МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Ю. А. Загорулько, Г. Б.Загорулько

ИНЖЕНЕРИЯ ЗНАНИЙ

Учебное пособие

Новосибирск
2016

УДК 004.891(075.8) + 004.82(075.8)
ББК 32.813.5я 73-1
З-143

Рецензент
к.ф.-м.н., доц. *Ф. А. Мурзин*

Издание подготовлено в рамках реализации *Программы развития государственного образовательного учреждения высшего профессионального образования «Новосибирский государственный университет» на 2009–2018 годы.*

З-143 Загорулько, Ю. А., Загорулько, Г. Б.
Инженерия знаний : учеб. пособие. / Ю. А. Загорулько, Г. Б. Загорулько ; Новосиб. гос. ун-т. – Новосибирск : РИЦ НГУ, 2016. – 93 с.

ISBN 978-5-4437-0452-4

Пособие посвящено инженерии знаний – научной дисциплине, включающей в круг изучения научные, технологические и методологические вопросы создания программных систем, основанных на знаниях. В пособии рассматриваются основные модели и средства извлечения, представления, структурирования и использования знаний. В связи с большой практической значимостью экспертных систем подробно описываются принципы их построения, архитектура и технология их разработки.

Данное учебное пособие рекомендуется студентам, изучающим курс «Инженерия знаний», а также курсы, в которых изучаются модели, методы и системы искусственного интеллекта.

УДК 004.891(075.8)+004.82(075.8)
ББК 32.813.5я 73-1

ISBN 978-5-4437-0452-4

1. Введение

В начале 80-х гг. XX в. в рамках исследований по искусственному интеллекту сформировалось самостоятельное направление – "инженерия знаний", в задачу которого входят разработка, исследование и использование экспертных систем (ЭС).

Огромный интерес к экспертным системам (ЭС) был вызван следующими основными причинами. Во-первых, ЭС ориентированы на решение широкого круга задач в неформализованных областях, т. е. на приложения, которые ранее считались малодоступными для вычислительной техники. Во-вторых, экспертные системы позволяют специалистам, не имеющим навыков программирования, создавать практически значимые приложения, что резко расширяет сферу использования вычислительной техники. В-третьих, экспертные системы при решении практических задач позволяют получать результаты, сравнимые, а иногда и превосходящие те, которые может получить эксперт-человек. В-четвертых, экспертные системы легко объединяются с традиционными программными системами (системами управления базами данных, системами поддержки принятия решений и т. д.) в интегрированные приложения.

1.1. Введение в инженерию знаний

Название рассматриваемой в этом учебнике дисциплине дал Е. Фейгенбаум [32], который определил задачу «инженерии знаний» (knowledge engineering) как «привнесение принципов и инструментария исследований из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов».

Инженерия знаний как научная дисциплина включает в круг изучения собственно научные, технологические и методологические вопросы создания программных систем, основанных на знаниях. Именно к такому классу систем относятся экспертные системы.

Таким образом, предметом инженерии знаний является разработка программных систем, которые при решении задач, трудных для эксперта-человека, получают результаты, не уступающие по качеству и эффективности решениям, получаемым экспертом.

Вычислительные возможности каждой экспертной системы определяются в первую очередь ее наращиваемой базой знаний (БЗ) и только во вторую очередь используемыми в ней методами.

Так как база знаний – главный компонент ЭС, источник ее мощности, то приобретение и представление знаний в БЗ ЭС является важнейшей частью процесса построения ЭС.

При построении базы знаний ЭС инженер знаний должен найти ответы на два взаимосвязанных вопроса:

1) ЧТО представлять?

2) КАК представлять?

Вопрос «ЧТО представлять?» определяется сутью решаемых задач, т. е. нужно определить, какие знания необходимы системе для решения поставленных перед ней задач.

Вопрос «КАК представлять?» – это вопрос о выборе способа представления знаний: какой выбрать формализм и как представить знания в выбранном формализме.

Остановимся подробнее на решении этих вопросов в следующем разделе.

1.2. Проблемы представления знаний

Проблема представления знаний является одной из самых старых проблем не только в искусственном интеллекте, но и в науке и образовании в целом. Она решается всякий раз, когда требуется передать кому-то знания и научить ими пользоваться.

Как уже говорилось выше, при решении данной проблемы возникают два вопроса: "ЧТО представлять?" и "КАК представлять?". Если ответ на первый вопрос в основном определяется той конкретной задачей (классом задач), которую мы хотим решить, то с ответом на второй вопрос дело обстоит гораздо сложнее.

Главная трудность здесь, по-видимому, состоит в том, что при решении этого вопроса приходится сталкиваться с различными противоречиями. Так, например, оказывается, что средства представления, удобные для человека, неэффективно реализуются на ЭВМ, а средства, эффективно реализуемые на ЭВМ, не всегда удобны для человека. Кроме того, для представления одних видов знаний больше подходят одни модели и средства, для других – другие.

В связи с этим разработано много различных моделей и средств представления знаний. Более того, они продолжают разрабатываться. Несмотря на такое многообразие средств представления знаний, их можно классифицировать, выделив три основные модели представления знаний: логическую, сетевую и продукционную.

2. Логическая модель представления знаний

Одним из достаточно известных средств представления знаний является аппарат формальной логики. В логике разрабатываются методы правильных рассуждений, представляющих собой цепь умозаключений в логически последовательной форме. Рассуждения в ней изучаются с точки зрения формы, а не смысла, и с этой целью в обычных рассуждениях выделяются определенные элементы, которые могут замещаться в них произвольным образом какими-то другими элементами.

Например, в хорошо известном силлогизме «Люди – смертны, Сократ – человек, следовательно, Сократ смертен», приписываемом Аристотелю, в обоих случаях вхождения слов «человек (люди)», «смертен (смертны)», «Сократ» могут быть заменены любыми другими словами, и при этом само рассуждение останется формально допустимым. Таким образом, уже простая замена таких слов в рассуждениях символьными обозначениями позволяет строить обобщенные суждения на основе подобных рассуждений. Так, в приведенном примере абстрактная модель этого рассуждения принимает следующий вид:

«Если все *x* суть *y* и если *z* есть *x*, то *z* есть *y*».

2.1. Базовые понятия

Основное преимущество базирующихся на логике формализмов состоит в том, что с их помощью проще обеспечить корректность структур и решений системы, чем с помощью других способов представления. Решаемая задача записывается в виде утверждений некоторой формальной системы.

Формальная система представляет собой совокупность абстрактных объектов (обычно обозначаемых символами) и правил оперирования множеством таких символов в чисто синтаксической трактовке без учета их семантики (или смыслового содержания) [14].

Формальная система определена, если:

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

2) определена процедура построения правильных формул β;

3) выделено некоторое множество формул $\boldsymbol{A}$, называемых аксиомами;

4) задано конечное множество правил вывода $\boldsymbol{P}$, которые позволяют получать из некоторого конечного множества формул другое множество формул.

Справедливость утверждения (или формулы) $\boldsymbol{F}$ устанавливается или опровергается на основании аксиом $\boldsymbol{A}$ и правил вывода $\boldsymbol{P}$ этой формальной системы.

На практике в качестве формальной системы обычно используется исчисление предикатов первого порядка.

В последнее время для представления знаний стали использовать те или иные подмножества дескриптивной логики, которая положена в основу популярного в настоящее время языка описания Web-онтологий – OWL.

С помощью языка OWL описываются информационные ресурсы и документы, представленные в сети Интернет. Благодаря тому, что OWL базируется на логике, оказывается возможным вывод фактов, не представленных явно в сети Интернет.

2.2. Исчисление предикатов первого порядка

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

Для построения атомов разрешается использовать следующие четыре типа символов:

1) **Индивидные символы, или константы**. Это обычно имена объектов такие, как *Мэри*, *Джон*, или числа *3, 5* и др.

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

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

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

Всякая функция, или функциональный символ, имеет определенное число аргументов.

Если функциональный символ *f* имеет *n* аргументов, то *f* называется *n-местным функциональным символом.* (Индивидный символ, или константа, может рассматриваться как функциональный символ без аргументов.)

Аналогично, если предикатный символ *P* имеет *n* аргументов, то *P* называется *n-местным предикатным символом.* Например, *отец* – одноместный функциональный символ, а *БОЛЬШЕ* и *ЛЮБИТ* – двухместные предикатные символы.

Функция есть отображение, которое отображает список констант в данную константу. Например, *отец* – функция, которая отображает человека по имени Джон в человека, который есть отец Джона. Следовательно, *отец* (Джон) представляет человека, даже если его имя неизвестно. В логике первого порядка мы называем выражение *отец (Джон)* термом.

Определение 2.1. *Термы* определяются рекурсивно следующим образом:

1) Константа есть терм.

2) Переменная есть терм.

3) Если f есть n-местный функциональный символ и $t_1, \ldots, t_n$ – термы, то $f(t_1, \ldots, t_n)$ – терм.

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

Так как *x* и *1* – термы, а *плюс* – двухместный функциональный символ, то *плюс (x, 1)* есть терм согласно приведенному определению.

Легко видеть, что *плюс (плюс (x, 1), x)* и *отец (отец (Джон))* – также термы; первый обозначает *((x+1)+x),* а второй – дедушку Джона.

Предикат есть отображение, которое отображает список констант в *И* (истину) или *Л (*ложь*).*

Например, если *БОЛЬШЕ* есть предикат, то *БОЛЬШЕ (*5,3*)* есть *И,* но *БОЛЬШЕ (1, 3)* есть *Л.*

Определив термы, мы можем теперь дать формальное определение атома логики первого порядка.

Определение 2.2. Если P – n–местный предикатный символ и $t_1, \ldots, t_n$ – термы, то $P(t_1, \ldots, t_n)$ – атом *(*или атомарная формула*).*

Для построения формул мы можем использовать пять логических связок, приведенных выше, а также два специальных символа $\forall$ и $\exists$, чтобы характеризовать переменные. Символы $\forall$ и $\exists$ называются соответственно *кванторами (все) общности* и *существования.*

Если *x* – переменная, то $(\forall x)$ читается как *«для всех x», «для каждого x»* или *«для всякого x»*, тогда как $(\exists x)$ читается *«существует x», «для некоторых x»* или *«по крайней мере для одного x».*

Определение 2.3. *Правильно построенные формулы (п.п.ф.)* или *формулы* логики первого порядка рекурсивно определяются следующим образом:

1) Атом есть формула.

2) Если *F* и G – формулы, то $\neg(F)$, $(F \vee G)$, $(F \wedge G)$, $(F \rightarrow G)$ и $(F \leftrightarrow G)$ – формулы.

3) Если *F* – формула, а *x* – свободная переменная в *F*,

то $(\forall x)F$ и $(\exists x)F$ – формулы.

4) Формулы порождаются только конечным числом применений правил 1), 2) и 3).

Пример 2.1

Переведем утверждение «Каждый человек смертен. Конфуций – человек. Следовательно, Конфуций смертен» в формулу.

Обозначим «*x* есть человек» через *ЧЕЛОВЕК (x)* и *«x* смертен» через *СМЕРТЕН (x).* Тогда утверждение «каждый человек смертен» может быть представлено формулой

$$(\forall x)\,(\text{ЧЕЛОВЕК}\,(x) \rightarrow \text{СМЕРТЕН}\,(x)),$$

а утверждение «Конфуций – человек» – формулой

ЧЕЛОВЕК (Конфуций)

и «Конфуций смертен» – формулой

СМЕРТЕН (Конфуций).

Утверждение в целом теперь может быть представлено формулой

$$(\forall x)\,(\text{ЧЕЛОВЕК}\,(x) \rightarrow \text{СМЕРТЕН}\,(x)) \wedge \text{ЧЕЛОВЕК (Конфуций)} \rightarrow \text{СМЕРТЕН (Конфуций).}$$

Формальное доказательство определяется как конечная последовательность формул *F1, F2, ..., Fr* такая, что каждая формула *Fi* либо является аксиомой, либо выводима при помощи одного из правил вывода из предшествующих ей формул *Fj*, где $j < i$.

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

В частности, всякая аксиома является теоремой.

Использование исчисления предикатов для представления знаний будет эффективным в том случае, если существует средство для автоматического доказательства теорем [12, 30].

Формальные системы (ФС) всегда представляют собой модель какой-то реальности (либо конкретной, либо математической). **Интерпретация** представляет собой распространение исходных положений формальной системы на реальный мир. Интерпретация придает смысл каждому символу ФС и устанавливает взаимно однозначное соответствие между символами ФС и реальными объектами. Теоремы ФС, будучи однажды интерпретированы, становятся после этого **утверждениями** в обычном смысле слова, и в этом случае уже можно делать выводы об их *истинности* или *ложности*.

Если некоторая формула истинна при всех интерпретациях, то ее называют **общезначимой**.

Если формула содержит кванторы, общезначимость или ее отсутствие не всегда можно установить. Если некоторое множество формул не удовлетворяется ни при какой интерпретации, то оно называется **противоречивым** (или **невыполнимым**).

Первый вопрос, возникающий при задании формальной системы, состоит в том, чтобы определить, является данная формула общезначимой или нет, и как это доказать. В математике предполагается, что при задании формальной системы существует хорошо определенный способ действий,

который за конечное число шагов позволит получить ответ на данный вопрос. Такой способ, если он существует, называется **процедурой решения**, а соответствующую формальную систему называют **разрешимой**. Однако основная трудность заключается в том, что такие процедуры существуют далеко не всегда даже для таких простых теорий, как исчисление предикатов первого порядка.

2.3. Метод резолюций

В 1930 г. Ж. Эрбран предложил метод доказательства теорем в формальных системах первого порядка [20, 29], идея которого состоит в следующем: чтобы получить некоторое заключение *C* исходя из гипотез *H1, H2, ..., Hn*, т. е. чтобы доказать теорему

$$T: H1 \wedge H2 \wedge ... \wedge Hn \rightarrow C,$$

достаточно доказать противоречивость формулы

$$F: H1 \wedge H2 \wedge ... \wedge Hn \wedge \neg C,$$

в которой отрицание заключения добавлено к гипотезам.

Такое доказательство может быть существенно проще прямого вывода.

Был разработан метод доказательства теорем, базирующийся на описанной выше идее, который получил название **метод резолюций**.

Кроме того, было доказано, что метод резолюций является исчерпывающим методом доказательства теорем [29]. Это значит, что если формула противоречива, то с помощью метода резолюции всегда можно обнаружить это противоречие.

Для более легкого усвоения материала мы сначала рассмотрим основные положения метода резолюций на примере логики высказываний, а затем распространим эти положения на логику первого порядка.

Определение 2.4. Резольвента.

Введем понятие резольвенты**.** Для этого рассмотрим следующие дизъюнкты (предложения):

$C1: P$

$C2: \neg P \vee Q.$

Тогда, $C3: Q$ есть *резольвента* дизъюнктов $C1$ и $C2$, полученная как дизъюнкция $C1 \vee C2$, из которой вычеркнуты контрарные пары (P и $\neg P$).

Пример 2.2

1)	2)	3)
$P \vee R$	$\neg P \vee Q \vee R$	$\neg P \vee Q$
$\neg P \vee Q$	$\neg Q \vee S$	$\neg P \vee R$
---	---	---
$R \vee Q$	$\neg P \vee R \vee S$	нет резольвенты

Зачем же понадобилось вводить понятие резольвенты? Оказывается, резольвента обладает одним замечательным свойством, о котором говорится в следующей теореме.

Теорема 1

Пусть даны два дизъюнкта *C1* и *C2*. Тогда резольвента *C* дизъюнктов *C1* и *C2* есть *логическое следствие C1* и *C2*.

Благодаря тому, что резольвента является логическим следствием своих родителей-дизъюнктов, можно ввести понятие резолютивного вывода.

Определение 2.5. Резолютивный вывод.

Пусть *S* – множество дизъюнктов. *Резолютивный вывод C* из *S* есть такая конечная последовательность дизъюнктов *C1, C2, … Ck* , что каждый *Ci* либо принадлежит *S*, либо является резольвентой дизъюнктов, предшествующих *Ci* , и *C = Ck* .

Вывод □ (противоречия) из *S* называется опровержением (доказательством невыполнимости) *S* .

Говорят, что *C* выводимо из *S*, если существует вывод *C* из *S* .

Пример 2.3

Пусть $S = \{\neg P \vee Q, \neg Q, P\}$.

Дизъюнкты $\neg P \vee Q$ и $\neg Q$ дают $\neg P$,

а P и $\neg P$ дают □ – противоречие.

Значит, множество *S* невыполнимо (противоречиво).

Покажем, как метод резолюций может использоваться для доказательства теорем в исчислении высказываний.

Пример 2.4

Пусть даны 4 утверждения:

(1) $P \rightarrow S$

(2) $S \rightarrow U$

(3) P

(4) U

Докажем путем опровержения, что *U* есть логическое следствие формул (1), (2) и (3). Для этого возьмем *U* с отрицанием и получим следующее множество утверждений:

(1) $P \rightarrow S$		(1) $\neg P \vee S$
(2) $S \rightarrow U$	*<преобразуем в*	(2) $\neg S \vee U$
(3) P	*стандартную форму>*	(3) P
(4) $\neg U$		(4) $\neg U$

Далее выполним следующий резолютивный вывод:

(1) $\neg P \vee S$

(2) $\neg S \vee U$

(3) P

(4) $\neg U$

(5) S (резольвента (3) и (1))

(6) U (резольвента (5) и (2))

(7) $\square$. (резольвента (6) и (4))

Таким образом, мы получили противоречивое множество дизъюнктов. Значит, предположение о том, что имеет место $\neg U$, неверно. Следовательно U есть логическое следствие формул (1), (2) и (3).

2.4. Использование метода резолюции для доказательства теорем в логике первого порядка

При доказательстве теорем в логике первого порядка необходимо использовать преобразование формулы в нормальную форму и подстановку (или унификацию) переменных.

Рассмотрим следующий пример.

Пример 2.5

Дано:

(1) $(\forall x)$ *(Человек(x)* $\rightarrow$ *Смертен(x))*

(2) *Человек(Сократ)*

Доказать:

(3) *Смертен(Сократ)*

Доказательство снова будем вести путем опровержения. Для этого предположим, что верно $\neg$ *Смертен(Сократ)*:

(1) $\neg$ *Человек(x)* $\vee$ *Смертен(x)*

(2) *Человек(Сократ)*

(3) $\neg$ *Смертен(Сократ)*

Выполнив подстановку θ = *{x/Сократ}*, получим

(1) $\neg$ *Человек(Сократ)* $\vee$ *Смертен(Сократ)*

(2) *Человек(Сократ)*

(3) $\neg$ *Смертен(Сократ)*

Далее получаем следующую последовательность резольвент:

(4) *Смертен(Сократ)* (из (1) и (2))

(5) $\square$ *– противоречие* (из (4) и (3))

Мы получили противоречие, а это значит, что наше предположение о том, что верно утверждение $\neg$ *Смертен(Сократ),* неверно. Следовательно, верно утверждение *Смертен(Сократ).*

Выше было показано, как можно применить метод резолюции к конкретной формуле исчисления предикатов первого порядка. Для того чтобы применить метод резолюции к произвольному множеству формул, требуется представить эти формулы в специальном виде – в виде предложений.

Любую формулу исчисления предикатов можно представить в виде предложений, применяя к ней следующую последовательность операций.

1. Стандартизация переменных.

В области действия любого квантора переменную, связываемую им, можно заменить другой переменной, и это не приведет к изменению значения истинности формулы.

Стандартизация переменных означает их переименование с тем, чтобы каждый квантор имел свою собственную переменную.

Так, вместо

$$(\forall x)\{P(x) \rightarrow (\forall x)\, Q(x)\}$$

следует написать

$$(\forall x)\{P(x) \rightarrow (\forall y)\, Q(y)\}.$$

2. Исключение знаков импликации.

На этом шаге используется подстановка:

$\neg A \vee B$ вместо $A \rightarrow B$,

которая применяется столько раз, сколько необходимо.

3. Уменьшение области действия знаков отрицания.

Необходимо преобразовать формулу таким образом, чтобы знак отрицания ($\neg$) применялся не более чем к одному предикатному символу.

Для этого применяются следующие подстановки:

$\neg A \vee \neg B$ вместо $\neg (A \wedge B)$

$\neg A \wedge \neg B$ вместо $\neg (A \vee B)$

A вместо $\neg \neg A$

$(\exists x)\{\neg A\}$ вместо $\neg (\forall x)A$

$(\forall x)\{\neg A\}$ вместо $\neg (\exists x)A$

4. Исключение кванторов существования.

Рассмотрим формулу $(\forall y \exists x)\{P(x,y)\}$, которую можно интерпретировать следующим образом: «Для всех y существует такой x (возможно, зависящий от y), что x больше y».

Пусть эта зависимость определяется явно с помощью некоторой функции $g(y)$, отображающей каждое значение y в x, который «существует». Такая функция называется **функцией Сколема**.

Если вместо x, который «существует», взять функцию Сколема, то можно исключить квантор существования: $(\forall y)\{P(g(y),y)\}$.

Функциональные символы для функций Сколема должны быть «новыми», т. е. не должны совпадать с теми символами, которые уже имеются в формуле.

Заметим, что при исключении кванторов существования переменные, не связанные никакими кванторами всеобщности, заменяются 0-местными функциями Сколема (т. е. константными символами).

Если в описанной выше формуле поменять местами кванторы, то получим формулу $(\exists x \forall y)\ \{P(x, y)\}$, где переменная x связана только квантором существования, поэтому она может быть заменена на константу a.

В результате получим: $(\forall y)\ \{P(a, y)\}$.

5. Приведение к предваренной нормальной форме.

Теперь можно перенести все кванторы всеобщности в начало формулы и считать областью действия каждого квантора всю часть формулы, расположенную за ним.

Про полученную в результате формулу говорят, что она имеет **предваренную нормальную форму**.

Предваренная нормальная форма имеет вид

$(\forall x \forall y \forall z)\{F(x, y, z)\ldots\}$.

6. Приведение к конъюнктивной нормальной форме.

Пользуясь законом дистрибутивности операций конъюнкции и дизъюнкции, любую формулу можно представить в виде конъюнкции конечного множества дизъюнкций предикатов и (или) их отрицаний.

Говорят, что такая формула имеет **конъюнктивную нормальную форму.** Она выглядит следующим образом:

$(\forall x \forall y \forall z)\{F1(x,y,z) \wedge F2(x,y,z) \wedge \ldots\}$.

