \mu\alpha_{z_{ji}} \ldots \rangle .$$

Высказывание $E_{ij}$ представляет собой $i$-ю входную нечёткую ситуацию, которая может иметь место, если лингвистическая переменная $\beta_v$ примет значение $\alpha_{vj}$. Значения $\alpha_{Xji}$, $\alpha_{Yji}$, $\alpha_{Zji}$ , … $\alpha_{Vji}$ – нечёткие переменные с функциями принадлежности соответственно: $\mu_{Xji}(x)$, $\mu_{Yji}(y)$, $\mu_{Zji}(z)$, … $\mu_{Vji}(v)$ $(x \in X, y \in Y, z \in Z, v \in V)$.

Обе приведённые системы нечётких высказываний, так же как и ранее рассмотренные чёткие системы, отражают два разных случая взаимосвязи между значениями входных и выходных параметров процесса проектирования. В первом случае в зависимости от базовых значений входных лингвистических переменных делается вывод о базовом значении выходной лингвистической переменной. Во втором случае в зависимости от возможных значений выходного параметра делается предположение о возможных значениях входных параметров.

Представим системы в более компактном виде.

Используя правило преобразования конъюнктивной формы, высказывание $Eji$ можно записать в более компактном виде:

$$E_{ji} : \langle \beta_W \text{ есть } \alpha_{Eji} \rangle,$$

где $\beta_W$ – лингвистическая переменная, определённая на множестве $W = XYZ$ … и принимающая базовые значения $\alpha_{Eji}$ с функцией принадлежности: $\mu_{Eji}(w) = \min\{\mu_{Xji}(x), \mu_{Yji}(y), \mu_{Zji}(z), \ldots\}$.

Далее согласно правилу преобразования дизъюнктивной формы высказывания $L_j^{(1)}$ и $L_j^{(2)}$ могут быть представлены в виде

$$L_j^{(1)} = \langle ЕСЛИ\ \beta_W \text{ есть } \alpha_{Wj}\ \ ТО\ \beta_V \text{ есть } \alpha_{Vi} \rangle,$$

$$L_j^{(2)} = \langle \text{ЕСЛИ } \beta_V \text{ есть } \alpha_{Vj}\ \ ТО\ \beta_W \text{ есть } \alpha_{Wi} \rangle,$$

Здесь $\alpha_{Wi}$ – значение лингвистической переменной $\beta_W$ с функцией принадлежности: $\mu_{Wj}(w) = \max \mu_{Eji}(w)$.

Обозначим через $A_j$ и $B_j$ высказывания $\langle \beta_W \text{ есть } \alpha_{Wj} \rangle$ и $\langle \beta_V \text{ есть } \alpha_{Vj} \rangle$.

Тогда системы нечётких высказываний запишутся в виде

$$\widetilde{L}^{(1)} = \begin{cases} \widetilde{L}_1^{(1)} : \langle ЕСЛИ\ \widetilde{A}_1\ ТО\ \widetilde{B}_1 \rangle \\ \widetilde{L}_2^{(1)} : \langle ЕСЛИ\ \widetilde{A}_2\ ТО\ \widetilde{B}_2 \rangle \\ \ldots \\ \widetilde{L}_m^{(1)} : \langle ЕСЛИ\ \widetilde{A}_m\ ТО\ \widetilde{B}_m \rangle \end{cases} \quad (35)$$

Эту систему назовем нечёткой системой первого типа.

$$\widetilde{L}^{(2)} = \begin{cases} \widetilde{L}_1^{(2)} : \langle ЕСЛИ\ \widetilde{B}_1\ ТО\ \widetilde{A}_1 \rangle \\ \widetilde{L}_2^{(2)} : \langle ЕСЛИ\ \widetilde{B}_2\ ТО\ \widetilde{A}_2 \rangle \\ \cdots \\ \widetilde{L}_m^{(2)} : \langle ЕСЛИ\ \widetilde{B}_m\ ТО\ \widetilde{A}_m \rangle \end{cases} \quad (36)$$

Эту систему назовём нечёткой системой второго типа.

Системы нечётких экспертных высказываний представимы в виде соответствий:

1. Система высказываний первого типа может быть задана соответствием:

$$\Gamma^{(1)} = (T_V, T_W, F_1), \quad (37)$$

где $T_W$ – область отправления (множество входных ситуаций); $T_V$ – область прибытия (множество выходных ситуаций); $F_1 \subseteq T_W T_V$ – график соответствия.

2. Система высказываний второго типа задаётся соответствием:

$$\Gamma^{(2)} = (T_V, T_W, F_2), \quad (38)$$

где $F_2 \subseteq T_V T_W$.

Графики соответствия представляются в виде графа, в левой части которого вершинам соответствуют области отправления, а в правой – области прибытия.

Пример приведён на рис. 12.

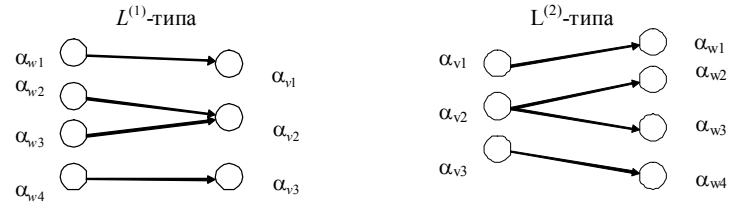



**Рис. 12**

Для анализа нечёткой информации вводится ряд понятий:

1. Система нечётких высказываний называется лингвистически не избыточной, если граф соответствия не содержит повторяющихся пар вершин.

2. Система нечётких высказываний называется лингвистически полной, если граф системы первого типа в правой части, а системы второго типа в левой части не содержит изолированных вершин. В противном случае система является лингвистически вырожденной (пример на рис. 13):

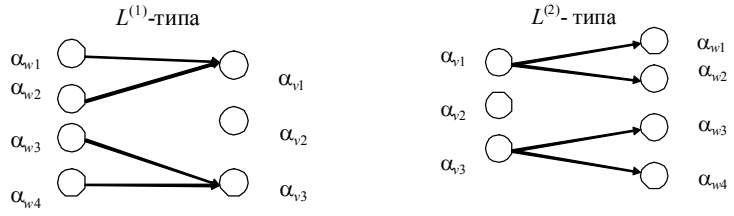



**Рис. 13**

3. Система нечётких высказываний называется лингвистически непротиворечивой, если в графе соответствия системы первого типа из каждой вершины левой части выходит не более одного ребра, а для системы второго типа в каждую вершину правой части входит не более одного ребра. Примеры противоречивых систем приведены на рис. 14.

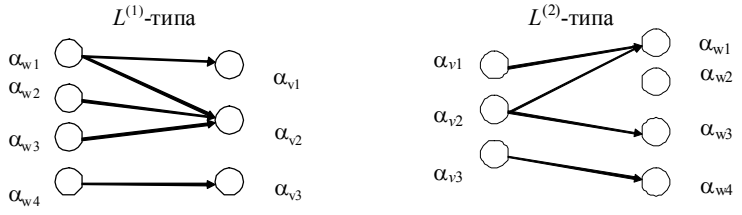



**Рис. 14**

Рассмотренные понятия позволяют качественно оценить экспертную информацию. Естественным требованием к ней является то, что система нечётких высказываний должна быть лингвистически полной невырожденной и непротиворечивой.

# 4. ПРИНЯТИЕ РЕШЕНИЙ В СИСТЕМАХ, ОСНОВАННЫХ НА ПРАВИЛАХ

## 4.1. ВЫБОР РЕШЕНИЙ НА ОСНОВЕ ЧЁТКОЙ ЭКСПЕРТНОЙ ИНФОРМАЦИИ

Использование моделей и алгоритмов принятия решений на основе экспертной информации связано с решением задачи представления данной информации в виде, пригодном для использования.

При выборе решений в чётких условиях экспертная информация представила в виде системы условных высказываний, устанавливающих взаимосвязь между чёткими значениями входных и выходных параметров процесса принятия решения.

Если в зависимости от возможных чётких значений входных параметров делается вывод о значениях выходного параметра, то такая система называется системой $L^{(1)}$-типа. Данная система представляется в виде:

$$L^{(1)} = \begin{cases} L_1^{(1)} : \langle ЕСЛИ\ A_1\ ТО\ B_1 \rangle \\ L_2^{(1)} : \langle ЕСЛИ\ A_2\ ТО\ B_2 \rangle \\ \dots \\ L_m^{(1)} : \langle ЕСЛИ\ A_m\ ТО\ B_m \rangle \end{cases} \tag{39}$$

где $m$ – число экспертных высказываний; $Aj$ – чёткое значение входного параметра; $Bj$ – чёткое значение выходного параметра или некоторое конкретное действие процесса проектирования.

В случаях, когда в зависимости от возможных значений выходной ситуации ($Bj$) экспертом делается предположение о возможной входной ситуации ($Aj$), система экспертных высказываний называется системой $L^{(2)}$-типа и представляется в виде:

$$L^{(2)} = \begin{cases} L_1^{(2)} : \langle ЕСЛИ\ B_1\ ТО\ A_1 \rangle \\ L_2^{(2)} : \langle ЕСЛИ\ B_2\ ТО\ A_2 \rangle \\ \dots \\ L_m^{(2)} : \langle ЕСЛИ\ B_m\ ТО\ A_m \rangle \end{cases} \tag{40}$$

Рассмотрим теперь механизм выбора решений в чётких условиях.

(1) При задании экспертной информации системой высказываний $L^{(1)}$-типа выбор решения основывается на правиле modus ponens.

Пусть $A$ и $B$ – произвольные чёткие высказывания. Правило $\langle ЕСЛИ\ Aj\ ТО\ Bj \rangle$ – правило из системы $L^{(1)}$. Обозначим через $T(A/A_j)$ – истинность высказывания $A$ относительно $A_j$; $T(B/B_j)$ – истинность высказывания $B$ относительно $B_j$:

$$T(A/A_j) = \begin{cases} 1, \ при\ A = A_j; \\ 0, \ в\ других\ случаях. \end{cases} \tag{41}$$

$$T(B/B_j) = \begin{cases} 1, при\ B = B_j; \\ 0, в\ других\ случаях. \end{cases} \tag{42}$$

(В данном случае значения истинностей являются частным случаем значений истинности для нечётких высказываний).

Согласно правилу modus ponens из высказываний $\langle ЕСЛИ\ Aj\ ТО\ Bj \rangle$ $И$ $\langle A \rangle$ выводимо высказывание $\langle B \rangle$.

Формально правило modus ponens записывается в виде

$$\frac{\langle ЕСЛИ\ Aj\ \ ТО\ \ Bj \rangle \qquad \underline{\langle A - \text{истинно} \rangle}}{\langle B - \text{истинно} \rangle}. \tag{43}$$

Истинностью правила modus ponens для схемы вывода (43) называется величина

$$T(L_j^{(1)}, A, B) = \begin{cases} 1, \ если\ A = A_j\ и\ B = B_j \\ 0, \ в\ других\ случаях \end{cases} \tag{44}$$

(2) При задании экспертной информации системой высказываний $L^{(2)}$-типа выбор решения основывается на индуктивной схеме вывода. Согласно ей из высказываний $\langle ЕСЛИ\ Aj\ ТО\ Bj \rangle$ $И$ $\langle A \rangle$ следует правдоподобность высказывания $\langle B \rangle$. Формально такая схема запишется в виде

$$\frac{\langle ЕСЛИ\ Aj\ ТО\ Bj \rangle; \qquad \underline{\langle A - \text{истинно} \rangle;}}{\langle B - \text{более правдоподобно} \rangle}. \tag{45}$$

Понятие истинности данной схемы вывода выводится аналогично схеме modus ponens (44).

Тогда при выборе решений в чётких условиях в случае, когда информация задана полной непротиворечивой системой первого типа, правило modus ponens соответствует выбору такого выходного высказывания, при котором истинность (44) схемы вывода (43) достигает своего наибольшего значения, т.е. единицы. Для принятия решения используется алгоритм прямой цепочки рассуждений.

Аналогично, при задании экспертной информации системой чётких высказываний второго типа индуктивная схема вывода соответствует выбору такого высказывания *B*, при котором истинность (44) схемы вывода (45) также достигает своего максимума (=1). В данном случае для принятия решения используется алгоритм обратной цепочки рассуждений.

