В.А. Лобанова
О.А. Воронина

ПРИНЦИПЫ ОРГАНИЗАЦИИ САПР С ЭЛЕМЕНТАМИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

ОРЛОВСКИЙ
ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ
имени И.С. Тургенева

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

В.А. Лобанова, О.А. Воронина

ПРИНЦИПЫ ОРГАНИЗАЦИИ САПР С ЭЛЕМЕНТАМИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Орёл
ОГУ имени И.С. Тургенева
2021

УДК 004.896(075.8)
ББК 32.813я73
Л68

Рецензенты:

кандидат технических наук, доцент
заведующий кафедрой электроники, радиотехники и систем связи
федерального государственного бюджетного
образовательного учреждения высшего образования
«Орловский государственный университет имени И.С. Тургенева»
В.В. Мишин,

кандидат технических наук, доцент
заведующий кафедрой «Системы информационной безопасности»
федерального государственного бюджетного
образовательного учреждения высшего образования
«Брянский государственный технический университет»
М.Ю. Рытов

Лобанова, В.А.

Л68 Принципы организации САПР с элементами искусственного интеллекта: учебное пособие / В.А. Лобанова, О.А. Воронина. – Орёл: ОГУ имени И.С. Тургенева, 2021. – 169 с.

ISBN 978-5-9929-1016-2

Учебное пособие посвящено теории разработки интеллектуальных подсистем САПР в рамках теории искусственного интеллекта. Пособие состоит из четырех разделов. Первый раздел посвящен рассмотрению основных принципов создания САПР, второй – системному подходу к проектированию электронных средств, в третьем разделе изучается математическое обеспечение интеллектуального проектирования, в четвертом – информационное обеспечение интеллектуальных САПР.

Пособие составлено с использованием соответствующих литературных источников, а также специализированных интернет-ресурсов.

Предназначено студентам, обучающимся по направлениям 11.03.03 «Конструирование и технология электронных средств» и 11.04.03 «Конструирование и технология электронных средств», изучающим дисциплины «Интеллектуальные системы автоматизированного проектирования изделий электроники и микроэлектроники» и «Современные технологии проектирования электронных средств». Также может быть полезно студентам других направлений (специальностей).

УДК 004.896(075.8)
ББК 32.813я73

ISBN 978-5-9929-1016-2

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Разработка интеллектуальных подсистем САПР производится в рамках теории искусственного интеллекта. Искусственный интеллект (ИИ) – это область исследований, находящаяся на стыке наук. Специалисты, работающие в этой области, пытаются понять, какое поведение считается разумным (анализ), и создать работающие модели этого поведения (синтез). Практической целью является создание методов и техники, необходимой для программирования «разумности» и ее передачи вычислительным машинам, а через них всевозможным системам и средствам.

Искусственный интеллект как научное направление, связанное с попытками формализовать мышление человека, имеет длительную историю. Еще Платон, Аристотель, Р. Декарт, Г.В. Лейбниц, Дж. Буль и многие другие исследователи на уровне современных им знаний стремились описать мышление как совокупность некоторых элементарных операций, правил и процедур. Качественно новый период развития ИИ связан с появлением в научных лабораториях ЭВМ и с публикацией книги Н. Винера «Кибернетика, или Управление и связь в животном и машине», появившейся еще ранее книги У. Росс Эшби «Введение в кибернетику».

В нашей стране идеи создания ИИ получили признание после выхода целой серии переводных работ Н. Винера, У. Росс Эшби, Ст. Бира и Э. Беркли в конце 50-х – начале 1960-х годов прошлого века и нашего соотечественника Берга. Однако к непосредственному использованию идей ИИ российские ученые обратились только с развитием информационных технологий, с необходимостью повышения интеллектуальности систем автоматизированного проектирования современных конструкций и технологических процессов изготовления изделий в различных областях.

Искусственный интеллект появился на базе вычислительной техники, математической логики, программирования, психологии, лингвистики, нейрофизиологии и других отраслей знаний. Искусственный интеллект – это образец междисциплинарных исследований, где соединяются профессиональные интересы специалистов разного профиля. Само название науки возникло в конце 60-х гг., а в 1969 г. в Вашингтоне (США) состоялась первая Всемирная конференция по искусственному интеллекту.

Известно, что совокупность научных исследований обретает права науки, если выполнены два необходимых условия: у этих исследований должен быть объект изучения, не совпадающий с теми, которые изучают другие науки; должны существовать специфические методы исследования этого объекта, отличные от методов других, уже сложившихся наук. Исследования, которые объединяются сейчас термином «искусственный интеллект», имеют свой специфический объект изучения и свои специфические методы.