7. Исключение кванторов всеобщности.

Так как все переменные, оставшиеся на данном этапе, относятся только к кванторам всеобщности и порядок расположения этих кванторов несущественен, то их можно явным образом не указывать.

Теперь формула имеет вид

$F1(x, y, z) \wedge F2(x, y, z) \wedge \ldots$

8. Исключение связок «и».

Теперь можно исключить знак $\wedge$, заменяя $A \wedge B$ двумя формулами A, B. Результатом многократной замены будет конечное множество формул, каждая из которых представляет собой дизъюнкцию предикатных символов или их отрицаний. (Такие формулы называются предложениями.)

Покажем теперь на примере, как можно эффективно использовать метод резолюции для доказательства теорем.

Пример 2.6

Пусть нужно доказать, что из посылки «Студенты суть граждане» должно следовать заключение «Голоса студентов суть голоса граждан».

Итак, мы имеем следующее:

Посылка: Студенты суть граждане.

Заключение: Голоса студентов суть голоса граждан.

Пусть предикаты *S(x), C(x)* и *V(x, y)* означают «*x* – студент», «*x* – гражданин» и «*x* есть голос *y*» соответственно. Тогда, посылка и заключение запишутся следующим образом

$(\forall y)\{S(y) \to C(y)\}$ (посылка),

$(\forall x)\{(\exists y)\{S(y) \wedge V(x,y)\} \to (\exists z)\{C(z) \wedge V(x,z)\}\}$ (заключение).

Преобразуя посылку в предложение, получим:

$\neg S(y) \vee C(y)$. (п.1)

Далее, отрицая заключение, получаем:

$\neg\{(\forall x)\{(\exists y)\{S(y) \wedge V(x, y)\} \to (\exists z)\{C(z) \wedge V(x, z)\}\}\} =$

$= \neg\{(\forall x)\{(\forall y)\{\neg S(y) \vee \neg V(x, y)\} \vee (\exists z)\{C(z) \wedge V(x, z)\}\}\} =$

$= \neg\{(\forall x)(\forall y)(\exists z)\{\neg S(y) \vee \neg V(x, y) \vee \{C(z) \wedge V(x, z)\}\}\} =$

$= (\exists x)(\exists y)(\forall z)\{S(y) \wedge V(x, y) \wedge \{\neg C(z) \vee \neg V(x, z)\}\}.$

«Рассыпая» полученную формулу, получим три предложения:

$S(b)$ (п.2),

$V(a,b)$ (п.3),

$\neg C(z) \vee \neg V(a, z)$ (п.4).

Наконец, доказательство методом резолюций выглядит следующим образом

$C(b)$ из (п.1) и (п.2),

$\neg V(a ,b)$ из (п.5) и (п.4),

□ (противоречие) из (п.6) и (п.3).

Таким образом, мы предположили, что заключение неверно, и пришли к противоречию. Значит, наше предположение оказалось неверным и из указанной выше посылки следует данное заключение.

Впервые метод резолюции был реализован на ЭВМ Дж. Робинсоном в 1963 г.

В 1970-е гг. был разработан язык логического программирования ПРОЛОГ, в основу которого наряду с бэктрекингом был положен метод резолюции. Это позволило определять задачи в форме исчисления предикатов первого порядка [11]. В настоящее время существуют многочисленные версии языка ПРОЛОГ, которые завоевали популярность как языки разработки интеллектуальных систем.

Контрольные вопросы

1. Назовите основные компоненты формальной системы.
2. В чем суть метода резолюции?
3. Что такое резольвента? Логический смысл резольвенты.
4. Что такое резолютивный вывод?

3. Сетевая модель

В основе сетевых моделей лежит понятие сети. В этих моделях в явной форме выделены все отношения, составляющие каркас знаний моделируемой предметной области, и учитывается их семантика. К этому классу моделей относятся семантические сети, функциональные сети и фреймы (фрейм-представление).

3.1. Семантическая сеть

Определение 3.1.

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

На рис. 3.1 приведен пример небольшой семантической сети.

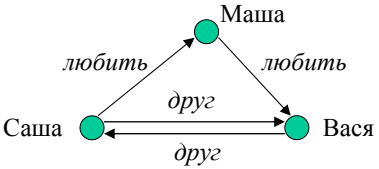

Рис. 3.1. Пример семантической сети

Более строгое определение семантической сети с использованием терминологии, принятой в реляционной модели данных, дано в [13] В. С. Лозовским. Приведем его.

Определение 3.2.

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

$INT(R_i) = \{\dots <A_j, DOM(A_j)> \dots\}$,

где R_i – имя отношения местности n_i, $A_j \in A$, $j = 1, \dots, n_i$ – атрибуты отношения R_i, $DOM(A_j)$ – домен атрибута A_j (множество значений атрибута A_j отношения R_i).

Экстенсионалом отношение R_i будем называть множество

$EXT(R_i)=\{\dots F_k. \dots\}$, $k= 1, \dots, p_i$,

где p_i – кардинальность множества $EXT(R_i)$, F_k – факты отношения R_i, записываемые в виде

$F_k = M_k : R_i (A_1 v_{i1k} \dots A_j v_{ijk} \dots)$,

где M_k – метка факта, $v_{ijk} \in DOM(A_j)$ – значение атрибута A_j.

Порядок записи пар «атрибут-значение» (атрибутивных пар) и фактов роли не играет. Все факты и атрибутивные пары внутри каждого факта попарно различны.

Тогда, семантическая сеть есть совокупность пар

{…< INT(R_i) EXT(R_i) >…} (для $i = 1, \ldots, m$),

где m – число различных отношений R_i.

Введенное выше понятие семантической сети естественным образом распадается на *интенсиональную {… INT(R_i) …}* и *экстенсиональную {… EXT(R_i) …}* семантические сети.

Большинство известных моделей семантических сетей с той или иной степенью точности описывается моделью, заданной Определением 3.2, поэтому примем эту модель за базовую и с учетом этого рассмотрим классификацию семантических сетей.

В зависимости от вида используемых вершин и дуг сети делятся на различные классы.

По виду вершин различают (1) простые и (2) иерархические семантические сети (рис. 3.2).

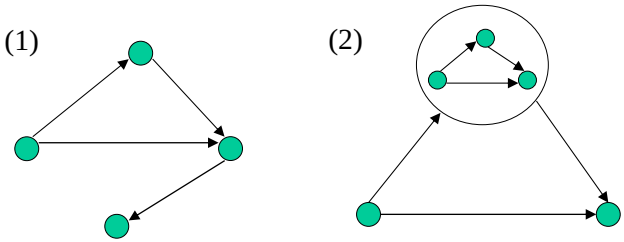

Рис. 3.2. Простая (1) и иерархическая (2) семантическая сеть

Сети, включающие вершины, не имеющие внутренней структуры, называются **простыми**. Если сеть содержит вершины, обладающие некоторой структурой, например, в виде сети (процесс этот можно рекурсивно продолжать), то она называется **иерархической** [28].

Частным случаем иерархических сетей являются **объектно-ориентированные сети,** вершинами которых являются объекты, обладающие некоторыми свойствами (например, имеющие атрибуты).

Рассмотрим теперь классификацию семантических сетей по типу дуг.

Сети, в которых используются только бинарные отношения, называются **бинарными**.

Бинарные сети, в свою очередь, можно разбить ещё на два класса: *семантически-бинарные* и *синтаксически-бинарные*.

Семантически-бинарные сети строятся только на основе бинарных отношений, семантика которых определяется конкретной предметной областью.

В синтаксически-бинарных сетях отношение выступает в качестве синтаксической конструкции, поэтому допускается использование бинарных отношений с произвольной семантикой. Выразительная сила такой сети в точности эквивалентна сети, данной в Определении 3.2 [13].

Изобразительные возможности семантически-бинарных сетей значительно уже, так как они, например, не позволяют представлять многоместные отношения.

Если отношения между вершинами сети одинаковы, то такая сеть называется **однородной**, в противном случае – **неоднородной.**

Наиболее часто используемым видом однородных сетей являются **сценарии**. В сценариях в качестве единственного отношения выступает отношение нестрогого порядка [31]. Его семантика может быть различной. Это может быть каузальное отношение, временное отношение следования, классифицирующее отношение типа *"род-вид"* или *"элемент-класс"* и т. п. Сценарии, как и любые другие сети, могут быть простыми и иерархическими (вложенными). В системах искусственного интеллекта они часто служат для формирования допустимых планов по достижению цели.

Однако на практике чаще всего применяются неоднородные семантические сети. Используемые в них отношения можно разбить на следующие классы:

класс-подкласс (*SUB*) – отношение, устанавливаемое между понятиями;

элемент-класс (*ISA*) – отношение, устанавливаемое между экземплярами понятий и понятиями;

часть-целое (*part_of*), или **целое-часть** – отношение включения одного объекта в другой;

атрибутивные отношения или **отношения типа свойство-значение** (*"цвет", "вес", "рост"* и т. п.);

темпоральные отношения (*"раньше", " позже", "одновременно"* и т. п.);

логические отношения (*"и", "или", "не", "следование"*);

глубинно-падежные семантические отношения Филмора [26] – отношения, пришедшие из лингвистики и служащие для выражения в предложении глубинных семантических отношений между группой существительного и действием (*"агент", "соагент", "объект", "инструмент", "время действия", "место действия"* и т. п.).

Для повышения изобразительных возможностей сети некоторые отношения интерпретируются особым образом. К последним относятся таксономические отношения *"класс-подкласс"* (*"множество-подмножество"*

или *“общее-частное”*) и *“элемент-класс”* (*“элемент-множество”*), по которым устанавливается иерархия понятий семантической сети и организуется наследование свойств понятий более высокого уровня понятиями (объектами) более низкого уровня. Это делает описание модели более компактным, поскольку информацию о наследуемых свойствах не нужно повторять в объектах и понятиях более низкого уровня.

Достоинствами семантических сетей является их наглядность, высокая ассоциативность и гибкость представления знаний. Основным недостатком является потеря наглядности и высокая трудоемкость обработки при переходе к семантическим сетям большого размера.

Приведем содержательный пример семантической сети (рис. 3.3), которая описывает контекст действия, выражаемого следующим предложением: «1 сентября 2010 года Вася с Машей разбили кирпичом окно в школе № 130, расположенной в Академгородке». (Заметим, что вершины-понятия на рисунке выделены серым цветом.)

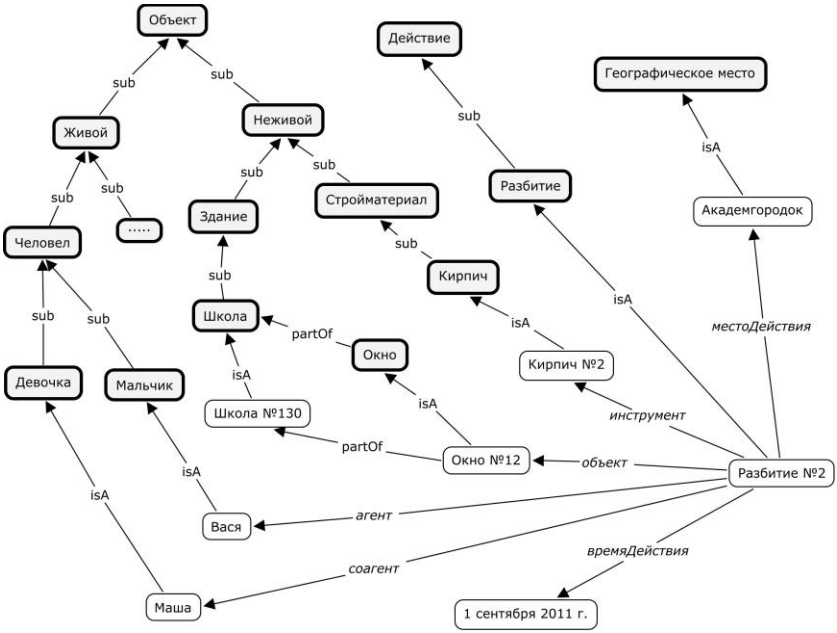

Рис. 3.3. Содержательный пример семантической сети

3.2. Функциональная сеть

Важным видом сетевой модели, имеющим большое практическое значение, является функциональная сеть. Такая сеть представляет собой дву-

дольный ориентированный граф, включающий вершины двух типов – объекты и операторы (функции). Дуги отражают функциональные связи между операторами и объектами. Дуга, направленная от объекта к оператору, предписывает рассматривать этот объект как аргумент данного оператора, дуга обратной ориентации указывает на то, что объект выступает по отношению к оператору в качестве результата.

Например, функциональная сеть, описывающая уравнение U = I * R, показана на рис. 3.4.

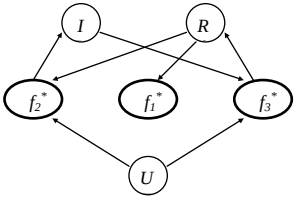

Рис. 3.4. Пример функциональной сети

Функциональные сети отражают некоторую декомпозицию определенной вычислительной процедуры, а дуги показывают функциональную связь, устанавливаемую между ее частями, возникшими в результате декомпозиции.

Если в функциональной сети операторы заменить отношениями, а дуги сделать неориентированными, то получится сеть, которую можно использовать для представления знаний о возможности вычислений, т. е. модель. Модели такого вида называются вычислительными. Особый интерес представляют вычислительные модели, предложенные Э. Х. Тыугу [25] и А. С. Нариньяни [18].

3.3. Фрейм-представление

Еще одним популярным средством представления знаний, разработанным в рамках сетевой модели, является *фрейм-представление*. Автор этого термина, М. Минский [15], первоначально определял фрейм как структуру данных, предназначенную для представления стереотипной ситуации. В современных фрейм-системах с каждым фреймом ассоциируется разнообразная информация о некотором понятии, явлении или объекте [23]. Фрейм может быть представлен в виде сети из узлов и отношений (по этой причине он и отнесен к сетевым моделям). «Верхние уровни» фрейма фиксированы и включают сущности, всегда истинные, в ситуации, описываемой данным фреймом. «Нижние уровни» содержат терминалы или слоты, т. е. «ячейки», которые заполняются конкретной информацией (данными) при активации фрейма.

Фрейм состоит из имени и отдельных информационных единиц, называемых слотами:

$F: [<r_1, v_1>, <r_2, v_2>, \ldots, <r_n, v_n>]$,

где F – имя фрейма, r_i – имя слота, а v_i – значение слота.

В отличие от семантических сетей во фрейм-представлении фиксируется жесткая структура информационных единиц, которая задается в фрейме-прототипе, или протофрейме. Конкретные фреймы, или фреймы экземпляры, задаются путем конкретизации протофрейма и, следовательно, имеют такую же структуру, как и соответствующий протофрейм.

В качестве значений слотов фрейма могут выступать имена других фреймов, что обеспечивает связывание фреймов между собой. В частности, это позволяет организовать иерархию фреймов и наследование свойств по этой иерархии. Кроме того, значением слотов могут быть имена процедур, каждая из которых вызывается в определенной ситуации, например, при присваивании слоту нового значения, удалении старого значения слота, выдачи значения слота и т. п., что делает фреймы активными участниками процесса обработки информации.

Примером слота с пустым значением, из которого запрашивается информация, может служить слот **Возраст**. Значение возраста в слоте нет смысла хранить, так как он постоянно меняется. Гораздо удобнее связать с этим слотом процедуру, которая будет вычислять актуальный возраст по заданной в другом слоте дате рождения.

Включение во фрейм-представление аппарата предположений и ожиданий усиливает его изобразительные и операционные возможности. Так, слотам фрейма по умолчанию могут быть приписаны некоторые стандартные значения. Это позволяет анализировать с помощью фреймов ситуации, в которых отсутствует упоминание о целом ряде деталей. В дальнейшем стандартные значения, присвоенные по умолчанию, могут быть заменены более подходящими значениями, которые будут получены в процессе обработки текущей ситуации.

Родственные фреймы связываются в систему фреймов. Важно, что различные фреймы одной и той же системы используют общее множество слотов. Благодаря этому становится возможным координирование информации, полученной из различных источников.

Системы фреймов, в свою очередь, организуются в информационно-поисковую сеть, которая используется, когда предложенный фрейм оказывается непригодным для данной ситуации, т. е. его слотам не могут быть присвоены значения, удовлетворяющие связанным с ними условиям. В подобных ситуациях сеть используется для того, чтобы предложить какой-нибудь другой фрейм.

Проиллюстрируем работу системы, основанной на фреймах.

Предположим, что понятие письменного отчета организовано так, как показано на рис. 3.5. Допустим, что узел **Отчет № 2** имеет структуру, представленную на рис. 3.6, и предположим, что делается запрос: *Дайте отчет о продвижении проекта по теме ВС-32*. Тогда информационно-поисковая система проанализирует запрос и внесет *ВС-32* в слот **Тема** экземпляра фрейма **ОТЧЕТ О ПРОДВИЖЕНИИ** (в данном случае узла **Отчет № 2**).

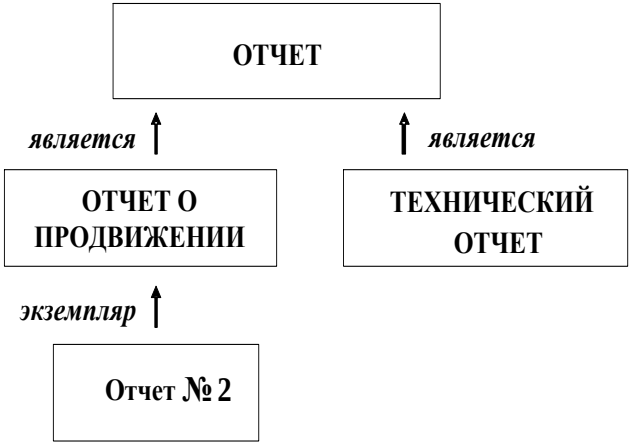

Рис. 3.5. Понятие письменного отчета

Далее все действия выполняются автоматически.

1. Выполняется процедура ***Если-добавлено***, связанная со слотом **Тема**, поскольку в этот слот было введено значение. Эта процедура осуществляет поиск (в базе данных системы) руководителя проекта по теме ВС-32. Допустим, что имя руководителя проекта *Сергей Петров*. Процедура вписывает это имя в слот **Автор** отчета о продвижении работ (**Отчет № 2**).

2. Выполняется процедура ***Если-добавлено***, связанная со слотом **Автор.** Она начинает составлять сообщение для Петрова, но обнаруживает, что у нее нет нужного значения слота **Дата представления**.

3. Указанная выше процедура активирует процедуру ***Если-требуется***, связанную со слотом **Дата представления**. Процедура ***Если-требуется*** определяет текущую дату, используя календарь базы данных, и решает, что дата *30 сентября* ближайшая к ней. Эта процедура вписывает *30 сентября* в слот **Дата представления**.

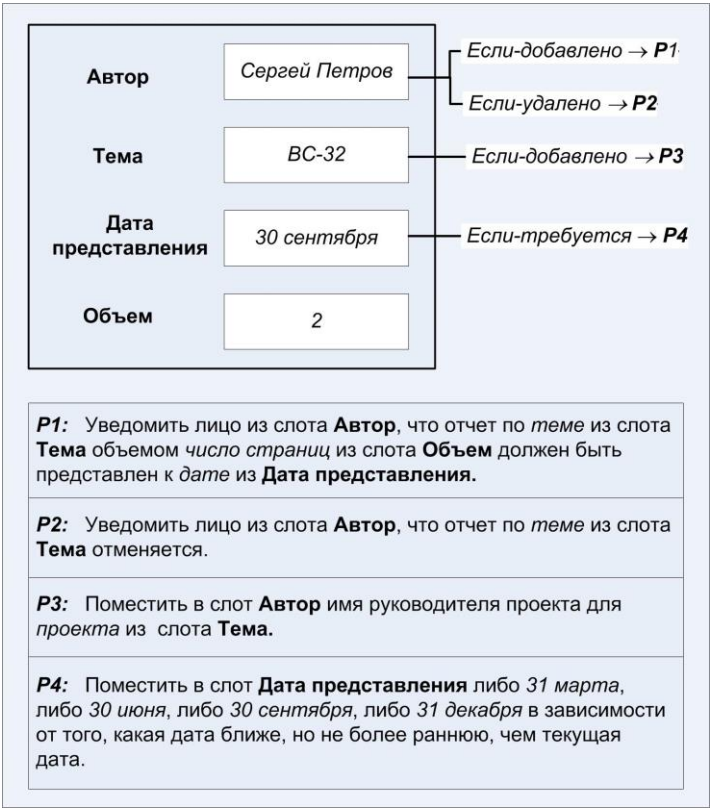

Рис. 3.6. Фрейм отчета о продвижении

4. На следующем шаге процедура ***Если-добавлено***, связанная со слотом **Автор**, обнаружит, что еще одно значение, которое нужно включить в сообщение, – *объем* (число страниц) отчета, отсутствует. Слот **Объем** не связан с процедурами, однако выше узла **Отчет № 2** существует узел общей концепции отчета о продвижении работ (**ОТЧЕТ О ПРОДВИЖЕНИИ**). Предположим, что он содержит значение объема отчета – 2 страницы. Процедура использует это значение и составляет следующее сообщение: *Сергей Петров, представьте, пожалуйста, отчет о продвижении работ по проекту темы ВС-32 к 30 сентября. Предполагаемый объем отчета – 2 страницы*.

Если в какой-то момент имя *Сергей Петров* будет удалено из слота **Автор**, то система автоматически отправит ему сообщение о том, что отчет о продвижении темы ВС-32 не требуется (поскольку сработает процедура ***Если-удалено***).

Основным недостатком фрейм-представления является отсутствие общей теории. В нем много эвристик и вариантов решения.

К главным достоинствам фрейм-представления можно отнести иерархичность описания понятий предметной области и достаточно естественное объединение в рамках одного средства как декларативного, так и процедурного компонентов представления знаний.

Следует заметить, что фрейм-представление является не конкретным языком представления знаний, а идеологической концепцией, реализуемой в каждом конкретном языке.

Наиболее известными языками представления знаний на основе фреймов являются KRL (knowledge representation language) и FRL (frame representation language).