### 4.1.1. Алгоритм прямой цепочки рассуждения

Прямой алгоритм в ходе рассуждения движется от исходных данных через систему правил к результату. При этом из исходных данных поочередно выбираются все факты, последовательно сравниваются с частью *ЕСЛИ* всех правил базы знаний. Прямая цепочка рассуждений позволяет пользователю экспертной системы получить всю информацию, сходящую из входных данных.

Базу знаний экспертной системы можно условно разбить на две части: базу правил и базу данных. База правил включает в себя статические знания о предметной области и формируется на основе экспертной информации. Процесс построения базы правил является наиболее трудоёмким и трудно формализуемым этапом разработки экспертной системы. База данных – знания в виде фактов, которые в свою очередь подразделяются на постоянные и временные факты. Временные факты описывают определённую ситуацию, в процессе работы экспертной системы и зависят от входных данных. Постоянные факты определяют логические зависимости - между объектами, предметной области и не меняются в ходе эксплуатации экспертной системы.

Работу алгоритма рассмотрим на основе знаний, представленных в виде фактов и правил, являющихся фрагментом базы знаний системы проектирования оборудования автоматизированной линии гальванопокрытий.

База знаний содержит три факта:

1) кислота = агрессивная жидкость;
2) азотная кислота = кислота;
3) сернистая кислота = кислота.

и 9 правил:

1. *ЕСЛИ* среда = кислота *И* концентрация > 70% *ТО* среда концентрированная кислота.
2. *ЕСЛИ* среда = кислота *И* концентрация < 70% *ТО* среда = разбавленная кислота.
3. *ЕСЛИ* среда = концентрированная кислота *ТО* материал ванны = хромоникелевая сталь.
4. *ЕСЛИ* среда = разбавленная кислота *ТО* материал ванны = углеродистая сталь.
5. *ЕСЛИ* среда = агрессивная жидкость *ТО* футеровка = есть.
6. *ЕСЛИ* футеровка = есть *И* среда = сернистая кислота *ТО* материал футеровки – свинец.
7. *ЕСЛИ* футеровка = есть *И* среда = азотная кислота *ТО* материал футеровки = винипласт.
8. *ЕСЛИ* температура = меньше 100 °С *ТО* тип обогрева = пароводяная рубашка.
9. *ЕСЛИ* температура = больше 100 °С *ТО* тип обогрева = электронагреватель.

После формирования базы знаний определяются вспомогательные структуры данных, необходимые для работы алгоритма рассуждений. Для удобства работы представим эти структуры в виде табл. 3 – 8.

В эту таблицу 3 заносятся имена переменных, присутствующих в условной части правил.

Способ организации: для каждого правила базы знаний в списке резервируется $N$ позиций, где $N = \max N_i$, $i = 1 \ldots k$;

$N_i$ – количество переменных в условной части $i$-го правила;

$k$ – количество правил.

**3. Список переменных условия**

| Переменная | Переменная |
|---|---|
| 1) среда | 11) футеровка |
| 2) концентрация | 12) среда |
| 3) среда | 13) футеровка |
| 4) концентрация | 14) среда |
| 5) среда | 15) температура |
| 6) | 16) |
| 7) среда | 17) температура |
| 8) | 18) |
| 9) среда | |
| 10) | |

В графу “Имя” заносятся имена переменных условия, причём каждая переменная заносится только один раз. Графа “Признак инициализации” заполняется до начала работы экспертной системы значением “не определена”. Графа “Значение” в начале работы пуста.

### 4. Список переменных условия расширенный

| Имя | Признак инициализации | Значение |
|---|---|---|
| среда | не определена | |
| концентрация | не определена | |
| футеровка | не определена | |
| температура | не определена | |

Указатель (табл. 5) предназначен для отслеживания текущего события в цепочке рассуждений и состоит из двух частей:

а) номера правила, с которым и настоящий момент работает система;

б) номера условия соответствующего правила: данная графа необходима в связи с тем, что правило в общем случае может содержать более одного условия.

### 5. Указатель переменных условия

| Номер правила | Номер условия |
|---|---|

### 6. Очередь переменных логического вывода

| Переменная |
|---|

Таблица 6 представляет собой список переменных условия и определяет последовательность их сопоставления с условными частями правил.

Определив систему фактов и правил и сформировав все необходимые вспомогательные таблицы, получим состав базы знаний, обеспечивающий работу алгоритма прямой цепочки рассуждений. Теперь система готова принять от пользователя входные данные. Полученная информация интерпретируется системой и заносится в соответствующие таблицы.

Пусть пользователю требуется получить консультацию о конструктивных особенностях проектируемой ванны при условии, что в ней будет находиться сернистая кислота с концентрацией до 75% при температуре от 80 до 90 °С.

Входная информация представляется следующим образом:

среда = сернистая кислота;

концентрация = более 70%;

температура = меньше 100 °С.

На основе списка переменных условия и системы правил можно определить допустимые значения каждой переменной условия, которые используются при интерпретации исходной информации во внутреннее представление системы. Эта информация заносится в очередь переменных логического вывода и в расширенный список переменных условия (табл. 4).

Далее выбирается переменная, стоящая в очереди первой (“Среда”) и отыскивается правило, содержащее в условной части эту переменную. Поиск осуществляется по списку переменных условия.

Первым правилом, содержащим переменную “Среда”, является правило 1. При этом указатель переменных условия устанавливается как первое условие первого правила (табл. 7).

### 7. Указатель переменных условия при анализе первого правила

| 1 | 1 |
|---|---|

Из списка переменных условия следуют такие, что для правила 1 в условной части содержится две переменных (“Среда” и “Концентрация”). Обратившись к расширенному списку переменных условия, определяем, что обе эти переменные проинициализированы. При этом имеется логическое несоответствие в значениях переменной “Среда” в расширенном списке и в условной части правила. Такое несоответствие возможно по двум причинам:

1) по причине смыслового различия (сернистая кислота – азотная кислота, концентрированная кислота – разбавленная кислота);

2) по причине того, что одно из значений является более общим и включает в себя другое (агрессивная жидкость = концентрированная кислота, кислота = азотная кислота).

Выявление причин несоответствия значений выполняется на основе анализа группы фактов базы знаний. В рассматриваемом примере в базе знаний имеется факт: сернистая кислота = кислота, откуда следует, что понятие “сернистая кислота” является частным случаем понятия “кислота”. В результате этого первое условие первого правила является выполненным. Указатель переменных условия устанавливается на второе условие первого правила. Значения переменной “концентрация” в расширенном списке переменных условия и в условной части правила совпадают, следовательно, все правило считается выполненным. Согласно части “*ТО*” этого правила, переменной “среда” присваивается значение “концентрированная кислота”. Изменение значения переменной отражается в расширенном списке переменных условия (табл. 8).

**8. Изменения в расширенном списке**

| | | |
|---|---|---|
| среда | определена | концентрированная кислота |

В части *ТО* правила содержится переменная условия. Она должна быть помещена в очередь переменных логического вывода. В данном случае этого не происходит, так как переменная “среда” уже есть в очереди. Кроме того, в базу знаний добавляется новый факт: сернистая кислота = концентрированная кислота. Этот факт является временным. Он необходим только для работы: при конкретных исходных данных и удаляется из базы знаний при завершении работы системы. Временные факты формируют логическую цепочку, на основе которой система в дальнейшем выясняет причины возникшего несоответствия значений переменных.

Далее система проверяет, имеется ли ещё какое-либо правило, в условной части которого содержится переменная “среда”. Если такого правила не существует, то переменная удаляется из очереди. В противном случае, указатель устанавливается на первое условие найденного правила (в данном случае правило 2). Анализ найденного правила выполняется системой аналогично схеме, рассмотренной выше. Такой анализ показывает, что правило 2 не выполняется. Рассмотрим теперь правило 3. Око считается выполненным. Согласно части *ТО* этого правила переменной “материал ванны” присваивается значение “хромоникелевая сталь”. Эта переменная не содержится в списке переменных условия, следовательно она является выводом.

Переменные вывода помещаются в список переменных вывода, который при завершении работы система определяет результат её работы.

После анализа правила 3 состояние структур данных в базе знаний следующее:

- очередь не изменилась;
- расширенный список переменных условия не изменился;
- список переменных вывода содержит одну строку;
- число фактов в базе знаний – 4;
- указатель переменных условия (табл. 9).

**9. Указатель переменных условия** (правило 3)

| | |
|---|---|
| 3 | 1 |

Дальнейший анализ показывает, что правило 4 не выполняется из-за несоответствия значения переменной “среда” (“разбавленная кислота” – “концентрированная кислота”). Выполняем рассмотрение следующего правила, содержащего переменную условия “среда” (правило 5). Это правило выполняется. В структуре данных возникли изменения:

- в очередь переменных логического вывода добавляется “футеровка”;
- изменяется значение переменной в расширенном списке переменных условия (табл. 10);
- указатель переменных условия.

**10. Изменения в расширенном списке**

| футеровка | Определена | Есть |
|---|---|---|
| | | |

- в список переменных вывода добавляется переменная “материал футеровки” со значением “свинец”;
- очередь не изменилась;
- списки переменных условия не изменились;
- указатель переменных условия (табл. 11).

**11. Указатель переменных условия** (правило 6)

| | |
|---|---|
| 6 | 2 |

Правило 7 не выполняется.

После этого система не находит больше правила, содержащего переменную “среда”. Эта переменная исключается из очереди. Первой в очереди становится переменная “концентрация”. Для неё должны быть выполнены все предыдущие шаги. В нашем примере правила, содержащие в условной части переменную “концентрация”, уже были проверены (правила 1 и 2), действия, предусмотренные частью *ТО* были выполнены, поэтому переменная условия “концентрация” исключается из очереди.

Теперь первая в очереди переменная “температура”. Анализ правил, содержащих переменную “температура” приводит к следующему состоянию структур данных:

– очередь пустая (после рассмотрения переменной “температура” в очереди остаётся переменная условия “футеровка”, которая исключается аналогично переменной “концентрация”);

- списки переменных условия не изменились;
- указатель (табл. 12);
- к списку переменных вывода добавили “тип обогрева”, равную “пароводяная рубашка”.

**12. Указатель переменных условия** (правило 9)

| 9 | 1 |
|---|---|

Так как очередь пуста, процесс рассуждения заканчивается. Результатом работы алгоритма являются следующие решения:

- материал ванны – хромоникелевая сталь;
- ванна имеет защитную футеровку;
- материал футеровки – свинец;
- тип обогрева – пароводяная рубашка.

В простейшем случае результатом работы является список логических выводов, содержащий переменные вывода и их значения. Однако существует ряд случаев, когда в качестве выходной информации используются и переменные условия. Это позволяет обеспечить более логичное и наглядное представление исходных данных системы в виде, удобном пользователю. Так в рассмотренном выше примере переменная вывода “материал футеровки” входит в список выводов только при определённых значениях переменной условия “футеровка”. Поэтому переменная условия “футеровка” также может быть рассмотрена как результат работы системы.

Предложенный алгоритм не является самым универсальным и допускает ряд модификаций. В частности, допускается различное толкование недостатка входных данных. В ходе работы алгоритма может возникнуть ситуация, когда при обращении к некоторому правилу выяснится, что переменная условия в нём не была определена.

В данной ситуации система может либо запросить недостающие данные у пользователя, либо выдать сообщение о недостающей информации и прекратить рассмотрение данного правила. Каждый из этих подходов имеет свои преимущества и недостатки. В первом случае достигается более полный список выводов, но пользователю приходится вводить большее количество информации. В противном случае недостаток входной информации может сделать невозможным получение достаточно важных выводов, поэтому возникает задача выявления минимально необходимого комплекта входной информации.

Существенного ускорения работы системы можно достичь за счёт исключения из дальнейшего рассмотрения уже проверенных правил. Для тех правил, часть *ТО* которых была выполнена, устанавливается признак “выполнено” и они в дальнейшем не обрабатываются системой. При этом следует помнить, что правило может не выполняться из-за недостатка входной информации. По ходу работы алгоритма может быть найдена дополнительная информация, что позволит обработать эти правила.

### 4.1.2. Алгоритм обратной цепочки рассуждений

В отличие от алгоритма прямой цепочки рассуждений при обратной цепочке рассуждений система ведёт работу от гипотез и результатов к фактам и входной информации. Выдвинутая гипотеза проверяется на основе правил базы знаний. В случае, если входной информации недостаточно для подтверждения гипотезы, происходит обращение к пользователю.

Работу алгоритма обратной цепочки рассуждений рассмотрим на основе примера, приведённого в предыдущем разделе на основе фрагмента базы знаний проектирования гальванического оборудования. Набор фактов и правил в данном случае остаётся без изменений.