Когда в конце 1940-х – начале 1950-х гг. появились компьютеры, стало ясно, что инженеры и математики создали не просто быстро работающее устройство для вычислений, а нечто более значительное. Оказалось, что с помощью компьютера можно решать различные головоломки, логические задачи, играть в шахматы, создавать игровые программы. Компьютеры стали принимать участие в творческих процессах: сочинять мелодии, стихотворения и даже сказки. Появились программы для перевода с одного языка на другой, для распознавания образов, доказательства теорем. Это свидетельствовало о том, что с помощью компьютера и соответствующих программ можно автоматизировать такие виды человеческой деятельности, которые называются интеллектуальными и считаются доступными лишь человеку.

Несмотря на большое разнообразие невычислительных программ, созданных к началу 1960-х гг., программирование в сфере интеллектуальной деятельности находилось в гораздо худшем положении, чем решение расчетных задач. Программирование для задач расчетного характера опиралось на соответствующую теорию – вычислительную математику. На основе этой теории было разработано много методов решения задач. Эти методы стали основой для соответствующих программ. Ничего подобного для невычислительных задач не было. Любая программа была здесь уникальной, как произведение искусства. Опыт создания таких программ никак не обобщался, умение их создавать не формализовалось.

Никто не станет отрицать, что в отличие от искусства у науки должны быть методы решения задач. С помощью этих методов все однотипные задачи должны решаться единообразным способом, и, «набив руку» на решении задач определенного типа, легко решать новые задачи, относящиеся к тому же типу. Но именно такие методы и не смогли придумать те, кто создавал первые программы невычислительного характера.

Первые шаги кибернетики были направлены на изучение и осмысление процессов, протекающих в сложных, прежде всего, в живых системах, включая и мыслящие. Исследования имели ярко выраженный познавательный характер. Но уже тогда стали появляться разработки, направленные на воспроизведение в ЭВМ определенных процессов и феноменов мышления. Позднее именно это направление и оформилось в самостоятельную область, разрабатывающую проблему ИИ.

В ходе последующего развития исследований по ИИ, в 1960 – 1970 годах, произошло их разделение на два самостоятельных направления. Это разделение сохраняется и до настоящего времени. Без четкого понимания различий и тенденций развития обоих направлений нельзя правильно разобраться и в современном состоянии проблемы.

1. ОСНОВНЫЕ ПРИНЦИПЫ СОЗДАНИЯ САПР

1.1. Повышение интеллектуальности подсистем проектирования

При создании программы для игры в шахматы программист использует собственные знания о процессе игры. Человек преобразовывает их в программу, а компьютер лишь механически исполняет эту программу. Можно сказать, что компьютер при этом отличал вычислительные программы от невычислительных. В памяти компьютера не было знаний о том, что он на самом деле делает, так как он одинаковым образом находил корень квадратного уравнения или писал стихи.

Об интеллекте компьютера можно было бы говорить, если бы он сам, на основании собственных знаний о том, как протекает игра в шахматы и как играют в эту игру люди, сумел составить шахматную программу или синтезировал программу для написания несложных вальсов и маршей.

Именно не сами процедуры, с помощью которых выполняется та или иная интеллектуальная деятельность, а понимание того, как их создать, научиться новому виду интеллектуальной деятельности, – вот где скрыто то, что можно назвать интеллектом. Специальные метапроцедуры обучения новым видам интеллектуальной деятельности отличают человека от компьютера. Следовательно, при создании искусственного интеллекта основной задачей становится реализация машинными средствами тех метапроцедур, которые используются в интеллектуальной деятельности человека.

В психологии мышления есть несколько моделей творческой деятельности. Одна из них называется лабиринтной. Суть лабиринтной гипотезы, на которой основана лабиринтная модель, состоит в следующем: переход от исходных данных задачи к ее решению лежит через лабиринт возможных альтернативных путей. Не все пути ведут к желаемой цели – многие из них заводят в тупик, из которого надо уметь возвращаться к тому месту, где потеряно правильное направление. Это напоминает попытки не слишком умелого школьника решить задачу об упрощении алгебраических выражений. Для этой цели на каждом шагу можно применять некоторые стандартные преобразования или придумывать искусственные приемы. Но весьма часто

вместо упрощения выражения происходит его усложнение, и возникают тупики, из которых нет выхода. По мнению сторонников лабиринтной модели мышления, решение всякой творческой задачи сводится к целенаправленному поиску в лабиринте альтернативных путей с оценкой успеха после каждого шага [3].

С лабиринтной моделью связана первая из метапроцедур – целенаправленный поиск в лабиринте возможностей. Программированию этой метапроцедуры соответствуют многочисленные процедуры поиска, основанные на соображениях «здравого смысла» (человеческого опыта решения аналогичных задач). В 1960-х гг. было создано немало программ на основе лабиринтной модели, в основном игровых и доказывающих теоремы «в лоб», без привлечения искусственных приемов. Соответствующее направление в программировании получило название *эвристического программирования*. Высказывались даже предположения, что целенаправленный поиск в лабиринте возможностей – универсальная процедура, пригодная для решения любых интеллектуальных задач.