Контрольные вопросы

1. Какие модели представления знаний Вы знаете?
2. Особенности сетевой модели. Какие средства представления знаний можно причислить к сетевой модели?
3. Что такое семантическая сеть как математический объект?
4. На каких принципах основана классификация семантических сетей? Приведите примеры различных видов семантических сетей.
5. Отличие простых и иерархических семантических сетей.
6. Отличие однородных и неоднородных семантических сетей.
7. Назовите основные типы отношений в семантической сети.
8. Что такое фрейм? Приведите типичную структуру.
9. Назовите самые существенные особенности фрейм-представления. Что общего у фреймов с семантическими сетями и каковы отличия?
10. Что такое присоединенные процедуры? Их роль в фрейме.
11. Приведите пример фрейма с присоединенными процедурами.

4. Продукционная модель

Продукционная модель предполагает такой способ организации вычислительного процесса, при котором программа преобразования некоторой информационной структуры *С* задается в виде системы правил вида:

УСЛОВИЕ → ДЕЙСТВИЕ,

где *УСЛОВИЕ* описывает определенные требования к текущему состоянию структуры *С,* а *ДЕЙСТВИЕ* содержит набор тех операций над *С*, которые надо выполнить, если *С* удовлетворяет этим требованиям.

Описанная форма программы характерна для многих известных формальных и программных (инструментальных и прикладных) систем.

4.1. Формальные системы продукций

Формальными системами, удовлетворяющими требованиям продукционной модели, являются системы подстановок и формальные грамматики.

Системы подстановок представляют собой правила преобразования цепочек (слов), заданных в некотором алфавите. К таким системам относятся системы продукций Поста (именно этим системам мы обязаны появлению термина *«система продукций»,* который получил со временем более широкое употребление) и нормальные алгоритмы Маркова. Особенностью таких систем является то, что в них по сравнению с машиной Тьюринга богаче возможности преобразования данных (слово любой сложности преобразуется в любое произвольное слово) при менее развитом управлении [23].

Системы Поста определяются алфавитом ***S*** и набором правил-продукций вида:

$a_i W \to W b_i \quad (i = 1,...,m),$ (1)

где a_i и b_i – некоторые слова в алфавите *S*.

Правило (1) применимо к слову *d*, если *d* начинается с a_i . В этом случае правило вычеркивает в *d* префикс a_i и приписывает к нему справа слово b_i.

Например, применяя к слову *ababc* продукцию $abW \to Wd$, получим слово *abcd*, к которому еще раз может быть применена та же продукция.

Нормальные алгоритмы Маркова определяются алфавитом *S* и последовательностью подстановок вида:

$a_i \to b_i$

$a_i \to . b_i$,

где a_i и b_i – некоторые слова в алфавите *S*,

а подстановка, отмеченная точкой (→.), является заключительной.

При этом *(n + 1)*-й шаг обработки начального слова d_0 определяется следующим образом.

Пусть в последовательности подстановок имеется первая, у которой левая часть a_i хотя бы раз входит в слово d_n (d_n – слово, полученное в результате первых *n* шагов), тогда в d_n вместо самого левого вхождения a_i подставляется b_i, порождая слово d_{n+1}. Если использованная подстановка не была заключительной, осуществляется переход к шагу *(n+2),* в противном же случае d_{n+1} объявляется результатом работы системы. Если на шаге (*n + 1*) не нашлось ни одной применимой подстановки, то результатом будет слово d_n.

Дадим краткое сравнение этих систем.

Заметим, что в **системе продукций Поста** фиксируется место вхождения левой части правила (начало обрабатываемого слова), но не оговари-

вается порядок применения правил, что делает систему Поста **недетерминированной** и порождает в общем случае не один, а множество процессов обработки, дающих, возможно, различные результаты.

В **алгоритмах Маркова** правила применимы к любому участку слова (хотя эта свобода и ограничена самым левым вхождением подслова a_i), но при этом порядок просмотра правил фиксирован. Вследствие этого система Маркова является **детерминированной** и порождает для каждого конкретного входного слова единственный процесс, приводящий к однозначному результату.

Формальные грамматики, служащие для описания синтаксиса и семантики различных языков, являются как раз тем классом формальных систем, который в своем развитии привел к появлению современных систем продукций (СП). Первым продукционным языком программирования был язык Флойда, основанный на грамматике и предназначавшийся для синтаксического анализа.

4.2. Программные системы продукций

Наибольший практический интерес представляют программные системы продукций, т. е. системы, имеющие программную реализацию. Такие системы далее называются системами продукций (СП).

4.2.1. Структура программной СП

Программная СП состоит из трех основных компонентов (рис. 4.1): базы данных, множества правил-продукций и интерпретатора [23, 36].

Рис. 4.1. Структура программной системы продукций

База данных (БД) представляет собой рабочую память, над которой работает множество правил. Организация рабочей памяти может быть самой разной – от простой памяти в виде совокупности именованных переменных до списков, семантических сетей и фреймов.

Правила также могут иметь произвольную сложность, сохраняя при этом вид *УСЛОВИЕ → ДЕЙСТВИЕ*.

Работу **интерпретатора** можно рассматривать как модуль, реализующий поисковый процесс, состоящий, по крайней мере, из двух фаз: выбора продукции и ее применения.

СП могут различаться как организацией рабочей памяти, так и видом правил, но существенным различием для них является различие в интерпретаторе, т. е. в способе выбора и применения продукций.

4.2.2. Проблема выбора продукций

Проблема выбора интерпретатором продукций сводится к двум подзадачам: (1) максимально ограничить число продукций, условия применимости которых будут проверяться, и (2) из полученного множества продукций выбрать одну с истинным условием применимости.

Первая задача чаще всего решается не путем ограничения множества продукций, а путем активации какого-либо его подмножества.

Вторая задача возникает в связи с тем, что в выделенном подмножестве могут оказаться истинными условия более чем одной продукции. Совокупность таких продукций называется **конфликтным множеством**, а процедура выбора продукции из этого множества – процедурой разрешения конфликта.

Перечислим основные **способы выбора продукций** из конфликтного множества:

- случайный выбор;
- выбор по заранее оговоренному критерию, не меняемому в ходе вычислений (например, первой применяется продукция с самыми жесткими требованиями – продукция с самым длинным списком условий, либо на продукциях задан полный порядок или иерархия, при этом первой применяется самая «старшая» продукция и т. п.);
- выбор по критерию, формируемому динамически по ходу процесса, например, приписыванием правилам и/или компонентам базы динамически вычисляемых весов (приоритетов) (в этом случае первой для исполнения выбирается продукция с наивысшим приоритетом, или продукция, условия которой удовлетворяются на данных, имеющих максимальный приоритет).

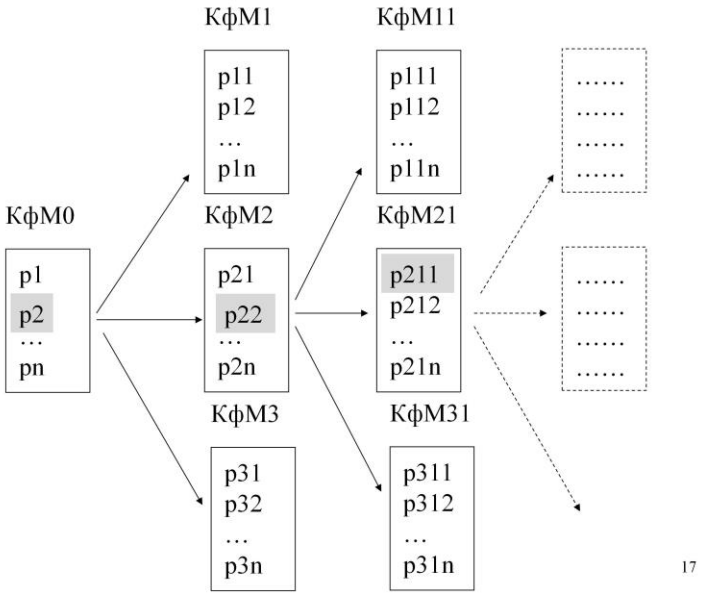

Рис. 4.2. Стратегии выбора продукций

Процедура разрешения конфликта может использовать одну из стратегий: безвозвратную или пробную (рис. 4.2).

При **безвозвратной стратегии** на каждом шаге вычислений из конфликтного множества для выполнения выбирается одна из подходящих продукций, и в дальнейшем вернуться к этой точке вычислений и применить другую продукцию невозможно.

При **пробной стратегии,** называемой также **бэктрекингом** [20], обеспечивается возможность возврата к уже пройденной точке вычислений и применения другой (альтернативной) продукции из конфликтного множества.

4.2.3. Стратегии применения СП

По способу применения продукций выделяют два вида систем продукций: прямые и обратные.

В СП, работающей в **прямом направлении**, образцом для поиска служит левая часть продукции (***УСЛОВИЕ***). Задача решается в направлении от исходного состояния к целевому. Продукции, применяясь к текущему состоянию, порождают новые состояния.

В **обратных СП** задача решается в обратном направлении – от цели к начальному состоянию. Каждый шаг обратного движения, т. е. применение продукции в обратном направлении, когда ***УСЛОВИЕ*** и ***ДЕЙСТВИЕ*** меняются местами, производит подцелевое состояние, из которого целевое может быть получено при прямом движении.

В качестве примера рассмотрим СП, включающую несколько простых правил:

(1) $y \,\&\, w \rightarrow x$

(2) $u \,\&\, z \rightarrow y$

(3) $r \rightarrow z$

В приведенных правилах стрелка ($\rightarrow$) означает, что если верно то, что написано слева, то верно написанное справа.

Эта СП работает над базой данных, содержащей следующие факты:

r, s, t, u, v, w,

где, например, факт *r* означает «зажигание исправно», а факт *s* – «шумы в коробке передач».

Первое правило (1) может интерпретироваться следующим образом: «Если имеет место перегрев двигателя (*y*) и есть шумы в двигателе (*w*), то перебит маслопровод (*x*)».

Прямой вывод предполагает использование правил для вывода новых фактов из имеющихся. Для этого Интерпретатор по очереди просматривает все правила с целью выяснения, являются ли факты в левой части правил истинными. Если у очередного правила левая часть истинна, то добавляется факт из правой части правила к хранимым фактам (в базу данных). Затем Интерпретатор переходит к следующему правилу и повторяет тот же процесс. Проверив все правила, он начинает проверку правил сначала. Эта работа продолжается до тех пор, пока в базу данных добавляются новые факты.

Вернемся к нашему примеру.

Сначала может примениться только одно правило

(3) $r \rightarrow z$,

которое добавляет *z* в базу данных.

После этого может примениться правило

(2) $\underline{u \,\&\, z} \rightarrow y$,

так как факт *u* задан как истинный с самого начала, а факт *z*

только что был выведен.

В результате в БД будет добавлен факт *y*.

Теперь в базе фактов есть и *y*, и *w*, поэтому может сработать правило

(1) $y \,\&\, w \rightarrow x$,

которое выведет факт *x*.

После этого работа Интерпретатора завершится.

При втором способе (**обратном выводе**) применения продукций вывод начинается не с посылок правил, а сразу с интересующего нас заключения (будем называть его целевым, или **целью**). Как правило, такое заключение не находится среди известных фактов, поэтому его истинность нужно доказать. Механизм вывода в этом случае состоит в нахождении тех правил, которые содержат данный факт в качестве заключения. Затем просматриваются посылки этих правил. Если они уже хранятся в базе фактов, то цель доказана.

Если посылки рассматриваемых правил не являются истинными фактами, то делается проверка: не являются ли они заключениями других правил. Если это так, то предпринимается попытка доказать истинность уже посылок этих правил. Этот процесс продолжается до тех пор, пока в качестве посылок не окажутся факты, которые являются истинными. Если такие факты найдутся, значит, цель доказана, в противном случае целевое заключение не доказано.

Предположим, нам необходимо доказать истинность факта *x* на основе представленных выше правил и фактов.

Факт *x* не задан явно в базе фактов, поэтому цель *x* активирует правило

(1) $y \,\&\, w \rightarrow x$,

w есть в базе данных, тогда нужно доказать подцель *y*.

Подцель *y* активирует правило

(2) $u \,\&\, z \rightarrow y$

u есть в БД, тогда нужно доказать подцель *z*.

Подцель *z* активирует правило

(3) $r \rightarrow z$,

r есть в базе данных, тогда подцель *z* истинна, а значит, истинна и подцель *y*, а тогда истинна и цель *x*. Что и требовалось доказать.

Выбор стратегии вывода зависит от типа решаемой задачи.

Прямой вывод применим в тех ситуациях, когда *пространство* возможных *решений необозримо*, в то время как *количество исходных данных невелико*. Например, имеется огромное число способов сборки сложного компьютера из модульных компонентов, набор которых ограничен. В связи с этим прямой вывод чаще всего применяется в задачах планирования и проектирования.

Обратный вывод применяется в тех задачах, где *число возможных решений невелико*, но присутствуют *большие объемы исходных данных*. К таким задачам относятся задачи классификации и диагностики, в которых число видов (например, животных) или диагностируемых ситуаций (например, заболеваний, технических неисправностей) невелико. Поэтому обратный вывод чаще всего применяется при диагностике и классификации.

Ярким представителем систем, использующих обратную стратегию, является язык ПРОЛОГ. Прямая стратегия используется в таких системах, как OPS-5 [27], Semp-TAO [9]. В некоторых системах, например в системе доказательства теорем Нильсона, языке Плэнер [22], одновременно используются обе стратегии.

4.3. Классификация систем продукций

В зависимости от отношения к проблеме активации продукций СП делятся на два больших класса: простые СП и управляемые СП:

- Простые системы продукций
- Управляемые системы продукций
  - СП с независимым управляющим языком
  - Иерархические СП
  - Последовательные СП
  - Параллельно-последовательные СП

В простых СП активными считаются все продукции. Обычно такой способ активации продукций применяется в СП, небольших по объему, а также в СП, характеризующихся тем, что структуризация множества продукций и принудительная активация противоречат положенным в их основу принципам или используемой ими стратегии выбора продукций. Это относится, например, к семейству языков OPS [27], использующих специальный алгоритм быстрого сопоставления образцов (Rete-алгоритм), и языку ПРОЛОГ [11], в основе которого лежит бэктрекинг и обратный вывод.

В *управляемых СП* предусмотрены средства управления активацией продукциями. Выделяется четыре вида управляемых систем продукций: *СП с независимым управляющим языком, иерархические СП, последовательные* и *параллельно-последовательные СП.*

Системы продукций с независимым управляющим языком состоят из «чистой» СП и *управляющего языка*, с помощью которого на каждом шаге вычислений явно и независимо от эвристических процедур разрешения конфликта указывается подмножество продукций, подлежащих активации.

Например, если «чистая» система продукций включает следующее множество правил: *p1, p2, ..., pn*, то стратегия активации правил может быть задана выражением

$p1, (p2,p3)^2, (p4,p5)^*, (p6)^3, p7,$

где круглые скобки задают группы правил, исполняемых на одном этапе, числа в степени задают количество повторений, а звездочка означает, что данная группа исполняется до тех пор, пока ее правила применяются.

Достоинством таких СП является то, что они позволяют повысить эффективность СП без повышения сложности продукций, недостатком – статичность и жесткость управления.

Другой способ управления активацией продукций – использование *метапродукций,* т. е. продукций, обладающих информацией о других продукциях, и активирующих (деактивирующих) на основе этих знаний и анализа текущей ситуации определенное подмножество продукций. Системы продукций могут включать несколько уровней метапродукций (рис. 4.3), поэтому такие СП называются *иерархическими.*

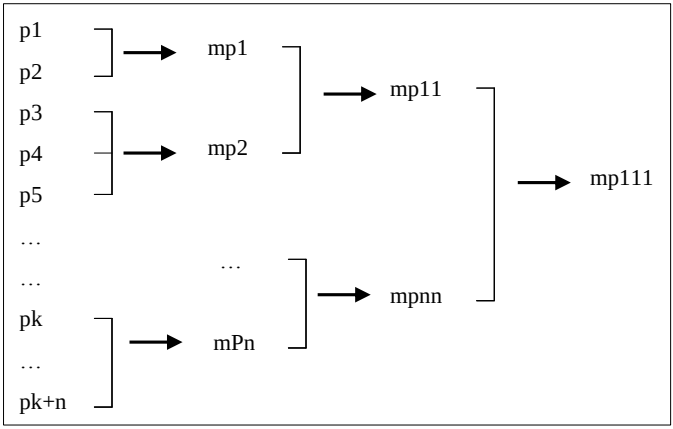

Рис. 4.3. Структура иерархической СП

Таким образом, аппарат метапродукций позволяет организовать довольно гибкое динамическое управление активацией продукций.

В *последовательных СП* (рис. 4.4) множество продукций разбивается на несколько подмножеств, каждое из которых представляет собой автономный модуль обработки данных (продукционный модуль). При этом фиксируется следующий порядок применения правил: сначала применяются правила из первого подмножества, затем из второго и т. д.

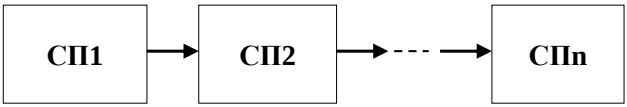

Рис. 4.4. Структура последовательной СП

Такая организация СП может использоваться в системах, в которых возникает необходимость в многоуровневом представлении знаний, причем эти уровни могут быть четко разделены. Например, в лингвистических процессорах семейства ЗАПСИБ [16, 17], обрабатывающих запросы к ре-

ляционной базе данных на естественном языке, каждому уровню знаний соответствует определенный продукционный модуль.

Параллельно-последовательные СП (рис. 4.5) характеризуются тем, что в них множество правил разбито на непересекающиеся подмножества, каждое из которых имеет свою рабочую память, т. е. фактически СП состоит из нескольких систем продукций, процессы обработки информации в которых протекают независимо. Взаимодействие между этими процессами осуществляется через общую память и только в строго определенные моменты времени посредством включенных в систему двух множеств так называемых «параллельных продукций». СП такого типа получили распространение на многопроцессорных ЭВМ.

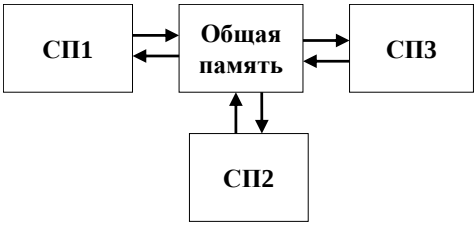

Рис. 4.5. Структура параллельно-последовательной СП

4.4. Достоинства и недостатки систем продукций

Достоинства:

- *Универсальность СП*, как метода описания широкого класса задач.
- *Естественность спецификации знаний*. Для многих предметных областей естественно представлять знания в виде правил вида *УСЛОВИЕ* $\rightarrow$ *ДЕЙСТВИЕ*.
- *Высокая и естественная модульность СП*: (1) каждая продукция представляет собой автономное действие, снабженное индивидуальной функцией управления, самостоятельно определяющей момент выполнения действия; (2) все множество продукций может естественным образом структурироваться путем разбиения на подмножества, объединяющие продукции, которые относятся к одним и тем же компонентам знаний.

Недостатки:

- При большом количестве правил в СП *трудно отслеживать* их *непротиворечивость*.
- *Повышенная сложность контроля правильности* СП-процесса.
- Существенно *более низкая эффективность вычислительного процесса* по сравнению с традиционным программированием.

4.5. Применение продукционной модели

На основе продукционной модели разработан ряд языков и систем представления знаний.

Одним из первых языков продукционного типа является Рефал [1]. Этот язык ориентирован на решение задач, связанных с обработкой текстов, и в основном использовался для разработки трансляторов со специализированных языков. К этому классу языков относятся также уже упоминавшиеся языки ПРОЛОГ и OPS-5.

Одной из первых систем, в которой использовалась продукционная модель, была экспертная система MYCIN [23, 27], дающая консультации при диагностике и лечении инфекционных заболеваний. Вообще для многих экспертных систем характерно представление знаний в виде продукционных правил.

Таким образом, продукционная модель используется в следующих областях:

- Построение компиляторов,
- Автоматическая обработка текстов,
- Распознавание и синтез речи,
- Экспертные системы.

Контрольные вопросы

1. Что такое продукционная модель?
2. Приведите структуру программной системы продукций.
3. Каков цикл работы системы продукций?
4. Что такое конфликтное множество правил? Каковы основные способы разрешения конфликтов в системе продукций?
5. Что такое бэктрекинг? Поясните смысл этого понятия применительно к продукционной модели представления знаний.
6. Дайте классификацию систем продукций.
7. Что представляет собой простая система продукций?
8. Какие системы продукций называются управляемыми? Перечислите типы управляемых систем продукций.
9. Дайте характеристику систем продукций с независимым управляющим языком.
10. Что такое метапродукция? Для каких целей используются метапродукции?
11. Что такое иерархические системы продукций?
12. Какие системы продукций называются последовательными?
13. Что такое параллельно-последовательные системы продукций?

5. Представление нечетких знаний

Часто, при представлении знаний о сложных предметных областях приходится сталкиваться с их неполнотой, неточностью, неоднозначностью и нечеткостью.

Нечеткость связана с отсутствием точных границ области определений и свойственна большинству понятий. Эта нечеткость границ приводит к тому, что не всегда оказывается возможным решить вопрос о соответствии данного объекта данному понятию по принципу *да/нет*. Часто можно говорить только о степени соотнесенности одного другому, оценивая ее, например, в интервале от 1 (определенное да) до 0 (определенное нет).

Это означает, что переход от полной принадлежности объекта классу к полной его непринадлежности происходит не скачком, а плавно, причем принадлежность объекта классу может быть выражена числом из интервала [0,1].

Аналогичные рассуждения можно отнести и к отдельным свойствам объектов. Не всегда можно четко рассуждать о таких свойствах объектов, как вес, цвет, температура, размер и т. п. Нет четкой границы между тяжелым и легким, темным и светлым, холодным и горячим, большим и маленьким и т. п.

Методы представления нечетких знаний были предложены американским профессором Л. Заде в 1965 г. [10].

Л. Заде ввел два фундаментальных понятия: лингвистическая переменная и нечеткое множество.

5.1. Понятие лингвистической переменной

Сначала дадим неформальное определение лингвистической переменной.

Лингвистическая переменная (ЛП) – это переменная, значениями которой являются слова или выражения естественного (иногда искусственного) языка. Переменную ***Возраст*** можно рассматривать как лингвистическую переменную, если она принимает не числовые значения (например, от 0 до 100), а лингвистические, такие как *молодой, старый, очень молодой, очень старый* и т. п. Аналогично можно ввести ЛП *Температура_тела_больного* со значениями *нормальная, повышенная, высокая, очень высокая* и т.п.