Как и в прямом алгоритме, введём ряд вспомогательных таблиц, необходимых для работы алгоритма:

1) список переменных условия (табл. 4);
2) расширенный список переменных условия (табл. 5); построение этих таблиц рассмотрено ранее;
3) список логических выводов. В список заносятся последовательно номера всех правил базы знаний и имена переменных, расположенных в части *ТО* соответствующих правил.
4) стек логических выводов (табл. 13).

**13. Стек логических выводов**

| Номер правила | Номер условия |
|---|---|

Пусть теперь системе требуется найти ответ на вопрос: какой материал необходим для изготовления защитной футеровки проектируемой ванны?

Алгоритм ответа на вопрос будет включать в себя следующие шаги:

1. Определяем переменную, значение которой должно быть найдено. В нашем примере это “материал футеровки”.

2. Находим переменную “материал футеровки” в списке логических выводов. В стек логических выводов (табл. 14) заносим номера правила и условия, содержащие вывод по материалу. Данный шаг алгоритма рассуждений допускает некоторые модификации:

а) В стек заносятся в обратном порядке номера всех правил, содержащих в части *ТО* искомую переменную:

Затем выбираем из стека верхнее правило и начинаем его обработку. Если правило доказать не удалось, из стека выбираем следующее правило и т.д. В дальнейшем будем использовать данную модификацию алгоритма.

б) В стек заносится номер первого найденного правила, а на соответствующую строку в списке логических выводов установим указатель. Если правило недоказуемо, повторяем поиск в базе знаний, начиная его с текущей позиции указателя (табл. 15).

**14. Стек логических выводов после первого обращения к базе знаний**

| | |
|---|---|
| 6 | 1 |
| 7 | 1 |

Дно стека

**15. Стек и список логических выводов после первого обращения к базе знаний**

Стек

| | |
|---|---|
| 6 | 1 |

Список

| № Правила | Переменная |
|---|---|
| 6 | материал футеровки |

После этого система приступает к рассмотрению найденного правила. Из списка переменных условия определяем, что переменными условия в правиле 6 являются “футеровка” и “среда”. Выбираем первую переменную – “футеровка”. Из расширенного списка переменных условия определяем, что переменная не была проинициализирована. Переменная “футеровка” есть в списке логических выводов (правило 5). В этом случае система помещает номер правила в стек логических выводов (табл. 16).

**16. Стек логических выводов** (шаг 2)

| | |
|---|---|
| 7<br>5 | 1 |
| 6 | 1 |
| 7 | 1 |

3. В правиле 5 через список переменных условия система определяет переменную условия “среда” и через расширенный список переменных условия проверяет, имеет ли она какое-либо значение. В данном примере, переменная “среда” не проинициализирована, но в то же время она содержится в списке логических выводов (правило 1). Заносим в стек логических выводов информацию по найденному правилу (табл. 17).

**17. Стек логических выводов (**шаг 3)

| | |
|---|---|
| 1 | 1 |
| 5 | 1 |
| 6 | 1 |
| 7 | 1 |

4. Через список переменных условия определяем, что для первого правила переменными условия являются “среда” и “концентрация”. Выбираем первую переменную – “среда”. Из расширенного списка видно, что она не проинициализирована, но в то же время она содержится в другой строке списка логических выводов (правило 2). Информация по найденному правилу заносится в стек логических выводов (табл. 18).

**18. Стек логических выводов** (шаг 4)

| | |
|---|---|
| 2 | 1 |
| 1 | 1 |
| 5 | 1 |
| 6 | 1 |
| 7 | 1 |

Первой переменной условия правила 2 является переменная “Среда”, Она не проинициализирована и в списке логических выводов нет больше строк, содержащих её. Система должна задать вопрос о конкретном значении этой переменной. Предположим, что на вопрос системы был дан ответ “сернистая кислота”. После этого в расширенном списке переменных условия появляется информация, представленная в табл. 19.

**19. Изменения в расширенном списке** (шаг 4)

| | | |
|---|---|---|
| среда | определена | сернистая кислота |

5. Далее система проверяет значение переменной “среда” в условной части правила и в расширенном списке переменных условия. Эти значения не совпадают. О причине несовпадения можно узнать из анализа фактов базы знаний. В нашем примере база знаний содержит факт

сернистая кислота = кислота.

Следовательно, первое условие правила 2 можно считать выполненным. Если же подобного факта найти не удаётся, условие и все правило считается невыполненным.

6. В стеке логических выводов для правила 2 меняется указатель условия (табл. 20).

**20. Изменения в стеке логических выводов**

| | |
|---|---|
| 2 | 2 |

В этом случае система приступает к анализу второго условия правила 2. Анализ этого условия приводит к необходимости запросить пользователя о значении переменной “концентрация”. Предположим, что на запрос был получен ответ:

концентрация = более 70%.

При этом в расширенном списке переменных условия изменяется значение переменной “концентрация” (табл. 21).

**21. Изменения в расширенном списке** (шаг 6)

| | | |
|---|---|---|
| концентрация | определена | более 70% |

Анализ значений переменной “концентрация” в условной части второго правила и в расширенном списке переменных условия и далее на группе фактов показывает, что условие 2, а следовательно, и все правило 2 не выполняется. При этом из стека логических выводов исключается верхняя строка. Теперь правилом, требующим дальнейшего рассмотрения, является правило 1 (согласно стеку).

7. Анализ правила 1 показывает, что оно выполняется. При этом значение переменной “среда” в расширенном списке переменных условия изменяется (табл. 22).

**22. Изменения в расширенном списке** (шаг 7)

| | | |
|---|---|---|
| Среда | определена | концентрированная кислота |

В базу знаний добавляется временный факт:

сернистая кислота = концентрированная кислота

После этого исключаем из стека строку, содержащую правило.

8. Продолжается рассмотрение первого условия правила 5. Анализ показывает, что оно выполняется (с помощью группы фактов устанавливается соответствие между “концентрированной кислотой” из расширенного списка и ”агрессивной жидкостью” из условной части правила). При этом в расширенном списке переменных условия переменной “футеровка” присваивается значение “есть” (табл. 23).

**23. Изменения в расширенном спис**ке (шаг 8)

| | | |
|---|---|---|
| футеровка | определена | есть |

Правило 5 также исключается из стека к система переходит к анализу правила 6. Анализ этого правила показывает, что первое условие выполняется (“футеровка” = “есть”); переменной условия 2 правила 6 является переменная “среда”. С помощью группы фактов устанавливается соответствие между “концентрированной кислотой” из расширенного списка и “сернистой кислотой” из условной части правила. Условие выполняется, следовательно, выполняется и все правило. Согласно этому правилу

материал футеровки = свинец.

Ответ на поставленный вопрос получен и система завершает свою работу.

Вообще говоря, работа системы завершается в двух случаях:

а) Проинициализирована переменная вывода, значение которой требовалось определить. В этом случае система сработала корректно, значение переменной есть результат работы.

б) Переменная не проинициализирована, и при очередном обращении к стеку логических выводов выявляется, что он пуст. В этом случае результата достичь не удалось, выдаётся сообщение об ошибке.

Отличие в алгоритме работы с базой знаний при обратной цепочке рассуждений по сравнению с алгоритмом при прямой цепочке рассуждений состоит, прежде всего, в том, что при прямой цепочке анализ выполняется от переменных условия к переменным вывода; в обратной же все иначе – от переменных вывода к переменным условия. В соответствии с этим несколько изменяется и структура данных базы знаний.

### 4.2. ВЫБОР ЗНАЧЕНИЙ ПАРАМЕТРОВ НА ОСНОВЕ НЕЧЁТКОГО ПРАВИЛА MODUS PONENS

Итак, после построения функции принадлежности, которые определяют все нечёткие значения входных и выходных параметров, переходим к формализации отношений между входными и выходными параметрами. Наиболее простой способ задания таких отношений – это представление их в виде системы нечётких высказываний вида

$$\widetilde{L} = \begin{cases} \widetilde{L}_1 : \langle \textit{ЕСЛИ } \widetilde{E}_{11} \textit{ ИЛИ } ... \textit{ ИЛИ } \widetilde{E}_{1n_1} \textit{ ТО } \widetilde{B}_1 \rangle; \\ ... \\ \widetilde{L}_m : \langle \textit{ЕСЛИ } \widetilde{E}_{m1} \textit{ ИЛИ } ... \textit{ ИЛИ } \widetilde{E}_{mn_m} \textit{ ТО } \widetilde{B}_m \rangle, \end{cases} \tag{46}$$

где $m$ – число базовых нечётких значений выходной лингвистической переменной $\beta_v$; $\widetilde{B}_i (i = \overline{1,m})$ – нечёткое высказывание $\langle \beta_i \textit{ есть } \alpha_{vi} \rangle$, где $\alpha_{vi}$ – нечёткое значение (переменная) лингвистической переменной $\beta_v$; $\widetilde{E}_{ji} \left( i = \overline{1,n_i}; j = \overline{1,m} \right)$ – нечёткое высказывание, которое отражает $i$-ю входную ситуацию, которая может возникнуть, если выходная лингвистическая переменная примет нечёткое значение $\alpha_{vi}$;

$$\widetilde{E}_{ji} : \langle \beta_x \textit{ есть } \alpha_{xij} \textit{ и } \beta_y \textit{ есть } \alpha_{yij} \textit{ и } \beta_z \textit{ есть } \alpha_{zij} \textit{ и} ... \rangle;$$

где $\beta_x, \beta_y, \beta_z$ ... – входные лингвистические переменные, а $\alpha_{xji}, \alpha_{yji}, \alpha_{zij}$ ... – их нечёткие значения.

Возможно представить исходную систему нечётких высказываний в более компактной форме, используя правила преобразования нечётких высказываний. Вводится нечёткое высказывание $\widetilde{A}_j$, вида $\widetilde{A}_j : \langle \beta_W is \alpha_{Wj} \rangle$, где $\beta_W$ – обобщённая входная лингвистическая переменная; $\alpha_{Wj}$ – её нечёткое значение с функцией принадлежности:

$$\mu_{Wj}(\omega) = \max \left\{ \min \left[ \mu_{xij}(x), \mu_{yij}(x), \mu_{zij}(x), ... \right] \right\} \tag{47}$$

Тогда система нечётких высказываний примет вид

$$\widetilde{L}^{(1)} = \begin{cases} \widetilde{L}_1 : \langle \textit{ЕСЛИ } \widetilde{A}_1 \textit{ ТО } \widetilde{B}_1 \rangle \\ ..... \\ \widetilde{L}_m : \langle \textit{ЕСЛИ } \widetilde{A}_m \textit{ ТО } \widetilde{B}_m \rangle \end{cases} \tag{48}$$

Таким образом, получили нечёткую систему, которая связывает обобщённый входной параметр $\omega \in W = XYZ...$ и выходной параметр $v \in V$.

Для выбора решений в нечётких условиях, когда экспертная информация представлена нечёткой системой первого типа (46), можно использовать схему вывода, основанную на нечётком правиле modus ponens.

Нечётким правилом modus ponens называется следующая схема вывода:

$$\begin{array}{l} \widetilde{L}^{(1)}; \\ \widetilde{A}' – \textit{истина}; \\ \hline \widetilde{B}' – \textit{истина}. \end{array} \tag{49}$$

Истинность данной схемы представляет собой нечёткое множество

$$T_{mp}(\widetilde{L}^{(1)}, \widetilde{A}', \widetilde{B}') = \{ < \mu_{mp}^{(1)}(\tau) / \tau > \} = \bigcap_{j=\overline{1,m}} T_{mp}(\widetilde{L}_j^{(1)}, \widetilde{A}', \widetilde{B}'),$$

где $\forall \tau \in [0,1] \mu_{mp}^{(1)}(\tau) = \min\limits_{j=\overline{1,m}} \mu_j^{(1)}(\tau)$; $\min \mu_j^{(1)}(\tau)$ – функция принадлежности

нечёткого множества $T_{mp}(\widetilde{L}_j^{(1)}, \widetilde{A}', \widetilde{B}')$ – определяется как

$$\mu_j^{(1)}(\tau) = 1 \cap (1 - \mu_{TA}(\tau) + \mu_{Tb}(\tau)),$$

где $TA$ – истинность высказывания $\widetilde{A}'$ относительно $\widetilde{A}_j$; $TB$ – истинность высказывания $\widetilde{B}'$ относительно $\widetilde{B}_j$.