Лингвистическая переменная описывается следующим набором:

(N, T(N), U, G, M), где

N – название лингвистической переменной,

T(N) – терм-множество *N*, т. е. совокупность ее лингвистических значений,

U – универсальное множество,

G – синтаксическое правило, порождающее терм-множество *T(N)*,

M – семантическое правило, которое каждому лингвистическому значению *X* ставит в соответствие его смысл *M(X)*, причем *M(X)* обозначает нечеткое подмножество множества *U* (т. е. подмножество, границы которого размыты).

Смысл лингвистического значения *X* характеризуется функцией совместимости

$c : U \to [0,1]$,

которая каждому элементу $u \in U$ ставит в соответствие значение совместимости этого элемента с *X* .

Так, например, если функция совместимости для значения *молодой* имеет вид, как показано на рис. 5.1, совместимость возраста 27 лет со значением *молодой* может быть равна 0.8, а 35 лет – 0,5.

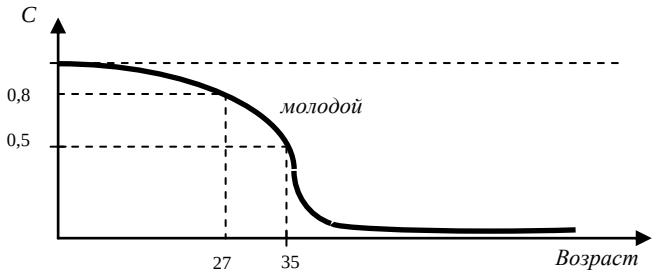

Рис. 5.1. Функция совместимости для значения *молодой*

Таким образом, с помощью лингвистических переменных можно приближенно описывать понятия и явления (свойства) не поддающиеся точному описанию.

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

Пример 5.1

Пусть *x* – мало,
x и *y* – примерно равны,
тогда *y* – более или менее мало.

5.2. Нечеткие множества

При рассмотрении смысла лингвистической переменной мы уже столкнулись с нечетким подмножеством, определив его как множество с размытыми или нечеткими границами. По-английски Fuzzy означает нечеткий,

размытый. Поэтому иногда нечеткие множества называют размытыми множествами, или множествами Заде (Zadeh set) – по имени их автора.

Дадим более строгое определение нечеткого множества, а также связанных с ним понятий.

Нечеткое множество *(НМ)*

$$A = \{(x, \mu_A(x)\} \quad (1)$$

определяется как совокупность упорядоченных пар, составленных из элементов x универсального множества X и соответствующих степеней принадлежности $\mu_A(x)$, или непосредственно в виде функции

$$\mu_A : X \to [0,1]$$

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

Носителем НМ А называется множество таких точек в X, для которых

$\mu_A(x) > 0$.

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

Точкой перехода НМ *A* называется такой элемент множества X, степень принадлежности которого множеству A равна *0,5*.

Пример 5.2

Пусть УМ X представляет собой интервал *[0,100]*, и переменная x, принимающая значения из этого интервала, интерпретируется как возраст. Нечеткое подмножество универсального множества X, обозначаемое термином *старый*, можно определить функцией принадлежности вида

$$\mu_A(x) = \begin{cases} 0 \text{ при } 0 \le x \le 50, \\ \left(1 + \left(\frac{x-50}{5}\right)^{-2}\right)^{-1}, \text{ при } 50 < x \le 100 \end{cases}$$

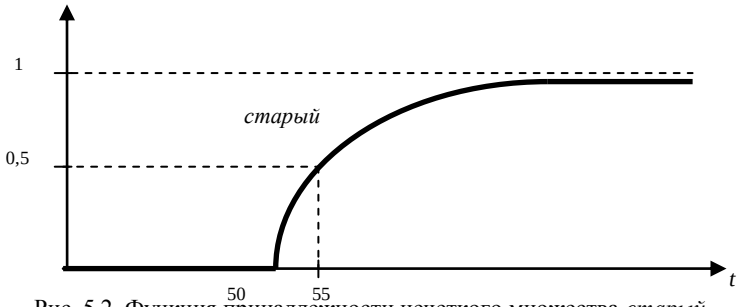

Рис. 5.2. Функция принадлежности нечеткого множества *старый*

В этом примере (рис. 5.2) носителем НМ *старый* является интервал *[50,100]*, высота близка к *1*, а точкой перехода является значение $x = 55$.

Обычно НМ А универсального множества *X* записывается в виде

$$A = \mu_1 | x_1 + \mu_2 | x_2 + \ldots + \mu_n | x_n \qquad (2)$$

где $\mu_i, i = 1, \ldots, n$ – степень принадлежности элемента x_i НМ A.

Пример 5.3

НМ *Несколько = 0,5|2 + 0,8|3 + 0,9|4 + 1|5 + 1|6 +1|7 +0,8|8 + 0,5|9*

Если носитель нечеткого множества имеет мощность континуума, то используется следующая запись:

$$A = \int_X \mu_A(x) | x$$

где знак $\int$ обозначает объединение одноточечных НМ $\mu_A(x) | x, x \in X$.

Пример 5.4.

$$НМ\ \textbf{старый} = \int_{50}^{100} (1 + (\frac{x-50}{5})^{-2})^{-1} | x .$$

Рассмотрим некоторые из основных операций, которые можно выполнить над нечетким множеством.

1. Дополнение НМ A:

$$\neg A = \int_X (1 - \mu_A(x)) | x .$$

Операция дополнения соответствует логическому отрицанию.

2. Объединение НМ A и B:

$$A + B = \int_X (\mu_A(x) \vee \mu_B(x)) | x .$$

Объединение соответствует логической связке «или».

3. Пересечение НМ A и B:

$$A \cap B = \int_X (\mu_A(x) \wedge \mu_B(x)) \mid x .$$

Пересечение соответствует логической связке «и».

4. Произведение НМ A и B:

$$A * B = \int_X (\mu_A(x) * \mu_B(x)) \mid x .$$

Таким образом, любое НМ A^α, где α – положительное число, следует понимать как $A^\alpha = \int_X (\mu_A(x))^\alpha \mid x$

Основным равенством, с помощью которого для НМ можно расширить область определения X отображения или отношения, включив в нее наряду с точками произвольные нечеткие подмножества множества X, является **принцип обобщения**.

Предположим, что f – отображение $X \to Y$, а A – нечеткое подмножество вида (2). Тогда согласно принципу обобщения

$$f(A) = f(\mu_1 \mid x_1 + \mu_2 \mid x_2 + \ldots + \mu_n \mid x_n) =$$

$$= \mu_1 \mid f(x_1) + \mu_2 \mid f(x_2) + \ldots + \mu_n \mid f(x_n)$$

Пример 5.5.

Пусть $X_1 = X_2 = 1 + 2 + \ldots 10$ и

$A_1 = \approx 2 =$ *примерно 2* $= 1|2 + 0.6|1 + 0.8|3$,

$A_2 = \approx 6 =$ *примерно 6* $= 1|6 + 0.8|5 + 0.7|7$,

$f(x_1,x_2) = x_1 * x_2 =$ арифметическое произведение x_1 и x_2.

Применяя принцип обобщения, имеем

$(\approx 2)*(\approx 6) = (1|2+0,6|1+0,8|3)*(1|6+0,8|5+0,7|7) =$

$= 1|12+0,8|10+0,7|14+0,6|6+0,6|5+0,6|7 +0,8|18+0,8|15+0,7|21 =$

$= 0,6|5+0,6|6+0,6|7+0,8|10+1|12+0,7|14+0,8|15+0,8|18+0,7|21.$ (3)

Таким образом, арифметическое произведение нечетких чисел *примерно 2* и *примерно 6* есть нечеткое число, выраженное формулой (3).

Используя принцип обобщения, можно переходить к нечетким множествам более высокого уровня.

НМ есть **множество типа n**, n = 2,3,..., если значениями его функции принадлежности являются НМ типа n–1. Функция принадлежности типа 1 принимает значения из интервала [0,1].

Операции дополнения, объединения, пересечения и т.п. для НМ типа 2 определяются, если использовать принцип обобщения [10, 19].

5.3. Нечеткие отношения

При выполнении нечетких выводов необходимо знать нечеткие отношения.

Если X – декартово произведение n универсальных множеств X_1, X_2, ..., X_n, то n–арное **нечеткое отношение** (НО) R в X определяется как нечеткое подмножество X:

$$R = \int_{X_1 x X_2 x \ldots X_n} \mu_R(x_1, x_2, \ldots, x_n) \,|\, (x_1, x_2, \ldots, x_n)$$

где μ_R – функция принадлежности НМ R.

Распространенными примерами нечетких отношений являются "*много больше чем", "имеет сходство", "близко к"* и т. д.

Например, если $X_1 = X_2 = (-\infty, +\infty)$, то отношение "*близко к"* можно определить следующим образом:

$$\text{"близко к"} = \int_{X_1 x X_2} e^{-\alpha * |x_1 - x_2|} \,|\, (x_1, x_2),$$

где a - коэффициент масштабирования.

Контрольные вопросы

1. Что такое лингвистическая переменная? Дайте неформальное определение лингвистической переменной. Приведите пример лингвистической переменной.
2. Дайте формальное определение лингвистической переменной.
3. Приведите основные способы задания лингвистической переменной. Покажите на примерах.
4. Что такое нечеткое множество? Приведите пример нечеткого множества
5. Приведите основные операции над нечеткими множествами.
6. Для чего нужны нечеткие отношения и как они задаются?
7. Приведите пример нечеткого отношения.

6. Использование нечеткой логики в системах, основанных на знаниях

6.1. Особенности нечеткой логики

В булевой логике 1 представляет *истину*, а 0 – *ложь*. То же имеет место и в нечеткой логике, но кроме того, здесь используются также дроби между 0 и 1 для указания «частичной» истины. Так, запись *р(высокий(X)) = 0,75* означает, что предложение «*X – высокий*» в некотором смысле на три четверти истинно. Точно так же оно на одну четверть ложно.

В нечеткой логике определены эквиваленты операций *И, ИЛИ* и *НЕ*:

p1 И p2 = min (p1, p2) (т. е. меньшее)
p1 ИЛИ p2 = max (p1, p2) (т. е. большее)
НЕ p1 = 1 – p1 (т. е. «обратное значение»)

Таким образом, нечеткие сведения можно комбинировать на основе строгих логических методов. Поэтому нечеткая логика может применяться

в практических системах, например, в системах поддержки принятия решений.

Слабым местом в нечеткой логике является функция принадлежности, вернее ее выбор. Предположим, что Петру 35 лет. Насколько истинно предположение, что он *молодой*? Равна ли его истинность величине 0,5, поскольку он прожил примерно полжизни, или 0,6?

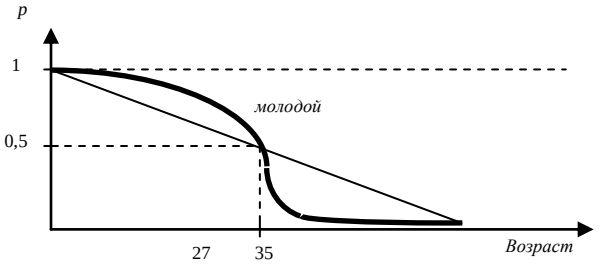

Рис. 6.1. Возможные функции принадлежности для нечеткого множества *молодой*

Какова должна быть функция принадлежности у нечеткого множества *молодой* (рис. 6.1), каков должен быть ее график (кривая или прямая)?

Для предпочтения одного вида функции другому нет серьезных рациональных обоснований, поэтому в реальной задаче могут присутствовать десятки и сотни подобных функций, каждая из которых до некоторой степени является произвольной.

Поэтому в практических системах, использующих нечеткую логику, например, в системе REVEAL, предусматриваются средства, позволяющие пользователю легко модифицировать различные принадлежности и/или устанавливать форму их графика.

Существует свыше десятка типовых форм кривых для задания функций принадлежности. Наибольшее распространение получили следующие функции:

- треугольная,
- трапецеидальная,
- гауссова (нормальное распределение).

Еще одной проблемой при использовании нечеткой логики является проблема взвешивания отдельных сведений и их использование в «нечетких правилах».

Предположим, что имеется два нечетких правила с одним и тем же следствием:

Правило 1: ***если*** *a* ***И*** *b*, ***то*** *c.*

Правило 2: ***если*** *e* ***ИЛИ*** *f*, ***то*** *c.*

При этом известны степени истинности (определенности) *a, b, e* и *f:*
p(a) = 1; p(b) = 0,8; p(e) = 0,5; p(f) = 0,4.
Тогда из *Правила 1* степень истинности *p(c) = min (1, 0,8) = 0,8,*
а из *Правила 2 p(c) = max (0,5, 0,4) = 0,5* .
Какое из этих значений *p(c)* выбрать? Первое, второе? А может быть взять их среднее арифметическое?

6.2. Схема Шортлиффа

Э. Шортлифф (E. Shortliffe) разработал схему [24], основанную на так называемых коэффициентах уверенности, которые он ввел для измерения степени доверия к любому данному заключению, являющемуся результатом полученных к этому моменту свидетельств.

Коэффициент уверенности – это разность между двумя мерами:
КУ [h : e] = МД [h : e] – МНД [h : e], (4)
где
КУ [h : e] – уверенность в гипотезе *h* с учетом свидетельств *e,*
МД [h : e] – мера доверия гипотезе *h* при заданных свидетельствах *e,*
МНД [h : e] – мера недоверия гипотезе *h* при свидетельствах *e.*

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

Таким образом, *КУ* – это простой способ взвешивания свидетельств «за» и «против».

Заметим, что приведенная формула не позволяет отличить случай противоречащих свидетельств (и МД, и МНД обе велики) от случая недостаточной информации (и МД, и МНД обе малы), что иногда бывает полезно.

Заметим также, что ни КУ, ни МД, ни МНД, не являются вероятностными мерами.

МД и МНД подчиняются некоторым аксиомам теории вероятности, но не являются выборками какой-нибудь популяции, и, следовательно, им нельзя дать статическую интерпретацию. Они просто позволяют упорядочить гипотезы в соответствии с той степенью обоснованности, которая у них есть.

Э. Шортлифф также ввел формулу уточнения для взвешивания свидетельств.

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

Формула для меры доверия выглядит следующим образом:
МД [h : e1, e2] = МД [h : e1] + МД [h : e2](1 – МД [h : e1]),* (5)
где запятая между свидетельствами *e1* и *e2* означает, что *e2* следует за *e1*.

Аналогичным образом уточняются значения *МНД.*

Смысл формулы (5) состоит в том, что эффект второго свидетельства *e2* на гипотезу *h* при заданном свидетельстве *e1* сказывается в смещении *МД* в сторону полной определенности на расстояние, зависящее от второго свидетельства.

Формула (5) имеет два важных свойства:

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

Вернемся к примеру

Правило 1: ***если*** *a* ***И*** *b*, ***то*** *c.*

Правило 2: ***если*** *e* ***ИЛИ*** *f*, ***то*** *c.*

При этом степени истинности *a, b, e* и *f:*

p(a) = 1; p(b) = 0,8; p(e) = 0,5; p(f) = 0,4.

Из *Правила 1 степень определенности p(c) = min (1, 0,8) = 0,8,*

из *Правила 2 p(c) = max (0,5, 0,4) = 0,5.*

Применяя формулу (2), получаем:

МД [c : Правило 1, Правило 2] =

МД [c : Правило 1] + МД [c : Правило 2](1 – МД [c : Правило 1])*

= 0,8 + 0,5(1 – 0,8) = 0,9.*

Итак *МД [c : Правило 1, Правило 2] = 0,9.*

Таким образом, объединенная мера доверия оказывается выше, чем при учете каждого свидетельства, взятого отдельно. Это согласуется с нашей интуицией, что несколько показывающих одно и то же направление свидетельств подкрепляют друг друга.

Кроме того, можно поменять порядок применения правил 1 и 2, но на результатах это не отразится.

Схема Шортлиффа допускает также возможность того, что правила, как и данные, могут быть ненадежными. Это позволяет описывать более широкий класс ситуаций.

Каждое правило снабжается *коэффициентом ослабления* (числом от 0 до 1), показывающим **надежность правила**.

Так, если в нашем примере мы снабдим *Правило 1* коэффициентом ослабления *0,6*, а *Правило 2* – коэффициентом *0,8*, получим следующее:

*МД [c : Правило 1] = min (1, 0,8)*0,6 = 0,48*

*МД [c : Правило 2] = max (0,5, 0,4)*0,8 = 0,4*

Применяя формулу уточнения (2), получаем:

МД [c : Правило 1, Правило 2] =

МД [c : Правило 1] + МД [c : Правило 2](1 - МД [c : Правило 1])*

= 0,48 + 0,4(1 – 0,48) = 0,48 + 0,208 = 0,688.*

Для обеспечения возможности исключения из вывода заключений, имеющих низкую достоверность, введен **порог уверенности** *(ПУ) – число*

от 0 до 1. Суть его состоит в следующем: если КУ некоторого заключения меньше *ПУ*, то таким заключением можно пренебречь.

В заключение следует отметить, что, хотя схема Шортлиффа не имеет строгого теоретического обоснования, она хорошо себя показала в практических приложениях, в частности в системе MYCIN и последовавшими за ней другими экспертными системами.

Контрольные вопросы

1. Каковы особенности использования нечеткой логики в экспертных системах?
2. Назовите три типовых формы кривых, используемых для задания функций принадлежности.
3. Перечислите основные компоненты схемы Шортлиффа.
4. Каковы смысл и свойства формулы уточнения?
5. Что такое надежность правила и как она задается?
6. Что такое порог уверенности и как он используется?

7. Онтологии

В настоящее время стало общепринятой практикой описывать предметные области с помощью онтологических моделей. В основе этих моделей лежит понятие онтологии, которое первоначально появилось в философии, но в конце XX-го в. стало широко использоваться в информатике (computer science) и инженерии знаний.

Основная цель создания онтологий заключается в обеспечении поддержки деятельности по накоплению, совместному использованию и повторному использованию знаний. Чтобы сделать такую деятельность возможной, создаются стандарты, регламентирующие структуру и процесс разработки онтологий, включая языки описания онтологий.

7.1. Основные определения

Несмотря на то, что существует множество определений онтологии, термин «онтология» имеет всего два основных значения:

1. Онтология – это раздел философии, учение о бытии (в отличие от гносеологии – учения о познании), в котором исследуются всеобщие основы, принципы бытия, его структура и закономерности [26].

2. Онтология – это некий инженерный артефакт, структура, описывающая некоторую реальность (или систему) с помощью слов заданного словаря и множества допущений на значения этих слов [21, 34].

Для инженерии знаний более подходящим является второе значение термина «онтология». Хотя не стоит отвергать и философский смысл этого термина, рассматривая онтологию как определенную систему категорий,

являющихся следствием определенного взгляда на мир [34]. Отмечая при этом такую важную деталь, что сама система категорий не зависит от конкретного языка: онтология Аристотеля всегда одна и та же, независимо от языка, использованного для ее описания.

Автором термина «онтология» в инженерии знаний является американский ученый Томас Грубер [33]. Согласно его определению, *онтология является явной спецификацией концептуализации*. При этом под концептуализацией понимается некоторая абстракция, т.е. упрощенное представление некоторой части мира, построенное для определенной цели. Другими словами, концептуализация – структура реальности, заданная независимо от словаря и конкретной ситуации.

Например, на столе лежат кубики. Концептуализация – это набор возможных положений кубиков (т. е. всевозможных пространственных отношений между кубиками), но не конкретное расположение кубиков.

Неформально, онтология представляет собой некоторое описание взгляда на мир применительно к конкретной области интересов. Это описание состоит из терминов и правил использования этих терминов, ограничивающих их значения в рамках конкретной области.

На формальном уровне, онтология – это система, состоящая из набора понятий и набора утверждений об этих понятиях, на основе которых можно строить классы, объекты, отношения, функции и теории.

Рассмотрим **формальную модель онтологии** [5].

Под формальной моделью онтологии O понимают тройку вида

$O = <C, R, F>$,

где C – конечное множество понятий (концептов) предметной области (ПО), которую определяет онтология O;

R – конечное множество отношений между понятиями ПО;

F – конечное множество функций интерпретации (аксиоматизация), заданных на понятиях и/или отношениях онтологии O.

Естественными ограничениями, накладываемыми на множество C, являются конечность и не пустота ($C \neq \varnothing$).

Множества R и F могут быть пустыми, что соответствует частным видам онтологии, когда она вырождается в простой словарь ($R = \varnothing$, $F = \varnothing$), таксономию понятий ($F = \varnothing$) и т. д.

Таким образом, онтологии на базовом уровне должны, прежде всего, обеспечивать словарь понятий (терминов) для представления и обмена знаниями о предметной области и множество связей (отношений), установленных между понятиями в этом словаре.

Основными компонентами современных онтологий являются:

- Классы (classes). Обычно организованы в таксономии.

- Отношения (relations). Представляют тип связей между концептами предметной области.
- Функции (functions). Специальный случай отношений, в которых n-й элемент определяется по значениям (n–1) предшествующих элементов: $F: C_1 \times C_2 \times \ldots \times C_{n-1} \Rightarrow C_n$.
- Аксиомы (axioms). Моделируют предложения, которые всегда истинны.
- Экземпляры (instances). Представляют конкретные объекты реального или абстрактного мира.

Скелетом любой онтологии является таксономия, т. е. древообразная структура, задающая классификации понятий (классов). В качестве таксономического отношения используется отношение «общее-частное», которое имеет различные варианты названия, но чаще всего используется "subclass-of".

Для таксономических отношений (*subclass_of*) имеют место следующие свойства:

- Транзитивность:
  A subclass_of B, B subclass_of C => A subclass_of C.
- Наследование свойств (*property)*:
  S property A, B subclass_of A => S property (B).

7.2. Классификация онтологий

Наряду с множеством определений онтологии, существует множество их различных классификаций. На практике основными критериями классификации являются: назначение онтологии, выразительность онтологии, уровень формальности онтологии.

Наиболее важной и в теоретическом, и практическом плане представляется классификация онтологий по назначению (или по цели создания), данная Н. Гуарино [34] (рис. 7.1):

- **Онтологии верхнего уровня (top-level ontology).** Включают самые общие понятия, которые не зависят от конкретных предметных областей и задач (являются общими для них). Такими понятиями могут быть «время», «пространство», «объект», «событие» и т. д.
- **Онтологии предметных областей (domain ontology).** Описывают понятия и отношения, характерные для конкретных предметных областей (например, медицины, археологии, энергетики).
- **Онтологии задач (task ontology).** Включают понятия и отношения, описывающие конкретную задачу или деятельность (например, диагностику или обучение).
- **Онтология приложения (application ontology).** Объединяет в себе онтологию задач и онтологию предметной области для того, чтобы

специализировать входящие в них понятия для применения в конкретном приложении.

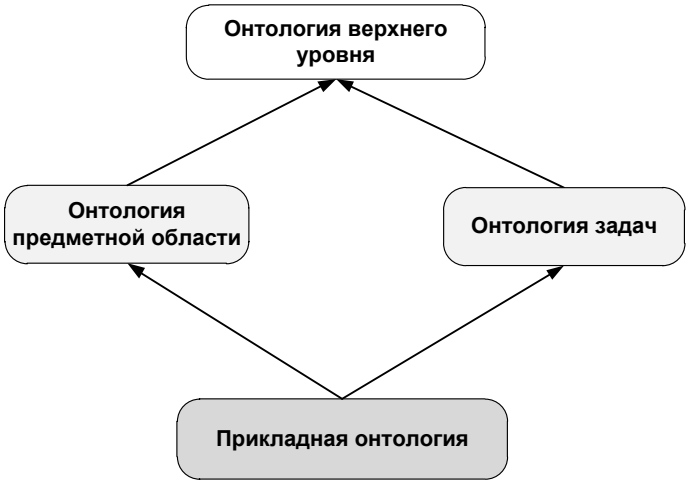

Рис. 7.1. Классификация онтологий по назначению

7.3. Онтологии верхнего уровня

Ввиду особой важности онтологий верхнего уровня для онтологического инжиниринга, рассмотрим эти онтологии подробнее.

Прежде всего, заметим, что каждая онтология верхнего уровня претендует на то, чтобы быть единой «правильной онтологией», фиксирующей знания, общие для всех предметных областей, и благодаря этому многократно использоваться при создании всех других онтологий более низкого уровня. Но все попытки создать онтологию верхнего уровня «на все случаи жизни» пока не привели к ожидаемым результатам. Поэтому разработка онтологий верхнего уровня продолжается до сих пор.

В то же время существует несколько серьезных проектов, посвященных созданию онтологий верхнего уровня: онтология Дж. Сова (Sowa's top-level ontology), онтология SUMO (The Standard Upper Ontology), онтология DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering), верхние уровни онтологии OpenCyc (Cyc's upper ontology), онтология универсалий и частиц (Top-level ontologies of universals and particulars).

Поскольку онтологии верхнего уровня описывают самые общие знания об окружающем мире, они во многом похожи. Так, во всех онтологиях проводится разделение сущностей на **абстрактные** (такие сущности не могут занимать положения ни в пространстве, ни во времени) и **реально существующие** (материальные, осязаемые).

Во всех онтологиях, так или иначе, присутствует деление на **постоянные** и **временные** (меняющиеся во времени) сущности, деление на **объекты** и **процессы**. В онтологии Дж. Совы это деление на «Континуальные» и «Происходящие», в онтологии DOLCE – «Постоянные» и «Происходящие», в SUMO – «Объекты» и «Процессы».

В то же время даже на верхних уровнях этих онтологий наблюдаются существенные различия.

В онтологии SUMO первично разделение на абстрактные и материальные сущности, а разделение на постоянные и временные – вторично.

В DOLCE на верхнем уровне производится разделение на постоянные, временные, абстрактные и качественные сущности.

В онтологии Дж. Совы иерархии сущностей в явном виде нет: в ней описаны только категории, по которым понятия разделяются или группируются.

В онтологии OpenCyc на верхнем уровне коллекция «Нечто» делится на «Неосязаемые» и «Индивиды», но экземпляры и тех, и других могут быть как абстрактными, так и материальными объектами.

Таким образом, онтологиям верхнего уровня присущи следующие характеристики:

- Нацеленность на многократное повторное использование онтологии
- Набор основных отношений
  - «Класс-подкласс»
  - «Часть-целое»
- Типичные концепты и принципы разделения
  - Сущность
  - Явление
  - Объект
  - Процесс
  - Роль

7.4. Применение онтологий

Онтологии широко применяются в таких классах систем, основанных на знаниях, как:

- Экспертные системы.
- Системы поддержки принятия решений.
- Порталы научных знаний.
- Обучающие и советующие системы.
- Системы управления знаниями предприятия.
- Лингвистические процессоры.

Наиболее важными задачами, решаемыми с помощью онтологий, являются:

- Семантическое (онтологическое) описание Интернет-ресурсов, доступное поисковым агентам (подход Semantic Web).
- Информационный поиск (расширение и специализация запросов на основе онтологии для повышения релевантности и полноты ответов).
- Интеграция разнородных источников данных (использование онтологии на концептуальном уровне и при поиске).
- Обработка текстов на естественном языке (модель предметной области задается в виде онтологии).

Системы, основанные на онтологиях, разработаны для самых разных областей человеческой деятельности: защита окружающей среды, медицина, промышленность, региональное управление, право и др.

Онтологии могут использоваться как при проектировании и разработке интеллектуальных систем (в управляемой онтологией разработке системы), так и в качестве полноценного компонента в процессе функционирования системы (в управляемой онтологией системе).

При разработке интеллектуальных систем онтология может использоваться для:

- формирования и фиксации общего разделяемого всеми экспертами знания о предметной области;
- явной концептуализации предметной области, позволяющей описывать семантику данных;
- обеспечения возможности переиспользования знаний;
- описания функциональности системы (типов решаемых задач).

В процессе функционирования интеллектуальной системы онтология может использоваться для обеспечения:

- совместного использования разнородных данных и знаний;
- процесса решения задач, составляющих функциональность системы;
- лучшего понимания предметной области пользователями системы.

Контрольные вопросы

1. Что такое онтология? Опишите два основных значения этого термина.
2. Приведите формальную модель онтологии.
3. Приведите общепринятую классификацию онтологий.
4. Дайте характеристику онтологий верхнего уровня.
5. Дайте характеристику онтологий предметных областей.
6. Что такое прикладная онтология?

7. Для каких целей используются онтологии?
8. Каковы наиболее важные приложения онтологий?

8. Визуальное представление знаний

При разработке и использовании баз знаний удобно применять наглядные представления, т. е. различные изображения, схемы, рисунки, наброски. Визуализация всегда считалась мощным инструментами познания, т. е. средством, предназначенным для организации и облегчения процесса познания. Визуальные модели, например, графы, обладают особенной когнитивной силой при структурировании информации [4].

Методы визуализации идей, процессов, проектов, текстов в форме сетевых графов достаточно традиционны. Считается, что они являются инструментом, позволяющим сделать видимыми понятийные (или семантические) сети памяти человека. В когнитивной психологии (или психологии познания, от cognition – познание) достаточно много теорий, основанных на предположении, что человеческая память наиболее адекватно представляется именно сетевой структурой. Данная трактовка является развитием моделей представления знаний типа семантических сетей, которые считаются наиболее близкими человеческой структуре памяти. Такие концептуально-когнитивные структуры отличаются индивидуальностью, связанной с личностными, интеллектуальными и профессиональными различиями носителей знаний, иначе это когнитивные модели индивида.

8.1. Интеллект-карты

Интеллект-карты (mind maps) – один из наиболее привлекательных и простых способов отображения понятийных структур. Хотя автором идеи интеллект-карт считается Т. Бьюзен [3], в России такой метод успешно использовался в педагогике уже в 1925 г.

Т. Бьюзен сформулировал эту идею еще в 1970-е гг. в качестве компактного средства организации конспектов, позднее он понял, что метод гораздо шире и может использоваться как мощное орудие мышления применительно к научной работе, инновациям, бизнес-идеям, политическим дискуссиям, мозговому штурму, педагогике и пр.

Идея интеллект-карт сильна еще и тем, что имеет серьезный нейрофизиологический базис. Человеческий мозг обладает выраженной сетевой структурой. Естественные нейронные сети мозга включают триллион нейронов, каждый из которых может связываться примерно с 10 000 ближайших соседей. И хотя природа их взаимодействия исследована далеко не полностью, так называемый «радиантный» характер передачи возбуждения от центра на периферию доказан.

Т. Бьюзен использовал также некоторые идеи из теории межполушарной асимметрии и гештальт-психологии.

Идея интеллект-карт заключается в использовании и совмещении функции левого и правого полушарий для достижения целостного и наглядного представления идеи. Фактически это переход от последовательного (текстового) изложения к сетевому (образному). Интеллект-карта – графическое выражение процесса радиантного мышления.

Интеллект-карты имеют 4 отличительные черты:

- объект внимания/изучения кристаллизован в центральном образе;
- основные темы, связанные с объектом изучения, расходятся от центрального образа в виде ветвей, которые поясняются ключевыми словами или образами;
- вторичные идеи также ветвятся;
- ветви формируют связанную узловую структуру.

Т. Бьюзен сформулировал несколько практических советов по рисованию интеллект-карт:

- соблюдайте иерархию мыслей;
- используйте размер шрифта, толщину букв и цвет для большей выразительности;
- используйте графические образы;
- стремитесь к ясности;
- не перегружайте интеллект-карту.

Для визуализации интеллект-карт можно использовать любой программный графический пакет. Однако сейчас все большую популярность приобретают специализированные методы и инструменты. На рынке программного обеспечения имеются десятки средств для быстрого и красивого мгновенного формирования интеллект-карт (майнд мэппинга), например:

- Inspiration
- Map it! (by Tony Buzan)
- MindMapper Pro
- MindGenius Business, www.mindgenius.com
- Visual Mind, www.visual-mind.com
- Mind Pad
- Mind manager, www.mindjet.com
- Freemind, www. freemind.com
- The Brain
- Mind meister
- Conception и др.

На рис. 8.1 представлен пример интеллект-карты, сформированной при помощи инструмента Mind Manager.

Рис. 8.1. Интеллект-карта понятия «Лингвистический проект»

Чтобы создать хорошую интеллект-карту, необходимо либо быть экспертом, либо глубоко погрузиться в предметную область, выявить свойства отображаемых понятий.

8.2. Концептуальные карты

Если интеллект-карты показывают связи и древовидную структуру произвольных фрагментов знаний, то концептуальные карты (concept maps) или графы позволяют детальнее рассмотреть предметную область и включают отношения между понятиями или концептами. Такие концептуальные графы (карты) состоят из узлов и направленных поименованных отношений, или связей, соединяющих эти узлы. Связи могут быть различного типа, например, «является», «имеет свойство» и т. п. Концепты и связи имеют универсальный характер для некоторого класса понятий предметной области. Поэтому любая разработка концептуального графа подразумевает анализ структурных взаимодействий между отдельными понятиями предметной области.

На рис. 8.2 представлен фрагмент концептуальной карты, предназначенной для описания процесса управления нефтегазодобывающим предприятием.

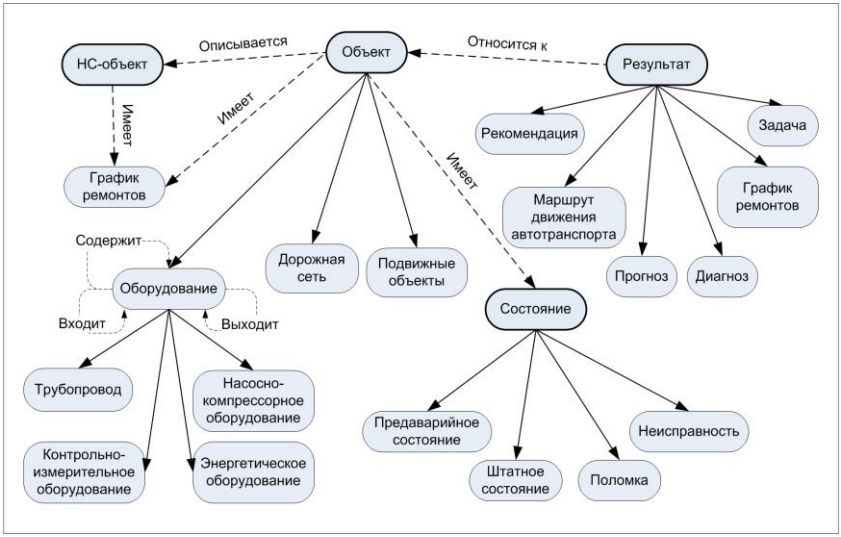

Рис. 8.2. Пример концептуальной карты

В процессе создания концептуальной карты эксперт и аналитик анализируют структуру отношений предметной области, что помогает им глубже понимать ее природу. Зачастую приходится генерировать новые, ранее невербализованные связи.

Впервые концептуальные карты были предложены Дж. Новаком в начале 1970-х гг. при изучении детского мышления и формирования первых научных понятий. Это исследование использовало идеи Д. Асубеля о формировании понятийного мышления. Концептуальные карты оказались эффективным инструментом отображения понятийной системы человека.

Визуальные спецификации в форме концептуальных карт могут использоваться не только при разработке баз знаний. Они широко используются в обучающих системах (E-learning) и в традиционном обучении.

И студенты, и преподаватели могут применять концептуальные карты в качестве инструментов для оценки изменений, произошедших в их мышлении. Р. Козма, один из разработчиков программы организации концептуальных карт – Learning Tool – считает, что эти средства являются инструментами познания (mind tool), усиливающими и расширяющими познания человека.

Разработка визуальных понятийных сетей требует от обучающихся:

- реорганизации знаний;
- исчерпывающего описания понятий и связей между ними;

- глубокой обработки знаний, что способствует лучшему запоминанию и извлечению из памяти знаний, а также повышает способности применять знания в новых ситуациях;
- связывания новых понятий с существующими понятиями и представлениями для улучшения понимания.

Полезность концептуальных карт, пожалуй, лучше всего демонстрируется их возможностью отображать формы мышления высшего порядка. Их применяют и для формальных обоснований в химии, и для аргументации высказываний в биологии.

Также было показано, что концептуальные карты полезны при описании процессов проведения научных исследований.

В простейшем случае построение концептуальной карты сводится к следующему:

- определению контекста путем задания конкретного фокусирующего вопроса (focus question), определяющего главную тему и границы концептуальной карты;
- выделению концептов – базовых понятий данной предметной области (обычно порядка 20 понятий);
- построению связей между концептами – определению отношений и взаимодействий базовых понятий;
- упорядочению графа – уточнению, удалению лишних связей, снятию противоречий.

«Хороший» граф обычно получается после 2–3 итераций.

Необходимо обратить внимание на типичные ошибки, которые совершают разработчики концептуальных карт:

- использование целых предложений вместо отдельных концептов в узле;
- построение линейных карт;
- наличие большого числа пересекающихся связей;
- наличие большого числа концептов на одной карте (часть концептов нужно вынести на другие карты);
- наличие неверно определенных типов отношений.

Еще одним преимуществом использования концептуальных карт в качестве средства структурирования знаний является системный подход к изучению предметной области. При этом достигаются*:*

- системность – концептуальная карта представляет целостный взгляд на предметную область;
- единообразие – материал, представленный в единой форме, гораздо лучше воспринимается и воспроизводится;
- научность – построение концептуальной карты позволяет восстановить недостающие логические связи во всей их полноте.

Стоит еще раз подчеркнуть, что концептуальный граф – не только цель, но и средство. В процессе построения, т. е. при взаимодействии семантических связей нашей памяти с визуальной информацией, связи перестраиваются, порождая, в свою, очередь новые знания.

8.3. Когнитивные карты

Когнитивная карта (от лат. *cognitio* – знание, познание) – образ знакомого пространственного окружения. Это понятие широко используется в психологии и было заимствовано в других областях. В инженерии знаний когнитивные карты используются как инструмент для м**етодологии когнитивного моделирования**, которая была предложена Р. Аксельродом и предназначена для анализа и принятия решений в плохо формализованных ситуациях.

При принятии решений в неструктурированных ситуациях у субъекта, т. е. лица, принимающего решение (ЛПР) или эксперта, возникает модель проблемной области, на основе которой он пытается объяснить происходящие в реальности процессы. При этом объективные закономерности реального мира представляются субъективными экспертными оценками. В результате образ наблюдаемой ситуации отражает не только законы и закономерности ситуации, но и мировоззрение субъекта, его систему убеждений, ценностей, уровень образования, опыт и т. д.

В этих условиях принятие решений – это искусство, включающее рациональные (логические) и интуитивные начала. В синтезе рационального и интуитивного возникает способность ЛПР принимать своевременные и адекватные решения.

Когнитивный подход к поддержке принятия решений ориентирован на то, чтобы активизировать интеллектуальные процессы субъекта и помочь ему зафиксировать свое представление проблемной ситуации в виде формальной модели. В качестве такой модели и используется **когнитивная карта ситуации**, которая представляет известные субъекту основные законы и закономерности наблюдаемой ситуации в виде ориентированного знакового графа, в котором вершины графа – это факторы (признаки, характеристики ситуации), а дуги между факторами – причинно-следственные связи между факторами. Дуги могут иметь вес, отражающий силу влияния факторов.

В когнитивной модели выделяют два типа причинно-следственных связей: положительные и отрицательные. При положительной связи увеличение значения фактора-причины приводит к увеличению значения фактора-следствия, а при отрицательной связи увеличение значения фактора-причины приводит к уменьшению значения фактора-следствия. Пример когнитивной карты приведен на рис. 8.3.

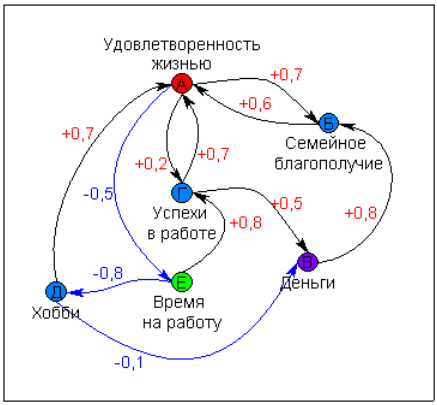

Рис. 8.3. Когнитивная карта ситуации «Удовлетворенность жизнью»

Причинно-следственный граф представляет собой упрощенную субъективную модель функциональной организации наблюдаемой системы и является «сырым» материалом для дальнейших исследований и преобразований – когнитивного моделирования.

Цель когнитивного моделирования заключается в генерации и проверке гипотез о функциональной структуре наблюдаемой ситуации до получения функциональной структуры, способной объяснить поведение наблюдаемой ситуации.

Создать когнитивную карту можно в любом графическом редакторе. Однако, основное ее предназначение – проведение когнитивного моделирования, а для этого нужны специальные инструменты.

8.4. Инструментарий IHMC CmapTools

Можно строить концептуальные карты на бумаге, на доске, в любом графическом или текстовом редакторе, но удобнее всего использовать свободно распространяемый инструментарий IHMC CmapTools. (Его можно скачать с сайта http://cmap.ihmc.us/).

Система CmapTools обладает интуитивно понятным интерфейсом (редактором) и позволяет пользователю легко строить концептуальные карты, вводя понятия – концепты и связывая их именованными отношениями. Редактор системы позволяет варьировать шрифты, цвета, толщину линий, устанавливать нужный фон.

Интересной особенностью системы является возможность связывать (ассоциировать) с любым понятием некий ресурс (текст, картинку, видео- или аудио-файл, страницу в сети Интернет).

Создаваемые с помощью инструментария IHMC CmapTools карты можно размещать как в закрытом пространстве, на жестком диске компьютера пользователя, так и в открытом пространстве, на удаленных серверах для совместного использования сообществом Cmap.

Программа может экспортировать концептуальные карты в графические образы, в формат XML, как web-страницу и др.

Контрольные вопросы

1. Что такое интеллект-карта? Назовите ее отличительные особенности.
2. Что такое концептуальная карта? Кратко опишите основные этапы построения концептуальной карты.
3. Что такое когнитивная карта? Какие типы связей задаются на когнитивной карте?
4. Для каких целей используются когнитивные карты?

9. Введение в экспертные системы

Целью инженерии знаний как прикладной науки является разработка программ, которые при решении задач, трудных для эксперта-человека, получают результаты, не уступающие по качеству и эффективности решениям, получаемым экспертом. Такие программы относятся к классу экспертных систем (ЭС).

9.1. Общее понятие экспертных систем

Что же такое экспертные системы? Дадим несколько определений. Первое из них отражает характеристики ЭС с точки зрения пользователя, второе – со стороны разработчика (инженера знаний).

Определение 1. Экспертные системы – это программные комплексы, аккумулирующие знания и опыт специалистов в некоторой предметной области с целью их тиражирования для консультации менее квалифицированных пользователей.

Определение 2. Экспертные системы – это класс программных систем, основанных на знаниях. Их вычислительные возможности определяются в первую очередь наращиваемой базой знаний и только во вторую очередь используемыми методами.

В дальнейшем с этих двух сторон и будем рассматривать экспертные системы.

9.2. Особенности и назначение экспертных систем

Экспертные системы предназначены для решения так называемых неформализованных задач.

Согласно А. Ньюэллу и М. Саймону [35], к неформализованным (ill-structured) относятся такие задачи, которые обладают одной или несколькими из следующих характеристик:

- задачи не могут быть заданы в числовой форме;
- цели не могут быть выражены в терминах точно определенной целевой функции;
- не существует алгоритмического решения задач; либо оно существует, но его нельзя использовать из-за ограниченности ресурсов (времени, памяти).

Следует подчеркнуть, что неформализованные задачи представляют большой и очень важный класс задач, представляющих большое практическое значение.

Таким образом, экспертные системы, как и системы искусственного интеллекта в целом, отличаются от систем обработки данных тем, что в них в основном используются символьный (а не числовой) способ представления, символьный вывод и эвристический поиск решения (а не простое исполнение известного алгоритма).

В связи с тем, что ЭС предназначены для решения неформализованных задач, для них характерны следующие особенности:

- Алгоритм решения задачи не известен заранее, а строится самой ЭС с помощью символических рассуждений, базирующихся на эвристических приемах;
- «Прозрачность» (ясность) полученных решений, т. е. система «осознает» в терминах пользователя, как она получила решение, и может проанализировать и объяснить свои действия и знания;
- Способность приобретения новых знаний от пользователя-эксперта и изменения в соответствии с ними своего поведения;
- Способность системы вести диалог о решаемой задаче на языке, понятном и удобном пользователю, т. е. ЭС должна обладать «дружественным» интерфейсом.