Для определения значений выходного параметра при конкретных значениях (чётких) параметров возможно использовать нечёткое правило modus ponens:

$$\begin{array}{l}\widetilde{L}^{(1)};\\ \underline{A'\text{– истина;}}\\ B'\text{ – истина.}\end{array} \qquad (50)$$

Здесь $A' = \langle \beta_W$ есть $w' \rangle$, $B' = \langle \beta_v$ есть $v' \rangle$ – чёткие высказывания о значении обобщённого входного и выходного параметров.

Так как чёткое значение $v$ можно рассмотреть как нечёткую переменную $\alpha_v$ характеризуемую нечётким множеством с функцией принадлежности:

$$\mu_{v'}(v) = \begin{cases} 1, если\, v = v'; \\ 0, в\, др.случ., \end{cases} \qquad (51)$$

то истинность чёткого высказывания относительно нечёткого высказывания полностью определяется одним значением $\mu_{v'}(v') = 1$. Тогда истинность правила modus ponens определяется:

$$T_{mp}(\widetilde{L}^{(1)}, A', B') = \{< \mu_{mp}^{(1)}(1)/1 >\}, \qquad (52)$$

где

$$\begin{array}{l}\mu_{mp}^{(1)}(1) = 1 \cap (1 - \mu_{W1}(w') + \mu_{v1}(v')) \cap (1 - \mu_{W2}(w') + \mu_{v2}(v')) \\ \cap \ldots \cap (1 - \mu_{Wm}(w') + \mu_{vm}(v'))\end{array} \qquad (53)$$

Величина (53) называется степенью истинности правила modus ponens для нечёткой системы высказываний первого типа. Данное понятие отражает степень соответствия чёткого значения $v'$ выходного параметра $V$ чёткому значению $w'$ обобщённого входного параметра $W$ при задании экспертной информации системой нечётких высказываний первого типа.

Введение понятия степени истинности нечёткого правила modus ponens позволяет сформулировать постановку задачи выбора выходного параметра: найти такие значения выходного параметра $V_0$, для которых степень истинности (53) схемы (49) на основе нечёткого правила modus ponens будет максимальной.

## 4.3. НЕЧЁТКАЯ МОДЕЛЬ ВЫБОРА ПАРАМЕТРОВ ПРИ ИНДУКТИВНОМ ЛОГИЧЕСКОМ ВЫВОДЕ

Рассмотрим теперь случай, когда экспертная информация задана системой нечётких высказываний второго типа (34). Задача заключается в том, чтобы для заданных значений входных параметров $x$, $y$, $z$, … входных параметров выбрать значения выходного параметра $V$.

Тогда индуктивную схему выбора можно записать:

$$\begin{array}{l}\widetilde{L}^{(2)};\\ \underline{A'\text{– истина;}}\\ B'\text{– истина.}\end{array} \qquad (54)$$

Здесь $A' = \langle \beta_W$ есть $w' \rangle$, $B' = \langle \beta_v$ есть $v' \rangle$ – чёткие высказывания о значении обобщённого входного и выходного параметров.

## 4.4. ВЫБОР ЗНАЧЕНИЙ ПАРАМЕТРОВ НА ОСНОВЕ НЕЧЁТКОГО ПРАВИЛА MODUS PONENS

Следует отметить, что дедуктивная (49) и индуктивная (54) схемы вывода принципиально отличаются друг от друга. В дедуктивной схеме высказывания о значениях входных параметров ( $\widetilde{A}'$ и $\widetilde{A}_j$ ) являются посылками как в самой схеме вывода, так и внутри системы $\widetilde{L}^{(1)}$, а высказывания о значениях выходных параметров ( $B'$ и $B_j$ ) являются следствиями.

В индуктивной схеме вывода высказывания означениях входных параметров являются посылкой для самой схемы ($A'$) и следствием внутри системы $\widetilde{L}^{(2)}$ ($A_j$), а высказывания о значениях выходного параметра являются следствием для схемы вывода ($B'$), но посылкой внутри системы $\widetilde{L}^{(2)}$ ($B_j$).

Поэтому для выбора значений выходного параметра $V$ на основе правила modus ponens необходимо индуктивную схему вывода преобразовать в дедуктивную. Для этого систему высказываний первого типа преобразуем в эквивалентную ей систему первого типа, используя правило контрапозиции:

$$ЕСЛИ\ A\ ТО\ B \equiv ЕСЛИ\ \neg B\ ТО\ \neg A\ . \qquad (55)$$

Применяя это правило к высказываниям из системы (34), получаем

$$ЕСЛИ\ \widetilde{B}_j\ ТО\ \widetilde{A}_j \equiv ЕСЛИ\ \neg\widetilde{A}_j\ ТО\ \neg\widetilde{B}_j\,, \tag{56}$$

где $\neg\widetilde{A}_j = \langle \beta_W есть \alpha^*_{Wj} \rangle$ ; $\neg\widetilde{B}_j = \langle \beta_V есть \alpha^*_{Vj} \rangle$ – нечёткие переменные с функциями принадлежности:

$$\forall w \in W = X \times Y \times Z \times \ldots \quad \mu^*_{Wj}(w) = 1 - \mu_{Wj}(w); \tag{57}$$

$$\forall v \in V \quad \mu^*_{Vj}(v) = 1 - \mu_{Vj}(v)\,. \tag{58}$$

Введём обозначения $\widetilde{A}^*_j = \neg\widetilde{A}_j$ и $\widetilde{B}^*_j = \widetilde{B}_j$. Тогда система нечётких высказываний второго типа (34) примет вид

$$\widetilde{L}^{(*)} = \begin{cases} \widetilde{L}^{(*)}_1 : \langle ЕСЛИ\ \widetilde{A}^*_1\ ТО\ \widetilde{B}^*_1 \rangle \\ \ldots\ldots\ldots \\ \widetilde{L}^*_m : \langle ЕСЛИ\ \widetilde{A}^*_m\ то\ \widetilde{B}^*_m \rangle \end{cases} \tag{59}$$

Тогда схема вывода (54) запишется в виде

$$\begin{array}{l} \widetilde{L}^{(*)}; \\ \qquad\qquad \underline{A' – истина;} \\ B' – истина. \end{array} \tag{60}$$

Степень истинности правила modus ponens (50) для этой схемы примет вид

$$\begin{array}{l} \mu^{(2)}_{mp}(1) = 1 \cap (1 - \mu^*_{W1}(w') + \mu^*_{v1}(v')) \cap (1 - \mu^*_{W2}(w') + \mu^*_{v2}(v')) \\ \cap \ldots \cap (1 - \mu^*_{Wm}(w') + \mu^*_{vm}(v')). \end{array} \tag{61}$$

Таким образом, при задании экспертной информации системой нечётких высказываний второго типа (34) математическая постановка задачи выбора значений выходного параметра формулируется так: при заданной системе нечётких высказываний второго типа (34) для значений *x*, *y*, *z*, … входных параметров найти такие значения выходного параметра v, для которых индуктивная схема вывода (60) имеет наибольшую степень истинности (61) правила modus ponens.

Для решения данной задачи выражение (60) в соответствии с (57), (59) приведём к виду

$$\mu^{(2)}_{mp}(1) = 1 \cap (1 + \xi_1 - \mu_{v1}(v)) \cap \ldots \cap (1 + \xi_m - \mu_{vm}(v))\,, \tag{62}$$

где $\xi_j = \mu_{Wj}$, $j = \overline{1,m}$.

Такая запись степени истинности несёт в себе много полезной информации.

Для достижения максимального значения степени истинности (=1), необходимо выполнение условий:

$$\begin{array}{c} 1 + \xi_1 - \mu_{v1}(v) \ge 1, \\ 1 + \xi_2 - \mu_{v2}(v) \ge 1, \\ \ldots \\ 1 + \xi_m - \mu_{vm}(v) \ge 1. \end{array}$$

Отсюда имеем

$$\begin{array}{c} \mu_{v1}(v) \le \xi_1, \\ \mu_{v2}(v) \le \xi_2, \\ \ldots \\ \mu_{vm}(v) \le \xi_m. \end{array} \tag{63}$$

Теперь можно из области *V* всех возможных значений выходного параметра определить значения, которые удовлетворяют всем этим условиям. Это и будут значения выходного параметра, оптимальные с точки зрения выбора.

Следует отметить, что применение правила modus ponens для системы нечётких высказываний второго типа имеет следующую особенность: множество рекомендуемых значений в произвольном случае состоит из нескольких отдельных интервалов, причём их количество может меняться в пределах от 1 до *m* + 1. Это связано с тем, что выбор решения максимизирует значение степени истинности (62), что в свою очередь связано с уменьшением значений $\mu_j(v)$ (63). В силу унимодальности функций принадлежности, уменьшение их значений возможно как влево, так и вправо от точки максимума, что и порождает указанную особенность.

## 4.4. ВЫБОР ЗНАЧЕНИЙ ПАРАМЕТРОВ НА ОСНОВЕ НЕЧЁТКОЙ ИНДУКТИВНОЙ СХЕМЫ ВЫВОДА

Нечёткой индуктивной схемой вывода называется схема

$$\frac{\langle ЕСЛИ\ \widetilde{B}\ ТО\ \widetilde{A}\rangle;\quad \widetilde{A}'-истина;}{\langle \widetilde{B}'-истина\rangle - более\ правдоподобно.} \tag{64}$$

Для системы нечётких высказываний второго типа индуктивная схема вывода принимает вид

$$\frac{\widetilde{L}^{(2)};\quad A'-истина;}{< B'-истина > -более\ правдоподобно,} \tag{65}$$

где $A' = \langle \beta_W$ есть $w' \rangle$, $B' = \langle \beta_v$ есть $v' \rangle$ – чёткие высказывания о значении обобщённого входного и выходного параметров.

Истинность данной схемы определяется как

$$T_{ис}(\widetilde{L}^{(2)}, A', B') = \{< \mu_{ис}^{(2)}(1)/1 >\},$$

где

$$\mu_{ис}^{(2)}(1) = 1 \cap (1 - \mu_{W1}(w') + \mu_{v1}(v')) \cap (1 - \mu_{W2}(w') + \mu_{v2}(v')) \cap \ldots \cap (1 - \mu_{Wm}(w') + \mu_{vm}(v')). \tag{66}$$

Величина (66) называется степенью истинности индуктивной схемы вывода для нечёткой системы высказываний второго типа. Данное понятие отражает степень соответствия чёткого значения $v'$ выходного параметра $V$ чёткому значению $w'$ обобщённого входного параметра $W$ при задании экспертной информации системой нечётких высказываний второго типа.

Тогда математическая постановка задачи выбора выходного параметра имеет вид: найти такие значения выходного параметра $V_0$, для которых степень истинности (66) схемы (65) на основе нечёткой индуктивной схемы вывода будет максимальной.

## 4.6. ВЫБОР ВАРИАНТА

В разделах 4.2, 4.3 рассматривались задачи выбора значений параметров объекта. Другим типом задач является выбор некоторого варианта из заранее заданного, достаточно небольшого числа вариантов. В этом случае экспертная информация может быть представлена одним из следующих способов:

$$\widetilde{L}^{(1)} = \begin{cases} \widetilde{L}_1^{(1)} : \langle ЕСЛИ\ \widetilde{A}_1 ТО\ B_1 \rangle \\ ..... \\ \widetilde{L}_m^{(1)} : \langle ЕСЛИ\ \widetilde{A}_m ТО\ B_m \rangle \end{cases} \tag{67}$$

или

$$\widetilde{L}^{(2)} = \begin{cases} \widetilde{L}_1^{(2)} : \langle ЕСЛИ\ B_1\ ТО\ \widetilde{A}_1 \rangle \\ \cdots \\ \widetilde{L}_m^{(2)} : \langle ЕСЛИ\ B_m\ ТО\ \widetilde{A}_m \rangle \end{cases} \tag{68}$$

Здесь высказывания $B_j$ выражают суть выбора того или иного варианта проектирования, т.е. являются чёткими высказываниями:

$B_j : \langle \beta_v\ есть\ v_j \rangle$, где $v_j$ – чёткое значение из конечного множества вариантов.