9.3. Структура и режимы работы экспертных систем

Экспертные системы строятся на следующих трех принципах.

1. Мощность ЭС обусловлена в первую очередь мощностью ее базы знаний и возможностью ее пополнения, и только во вторую очередь – используемыми ею методами (процедурами).

2. Знания, позволяющие экспертной системе получать качественные и эффективные решения задач, большой частью получены от экспертов и являются в основном эвристическими, экспериментальными, неопределенными, правдоподобными.

3. Учитывая неформализованность решаемых задач и эвристический, личностный характер используемых знаний, пользователь должен иметь возможность непосредственного взаимодействия с экспертной системой в виде диалога.

Архитектура экспертной системы вытекает из принципов, сформулированных выше.

Типовая ЭС состоит из следующих основных компонентов (рис. 9.1): решателя (машины вывода), рабочей памяти, базы знаний, подсистемы приобретения знаний, подсистемы объяснений и пользовательского интерфейса.

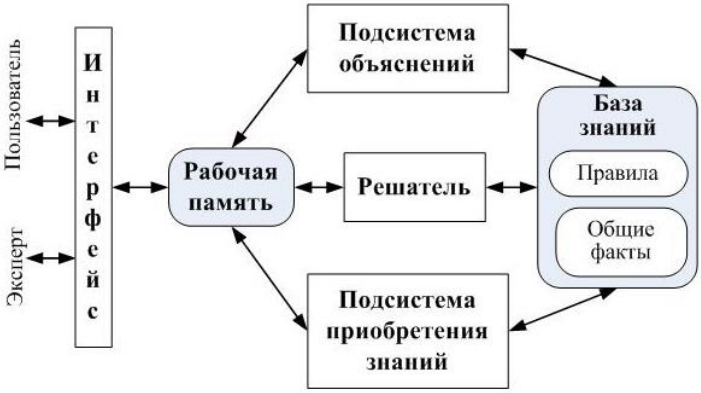

Рис. 9.1. Архитектура типовой экспертной системы

Рабочая память (РП) предназначена для хранения исходных и промежуточных данных решаемой в текущий момент задачи.

База знаний (БЗ) включает универсальные данные (факты), описывающие рассматриваемую область знаний, и множество правил, описывающих целесообразные преобразования данных в этой области.

Решатель, используя исходные данные из РП и знания из БЗ, формирует такую последовательность правил, которые, будучи примененными к исходным данным, приводят к решению задачи.

Подсистема приобретения знаний автоматизирует процесс наполнения ЭС знаниями, осуществляемый, как правило, пользователем-экспертом.

Подсистема объяснений отвечает на вопросы о том, как ЭС получила решение задачи (или почему она не получила решение) и какие знания при этом использовала, что повышает доверие пользователя к полученному результату и облегчает эксперту тестирование системы.

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

В разработке ЭС участвуют представители следующих специальностей:

- эксперт в проблемной области, задачи которой будет решать ЭС;
- инженер знаний – специалист по представлению знаний;
- программист по разработке инструментальных средств (ИС), предназначенных для ускорения разработки ЭС.

Необходимо отметить, что отсутствие среди участников разработки инженеров знаний (т. е. их замена программистами) либо приводит к неудаче процесс создания ЭС, либо значительно удлиняет его.

Эксперт определяет знания (данные и правила), характеризующие проблемную область, обеспечивает полноту и правильность введенных в ЭС знаний.

Инженер знаний помогает эксперту выявить и структурировать знания, необходимые для работы ЭС; осуществляет выбор того ИС, которое наиболее подходит для данной проблемной области, и определяет способ представления знаний в этом ИС; выделяет и программирует (традиционными средствами) стандартные функции (типичные для данной проблемной области), которые будут использоваться в правилах, вводимых экспертом.

Программист разрабатывает ИС (если ИС разрабатывается заново), содержащее в пределе все основные компоненты ЭС, и осуществляет его сопряжение с той средой, в которой оно будет использовано.

Экспертная система работает **в двух режимах**: режиме приобретения знаний и в режиме решения задачи (называемом также режимом консультации, или режимом использования ЭС).

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

В режиме консультации (или рабочем режиме, т. е. режиме решения задач) общение с ЭС осуществляет конечный пользователь, которого интересует результат и (или) способ его получения.

В режиме консультации данные о задаче пользователя после обработки их диалоговым компонентом поступают в рабочую память. Решатель на основе входных данных из рабочей памяти, общих данных о проблемной области и правил из БЗ формирует решение задачи. Если реакция системы не понятна пользователю, то он может запросить объяснения.

9.4. Классификация экспертных систем

Экспертные системы могут быть классифицированы по различным признакам. Рассмотрим классификацию ЭС по наиболее важным из них.

По назначению:

- Решение задач.
- Обучение специалистов.
- Тиражирование знаний экспертов.
- Автоматизация рутинных работ.

По типам задач:

- Интерпретация (символов или сигналов).
- Прогнозирование (предсказание).
- Диагностика.
- Конструирование.
- Планирование.
- Мониторинг (слежение).
- Управление.

По изменяемости/неизменяемости данных в ходе решения задачи:

- Статические.
- Динамические.

Статические ЭС используются в тех приложениях, где можно не учитывать изменения окружающего мира, происходящие за время решения задачи.

Динамические ЭС учитывают изменения, происходящие в окружающем мире во время исполнения приложения. Для учета динамики в архитектуру динамической ЭС дополнительно вводятся два компонента: подсистема моделирования внешнего мира и подсистема связи с внешним окружением. Последняя осуществляет связи с внешним миром через систему датчиков и контроллеров.

Рассмотрим еще одну классификацию ЭС, отражающую их «зрелость».

По степени проработанности и отлаженности различают**:**

- Демонстрационный прототип
  (решает часть требуемых задач, демонстрируя применимость метода ЭС)
- Исследовательский прототип
  (решает все задачи, но неустойчив в работе и не полностью оттестирован)
- Действующий прототип
  (надежно решает все задачи, но для решения сложных задач может потребоваться слишком много времени или памяти)
- Промышленная ЭС
  (обеспечивает высокое качество решений всех задач при минимуме затрат времени и памяти; получен из действующего прототипа пу-

тем расширения БЗ и/или перепрограммирования на более эффективных языках)

– Коммерческая ЭС
(хорошо отлажена и документирована, и может быть передана сторонним пользователям)

9.5. Примеры известных экспертных систем

Рассмотрим примеры наиболее известных экспертных систем [23]. При описании ЭС, в первую очередь, будем описывать ее назначение, способ представления знаний и инструментарий, с помощью которого ЭС была реализована.

1. Система MYCIN (Мицин).

Самая известная система, прототип многих последующих ЭС.

Назначение: постановка диагноза и определение методов лечения инфекционных заболеваний крови. Она «диагностирует» и «лечит» 100 известных ей заболеваний.

По качеству решений задач MYCIN не уступает человеку-эксперту.

Знания в MYCIN представлены в виде фактов и 400 правил вида «Условие → Действие».

«Условие» – это булево выражение из предикатных функций, применяемых к фактам.

«Действие» – выведенный факт или операция над фактом.

Выведенный факт имеет коэффициент определенности (КО), который вычисляется по КО фактов, входящих в «Условие».

Если КО меньше заданного в системе порогового значения, то его значение приравнивается нулю.

В системе есть метаправила, которые могут «включать» и/или «выключать» другие правила.

Процедура вывода реализуется в виде исчерпывающего поиска, направляемого целями.

Диалог с пользователем системы MYCIN ведется на ограниченном естественном языке (на основе шаблонов).

В системе MYCIN имеется **подсистема объяснений**, которая отвечает на вопросы ПОЧЕМУ и КАК:

ПОЧЕМУ был использован тот или иной факт?

КАК был получен данный факт?

Система обладает способностями **приобретать** новые и модифицировать имеющиеся **правила**.

На основе MYCIN была создана инструментальная система EMYCIN (Empty MYCIN), т. е. пустая MYCIN (с пустой базой знаний). Такие системы называются **оболочками ЭС.**

EMYCIN является предметно-независимой и так же, как и MYCIN ориентирована на решение задач диагностики. Заполняя ее БЗ новыми знаниями, можно получать новые экспертные системы.

Система MYCIN реализована на языке LISP.

2. Система PROSPECTOR.

Это промышленная ЭС. Ее область знаний – геология.

Назначение: оказание помощи геологу в определении наличия месторождения руды заданного вида на основе анализа геологических данных.

По качеству решений задач PROSPECTOR не уступает эксперту.

В системе используется нечеткая математика (нечеткие множества, которые дают меру оценки истинности данного утверждения).

При поиске решений используется как стратегия, направляемая целями, так и стратегия, направляемая данными.

PROSPECTOR обладает всеми свойствами классических ЭС.

С помощью данной ЭС было найдено месторождение молибдена, стоимость которого оценивалась в $ 100 млн.

На основе PROSPECTOR была разработана инструментальная система KAS, независимая от предметной области, т. е. оболочка ЭС.

Система **PROSPECTOR** реализована на языке INTERLISP.

3. **Система R1/XCON**.

Коммерческая ЭС. Область знаний – вычислительная техника.

Назначение:

На основании заказа пользователя, приобретающего требуемую ему конфигурацию вычислительной системы VAX-11/780 фирмы DEC, ЭС выполняет следующие функции:

- Проверяет заказ на совместимость компонент и выявляет недостающие компоненты;
- Выдает в виде диаграммы конечную конфигурацию VAX, которая используется техническими службами при установке системы заказчику;
- Учитывает при построении диаграммы ограничения, накладываемые заказчиком (порядок расположения компонент, тип и длина кабелей и т. п.).

Сложность решаемых R1 задач обусловлена сложностью системы VAX (420 компонент и множество правил их взаимодействия).

По качеству работы превосходит человека-эксперта (2,5 минуты против нескольких часов и допущения ошибок).

Система R1 разработана средствами языка OPS 5 и включает около 2500 правил.

4. **Система DENDRAL.**

Промышленная ЭС. Область знаний – химия.

Назначение: система определяет возможные структуры молекулы на основе химической формулы и масс-спектрограммы.

Система намного превосходит способности человека.

Разработана средствами языка INTERLISP.

5. **Система MACSYMA**.

Область знаний – математика.

Назначение: символьные математические преобразования. Выполняет символьное дифференцирование и интегрирование и упрощает выражения.

Создана средствами LISP.

6. **Система DI*GEN**.

DI*GEN – это оболочка для построения диагностических экспертных систем [7].

Проблемная область – диагностика. Может применяться в медицине и технике.

Объем базы знаний в оболочке DI*GEN может достигать 2000 понятий, которым соответствует несколько десятков тысяч продукционных правил. В описание базы знаний могут входить несколько десятков аномальных состояний, относительно которых экспертная система выполняет диагностику. В медицине аномальное состояние характерно для больного человека, в технических системах такое состояние, диагностируется по наличию отклонений от нормального режима работы системы (например, доменной печи).

С помощью оболочки DI*GEN совместно с НПО "Черметавтоматика" (г. Москва) была разработана промышленная ЭС ДОМНА, предназначенная для диагностики оборудования и хода производственного процесса доменной печи. Также с помощью оболочки DI*GEN совместно со специалистами Института экспериментальной и клинической медицины г. Новосибирска была создана ЭС КАРДИОЛОГ для диагностики сердечно-сосудистых заболеваний.

Разработана ЭС средствами языка C++.

Контрольные вопросы

1. Что такое экспертная система (ЭС)? Дайте определение ЭС.
2. Какой компонент экспертной системы в наибольшей степени влияет на ее мощность и полезность?
3. Какие задачи относят к неформализованным? Назовите их характеристики.
4. Приведите структуру типовой экспертной системы.
5. Приведите классификацию экспертных систем по степени проработанности и отлаженности.
6. В чем отличие статических экспертных систем от динамических?

7. Опишите назначение и основные принципы построения экспертной системы MYCIN.

10. Объяснения в экспертной системе

10.1. Принципы построения подсистемы объяснений

Как правило, в ЭС поддерживается три вида объяснений:

1) Объяснение действий (рассуждений) системы в ходе решения задачи;
2) Ответы на вопросы о динамических знаниях системы;
3) Ответы на вопросы о статических знаниях системы.

Рассмотрим сначала принципы объяснения действий (рассуждений) системы.

В связи с тем, что ЭС решает задачи с помощью поиска в дереве целей (рис. 10.1), возможны две разновидности вопросов:

1. Вопросы ПОЧЕМУ, т. е. какую текущую цель преследовала система, совершая данное действие.

2. Вопросы КАК X, (где X – ссылка на некоторую цель или подцель), т. е. с помощью каких действий система достигла цели X.

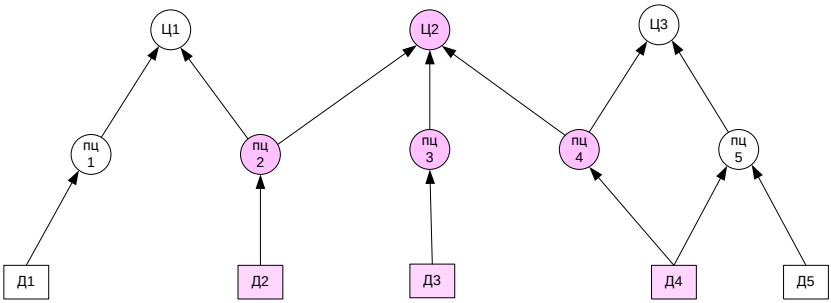

Рис. 10.1. Дерево целей

При ответе на вопрос ПОЧЕМУ движение по дереву целей (рис. 10.2) осуществляется вверх от текущей цели к цели, объясняющей, зачем достигается текущая цель (например, ПОЧЕМУ Вас интересует значение температуры?).

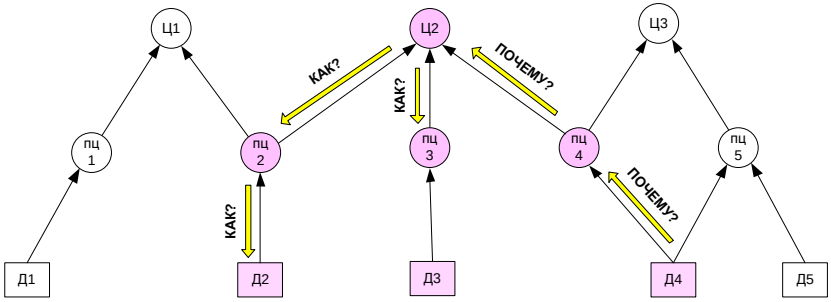

Рис. 10.2. Навигация по дереву целей

При ответе на вопрос КАК Х движение идет вниз от указанной цели – для рассмотрения того, как (каким способом) была достигнута цель Х.

Вопросы ПОЧЕМУ и КАК могут рассматриваться как средство для просмотра дерева целей. Комбинируя последовательность из вопросов ПОЧЕМУ и КАК, пользователь может получить объяснение на различных уровнях конкретности.

Вопрос ПОЧЕМУ также может иметь параметр, указывающий на сколько шагов осуществляется перемещение по дереву (по умолчанию – на 1 шаг).

Ответы на вопросы о динамических знаниях системы сводятся к поиску соответствующей информации в рабочей памяти ЭС.

Ответы на вопросы о статических знаниях ЭС требуют поиска всех правил, посылки и действия которых соответствуют обрабатываемому вопросу. Для этого анализируются списки, связанные с каждым параметром объекта экспертизы: в одном списке – перечень правил, использующих данный параметр в посылке, в другом – перечень правил, использующих параметр в заключении.

Также возможны вопросы о предметной области, о свойствах тех или иных сущностей (например, болезнях).

10.2. Основные достоинства и недостатки подсистемы объяснений

Основные достоинства подсистемы объяснений в ЭС:

1) объяснения помогают пользователю использовать систему для решения своих задач;
2) объяснения повышают степень доверия пользователя к ЭС, позволяя пользователю убедиться в правильности полученных результатов;
3) объяснения служат для обучения пользователя;
4) объяснения служат для отладки базы знаний ЭС.

Основные недостатки подсистемы объяснений в ЭС:

1) запросы на объяснение интерпретируются только в одном узком смысле (вопросы ПОЧЕМУ и КАК интерпретируются только в терминах целей и правил);
2) не все действия системы могут быть объяснены (например, почему сначала проверялась одна гипотеза, а потом другая);
3) объяснения основываются, фактически, на треке выполнения программы, поэтому при смене интерпретатора необходимо менять и систему объяснений.

Контрольные вопросы

1. Что такое система объяснений экспертной системы и на каких принципах она строится?
2. На какие вопросы отвечает система объяснений экспертной системы? Поясните смысл ответов.
3. Назовите достоинства и недостатки современных систем объяснений экспертной системы.

11. Построение баз знаний экспертных систем

Как было сказано выше, мощность ЭС определяется в первую очередь мощностью ее базы знаний и возможностью ее пополнения. Поэтому процесс построения базы знаний ЭС не только самый трудоемкий этап разработки ЭС, но и самый ответственный.

11.1. Приобретение знаний

Определение 11.1. Процесс получения знаний от эксперта (или другого источника знаний) и передача их экспертной системе называется **приобретением знаний.**

Источниками знаний для ЭС являются:

1) человек-эксперт,
2) тексты, в которых содержатся сведения о предметной области,
3) эмпирические данные (таблицы, базы данных).

Важность процесса приобретения знаний обусловлена тем, что качество и эффективность решения задач ЭС определяется качеством и количеством используемых ею знаний.

Сложность процесса приобретения знаний объясняется следующими факторами:

1) объем знаний, используемых экспертом, достаточно велик и
2) не все эти знания полностью осознаются экспертом.

Э. В. Попов [23] различает 3 фазы приобретения знаний, отражающие изменение функций участников разработки ЭС (эксперта (Э), инженера знаний (ИЗ)) и самой ЭС):

1) Предварительная фаза.
2) Начальная фаза.
3) Фаза накопления.

Предварительная фаза характеризуется тем, что ЭС еще не существует. На этой фазе инженер знаний должен получить основные знания от эксперта (основные понятия, отношения, задачи и т. п.) и на этой основе сформировать общее представление о структуре знаний и данных, а также принципах построения ЭС. Эта фаза соответствует таким этапам разработки ЭС, как *идентификация, концептуализация и формализация* (см. главу 13).

На **начальной фазе** осуществляется наполнение системы знаниями о представлении, т.е. определяются и задаются соответствующие структуры для организации и представления знаний в базе знаний ЭС. Эти знания может вводить только инженер знаний.

Эта фаза соответствует первой стадии этапа *реализации* экспертной системы.

В ходе **фазы накопления** осуществляется приобретение основных знаний о предметной области. Это делается совместно инженером знаний и экспертом, но возможно и без участия инженера знаний.

На этой фазе решаются такие задачи:

1) Обнаружение неполноты, некорректности или противоречивости знаний.
2) Извлечение новых знаний, устраняющих недостатки из п.1.
3) Преобразование новых знаний в вид, понятный ЭС.
4) Объединение «новых» знаний со «старыми».

11.2. Модели приобретения знаний.

Процесс приобретения знаний является не только наиболее важным, но и наиболее сложным этапом разработки экспертной системы. Это объясняется тем, что обычно инженер знаний плохо разбирается в предметной области, а эксперт не знает методов инженерии знаний. Здесь требуется совместная работа инженера знаний и эксперта по выработке и уточнению терминологии предметной области. Одной из важных задач для инженера знаний является помощь эксперту в структурировании его знаний о предметной области.

Процесс приобретения знаний можно свести к последовательности выполнения следующих задач:

1) определяется необходимость модификации (расширения) знаний;

2) при необходимости модификации знаний осуществляется извлечение новых знаний, в противном случае процесс приобретения знаний заканчивается;
3) новые знания преобразуются в форму, «понятную» экспертной системе;
4) знания системы модифицируются, и осуществляется переход к первой задаче.

Выделяются пять исторически сложившихся *моделей приобретения знаний*. Они различаются разной степенью вовлеченности экспертов в процесс приобретения знаний для ЭС, а иногда и в сам процесс создания ЭС.

11.2.1. Модель приобретения знаний ранними системами ИИ

В ранних системах ИИ **знания не отделялись от механизмов вывода** и взаимодействие с системой осуществлял только программист, Схема взаимодействия специалистов выглядела, как показано на рис. 11.1.

Рис. 11.1. Ранняя модель приобретения знаний

Программист должен был освоить с помощью эксперта предметную область и затем при разработке системы самому выступать в роли «эксперта».

Недостаток знаний о ПО не позволял программисту гарантировать полноту и непротиворечивость приобретенных знаний. Кроме того неизбежные модификации системы (при отсутствии разделения системы на базу знаний и механизм вывода) приводили к невозможности сохранения уже достигнутой непротиворечивости знаний.

11.2.2. Модель приобретения знаний ЭС с помощью инженера знаний

В этой и последующих моделях **знания отделены от механизмов вывода** и оформлены в виде **базы знаний.**

В модели, показанной на рис. 11.2., эксперт взаимодействует с ЭС либо непосредственно, либо через инженера знаний.

Преимущество данной модели по сравнению с предыдущей состоит в разделении базы знаний и механизма вывода, что упрощает модификацию знаний. В данной модели первую и вторую задачи приобретения знаний выполняет эксперт с помощью инженера знаний, третью задачу выполняет инженер знаний, а четвертую – экспертная система.

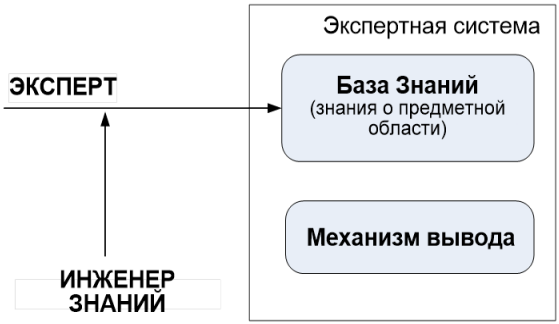

Рис. 11.2. Модель приобретения знаний ЭС с помощью инженера знаний

Данная модель приобретения знаний является самой популярной, так как позволяет строить хорошие по качеству базы знаний.

Важным недостатком этой модели является большая трудоемкость. Действительно, из четырех задач по приобретению знаний автоматизирована только одна.