### 4.6.1. Выбор варианта при дедуктивном выводе

В этом случае схема вывода имеет вид

$$\frac{\widetilde{L}^{(1)};\quad A'-истина;}{B'-истина.} \tag{69}$$

Степень истинности данной схемы для произвольного $j$-го высказывания системы (65) определяется:

$$\mu_{mp}^{(1)}(v_j) = 1 \cap [1 - \mu_{w1}(w') + \mu_{v1}(v')] \cap \ldots$$
$$\cap [1 - \mu_{wj}(w') + \mu_{vj}(v')] \cap \ldots \cap [1 - \mu_{wm}(w') + \mu_{vm}(v')]. \quad (70)$$

При этом учитываем, что $B_j$ – чёткие высказывания и, следовательно:

$$\mu_{vj}(v') = \begin{cases} 1 & при\ v' = v_j; \\ 0 & в\ др.случ. \end{cases} \quad (71)$$

Для выбора варианта $v_0 \in V$ на основе правила modus ponens необходимо:

1) для каждого $v_j \in V$ ($j = 1 \ldots m$) определить степень истинности по формуле (70);
2) в качестве решения ($v_0$) выбрать такое $v_j \in V$, при котором степень истинности (70) имеет максимальное значение.

### 4.6.2. Выбор варианта при индуктивном выводе

При выборе варианта при индуктивном выводе используется экспертная информация в виде (68).
Индуктивная схема вывода имеет вид

$$\frac{\begin{array}{c}\widetilde{L}^{(2)};\\ A' - истина;\end{array}}{B' - истина.} \quad (72)$$

Для выбора варианта проектирования на основе правила modus ponens преобразуем систему (68) второго типа в систему первого типа, согласно правилу контрапозиции (52):

$$\widetilde{L}^{(2)} \equiv \widetilde{L}^{*} = \{L_j^{*}\},\ j = \overline{1,m}, \quad (73)$$

где $\widetilde{L}_j^{*} : \langle если\ \widetilde{A}_j^{*}\ то\ B_j^{*} \rangle$; $\widetilde{A}_j^{*} : \langle \beta_W\ есть\ \alpha_{Wj}^{*} \rangle$; $B_j^{*} : \langle \beta_V\ есть\ \alpha_{Vj}^{*} \rangle$.

Значения $\alpha_{Wj}^{*}$ и $\alpha_{Vj}^{*}$ характеризуются функциями принадлежности:

$$\mu_{Wj}^{*}(w) = 1 - \mu_{Wj}(w) \text{ и } \mu_{Vj}^{*}(v) = 1 - \mu_{Vj}(v) = \begin{cases} 1 & при\ v \neq v_j; \\ 0 & в\ др.случ. \end{cases}$$

Задача принятия решения, как и ранее, заключается в выборе такого варианта, при котором степень истинности нечёткого правила modus ponens имеет наибольшее значение.

Для произвольного варианта $v_j \in V$ степень истинности имеет вид

$$\mu_{mp}^{(2)}(v_j) = 1 \cap [1 - \mu_{W1}^{*}(w) + \mu_{V1}^{*}(v_j)] \cap \ldots$$

$$\ldots \cap [1 - \mu_{Wj}^{*}(w) + \mu_{Vj}^{*}(v_j)] \cap \ldots \cap [1 - \mu_{Wm}^{*}(w) + \mu_{Vm}^{*}(v_j)] =$$

$$= 1 \cap [1 - (1 - \mu_{W1}(w)) + (1 - 0)] \cap \ldots$$

$$\ldots \cap [1 - (1 - \mu_{Wj}(w)) + (1 - 1)] \cap \ldots \cap [1 - (1 - \mu_{Wm}(w)) + (1 - 0)] =$$
$$= 1 \cap [1 + \mu_{W1}(w)] \cap \ldots \cap [\mu_{Wj}(w)] \cap \ldots \cap [1 + \mu_{Wm}(w)].$$

Таким образом, в качестве варианта выбирается такой вариант $v_k \in V$, для которого степень истинности $\mu_{Wk}(w)$ имеет наибольшее значение, т.е.:

$$v_0 = v_k \left| \mu_{Wk}(w) = \max_{j = \overline{1,m}} (\mu_{Wj}(w)) \right. . \quad (74)$$

## 5. ПРИНЯТИЕ РЕШЕНИЯ В НЕЙРОСЕТЕВЫХ СИСТЕМАХ

Очевидно, чтобы система хорошо работала и решала практические задачи, необходимо её обучить. Если говорить в общем, то обучение это относительно постоянный процесс изменения поведения при поступлении жизненного опыта. Если говорить о человеке, то результат его обучения оценивается по действиям и поступкам. Обучение же нейронных сетей – более прямой процесс.

Обучение нейронных сетей рассматривается как процесс аппроксимации непрерывной функции $y(X)$ другой функцией $Y(W, X)$, где $X = [x_1, x_2, ..., x_n]^t$ – входной вектор, а $W = [w_1, w_2, ..., w_n]^t$ – вектор весовых коэффициентов.

Задача обучения состоит в выборе вектора $W$, такого, что достигается лучшая аппроксимация, т.е.:

$$\rho(Y(W^*, X), y(X)) \le \rho(Y(W^*, X), y(X)), \qquad (75)$$

где $\rho(Y(W, X), y(X))$ – функция расстояния, которая определяет значение качества аппроксимации между $Y(W, X)$ и $y(X)$).

Все алгоритмы обучения делятся на две большие группы: с учителем и без учителя.

Рассмотрим алгоритм обучения с учителем многослойной нейронной сети. Предполагается, что в каждый момент времени вместе с входами формируется желаемое значение выхода $d$, которое поступает от учителя. Это иллюстрируется на рис. 15.

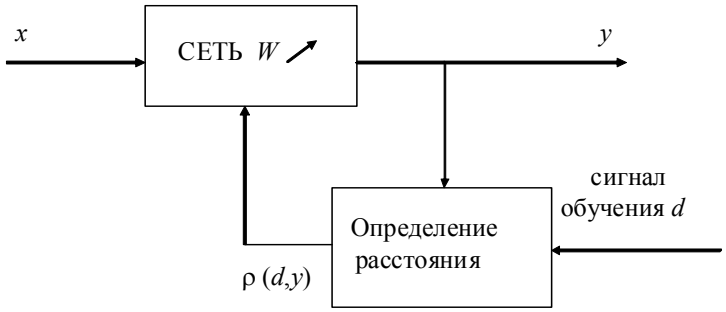



**Рис. 15**

По значениям реального выхода и желаемого выхода формируется ошибка, которая используется для корректировки параметров нейронной сети. Множество входных и выходных образцов, называемых обучающим множеством, является необходимым для такого способа обучения. “Учитель” оценивает негативное направление градиента ошибки и соответственно сеть уменьшает ошибку. Во многих ситуациях, входы, выходы и вычисляемые градиенты являются детерминистическими, однако минимизация осуществляется по случайному закону. И, как результат, большинство алгоритмов обучения с учителем используют стохастическую минимизацию ошибки в многомерном пространстве весов.

Рассмотрим стандартный алгоритм обучения многослойных нейронных сетей на основе обратного распространения ошибки (back-propagation).

Цель обучения состоит в определении всех весовых коэффициентов, при которых ошибка вычислений будет минимальной. Обучение сети осуществляется на основе множества пар “вход-выход”. Каждый пример обучения состоит из вектора $X = [x_1, x_2, ..., x_{n1}]$ входных сигналов и вектора $D = [d_1, d_2, ..., d_{n3}]$ желаемых результатов. Обучение состоит в определении всех весовых коэффициентов, таких что значение ошибки между желаемыми и действительными выходными сигналами будет минимальной (близкой к 0).

Рассматриваемый метод использует пошаговый градиентный подход для минимизации функции квадрата ошибки. С выводом всех необходимых формул можно ознакомиться в работах [8, 9]. Функциональная схема алгоритма обучения с обратным распространением ошибки представлена на рис. 16. Главным при данном способе обучения является определение локальной ошибки $\delta_j^s$ ($s$ = 1, 2). В выходном слое ошибка определяется как функция от желаемого и действительного результатов и сигмоидальной функции активации. Для скрытого слоя локальная ошибка определяется на базе ранее определённых локальных ошибок выходного слоя.

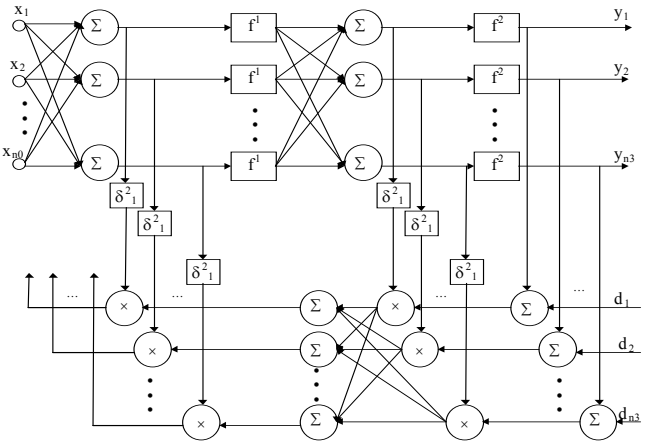


**Рис. 16**

Алгоритм реализации обучения с обратным распространением ошибки включает следующую последовательность шагов:

1. Инициализируются все синаптические веса $w_{ij}^{s}$ как малое случайное число.

2. Задаются все примеры обучения в виде пар “вход-выход”; вычисляются действительные значения выходов всех нейронов, используя заданные значения $w_{ij}^{s}$ и значения входов.

3. Используя значения желаемого и действительного выходов определяются локальные ошибки $\delta_j^s$ для всех уровней:

$$\delta_j^2 = \frac{\partial E_p}{\partial o_j^{/}} = -\frac{\partial E_p}{\partial e_{jp}} \frac{\partial e_{jp}}{\partial o_j^{/}} = e_{jp} \frac{\partial f^2}{\partial o_{jp}^{/}} = (d_p - y_p) \frac{\partial f^2}{\partial o_{jp}^{/}} \ ; \qquad (76)$$

$$\delta_j^1 = \frac{\partial f^1}{\partial h_j^{/}} \sum_{i=1}^{n_2} \delta_i^2 w_{ij}^2 \ . \qquad (77)$$

4. Пересчитываются синаптические веса по итерационной формуле

$$\Delta w_{ji}^s = \eta \delta_j^s x_i^s \ , \ s = 1, 2. \qquad (78)$$

5. Переходим ко второму примеру обучения и возвращаемся на пункт 2.

Заметим, что все тренировочные примеры обрабатываются циклически, пока ошибка обучения не станет малой. После обучения многослойный персептрон (нейронная сеть) обычно обладает свойствами объекта, для которого он обучался. Теперь можно вводить любые входные значения и получать выходные без дополнительного обучения.

## 6. ЯЗЫК ПРОЛОГ

Язык программирования Пролог появился в 1970 году одновременно с такими сейчас распространёнными языками, как Паскаль и Си. Он используется при “нетрадиционном” применении вычислительной техники: для создания систем понимания естественного языка, экспертных систем, базы знаний и т.д., т.е. задач, которые относят к задачам искусственного интеллекта.

Сила этого языка – в принципиально отличном от традиционных языков программирования подходе к описанию способа решения задачи: программа на Прологе описывает не процедуру решения задачи, а логическую модель предметной области:

- некоторые факты относительно свойств предметной области и отношений между этими свойствами;
- правила вывода новых свойств и отношений из уже заданных фактов.

Можно сказать, что Пролог – описательный язык, который используется для решения задач, которые могут быть выражены в терминах объектов и отношений между ними.

Программирование на языке Пролог состоит из следующих этапов:

- объявления некоторых фактов об объектах и отношениях между ними;
- объявления некоторых правил об объектах и отношениях между ними;
- формулировки вопросов об объектах и отношениях между ними.

## 6.1. ОПИСАНИЕ ЯЗЫКА

Пролог-программы состоят из термов. Терм – это либо константа, либо переменная, либо структура.

Каждый из этих термов записывается как последовательность символов алфавита:

1) прописных букв латинского алфавита A – Z;
2) строчных букв латинского алфавита a – z;
3) прописных букв русского алфавита А – Я;
4) строчных букв русского алфавита а – я;
5) цифр 0 – 9;
6) специальных знаков: +, –, *, /, \, ^, <, >, ~, :, ., ?, ,, @, #, &, $.

Рассмотрим каждый из термов.

*Константы* – это поименованные конкретные объекты или конкретные отношения. Существует два вида констант:

– атомы;
– числа.

Атомы бывают двух видов:

1) составленные из букв и цифр;
2) составленные из специальных знаков.

Атомы, составленные из букв и цифр, должны начинаться со строчной буквы: sreda, conc, ask, a1, b2 и т.д.

Иногда возникает необходимость иметь атом, начинающийся с прописной буквы или цифры. Если атом заключается в кавычки, то он может содержать любые литеры, например: “Prolog” “1причина”,”_ask” и т.д.

Атомы, составленные из спецзнаков, как правило, состоят только из специальных знаков, например: :–, ., ,.

Другой вид констант – числа – используется для представления числовых данных, что позволяет выполнять арифметические операции над ними, например: 0, 999, 54,23 и т.д.

*Переменные* могут менять своё значение в ходе работы программы. Имена переменных начинаются с прописной буквы или знака подчёркивания: X, Y, Z, Ask, _1_a.

Иногда возникает необходимость в использовании переменной, имя которой не будет нигде употребляться. В этом случае можно использовать анонимную переменную – одиночный знак подчеркивания (_). Анонимная переменная позволяет избавить программиста от необходимости придумывать разные имена переменным в тех случаях, когда эти имена в утверждении нигде больше не употребляются.

*Структура* – это единый объект, состоящий из совокупности других объектов, называемых компонентами. Компоненты группируются в структуру для удобства использования. Структура записывается на Прологе с помощью указания её функтора и компонент. Компоненты заключаются в круглые скобки и разделяются запятыми. Функтор записывается перед открывающейся круглой скобкой. Компоненты структуры – это константы, переменные или структуры. Если структура не содержит переменных ни на каком уровне вложенных подструктур, она называется константной, иначе – переменной, например:

book(X, avtor(“Александр Пушкин”))

Следует отметить, что факт в языке Пролог описывается как структура, функтором которой является некоторый предикат. Например, имеется факт: ″азотная кислота является кислотой″. На языке Пролог его можно записать: is(″азотная кислота″, ″кислота″).

## 6.2. ТИПЫ ДАННЫХ В ЯЗЫКЕ ПРОЛОГ

В языке Пролог используются следующие типы данных:

- symbol – последовательность букв, цифр и знаков подчеркивания, которая начинается со строчной буквы или заключена в кавычки, например person, a1, b2, “Prolog”.
- string – любая последовательность символов, которая заключена в кавычки, например “кислота”, “123/456”.
- char – отдельный символ, заключенный в апострофы, например ‘A’, ‘3’.
- integer – целое число.
- real – вещественное число.
- file – имя файла.

Данные типа symbol в отличие от данных типа string запоминаются в таблице символов, которая размещается в оперативной памяти, поэтому её использование обеспечивает наиболее быстрый поиск. Однако для построения таблицы символов требуется дополнительное время.

## 6.3. РАВЕНСТВО И УСТАНОВЛЕНИЕ СООТВЕТСТВИЯ

В прологе существует особый предикат равенство, обозначаемый литерой “ = ”, например X = Y.

При согласовании с данными цели X = Y, где X, Y – любые термы, действуют следующие правила:

1. Если X – неконкретизированная переменная, а переменная Y – конкретизирована, то выражение X = Y является оператором присваивания.

2. Если X, Y – конкретизированные термы, то выражение X = Y – операция сравнения.

3. Две структуры равны, если они имеют один и тот же функтор и одинаковое число аргументов, причём все соответствующие аргументы равны, например, при согласовании такого утверждения: is (“сернистая кислота”, “кислота”) = is (X, “кислота”), переменной X будет присвоено конкретное значение “сернистая кислота”.

Структуры могут быть вложены одна в другую на любую глубину. Если такие вложенные структуры проверяются на равенство, проверка займёт больше времени, поскольку необходимо проверить все структуры, например, попытка согласовать следующую цель:

a(b, C, d(e, F, g(h, i, J))) = a(B, c, d(E, f, g(H, i, j)))

будет успешной, а переменные B, C, F, E, J будут конкретизированы, им будут присвоены значения: b, c, f, e, j.

4. Если X, Y – неконкретизированные переменные, выражение X = Y означает, что переменные X, Y стали сцепленными. Если две переменные сцеплены, то при конкретизации одной из них, второй переменной будет автоматически присвоено то же самое конкретное значение, что и первой.

## 6.4. ОПЕРАЦИИ В ЯЗЫКЕ ПРОЛОГ

1. Операции отношения: =, <>, <, >, <=, >=.
2. Арифметические операции: +, –, *, /, mod.
3. Логические операции: AND, OR, NOT.

## 6.5. СТРУКТУРА ПРОГРАММЫ НА ЯЗЫКЕ ПРОЛОГ

Программа на Прологе может состоять из следующих разделов:

1. Описание нестандартных типов

[global] domains

Например,

domains
X = symbol
L = symbol*

Раздел global domains обеспечивает описание глобальных доменов, что позволяет организовать межмодульный интерфейс.

2. Хранение временного факта:

database

Например,

database
fact_sreda(X)

3. Описание предикатов, входящих в программу:

[global] predicates

В этом разделе определяются предикаты (отношения). Каждый предикат определяется своим именем и своими аргументами (параметрами). Если типы данных некоторых аргументов предопределены, то они должны быть описаны заранее в разделе domains. Предикат может не содержать параметров.

Раздел global predicates обеспечивает определение глобальных переменных предикатов, что позволяет обеспечить межмодульный интерфейс.

4. Определение целей:

goal

В этом разделе определяется цель, которая может состоять из нескольких подцелей.

5. Определение фактов и правил:

clauses

В этом разделе определяются факты и правила. Факт представляется именем предиката, за которым следуют аргументы, заключённые в круглые скобки. Заканчивается запись факта точкой. Все имена предикатов и константы должны начинаться со строчной буквы.

Правило состоит из заголовка правила и тела правила. Заголовок представляет собой предикат, тело состоит из термов, которые могут быть связаны между собой знаками логических операций (*NOT*, *AND*, *OR*). Между телом и заголовком стоит слово *if* (возможно использовать его встроенным предикатом ":–"). Каждое правило заканчивается точкой. Правило – аналог аксиомы вида $P1\&P2\&\ldots\&Pk \rightarrow Q$, где $Pi$ – условия; $Q$ – следствие. В Прологе обычно применяется инверсная запись правила: $Q$ :– $P1$, $P2$, …, $Pk$. Читается правило так: $Q$ удовлетворяется, если удовлетворяются подцели $P1$, $P2$, …, $Pk$.

6. Комментарии.

Располагаются в программе в произвольном месте, начинаются с символов /* и заканчиваются символами */.

***Пример.***

Пусть база знаний включает четыре факта и одно правило:

*Факты:*

1. население США составляет 520 млн. человек;
2. население России составляет 216 млн. человек;
3. площадь США составляет 8 млн. кв. км;
4. площадь России составляет 80 млн. кв. км.

*Правило:*

Если население страны $X$ составляет $Y$ млн. человек и площадь страны $X$ составляет $W$ млн. кв. км, то средняя плотность населения страны $X$ равна $Y/W$.

*Цель:* Определить плотность населения США и России.

```
predicates
count (string, real).
plosh (string, real).
density(string, real).
clauses
```

```
count("США",302).
count("Россия",142).
plosh("США",9).
plosh("Россия",17).
density(X,A) :-count(X,Y), plosh(X,W), A = Y/W, write("\n ", X, "=", A).
goal
X = "США", density(X, K), density("Россия", R).
```

## 6.6. СПИСКИ В ЯЗЫКЕ ПРОЛОГ

Список – это упорядоченная последовательность элементов, которая может иметь произвольную длину. Элементами списка могут быть любые термы: константы, переменные, структуры, а также другие списки.

В Прологе список – это просто один из частных видов структуры. Список – это любой пустой список, не содержащий ни одного элемента, или структура, имеющая две компоненты: голову и хвост. Пустой список записывают как []. Голова и хвост списка являются компонентами функтора, обозначаемого точкой. Например, список, состоящий из одного элемента "a", можно записать как:

.(a, [])

Аналогично, список, состоящий из трёх элементов: a, b, c, может быть записан как:

.(a, .(b, .(c, [])))

Списки являются упорядоченными последовательностями элементов, так что списки .(a, .(b, [])) и .(b, .(a, [])) являются разными.

Так как запись сложных списков с помощью функтора "." часто оказывается неудобной, то в Прологе предусмотрена другая синтаксическая форма, которая может быть использована для записи списков в программе. Это так называемая скобочная форма записи списка. Она представляет собой заключённую в квадратные скобки последовательность, например: [a], [a, b, c].

Списки могут содержать другие списки или переменные, например: [a, V1, b, [X,Y]]. Переменные, входящие в списки, ничем не отличаются от переменных в любой другой структуре. Они в любой момент могут быть конкретизированы.

Работа со списками основана на расщеплении их на голову и хвост. Голова списка – это его первый элемент; хвост списка – это список, состоящий из всех элементов исходного списка, за исключением первого. Например:

| Список | Голова | Хвост |
|---|---|---|
| [a, b, c] | A | [b, c] |
| [a] | A | [ ] |
| [a, V1, b, [X,Y]] | A | [V1, b, [X,Y]] |

Так как операция расщепления списка на голову и хвост очень широко используется в Прологе, то введена специальная форма для представления списка с головой X и хвостом Y. Эта форма записывается как [X|Y], где для разделения X и Y используется вертикальная черта. При конкретизации структуры подобного вида X сопоставляется с головой списка, а Y – с хвостом, например:

| Список 1 | Список 2 | Конкретизация |
|---|---|---|
| [1, 2, 3] | [X\|Y] | X = 1 |
| | | Y = [2,3] |
| [a, b, c, d] | [X, Y\|Z] | X = a |
| | | Y = b |
| | | Z = [c, d] |
| [A, b, c] | [a\|B] | A = a |
| | | B = [b, c] |
| [abc\|T] | [abc\|def] | T = [def] |

Синтаксически некорректной является запись: [X, Y|Z, W].

Возможны ситуации, когда, используя скобочную форму записи списков, создаётся структура, похожая на списки, но не заканчивающаяся пустым списком, например:

| Список 1 | Список 2 | Конкретизация |
|---|---|---|
| [a\|B] | [P\|c] | P = a |
| | | B = c |

Константа "c" не является ни списком, ни пустым списком. Обработка таких структур требует большой осторожности, когда они используются в качестве хвоста списка.

Одна из областей применения списков – это представление строк литер. Иногда возникает необходимость в использовании строк литер для ввода или вывода текста.

***Пример.***

В базе знаний содержатся два факта и четыре правила.

*Факты:*

1) сернистая кислота – кислота;

2) азотная кислота – кислота.

*Правила:*

1. *ЕСЛИ* среда – кислота И концентрация – менее 70%
*ТО* среда – разбавленная кислота
2. *ЕСЛИ* среда – кислота И концентрация – более 70%
*ТО* среда – концентрированная кислота
3. *ЕСЛИ* среда – разбавленная кислота
*ТО* материал ванны – хромоникелевая сталь
4. *ЕСЛИ* среда – концентрированная кислота
*ТО* материал ванны – углеродистая сталь

Требуется определить материал ванны при различных входных данных.

Программа на языке Пролог может иметь вид:

```
domains
A=symbol
D=symbol
predicates
is(A,A,D).
sreda(A,D,D).
conc(A,D,D).
mater(A,D,D).
ask_sreda(A).
view(D).
run.
database
fact_s(A).
fact_c(A).
goal
run.
clauses
is(“сернистая кислота“, “кислота“, L) :-
    L=[“\n сернистая кислота = кислота“].
is(“азотная кислота “, “кислота“, L) :-
  L=[“\n азотная кислота = кислота“].
sreda(“разбавленная кислота“,L,L2) :-
  ask_sreda(X), is(X,_,L), ask_conc(C), conc(“менее 70%“,L,L1),
  S=”\n среда=разбавленная кислота т.к.“,
  S1=”\n среда=кислота“,S2=”\n И концентрация= менее 70%“,
  L2=[S,S1,S2|L].
sreda(“концентрированная кислота“, L,L2) :-
  fact_s(X), is(X,_,L), conc(“более 70%“,L,L1),
   S=”\nсреда=концентрированная кислота т.к. “,
   S1=”\nсреда=кислота“,S2=”\n И концентрация= более 70%“,
   L2=[S,S1,S2|L].
ask_sreda(X) :-
  write(“\n среда? “), readln(X), asserta(fact_s(X)).
ask_conc(C) :-
  write(“\n концентрация? “), readln(C), asserta(fact_c(X)).
conc(C,L,L1) :-
  fact_c(C), S=”\nконцентрация=“,
  S1=C, S2=”\n т.к. было задано“, L1=[S,S1,S2|L].
mater(“хромоникелевая сталь“,L2,L3) :-
    sreda(“разбавленная кислота“,_,L2), write(“\n Получено решение“),
    S=”\nматериал ванны= хромоникелевая сталь“,
    S1=”\nт.к. среда = разбавленная кислота“, L3=[S,S1|L2].
 mater(“углеродистая сталь“,L2,L3):-
  sreda(“концентрированная кислота“,_,L2),
  write(“\n “), S=”\n материал ванны = углеродистая сталь“,
  S1=”\n т.к. среда = концентрированная кислота“, L3=[S,S1|L2].
run :-
  clearwindow, mater(_,_,L3), view(L3).
view([]).
view([S|L] :-
  write(S), view(L).
```

## 6.7. ПРИМЕРЫ ФУНКЦИЙ РАБОТЫ СО СПИСКАМИ

При работе со списками могут потребоваться дополнительные функции, которые требуют написания соответствующих предикатов.