11.2.3. Модель приобретения знаний ЭС с помощью интеллектуального редактора

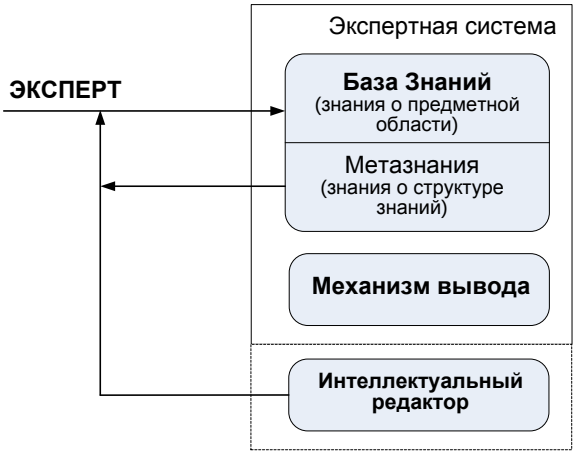

Рис. 11.3. Модель приобретения знаний ЭС с помощью интеллектуального редактора

В модели, изображенной на рис. 11.3, присутствует интеллектуальный редактор (И-редактор), обладающий развитыми диалоговыми способностями и значительными знаниями о структуре базы знаний (метазнаниями). И-редактор сам преобразует знания в форму, понятную ЭС, и модифицирует БЗ (объединяет новые знания со старыми).

В этой модели интеллектуальный редактор может быть включен в состав ЭС, а может быть отдельным приложением.

11.2.4. Модель приобретения знаний ЭС с помощью индуктивной программы

В этой модели ЭС приобретает знания аналогично человеку-эксперту.

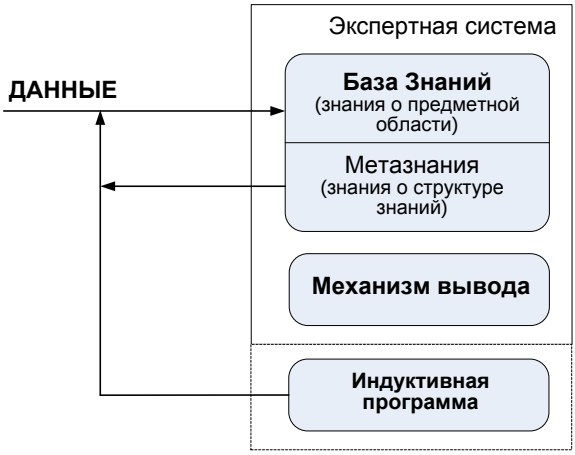

Рис. 11.4 Модель приобретения знаний ЭС с помощью индуктивной программы

Индуктивная программа анализирует данные, содержащие сведения о некоторой предметной области, автоматически формирует значимые отношения и правила, описывающие предметную область.

Основное достоинство этой модели – автоматизация всех основных фаз приобретения знаний.

11.3.5. Модель приобретения знаний ЭС с помощью программы понимания текста

В этой модели специальная программа «читает» и «понимает» тексты (книги, статьи и т. п.), извлекая из них знания.

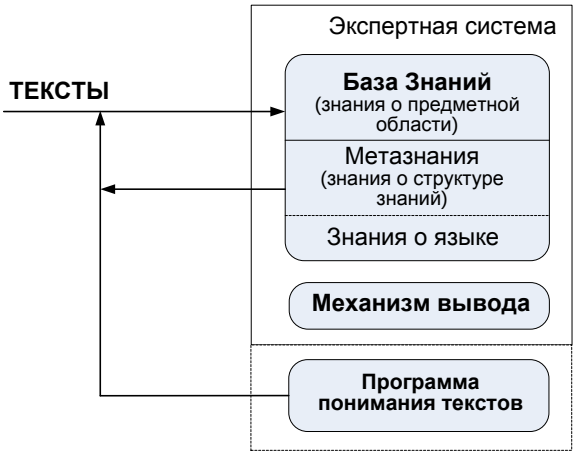

Рис. 11.5 Модель приобретения знаний ЭС с помощью программы понимания текста

Сложность задачи понимания текста состоит не только в обработке естественного языка, но и в необходимости воссоздать по тексту модель некоторой предметной области.

Контрольные вопросы

1. Что такое приобретение знаний? Дайте определение и приведите основные источники знаний.
2. Какие существуют фазы приобретения знаний?
3. Какие модели приобретения знаний Вам известны?
4. Назовите наиболее популярную (используемую) модель приобретения знаний. Расскажите о ее особенностях.

12. Методы извлечения знаний

12.1. Классификация методов извлечения знаний

Будем придерживаться классификации методов извлечения знаний (рис. 12.1), предложенной Т. А. Гавриловой [6]. Основной принцип деления методов извлечения знаний в этой классификации связан с источником знаний. Согласно этому принципу методы извлечения знаний делятся на *коммуникативные* и *текстологические*.

Коммуникативные методы охватывают все виды контактов с живым источником знаний – человеком–экспертом.

Текстологические методы предполагают извлечение знаний из документов (методик, пособий, руководств) и специальной литературы (статей, монографий, учебников).

Рис. 12.1. Классификация методов извлечения знаний

Разделение методов на эти две группы не означает их антагонистичности – обычно инженер знаний комбинирует различные методы: сначала изучает литературу, а затем идет беседовать с экспертом, и наоборот.

Коммуникативные методы, в свою очередь, делятся на *активные* и *пассивные*.

В **активных методах** ведущая роль, инициатива принадлежит инженеру знаний, который активно контактирует с экспертом различными способами – в диалогах, играх и т. п.

В **пассивных** методах ведущую роль играет эксперт, а инженер знаний только протоколирует рассуждения эксперта во время его реальной работы по принятию решений или записывает то, что эксперт считает нужным сказать во время беседы или лекции. Пассивные методы на первый взгляд просты, но они требуют от инженера знаний умения четко анализировать «поток сознания» эксперта и выявлять в нем значимые фрагменты знаний.

Активные и пассивные методы могут чередоваться даже в рамках одного сеанса извлечения знаний.

Активные методы делятся на две группы в зависимости от числа участвующих экспертов – *индивидуальные* и *групповые*.

Индивидуальные методы предполагают работу инженера знаний с одним экспертом. Они являются ведущими методами, поскольку подавляющая часть знаний извлекается из эксперта при личном общении.

Групповые методы играют вспомогательную роль. Они служат для активизации мышления участников дискуссий и позволяют выявлять весьма нетривиальные аспекты их знаний.

12.2. Критерии выбора метода извлечения знаний

На выбор метода извлечения знаний влияют 3 фактора:

- Личностные особенности инженера знаний.
- Личностные особенности эксперта.
- Характер предметной области.

По психологическим характеристикам люди делятся на 3 типа:

1. Мыслитель (познавательный тип).
2. Собеседник (эмоционально-познавательный тип).
3. Практик (практический тип).

Мыслители ориентированы на интеллектуальную работу, учебу, теоретические обобщения.

Собеседники – это общительные, открытые люди, готовые к сотрудничеству.

Практики предпочитают разговорам действие, хорошо реализуют замыслы других, нацелены на результативную работу.

Для характеристики предметной области используют такое ее качество как *структурированность*, т. е. степень теоретического осмысления и выявления основных закономерностей и принципов, действующих в данной предметной области.

Хорошо структурированная предметная область – теоретически устоявшаяся область, область с четкой аксиоматизацией и с широким применением математического аппарата.

Средне структурированная предметная область – область с определившейся терминологией, явными взаимосвязями между явлениями, но развивающейся теорией.

Слабо структурированная предметная область – область с размытыми определениями, скрытыми взаимосвязями между явлениями, богатой эмпирикой, с большим количеством «белых пятен».

12.3. Пассивные методы извлечения знаний

Пассивные методы извлечения знаний делятся на *наблюдения, вербальные отчеты, лекции*.

12.3.1. Наблюдение

Наблюдение обычно используется на начальных стадиях разработки базы знаний и применяются не самостоятельно, а в сочетании с другими методами. Суть метода: инженер знаний находится непосредственно рядом с экспертом во время его профессиональной деятельности или имитации таковой. При этом инженер знаний заранее просит эксперта максимально полно комментировать все его действия. Инженер знаний записывает все действия эксперта, его реплики, объяснения (возможна видеозапись сеанса).

Непременное условие – невмешательство инженера знаний в работу эксперта. Поэтому именно наблюдение – единственно «чистый» метод, исключающий вмешательство инженера знаний и навязывание им каких-то своих представлений эксперту.

Существует две разновидности проведения наблюдений:

- наблюдение за реальным процессом,
- наблюдение за имитацией процесса.

Обычно используют обе разновидности метода.

Заметим, что наблюдение очень важно для *разработки пользовательского интерфейса*, так как разработчикам ЭС (инженерам знаний) очень важно видеть и понимать реальный процесс работы эксперта.

Сеансы наблюдений требуют от инженера знаний следующих умений:

- овладения техникой стенографии,
- знаний методик хронометрирования производственного процесса,
- навыков «чтения по глазам»,
- серьезного предварительного знакомства с предметной областью.

Протоколы наблюдений тщательно *расшифровываются* и затем *обсуждаются* с экспертом, чтобы избежать ошибок и неточностей.

12.3.2. Вербальные отчеты

Вербальные отчеты, или протоколирование «мыслей вслух» отличается от наблюдений тем, что инженер знаний просит эксперта не просто комментировать свои действия и решения, но и объяснять, как это решение было найдено, т. е. продемонстрировать всю цепочку рассуждений. Все это протоколируется инженером знаний. Использование магнитофонов и диктофонов нежелательно – они сковывают эксперта.

Основная трудность метода состоит в том, что для любого человека принципиально сложно *объяснить, как он думает*. Люди не всегда способны достоверно описывать свои мыслительные процессы. Вообще часть

знаний у человека существует в невербальной форме, не поддается словесному описанию (различные процедурные знания типа «как завязывать шнурки». Хотя есть люди (эксперты), склонные к рефлексии, для которых эта работа вполне доступна.

Удачное протоколирование «мыслей вслух» делает этот метод наиболее эффективным методом извлечения знаний, поскольку эксперт может проявить себя максимально ярко и полно. Здесь эксперт может блеснуть свой эрудицией, показать всю глубину своих знаний.

Этот метод наиболее лестный и приятный для эксперта. Он требует от инженера знаний тех же навыков, что и *наблюдение*.

Также требуется расшифровка протоколов «мыслей вслух» – вербальных отчетов и их обсуждение с экспертом.

12.3.3. Лекции

Лекция – самый старый и известный способ передачи знаний.

Инженер знаний должен уметь слушать лекции, так как он не может учить эксперта, как нужно читать лекции. Для повышения эффективности данного метода, инженер знаний должен сформулировать для эксперта тему цикла лекций, а также тему и задачу каждой лекции (чтобы ограничить степень его свободы). Например, тема цикла лекций «Постановка диагноза – воспаление легких». Тема конкретной лекции – «Анализ рентгенограмм», задача лекции – научить слушателей по перечисленным экспертом признакам ставить диагноз «воспаление легких» и делать прогноз. При такой постановке эксперт может заранее структурировать свои знания и ход рассуждений. Инженеру знаний нужно только все внимательно записать.

Писать конспект – это искусство: нужно записывать главное, опускать второстепенное, выделять фрагменты знаний (параграфы, подпараграфы), уметь обобщать.

В конце или по ходу лекции необходимо задать уточняющие вопросы. Хороший вопрос по ходу лекции помогает и лектору, и слушателю.

Опытный лектор разбивает все вопросы условно на 3 группы:

- умные вопросы, углубляющие лекцию,
- глупые вопросы, или вопросы «не по существу»,
- вопросы «на засыпку», или провокационные вопросы.

Цикл (курс) лекций обычно включает 3–5 лекций. Продолжительность лекции 40–50 мин с 5–10-минутным перерывом. Этот метод, как и первые два, используется на начальной стадии разработки базы знаний для эффективного погружения в предметную область.

12.4. Активные индивидуальные методы

Активные индивидуальные методы извлечения знаний на сегодняшний день наиболее распространены. В той или иной степени к ним прибегают при разработке практически любой ЭС. К числу основных активных методов можно отнести *анкетирование, интервью, свободный диалог* и *игры с экспертами*.

Во всех этих методах активную роль играет инженер знаний, который пишет сценарий и режиссирует сеансы извлечения знаний. Первые три метода, которые можно назвать вопросными методами поиска знаний, схожи между собой и различаются лишь степенью свободы, которую может себе позволить инженер знаний при проведении сеансов извлечения знаний. Игры же имеют ряд существенных отличий.

12.4.1. Анкетирование

Анкетирование – наиболее жесткий метод, поскольку он наиболее стандартизирован. При использовании этого метода инженер знаний заранее составляет вопросник или анкету, размножает ее и использует для опроса нескольких экспертов.

Сама процедура может проводиться двумя способами:

1. Инженер знаний вслух задает вопросы и сам заполняет анкету по ответам экспертов.
2. Эксперт самостоятельно заполняет анкету после предварительного инструктирования.

Выбор способа зависит от конкретных условий (например, от оформления анкеты, ее понятности, готовности экспертов). Первый способ проще и для экспертов, и для инженеров знаний, однако, во втором способе эксперт не ограничен временем на обдумывание ответов.

Вопросник (анкета) заслуживает особого внимания. Существует несколько общих рекомендаций при составлении анкет, наибольший опыт работы с которыми накоплен в социологии и психологии. Вот несколько рекомендаций, которые достаточно универсальны, т. е. не зависят от предметной области.

1. Анкета не должна быть монотонной и однообразной, т. е. вызывать скуку или усталость. Это достигается вариациями формы вопросов, сменой тематики, вставкой вопросов-шуток и игровых вопросов.
2. Анкету следует приспосабливать к языку экспертов.
3. Вопросы влияют друг на друга, поэтому их последовательность должна быть строго продуманной.
4. Желательно стремиться к оптимальной избыточности. Известно, что в анкете всегда много лишних вопросов; часть из них необходима – это так называемые контрольные вопросы, а другая часть должна быть минимизирована.

5. Анкете необходимы «хорошие манеры», т. е. ее язык должен быть ясным, понятным, предельно вежливым.

12.4.2. Интервью

Под **интервью** понимается специфическая форма общения инженера знаний и эксперта, в которой инженер знаний задает серию заранее подготовленных вопросов с целью извлечения знаний о предметной области.

Интервью близко тому способу анкетирования, когда инженер знаний сам заполняет анкету, занося в нее ответы экспертов. Основное преимущество интервью в гибкости; в этих условиях инженер знаний может опускать ряд вопросов в зависимости от ситуации, вставлять новые, изменять темп, разнообразить ситуацию общения. Кроме того, у инженера знаний появляется возможность «взять в плен» эксперта своим обаянием, заинтересовать его самой процедурой и тем самым увеличить ее эффективность.

Роль вопросов в интервью очень важна. На качество интервью в значительной степени влияют следующие характеристики вопросов:

- язык вопроса (понятность, лаконичность, терминология);
- порядок вопросов (логическая последовательность и немонотонность);
- уместность вопросов (этика, вежливость).

Вопрос в интервью – это не просто средство общения, но и способ передачи мыслей и позиции инженера знаний. Отсюда необходимость в протоколах фиксировать не только ответы, но и вопросы, предварительно тщательно отрабатывая их форму и содержание.

Очевидно, что любой вопрос имеет смысл только в контексте. Поэтому вопросы может готовить только инженер знаний, уже овладевший спецификой работы экспертов в данной предметной области.

Важно заметить, что вопросы имеют для эксперта диагностическое значение – несколько откровенно глупых вопросов могут полностью разочаровать эксперта и отбить у него охоту к дальнейшему сотрудничеству.

Следует заметить, что для проверки достоверности и объективности информации, полученной в интервью ранее, необходимо применять контрольные вопросы. Контрольные вопросы должны быть составлены хитро, чтобы не обидеть эксперта недоверием (для этого используют повторение вопросов в другой форме, уточнения, ссылки на другие источники).

Важно также кроме вербальных вопросов (традиционных устных вопросов) применять вопросы с использованием наглядного материала. Такие вопросы не только разнообразят интервью и снижают утомляемость интервьюируемого, но и способствуют лучшему пониманию предметной области. В вопросах такого типа используют фотографии, рисунки и карточки. Например, эксперту предлагаются цветные картонные карточки, на которых выписаны признаки какого-нибудь заболевания. Затем ИЗ просит

разложить эти карточки в порядке убывания значимости каких-либо признаков, либо отнести их к выделенным группам понятий.

12.4.3. Свободный диалог

Свободный диалог – это метод извлечения знаний в форме беседы инженера знаний и эксперта, в которой нет жесткого регламентированного плана и вопросника.

Это определение не означает, что к свободному диалогу не надо готовиться. Напротив, внешне свободная и легкая форма требует серьезной профессиональной и психологической подготовки. Подготовка занимает разное время в зависимости от степени профессионализма инженера знаний, но в любом случае она необходима, так как несколько уменьшает вероятность самого нерационального метода — проб и ошибок.

Подготовка к диалогу, так же, как и к другим активным методам извлечения знаний, включает составление плана проведения сеанса, в котором необходимо предусмотреть следующие этапы.

1. Начало беседы (знакомство, создание у эксперта «образа» аналитика, объяснение целей и задач работы).
2. Диалог по извлечению знаний.
3. Заключительная стадия (благодарность за потраченное время, подведение итогов, договоренность о последующих встречах).

В свободном диалоге важно выбрать правильный темп или ритм беседы: без больших пауз, так как эксперт может отвлечься, но и без «гонки», иначе оба участника быстро утомятся, и нарастет напряженность; кроме того, некоторые люди говорят и думают медленно. Умение чередовать разные темпы, напряжение и разрядку в беседе существенно влияет на результат.

13.2.4 Экспертные игры

Игра – вид человеческой деятельности, который отражает (воссоздает) другие ее виды. При этом для игры характерна одновременно и условность и серьезность.

Понятие экспертной игры восходит к трем источникам – деловым играм, широко используемым при подготовке специалистов, диагностическим и компьютерным играм, применяемым в обучении.

Деловая игра – эксперимент, где участникам предлагается производственная ситуация, а они на основе своих опыта и знаний принимают решения. Принятые решения анализируются, и вскрываются закономерности мышления участников игры. Именно анализ игры «по горячим следам» и полезен для извлечения знаний. Если участниками деловой игры являются эксперты, то тогда игра становится *экспертной*.

Диагностическая игра – это та же деловая игра, но применяемая конкретно для диагностики принятия решений в медицине. Эти игры возникли при исследовании передачи опыта от опытных врачей к новичкам.

Экспертная игра, или *игра с экспертом* – это игра инженера знаний с экспертом, в которой инженер знаний берет на себя какую-нибудь роль в моделируемой ситуации. Например, в игре «Учитель-ученик» инженер знаний берет на себя роль ученика, а эксперт – учителя и поправляет ошибки ученика при выполнении роли эксперта в какой-нибудь предметной области.

Возможно и такое распределение ролей: «врач» (инженер знаний) – «консультант» (эксперт).

В экспертной игре инженер знаний может проверить свои гипотезы о предметной области, получить новые знания.

12.5. Активные групповые методы

Основное достоинство групповых методов – это возможность одновременного «поглощения» знаний от нескольких экспертов, взаимодействие которых вносит в этот процесс элемент принципиальной новизны. Однако следует отметить, что эти методы гораздо более трудоемки и дороги, чем индивидуальные, по причине сложности их организации.

Активные групповые методы сами по себе не могут служить источником более или менее полного знания. Их применяют как дополнение к традиционным индивидуальным методам для активизации мышления и поведения экспертов.

12.5.1. Круглый стол

Метод круглого стола (термин заимствован из журналистики) предусматривает обсуждение какой-либо проблемы из выбранной предметной области, в котором принимают участие с равными правами несколько экспертов. Обычно вначале участники высказываются в определенном порядке, а затем переходят к живой свободной дискуссии. Число участников дискуссии колеблется от трех до пяти-семи.

Большинство общих рекомендаций по извлечению знаний, предложенных ранее, применимо и к данному методу. Однако существует и специфика, связанная с поведением человека в группе.

Во-первых, подготовка круглого стола потребует дополнительных усилий от инженера знаний: как организационных (место, время, обстановка, минеральная вода, чай, кворум и т. д.), так и психологических (умение вставлять уместные реплики, чувство юмора, память на имена-отчества, способность гасить конфликтные ситуации и т. д.).

Во-вторых, большинство участников под воздействием «эффекта фасада» будут говорить совсем не то, что они сказали бы в другой обстановке,

т. е. желание произвести впечатление на других участников дискуссии будет существенно влиять на их высказывания.

Ход беседы за круглым столом необходимо записывать на магнитофон, а при расшифровке и анализе результатов учитывать «эффекта фасада», а также взаимные отношения участников.

Задача дискуссии за круглым столом – коллективно, с разных точек зрения, под разными углами обсудить и исследовать спорные моменты и гипотезы в рассматриваемой предметной области.

Для остроты и продуктивности дискуссии на круглый стол приглашают представителей разных школ и разных поколений. Это уменьшает опасность получения односторонних однобоких знаний.

Перед началом дискуссии ведущему полезно:

– убедиться, что все правильно понимают цель дискуссии (т. е. что происходит сеанс извлечения знаний);
– установить регламент выступлений (не более 5-7 мин);
– четко сформулировать тему дискуссии.

По ходу дискуссии важно проследить, чтобы слишком эмоциональные и разговорчивые эксперты не подменили тему, и чтобы критика высказанных утверждений и позиций была обоснованной.

12.5.2. Мозговой штурм

«Мозговой штурм» – один из наиболее распространенных методов раскрепощения и активизации мышления.

Впервые этот метод использовал в 1939 г. в США А. Осборн в качестве способа получения новых идей в условиях запрещения критики. Замечено, что боязнь критики мешает творческому мышлению, поэтому основная идея «мозгового штурма» – это отделение процедуры генерирования идей в замкнутой группе специалистов от процесса анализа и оценки высказанных идей.

Как правило, штурм длится недолго (около 40 мин.). Участникам (до 10 человек) предлагается высказывать любые идеи (шутливые, фантастические, ошибочные) на заданную тему (критика запрещена). Обычно высказывается более 50 идей. Регламент выступлений – до 2 мин. Самый интересный момент штурма – это наступление пика (ажиотажа), когда идеи начинают «фонтанировать», т. е. происходит непроизвольная генерация гипотез участниками.