Предположим, что имеется некоторый список, в котором X обозначает его голову, а Y – хвост (т.е [X|Y]).

### 6.7.1. Принадлежность элементов списку

Требуется определить, содержится ли некоторый объект в указанном списке. В Прологе это можно сделать, определив, совпадает ли этот объект с головой списка. Если – да, то наш поиск завершается успехом. Если – нет, то проверяем, есть ли объект в хвосте исходного списка, а значит снова проверяется голова, но уже хвоста списка. Затем проверяется голова очередного хвоста списка. Если мы доходим до конца списка, то наш поиск завершается неудачей: указанного объекта в списке нет.

Для того чтобы записать все это на Прологе, сначала необходимо установить отношение принадлежности между объектом и списком.

У нас есть два условия, которые надо проверить:

1) X является элементом списка Y, если X совпадает с головой списка Y.
2) X является элементом списка Y, если X входит в хвост списка Y. Для проверки этого условия можно использовать рекурсию.

Таким образом, можно записать:

```
domains
L=symbol*
predicates
yes(symbol, L).
clauses
yes(X, [X|_]) :- write("yes").
yes(X, [_|Y]) :- yes(X,Y).
goal
clearwindow, yes(d, [a,b,c,d,e,f]).
```

Результатом работы этой программы будет ответ "yes", так как элемент d входит в список [a, b, c, d, e, f]. Наиболее важный момент, о котором следует помнить, встретившись с рекурсивно определённым предикатом, заключается в том, что, прежде всего надо найти граничные условия и способ использования рекурсии.

Для рассмотренного предиката yes имеются два типа граничных условий. Либо объект, который мы ищем, содержится в списке, либо он не содержится в нём. Первое граничное условие для предиката yes распознается первым правилом, которое приведёт к прекращению поиска в списке, если первый аргумент предиката совпадает с головой списка, соответствующего второму аргументу.

Второе граничное условие не распознается ни одним из утверждений, но рекурсивные вызовы предиката yes во втором правиле будет продолжаться до тех пор, пока хвост списка не станет пустым. В этом случае поиск является неудачным.

### 6.7.2. Определение длины списка

Данную функцию также можно реализовать с помощью рекурсии:

```
domains
L=symbol*
predicates
len(L,integer).
clauses
len([],0).
len([_|HV],D) :- len(HV,T), D=T+1.
goal
len([a,b,c,d,e,f],D), write(D).
```

Результатом работы такой программы будет число 6.

### 6.7.3. Разделение строки на слова с занесением их в список

Данную функцию можно выполнить аналогично, используя рекурсию:

```
decompose("",[]).
decompose(X, [X1|T]) :- fronttoken(X,X1,X2), decompose(X2,T).
```

Граничное условие в данном случае – это пустая строка, аналогом которой является пустой список.

В данном примере используется встроенный предикат:

fronttoken(Строка, Лексема, Остаток),

который разделяет строку на лексему и остаток. В данном случае Строка, Лексема, Остаток имеют тип string. Например, в результате выполнения предиката

fronttoken("Мама мыла раму", X, Y),

переменной X будет присвоено значение "Мама", а переменной Y – значение "мыла раму".

### 6.7.4. Слияние списков

Данная функция выполняется рекурсивно:

append([],L,L).

append([X|L],L2,[X|L3]) :- append(L1,L2,L3).

Выход на граничное условие происходит, когда первый аргумент является пустым списком. Любой список, присоединённый к пустому списку, даёт тот же самый список. Во всех других случаях выполняется второе правило, смысл которого можно описать так:

1) первый элемент первого списка (X) всегда будет и первым элементом третьего списка;

2) хвост третьего списка (L3) всегда будет представлять результат присоединения второго списка (L2) к хвосту первого списка (L1). Для присоединения рекурсивно вызывается предикат append;

3) так как при каждом обращении к правилу удаляется голова первого списка, то постепенно этот список будет исчерпан и станет пустым так, что произойдет выход на граничное условие.

## 6.8. ОТСЕЧЕНИЕ

В Прологе используется специальный механизм, называемый отсечением (или сокращением). Отсечение позволяет указать, какие из сделанных ранее выборов не следует пересматривать при возврате по цепочке согласованных целевых утверждений.

Существуют две причины, побуждающие включать в программу такие указания:

1) программа будет выполняться быстрее, так как не будет тратиться время на попытки найти новые сопоставления для целей, о которых заранее известно, что они не внесут ничего нового в решение;

2) программа может занимать меньше места в памяти, так как отсутствует необходимость запоминать точки возврата для последующего анализа.

В некоторых случаях включение отсечения в программу будет означать переход от правила, которое не работает, к правилу, которое работает.

Отсечение записывается в программе с помощью предиката “!”

Например, в рассматриваемом ранее примере имелось правило:

Sreda (“разбавленная кислота”) :- ask_sreda (X), is(X, _),

conc(“менее 70“), write (“среда = разбавленная кислота“).

Переписав правило с использованием отсечения:

Sreda (”разбавленная кислота”) :- ask_sreda (X), is(X, _), !,

conc(“менее 70“), write (“среда = разбавленная кислота“).

Мы упростим процесс поиска. Если мы не найдём подтверждения факта is, то все остальные условия отсекутся и не потребуется дальнейших действий, тогда как в первоначальном варианте нам требуется проверить согласованность всех условий, входящих в правило.

Существуют различные способы описания того, что произошло с решениями, попавшими в область действия отсечения:

- во-первых, можно сказать, что оставшиеся альтернативы отбрасываются;
- во-вторых, можно смотреть на символ отсечения как на некий разделитель (забор), отделяющий целевые утверждения, например: foo:- a, b, c,!, d, e, f.

При обработке такого правила Пролог без каких-либо ограничений может выполнять возврат среди целей a, b и c до тех пор, пока не будет доказана цель *c*. Это приведет к тому, что Пролог перешагнёт через “забор !” и приступит к доказательству согласованности целевого утверждения d. Далее возврат может осуществляться между целевыми утверждениями d, e, f. При этом возможно, мы достигнем согласованности всех целей. Однако возможно этого не произойдёт и перешагнув “забор” обратно новых попыток доказательства утверждения *c* делаться не будет, а следовательно правило foo потерпит неудачу.

Итак, можно выделить три основных случая использования отсечения:

1. Когда мы хотим указать системе, заданного целевого утверждения, т.е. отсечение означает: если мы дошли до этого места, то выбрали именно то правило, которое нужно для данного целевого утверждения. Примером такого случая может служить задание граничных условий при рекурсии.

2. Когда мы хотим указать системе, что необходимо немедленно прекратить доказательство согласованности конкретного правила, не пытаясь найти для него альтернативные решения, т.е. это означает: если вы дошли до этого места, вам следует прекратить попытки доказать согласованность данного целевого утверждения (пример выше).

3. Когда мы хотим закончить порождение альтернативных решений. В этом случае отсечение означает: если вы дошли до этого места, то вы нашли единственное решение задачи.

Итак, подводя итог, отметим, что программа на языке Пролог включает цель, которую необходимо доказать, используя все имеющиеся факты и правила, т.е. необходимо доказать согласованность цели с базой знаний. В процессе доказательства база знаний просматривается сначала. Если цель сопоставляется с правилом, то требуется доказательство всех утверждений, входящих в это правило.

В случае удачного поиска, мы можем сделать попытку вновь доказать согласованность цели с базой знаний. В этом случае необходимо вернуть в исходное неопределённое состояние каждую переменную, конкретизированную при выборе предыдущего утверждения. Эти действия можно назвать “уничтожением” результатов, полученных ранее при доказательстве согласованности цели. Затем просмотр базы знаний возобновляется, но не сначала, т.е. ранее обнаруженные факты и правила не учитываются. Как и ранее цель, выбранная при возврате, может оказаться согласованной или не согласованной с базой знаний.

## 6.9. ПОРОЖДЕНИЕ МНОЖЕСТВЕННЫХ РЕШЕНИЙ

Простейшая ситуация, в которой некоторые множества факторов допускают несколько ответов на вопрос, возникает, когда в этом множестве имеется несколько фактов, сопоставимых с вопросом.

***Пример.*** На ужин приглашены шесть человек (три юноши и три девушки). Необходимо определить, все возможные пары:

```
predicates
men (symbol).
girl (symbol).
para (symbol, symbol).
goal
para (X, Y).
clauses
men  (garry).
men (pit).
men (nick)
girl (merry).
girl (terry).
para (x, y) :- men(X), girl(y), write(X,’,’, Y ), nl, fail.
```

(nl – встроенный предикат, обозначающий перевод строки).

В данном примере введён предикат – fail, который осуществляет вынужденное неудачное завершение выполнения предиката. Это выражение позволит продолжить поиск, когда найдено решение.

Аналогично, в рассмотренном ранее примере на принадлежность элемента списку также может потребоваться знать, сколько раз элемент повторяется в списке. Мы можем использовать этот же механизм:

```
domains
  L=symbol *
predicates
  yes(symbol, L).
clauses
yes (x,[x|_]) :- write(“yes”).
yes(x,[_|y]) :- yes(x, y).
goal
clearwindow, write(“\ символ ? ”), read(X), yes(X, [a, b, b, d, e, d, d ]), fail.
```

В этом случае, если мы введём символ d , то слово “yes” будет введено три раза, так как символ d трижды встречается в списке.

## 6.10. ФОРМИРОВАНИЕ СОСТАВНЫХ ЦЕЛЕВЫХ УТВЕРЖДЕНИЙ

В правилах вида x:- y, термин “y” может состоять из одного целевого утверждения; либо представлять конъюнкцию целевых утверждений или дизъюнкцию. Существуют предикаты, позволяющие реализовать конъюнкцию или дизъюнкцию.

**1. Конъюнкция целей**

Определяется функтором “,”, т.е. если A и B – целевые утверждения, то целевое утверждение A, B согласуется с базой знаний, если согласуется и A и B. Если A – согласуется , а B не согласуется, то делается попытка найти новое доказательство согласованности для A. Если A не согласуется, то не согласуется и конъюнкция в целом.

**2. Дизъюнкция целей**

Определяется функтором “;”. Если A и B – целевые утверждения то их дизъюнкция: A; B согласуется с базой знаний, если согласуется A или B.

***Пример:***

```
chelovek(X) :- X= “Адам”; X= “ева”;  мать (X, Y).
```

Будем считать, что объект X является человеком, если X – Адам, или X – Ева, или у X есть мать.

# 7. ПРОГРАММНО ЛАБОРАТОРНЫЙ КОМПЛЕКС ДЛЯ АНАЛИЗА ЗНАНИЙ В ЭКСПЕРТНОЙ СИСТЕМЕ НЕЧЁТКОГО ПРИНЯТИЯ РЕШЕНИЙ

В настоящее время большую актуальность приобретает использование экспертных систем для решения объёмных, трудно формализуемых задач в различных предметных областях. Эти задачи характеризуются, как правило, отсутствием или сложностью формальных алгоритмов решения, неполнотой и нечёткостью исходной информации, нечёткостью достигаемых целей. Данные особенности приводят к необходимости использования в процессе решения данных задач знаний, полученных от человека-эксперта в предметной области, и разработки экспертных систем, осуществляющих сбор и управление этими знаниями, принимающими решения об оптимальном способе достижения целей в условиях неполноты и нечёткости. Для выполнения данных задач был разработан программно-лабораторный комплекс.

Для принятия решений используется база знаний, которая содержит факты и знания.

Факты представляют собой информацию о лингвистических переменных (пример в табл. 24) и располагаются во внутренней базе данных экспертной системы (рис. 17).

**Таблица 24**

| Наименование | Терм-множество | Функции принадлежности* | Область определения | Синтаксическая процедура | Семантическая процедура |
|---|---|---|---|---|---|
| Вес ткани, *w* | малый | $\mu_1 = 0{,}5 - 0{,}5\sin\left(\frac{\pi}{15}(w-12{,}5)\right)$ | 5 … 20 | СЛЕГКА НЕ, И, ИЛИ | DIL(1-X), min(X,Y), max(X,Y) |
| | средний | $\mu_2 = 0{,}5 - 0{,}5\sin\left(\frac{\pi}{15}(\lvert w-12{,}5\rvert)\right)$ | | | |
| | большой | $\mu_3 = 0{,}5 + 0{,}5\sin\left(\frac{\pi}{15}(w-12{,}5)\right)$ | | | |
| Температура процесса, *T* | малая | $\mu_1 = \frac{1}{10\,000}(T-200)^2$ | 100 … 150 | И, ИЛИ | min(X,Y), max(X,Y) |
| | средняя | $\mu_2 = 1 - \frac{1}{1000}(T-125)^2$ | | | |
| | большой | $\mu_3 = \frac{1}{10000}(T-50)^2$ | | | |

Знания об использовании фактов и исходных данных представляются в виде нечётких правил в формате *ЕСЛИ…ТО…* . Например:

*1) ЕСЛИ* Вес_ткани = малый *И* Температура_процесса = малая
*ИЛИ* Вес_ткани = малый *И* Температура_процесса = средняя
*ИЛИ* Вес_ткани = средний *И* Температура_процесса = малая
*ТО* Схема = I ;
*2) ЕСЛИ* Вес_ткани = малый *И* Температура_процесса = большая
*ИЛИ* Вес_ткани = средний *И* Температура_процесса = средняя
*ИЛИ* Вес_ткани = большой *И* Температура_процесса = малая
*ИЛИ* Вес_ткани = большой *И* Температура_процесса = средняя
*ТО* Схема = II ;
*3) ЕСЛИ* Вес_ткани = средний *И* Температура_процесса = большая
*ИЛИ* Вес_ткани = большой *И* Температура_процесса = большая
*ТО* Схема = III .

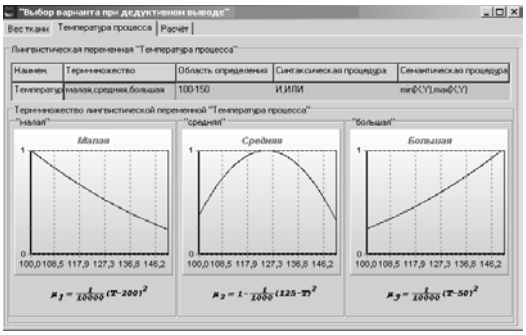



**Рис. 17**

Правила извлекаются от эксперта и располагаются во внешнем текстовом файле данных экспертной системы.

Процедура модуля извлечения знаний предназначена для предоставления экспертных знаний, их структурирования в виде пригодном для использования в компьютерной системе.

В разработанной системе извлечение знаний осуществляется с помощью программных средств, анализирующих грамматику описания знаний – лексического и синтаксического анализаторов, и генератора кода.

Грамматика описания знаний имеет вид:

⟨правило⟩ → {*ЕСЛИ* ⟨условие⟩ *ТО* ⟨ результат⟩ ;}
⟨условие⟩ → ⟨список переменных условия⟩
⟨список переменных условия⟩ → ⟨выражение⟩ {И/ИЛИ ⟨список переменных условия⟩} / (⟨список переменных условия⟩)
⟨результат⟩ → ⟨выражение⟩
⟨ выражение⟩ → ⟨переменная⟩ = ⟨значение⟩

⟨ переменная⟩ → ид.

⟨ значение⟩ → конст.

Лексический анализатор преобразует исходный текст правил с помощью таблиц элементов алфавита в последовательность лексем в виде кода слов, констант и символов.

Исходный текст считывается пословно из файла в строковую переменную, которая анализируется с помощью таблиц слов, констант алфавита и идентификаторов, заданных пользователем. Определяется тип переменной (слово, константа, идентификатор, символ), ищется её место в соответствующей таблице алфавита или, если там она отсутствует, в таблице идентификаторов пользователя. На выходе код каждой переменной записывается в трёхмерный массив (в первую ячейку – номер правила, во вторую ячейку – номер слова в правиле, в третью – тип таблицы и номер в таблице).

Примеры кодировочных таблиц представлены в табл. 25 – 28.

Например, результат работы лексического анализатора для первого правила (в скобках через запятую указаны номер кодировочной таблицы и номер слова в таблице) имеет вид:

***Правило 1***

(1,0) (3,0) (2,0) (4,0) (1,3) (3,1) (2,0) (4,1) (1,2) (3,0) (2,0) (4,0) (1,3) (3,1) (2,0) (4,2) (1,2) (3,0) (2,0) (4,3) (1,3) (3,1) (2,0) (4,1) (1,1) (3,2) (2,0) (4,4) (2,1)

**Таблица 25**

| Зарезервированные слова | |
|---|---|
| Номер | Наименование |
| 0 | ЕСЛИ |
| 1 | ТО |
| 2 | ИЛИ |
| 3 | И |

**Таблица 26**

| Символы | |
|---|---|
| Номер | Наименование |
| 0 | = |
| 1 | ; |
| 2 | . |
| 3 | ( |
| 4 | ) |

**Таблица 27**

| Идентификаторы | |
|---|---|
| Номер | Наименование |
| 0 | Вес_ткани |
| 1 | Температура_процесса |
| 2 | Схема |

**Таблица 28**

| Константы | |
|---|---|
| Номер | Наименование |
| 0 | малый |
| 1 | малая |
| 2 | средняя |
| 3 | средний |
| 4 | I |
| 5 | Большая |
| 6 | Большой |
| 7 | II |
| 8 | III |

Синтаксический анализатор, используя метод операторного предшествования, преобразует последовательность лексем во внутреннее представление знаний. В разработанной системе в качестве внутреннего представления используется постфиксная запись. Матрица отношений предшествования приведена в табл. 29.

**Таблица 29**

| | **ЕСЛИ** | **=** | **И** | **ИЛИ** | **ТО** | **;** | **.** | **ИД** | **КОНСТ** | **(** | **)** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **ЕСЛИ** | | < | < | < | = | | | < | < | | |
| **=** | | | > | > | > | > | > | | < | | > |
| **И** | < | < | | > | > | | | < | | < | |
| **ИЛИ** | < | < | < | > | > | | | < | | | > |
| **ТО** | | < | | | | > | > | < | | | |
| **;** | | | | | | | | | | | |
| **.** | | | | | | | | | | | |
| **ИД** | | > | | | | | | | | | |
| **К** | | | > | > | > | > | > | | | | > |
| **(** | | < | | < | | | | < | | | = |
| **)** | | | | | > | | | | | | |

Тогда, например правило 1 в результате работы метода операторного предшествования примет вид:

***Правило 1***

Внутреннее представление:

(3 0) (4 0) (2 0) (3 1) (4 1) (2 0) (1 3) (3 0) (4 0) (2 0) (3 1) (4 2) (2 0) (1 3) (1 2) (3 0) (4 3) (2 0) (3 1) (4 1) (2 0) (1 3) (1 2) (3 2) (4 4) (2 0) (1 0) (1 1)

На исходном языке:

Вес_ткани малый = Температура_процесса малая = *И* Вес_ткани малый = Температура_процесса средняя = *И ИЛИ* Вес_ткани средний = Температура_процесса малая = *И ИЛИ* Схема I = *ЕСЛИ ТО*

Генератор кода считывает внутреннее представление знаний и формирует объектный код в виде описания на языке экспертной системы. В разработанной системе реализована дедуктивная схема вывода на основе правила modus ponens (см. раздел 4.6).

Использование постфиксной записи в качестве внутреннего представления позволяет упростить процесс принятия решения. На рисунке 18 приведён алгоритм обработки знаний, представленных в постфиксной записи.

В результате работы процедуры генерации кода формируется массив, содержащий значения степеней истинности правила modus ponens для каждого правила системы нечётких высказываний. В качестве решения выбирается такое значение выходного параметра, для которого степень истинности имеет максимальное значение.


**Рис. 18**

В ходе работы процедуры принятия решения формируется система объяснений. Она предназначена для показа пользователю всего процесса рассуждений, в результате которого было найдено или не найдено решение. Пользователь получает возможность просмотреть все логические выводы, которые привели к результату.

Разработанная система позволяет просмотреть и отредактировать данные о лингвистических переменных: наименование, терм-множество, область определения, синтаксические и семантические процедуры, функции принадлежности.

Предусмотрен ввод исходных данных лингвистических переменных из области определения и отображение соответствующих значений функций принадлежности.

Результаты работы механизма логического вывода отображаются на экране.

Экспертная система позволяет просмотреть грамматику описания языка знаний, теоретический материал, систему объяснений работы механизма логического вывода.

## ЗАКЛЮЧЕНИЕ

Использование методов искусственного интеллекта становится всё более актуальным для синтеза проектных решений. В связи с этим необходимо правильно извлечь знания от эксперта; формализовать эти знания; представить в виде, пригодном для дальнейшего использования; применить знания для решения конкретных задач. Этим вопросам и посвящено данное учебное пособие.

В пособии представлены методы принятия решений для различных способов представления знаний. Приведено краткое описание языка Пролог, который является наиболее распространённым языком программирования для систем искусственного интеллекта. В качестве примера рассмотрен учебно-лабораторный комплекс для анализа знаний в условиях нечётко-заданной экспертной информации.

Все рассмотренные вопросы будут полезны студентам и магистрантам, изучающим дисциплины “Интеллектуальные подсистемы САПР”, “Системы искусственного интеллекта”, “Нейросетевые технологии в САПР”, а также использующим методы искусственного интеллекта в выпускных квалификационных работах.

## СПИСОК ЛИТЕРАТУРЫ


1. Искусственный интеллект : В 3-х кн. Кн. 2. Модели и методы: справочник / под ред. Э.В. Попова. – М. : Радио и связь, 1990. – 303 с.
2. Левин, Р. Практическое введение в технологию искусственного интеллекта и экспертных систем с иллюстрациями на Бейсике / Р. Левин, Д. Дранг, Б. Эделсон. – М. : Финансы и статистика, 1990. – 239 с.
3. Нильсон, Н. Принципы искусственного интеллекта / Н. Нильсон. – М. : Радио и связь. 1990. – 376 с.
4. Экспертные системы в САПР : лаб. работы / сост. : А.А. Кузнецов, О.П. Федосов. – Тамбов : Изд-во Тамб. гос. техн. ун-т, 1995. – 33 с.
5. Методы представления знаний : метод. указ. / сост. И.Л. Коробова. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2003. – 30 с.
6. Основы теории нечётких множеств : метод. указания / сост. И.Л. Коробова. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2003. – 30 с.
7. Малышев, Н.Г. Нечёткие модели для экспертных систем в САПР / Н.Г. Малышев, Л.С. Берштейн, А.В. Боженюк. – М. : Энергоатомиздат, 1991. – 264 с.
8. Горбань А.Н., Дунин-Барковский В.Л., Кирдин Н., Миркес Е.М., Новоходько А.Ю., Россиев Д.А., Терехов С.А., Сенашова М.Ю. Нейроинформатика. <http//www.bmstu.ru//facult/iu/iu4/rus/stst/book2/ann.htm>
9. Вопросы приближения функций : метод. указ. / сост. : Ю.В. Литовка, А.В. Романенко, И.Л. Коробова. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2001. – 20 с.
10. Заде, Л. Основы нового подхода к анализу сложных систем и процессов принятия решения / Л. Заде // Математика сегодня : сб. ст. – М. : Знание, 1974. – 48 с.
11. Zimmerman H. J. Fuzzy Set Theory and its Applications. – Boston etc. 1992.
12. Кафаров, Б.Б. Системный анализ процессов химической технологии. Применение метода нечётких множеств / Б.Б. Кафаров, И.Н. Дорохов, Е.П. Марков. – М. : Наука, 1986.
13. Кофман, Л. Введение в теорию нечётких множеств / Л. Кофман. – М. : Радио и связь, 1982.
14. Нечёткие множества в моделях управления и искусственного интеллекта / под ред. Д.А. Поспелова. – М. : Наука, 1986.
15. Вощинин, А.П. Оптимизация в условиях неопределенности / А.П. Вощинин, Г.Р. Сотиров. – Изд-во : МЭИ (Россия), Техника (НРБ), 1990.

## Оглавление