При последующем анализе всего лишь 10–15% идей оказываются разумными, но среди них бывают весьма оригинальные и полезные. Оценивает результаты обычно группа экспертов, не участвовавшая в генерации.

Ведущий «мозгового штурма» (инженер знаний) должен свободно владеть аудиторией. Он должен заранее подобрать активную группу экспертов – «генераторов идей», не зажимать плохие идеи, так как они могут

служить катализаторами хороших. Он должен искусно задавать вопросы аудитории, «подогревая» генерацию, так как хорошие вопросы служат «крючком», при помощи которого извлекаются идеи. Вопросы также могут останавливать излишне многословных экспертов и служить развитию идей других.

Основной девиз штурма – «Чем больше идей, тем лучше».

Фиксация хода штурма – традиционная (протокол или магнитофон).

12.5.3. Ролевые игры

В ролевые игры играют одновременно несколько экспертов. Заранее составляется сценарий, распределяются роли, к каждой роли готовится портрет-описание и разрабатывается система оценивания игроков.

В игре принимает участие от 3 до 6 человек. Если участников больше, то они разбиваются на конкурирующие бригады. Вводится элемент состязательности, например, «чей диагноз лучше», «кто быстрее найдет неисправность» и т. п. Возможно использование наглядных материалов, табличек («Директор», «Главный конструктор» и т. п.). Главное, чтобы эксперты в игре действительно заиграли, раскрепостились и «раскрыли свои карты».

12.6. Текстологические методы извлечения знаний

Группа текстологических методов объединяет методы извлечения знаний, основанные на изучении специальных текстов из учебников, монографий, статей, методик и других носителей профессиональных знаний [5].

Задачу извлечения знаний из текстов можно сформулировать как задачу **понимания** и **выделения смысла** текста. Под пониманием здесь подразумевается формирование семантической или понятийной структуры текста.

Следует иметь в виду, что сам текст на естественном языке является лишь проводником смысла, а замысел и знания автора лежат во вторичной структуре (смысловой структуре или макроструктуре текста), настраиваемой над естественным текстом.

При извлечении знаний из текста инженеру знаний приходиться решать задачу декомпозиции этого текста для выделения истинно значимых для реализации базы знаний фрагментов. Сложность интерпретации научных и специальных текстов заключается еще и в том, что любой текст приобретает смысл только в контексте, где под контекстом понимается окружение, в которое «погружен» текст. Различают микро и макроконтекст. *Микроконтекст* – это ближайшее окружение текста. Так, предложение получает смысл в контексте абзаца, абзац в контексте главы и т. д. *Макроконтекст* – это вся система знаний, связанная с предметной областью (т. е. знания об особенностях и свойствах, явно не указанных в тексте). Другими словами, любое знание обретает смысл в контексте некоторого метазнания.

Основными моментами понимания текста являются:

1. Выдвижение предварительной гипотезы о смысле всего текста (предугадывание).

2. Определение значений непонятных слов (т. е. специальной терминологии).

3. Возникновение общей гипотезы о содержании текста (о знаниях).

4. Уточнение значения терминов и интерпретация отдельных фрагментов текста под влиянием общей гипотезы (от целого к частям).

5. Формирование некоторой смысловой структуры текста за счет установления внутренних связей между отдельными важными (ключевыми) словами и фрагментами, а также за счет образования абстрактных понятий, обобщающих конкретные фрагменты знаний.

6. Корректировка общей гипотезы относительно содержащихся в тексте фрагментов знаний (от частей к целому).

7. Принятие основной гипотезы о смысле текста.

Таким образом, центральным моментом процесса понимания текста является формирование смысловой структуры текста в виде наборов «опорных» (ключевых) слов, связанных в единую семантическую структуру.

При анализе текста необходимо выявлять внутренние связи между отдельными элементами текста и понятиями. Традиционно выделяют два вида связей в тексте – *эксплицитные* (или явные связи), которые выражаются во внешнем дроблении текста, и *имплицитные* (скрытые связи). *Эксплицитные связи* делят текст на параграфы с помощью перечисления компонентов, вводных слов (или коннекторов) типа «во-первых..., во-вторых..., однако и т. д.». *Имплицитные*, или внутренние *связи* между отдельными «смысловыми фрагментами» вызывают основное затруднение при понимании.

Текстологические методы делятся на три группы (рис. 12.1):

1) анализ специальной литературы;
2) анализ учебников;
3) анализ методик.

Перечисленные три метода существенно отличаются, во-первых, по степени концентрированности специальных знаний в источнике, и, во-вторых, по соотношению специальных и фоновых знаний. Наиболее простым методом является анализ учебников, в которых логика изложения обычно соответствует логике предмета, и поэтому макроструктура такого текста будет, наверное, более значима, чем структура текста какой-нибудь специальной статьи. Анализ методик, или технической литературы, затруднен как раз сжатостью изложения и практическим отсутствием комментариев, т. е. фоновых знаний, облегчающих понимание для неспециалистов. Поэтому можно рекомендовать для практической работы комбинацию перечисленных методов.

Базовая методика анализа текстов предметной области с целью извлечения и структурирования знаний включает последовательность следующих шагов:

1. Составление «базового» списка литературы для ознакомления с предметной областью и чтение по списку.
2. Выбор текста для извлечения знаний.
3. Первое знакомство с текстом (беглое прочтение). Для определения значения незнакомых слов – консультации со специалистами или привлечение справочной литературы.
4. Формирование первой гипотезы о макроструктуре текста.
5. Внимательное прочтение текста с выписыванием ключевых слов и выражений (компрессия текста).
6. Определение связей между ключевыми словами, разработка макроструктуры текста в форме графа или «сжатого» текста (реферата).
7. Формирование концептуального описания – основных взаимосвязей между понятиями ПО. (Для этого рекомендуется использовать инструментарий для составления концептуальных карт, например, CmapTools.).

Контрольные вопросы

1. Каковы принципы классификации методов извлечения знаний?
2. Отличие коммуникативных методов извлечения знаний от текстологических методов извлечения знаний?
3. Каковы особенности пассивных методов извлечения знаний?
4. Каковы особенности активных методов извлечения знаний?
5. Дайте характеристику групповым методам извлечения знаний.
6. Дайте характеристику индивидуальным методам извлечения знаний.
7. Что такое активные групповые методы? Дайте их описание.
8. Каковы критерии выбора метода извлечения знаний?

13. Технология разработки экспертных систем

13.1. Особенности разработки экспертных систем

Разработка ЭС имеет существенные отличия от разработки обычного программного продукта [8]. Опыт создания ЭС показал, что использование при их разработке методологии, принятой в традиционном программировании, либо чрезмерно затягивает процесс создания ЭС, либо вообще приводит к отрицательному результату. Дело в том, что неформализованность задач, решаемых ЭС, отсутствие завершенной теории ЭС и методологии их разработки приводят к необходимости модифицировать принципы и способы построения ЭС в ходе процесса разработки по мере того, как увеличивается знание разработчиков о проблемной области.

Перед тем как приступить к разработке ЭС, инженер по знаниям должен рассмотреть вопрос, следует ли разрабатывать ЭС для данного приложения. В обобщенном виде ответ может быть таким; использовать ЭС следует только тогда, когда разработка ЭС *возможна, оправдана* и методы инженерии знаний *соответствуют* решаемой задаче.

Чтобы разработка ЭС была ***возможной*** для данного приложения, необходимо одновременное выполнение, по крайней мере, следующих требований:

1) существуют эксперты в данной области, которые решают задачу значительно лучше, чем начинающие специалисты;
2) эксперты сходятся в оценке предлагаемого решения, иначе нельзя будет оценить качество разработанной ЭС;
3) эксперты способны вербализовать и объяснить используемые ими методы, в противном случае трудно рассчитывать на то, что знания экспертов будут "извлечены" и вложены в ЭС;
4) решение задачи требует только рассуждений, а не действий;
5) задача не должна быть слишком трудной (т. е. ее решение должно занимать у эксперта несколько часов или дней, а не недель);
6) задача хотя и не должна быть выражена в формальном виде, но все же должна относиться к достаточно «понятной» и структурированной области, т. е. должны быть выделены основные понятия, отношения и известны (хотя бы эксперту) способы получения решения задачи;
7) решение задачи не должно в значительной степени использовать "здравый смысл" (т. е. широкий спектр общих сведений о мире и о способе его функционирования, которые знает и умеет использовать любой нормальный человек), так как подобные знания пока не удается (в достаточном количестве) вложить в системы искусственного интеллекта.

Применение ЭС может быть ***оправдано*** одним из следующих факторов:

- решение задачи принесет значительный эффект, например экономический;
- использование человека-эксперта невозможно либо из-за недостаточного количества экспертов, либо из-за необходимости выполнять экспертизу одновременно в различных местах;
- при передаче информации эксперту происходит недопустимая потеря времени или информации;
- задача должна решаться в окружении, враждебном для человека.

Приложение ***соответствует*** методам ЭС, если решаемая задача обладает совокупностью следующих характеристик:

1) задача может быть естественным образом решена посредством манипуляции с символами (т. е. с помощью символических рассуждений), а не манипуляций с числами, как принято в математических методах и в традиционном программировании;
2) задача имеет эвристическую, а не алгоритмическую природу, т. е. ее решение требует применения эвристических правил;
3) задача достаточно сложна, чтобы оправдать затраты на разработку ЭС, но не чрезмерно сложна, чтобы ЭС могла с ней справиться;
4) задача является достаточно узкой, чтобы решаться методами ЭС, но в то же время практически значимой.

При разработке ЭС, как правило, используется концепция «быстрого прототипа».

Суть этой концепции состоит в том, что разработчики не пытаются сразу построить конечный продукт. На начальном этапе они создают прототип (прототипы) ЭС.

Прототипы должны удовлетворять двум противоречивым требованиям: с одной стороны, они должны решать типичные задачи конкретного приложения, а с другой – время и трудоемкость их разработки должны быть весьма незначительны, чтобы можно было максимально запараллелить процесс накопления и отладки знаний (осуществляемый экспертом) с процессом выбора (разработки) программных средств (осуществляемым инженером знаний и программистом). Для удовлетворения указанным требованиям, как правило, при создании прототипа используются разнообразные средства, ускоряющие процесс проектирования.

Прототип должен продемонстрировать пригодность методов инженерии знаний для данного приложения. В случае успеха эксперт с помощью инженера знаний расширяет знания прототипа о проблемной области. При неудаче может потребоваться разработка нового прототипа или разработчики могут прийти к выводу о непригодности методов ЭС для данного приложения.

По мере увеличения знаний прототип может достигнуть такого состояния, когда он успешно решает все задачи данного приложения. Преобразование прототипа ЭС в конечный продукт обычно приводит к перепрограммированию ЭС на языках низкого уровня, обеспечивающих как увеличение быстродействия ЭС, так и уменьшение требуемой памяти. Трудоемкость и время создания ЭС в значительной степени зависят от типа используемого инструментария.

13.2. Основные этапы разработки

Разработка экспертной системы включает шесть этапов (рис. 13.1): *идентификацию, концептуализацию, формализацию, выполнение (реализацию), тестирование, опытную эксплуатацию*.

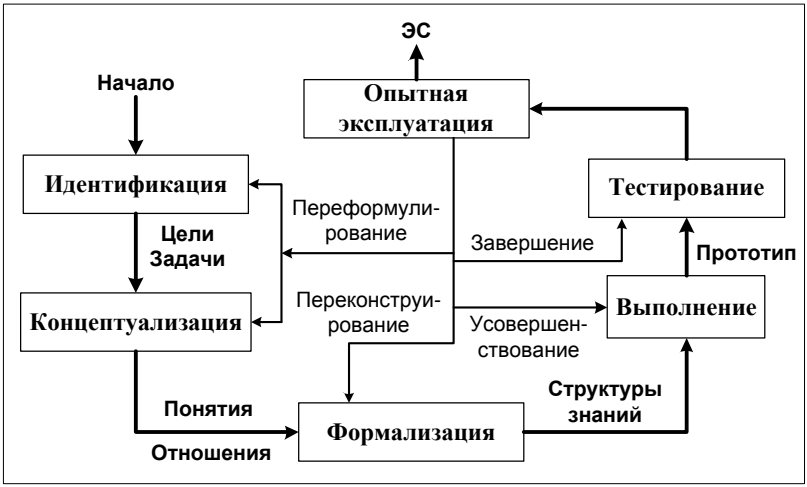

Рис. 13.1. Процесс разработки экспертной системы

На этапе **идентификации** определяются задачи, которые подлежат решению, выявляются цели разработки, определяются ресурсы, эксперты и типы пользователей.

На этапе **концептуализации** проводится содержательный анализ проблемной области, выявляются используемые понятия и их взаимосвязи, определяются методы решения задач.

На этапе **формализации** выбираются инструментальные средства и определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, моделируется работа системы, оценивается адекватность целям системы зафиксированных понятий, методов решений, средств представления и манипулирования знаниями.

На этапе **выполнения** разрабатывается один или несколько прототипов ЭС, решающих поставленные задачи. Затем на основе прототипов после их тестирования и опытной отладки будет создаваться конечный продукт – промышленная или коммерческая ЭС. На этапе выполнения осуществляется наполнение базы знаний, поэтому этот этап является наиболее важным и наиболее трудоемким этапом разработки ЭС.

На этапе **тестирования** в режиме диалога и с использованием подсистемы объяснений проверяется компетентность ЭС. При этом очень важно выбрать хороший набор тестовых примеров. Тестирование продолжается до тех пор, пока эксперт не решит, что ЭС достигла требуемого уровня компетентности.

На этапе **опытной эксплуатации** проверяется *пригодность* ЭС для конечных пользователей. Пригодность ЭС определяется ее *удобством* для пользователя и *полезностью*. По результатам опытной эксплуатации может потребоваться не только модификация программ и базы знаний, но и пользовательского интерфейса. Здесь же принимается решение о переносе ЭС на другие инструментальные средства и типы ЭВМ.

Процесс создания ЭС не сводится к строгой последовательности рассмотренных выше этапов. В ходе разработки приходится возвращаться на более ранние этапы и пересматривать принятые там решения (рис. 13.1).

13.3. Классификация инструментальных средств

Инструментальные средства разработки экспертных систем можно разделить на 4 категории:

1) Языки программирования
   - традиционные языки (C, C++, C#, Java и др.),
   - языки символьной обработки (LISP, INTERLISP, SMALLTALK).
2) Языки инженерии знаний (OPS-5, LOOPS, PROLOG).
3) Программные обстановки, автоматизирующие разработку ЭС и интеллектуальных систем ИИ (KEE, ART, AGE, RLL, HEARSAY-III, TEIRESIAS, Semp-TAO).
4) Оболочки экспертных систем – пустые ЭС, не содержащие никаких знаний о предметной области (EMYCIN, KAS, ЭКСПЕРТИЗА, ЭКО, ЭКСПЕРТ, DI*GEN).

В приведенной классификации инструментарии расположены в порядке убывания трудоемкости создания с их помощью ЭС или ее прототипа.

Действительно, при использовании инструментария первого типа в задачу разработчика входит программирование всех компонентов ЭС на языке довольно низкого уровня.

Использование инструментария второго типа позволяет значительно повысить уровень языка разработки ЭС, однако это может повлечь за собой снижение эффективности.

Инструментальные средства третьего типа позволяют разработчику не программировать некоторые или все компоненты ЭС, а выбирать их из заранее составленного набора. Обычно инструментарии этого типа подразделяют на средства, автоматизирующие построение ЭС (например, RLL, HEARSAY-III), и средства, автоматизирующие процесс приобретения знаний (например, TEIRESIAS).

При применении инструментария четвертого типа разработчик ЭС полностью освобождается от работ по программированию, так как он берет готовую базовую систему. Однако при использовании этого способа могут возникнуть следующие проблемы:

1) управляющие стратегии, вложенные в процедуры вывода базовой системы, могут не соответствовать методам решения, которые использует эксперт, взаимодействующий с данной системой, что может приводить к неэффективным, а возможно, и неправильным решениям;
2) язык представления знаний, принятый в базовой системе, может не подходить для данного приложения.

Контрольные вопросы

1. Какие требования к приложению должны удовлетворяться, чтобы разработка ЭС была возможной?
2. Чем (какими факторами) может быть оправдана разработка ЭС?
3. Расскажите, что Вы знаете о концепции «быстрого прототипа».
4. Назовите основные этапы разработки экспертной системы.
5. Какие задачи решаются на этапе идентификации при разработке экспертной системы?
6. Приведите общепринятую классификацию инструментальных средств разработки экспертных систем.
7. Что такое оболочка экспертной системы?

Литература

1. Базисный рефал и его реализация на вычислительных машинах (методические рекомендации). М. : ЦНИПИАСС, 1977.

2. Большая Советская Энциклопедия / Под ред. А.М. Прохорова. М. : Советская Энциклопедия, 1972. Т. 9. 555 с.

3. Бьюзен Т. Научите себя думать! 3-е издание. Минск : Попурри, 2008. 192 с.

4. Гаврилова Т. А., Муромцев Д. И. Интеллектуальные технологии в менеджменте: инструменты и системы. 2-е изд. Санкт-Петербург : Высшая школа менеджмента, Издательство СПбГУ, 2008. 488 с.

5. Гаврилова Т. А., Хорошевский В. Ф. Базы знаний интеллектуальных систем. Учебник. СПб. : Питер, 2001.

6. Гаврилова Т. А., Червинская К. Р. Извлечение и структурирование знаний для экспертных систем. Москва : Радио и связь, 1992.

7. Гринберг С. Я., Яхно Т. М. Решение задач технической диагностики с использованием оболочки ДИ*ГЕН // Техническая кибернетика, 1990. № 5, С.147–153.

8. Джарратано Дж., Райли Г. Экспертные системы: принципы разработки и программирования. Пер. с англ. М. : Издательский дом «Вильямс», 2006. 1152 с.

9. Загорулько Ю. А., Попов И. Г. Представление знаний в интегрированной технологической среде Semp-TAO // Проблемы представления и

обработка не полностью определенных знаний. Москва–Новосибирск, 1996. С. 59–74.

10. Заде Л. Понятие лингвистической переменной и его использование в принятии приближенных решении. М. : Мир, 1976.

11. Клоксин У., Меллиш К. Программирование на языке PROLOG. М. : Мир, 1987.

12. Логическое программирование / Под ред. В. Н. Агафонова. М. : Мир, 1988.

13. Лозовский В. Семантические сети // Представление знаний в человеко-машинных и робототехнических системах. М., 1984. Т. А. С. 84–121.

14. Лорьер Ж.–Л. Системы искусственного интеллекта. М. : Мир, 1991.

15. Минский М. Структура для представления знания // Психология машинного зрения. М. : Мир, 1978. С. 249–338.

16. Нариньяни А. С. Лингвистические процессоры ЗАПСИБ (часть I: задачи проекта). Новосибирск, 1979. (Препр./ АН СССР. Сиб. отд-ние. ВЦ; 199).

17. Нариньяни А. С. Лингвистические процессоры ЗАПСИБ (часть II: общая схема и основные модули). Новосибирск, 1979. (Препр./ АН СССР. Сиб. отд-ние. ВЦ; 202).

18. Нариньяни А. С. Недоопределенность в системах представления и обработки знаний // Изв. АН СССР. Техн. кибернетика. 1986. № 5. С. 3–28.

19. Нечеткие множества в моделях управления и искусственного интеллекта / Под ред. Поспелова Д. А. М. : Наука, 1986.

20. Нильсон Н. Искусственный интеллект. Методы поиска решений. М. : Мир, 1973.

21. Онтологии и тезаурусы: модели, инструменты, приложения: учебное пособие / Б. В. Добров, В. В. Иванов, Н. В. Лукашевич, В. Д. Соловьев. М. : Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2009. 173 с.

22. Пильщиков В. Н. Язык плэнер. М. : Наука, 1983.

23. Попов Э.В. Экспертные системы: Решение неформализованных задач в диалоге с ЭВМ. М. : Наука. 1987. 288 с.

24. Реннелс Г. Д., Шортлиф Э. Г. Вычислительные системы для медицины // В мире науки, 1987. № 12.

25. Тыугу Э. Х. Концептуальное программирование. М. : Наука, 1984.

26. Уинстон П. Искусственный интеллект. М. : Мир, 1980.

27. Хейес-Рот Ф., Уотерман Д., Ленат Д. Построение экспертных систем. М. : Мир, 1987.

28. Хендрикс Г. О расширении применяемости семантических сетей введением разбиений // Тр. IV Междунар. конф. по искус. интеллекту, Тбилиси, 1975. М., 1975, Т. 1. С. 190–206.

29. Хоггер К. Введение в логическое программирование. М.: Мир, 1988.

30. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. М. : Наука, 1983.

31. Шенк Р., Абельсон Р. Сценарии, планы, знание. // Тр. ГУ Междунар. конф. по искус. интеллекту, Тбилиси, 1975. М., 1975. Т. 6. С. 211–225.

32. Feigenbaum E. A. The art of artificial intelligence: Themes and case studies of knowledge engineering // The fifth International Joint Conference on Artificial Intelligence. Boston: MIT, 1977. P. 1014–1029.

33. Gruber T. Toward Principles for the Design of Ontologies Used for Knowledge Sharing // International Journal of Human–Computer Studies. November 1995. Vol. 43. Issues 5–6. P. 907–928.

34. Guarino N. Formal Ontology in Information Systems // Formal Ontology in Information Systems. Proceedings of FOIS'98, Trento, Italy, June 6–8, 1998 / Ed. N. Guarino. Amsterdam : IOS Press, 1998. P. 3–15.

35. Newell A., Simon M.A. Human problem solving. Englewood Cliffs, New Jersey : Prentice-Hall, 1972.

36. Newell A. Production systems: models of control structures // Visual information processing. New York: Academic Press, 1973. P. 463–526.

ОГЛАВЛЕНИЕ

Учебное издание

Загорулько Юрий Алексеевич,
Загорулько Галина Борисовна

ИНЖЕНЕРИЯ ЗНАНИЙ

Учебное пособие

Редактор *А. С. Штыкова*
Оригинал-макет *Ю. А. Загорулько*
Обложка *Е. В. Неклюдовой*

Подписано в печать 25.04.2016 г.
Формат 60 x 84 1/16. Уч.-изд. л. 5,75. Усл. печ. л. 5,35.
Тираж 70 экз. Заказ №

Редакционно-издательский центр НГУ
630090, Новосибирск-90, ул. Пирогова, 2.