Но исследователи отказались от этой идеи, когда столкнулись с задачами, в которых лабиринта возможностей либо не существовало, либо он был слишком велик для метапроцедуры поиска, как, например, при игре в шахматы. Конечно, в этой игре есть лабиринт возможностей – это все мыслимые партии игры. Но как в этом астрономически большом лабиринте найти те партии, которые ведут к выигрышу? Лабиринт столь велик, что никакие скорости вычислений не позволят целенаправленно перебрать пути в нем. И все попытки использовать для этого человеческие эвристики (в данном случае профессиональный опыт шахматистов) не указывают пути решения задачи. Поэтому современные шахматные программы уже давно применяют не только метапроцедуру целенаправленного поиска, но и иные метапроцедуры, связанные с другими моделями мышления.

Долгие годы в психологии изучалась ассоциативная модель мышления. Основной метапроцедурой этой модели являются ассоциативный поиск и ассоциативное рассуждение. Предполагается, что решение неизвестной задачи так или иначе основывается на уже решенных задачах, чем-то похожих на ту, которую надо решить. Новая задача рассматривается как уже известная, хотя и несколько отличающаяся от известной. Поэтому способ ее решения должен быть близок к тому, который когда-то помог решить подобную задачу.

Для этого надо обратиться к памяти и попытаться найти нечто похожее, что ранее уже встречалось. Это и есть ассоциативный поиск. Когда, увидев незнакомого человека, вы стараетесь вспомнить, на кого он похож, реализуется метапроцедура ассоциативного поиска. Но понятие ассоциации в психологии шире, чем просто похожесть. Ассоциативные связи могут возникнуть и по контрасту, как противопоставление одного другому, и по смежности, т. е. в силу того, что некоторые явления возникали в рамках одной и той же ситуации или происходили одновременно (или с небольшим сдвигом по времени).

Ассоциативное рассуждение позволяет переносить приемы, использованные ранее, на текущую ситуацию. К сожалению, несмотря на многолетнее изучение ассоциативной модели, не удалось создать стройную теорию ассоциативного поиска и ассоциативного рассуждения. Исключение составляет важный, но частный класс ассоциаций, называемых условными рефлексами. И все же метапроцедура ассоциативного поиска и рассуждения сыграла важную роль: она помогла создать эффективные программы в распознавании образов, в классификационных задачах и в обучении компьютеров. Но одновременно эта метапроцедура привела к мысли о том, что для ее эффективного использования надо привлечь результаты, которые были получены в другой модели мышления, опирающейся на идею внутреннего представления проблемной области, на знаниях о ее особенностях, закономерностях и процедурах действия в ней.

Это представление о мыслительной деятельности человека обычно называют модельной гипотезой. Согласно ей, мозг человека содержит модель проблемной ситуации, в которой ему надо принять решение. Для решения используются метапроцедуры, оперирующие с совокупностью знаний из той проблемной области, к которой принадлежит данная проблемная ситуация. Например, если проблемная ситуация – переход через улицу с интенсивным движением, то знания, которые могут помочь ее разрешить, касаются способов организации движения транспорта, сигналов светофоров, наличия дорожек для перехода и т. д. [18].

В модельной гипотезе основными метапроцедурами становятся представление знаний, рассуждения, поиск релевантной (связанной с данной проблемной ситуацией) информации в совокупности имеющихся знаний, их пополнение и корректировка. Эти метапроцедуры составляют ядро интеллектуальных возможностей современных программ и программных систем, ориентированных на решение творче-

ских задач. В совокупности с метапроцедурами целенаправленного поиска в лабиринте возможностей, ассоциативного поиска и рассуждения они образуют арсенал интеллектуальных средств, которым располагают современные интеллектуальные системы, часто называемые системами, основанными на знаниях.

Можно сформулировать основные цели и задачи искусственного интеллекта. Объектом изучения искусственного интеллекта являются метапроцедуры, используемые при решении человеком задач, традиционно называемых интеллектуальными или творческими. Но если психология мышления изучает эти метапроцедуры применительно к человеку, то искусственный интеллект создает программные (а сейчас уже и программно-аппаратные) их модели.

Цель исследований в области искусственного интеллекта – создание арсенала метапроцедур, достаточного для того, чтобы компьютеры (или другие технические системы, например роботы) могли находить по постановкам задач их решения. Иными словами, стали автономными программистами, способными выполнять работу профессиональных программистов-прикладников (создающих программы для решения задач в определенной предметной области). Разумеется, сформулированная цель не исчерпывает всех задач, которые ставит перед собой искусственный интеллект. Это цель ближайшая. Последующие цели связаны с попыткой проникнуть в области мышления человека, которые лежат вне сферы рационального и выразимого словесно (вербально) мышления. В поиске решения многих задач, особенно сильно отличающихся от ранее решенных, большую роль играет та сфера мышления, которую называют подсознательной, бессознательной или интуитивной.

Основными методами, используемыми в искусственном интеллекте, являются разного рода программные модели и средства, компьютерные эксперименты и теоретические модели. Однако современные компьютеры уже мало удовлетворяют специалистов по искусственному интеллекту. Их конструкция не имеет ничего общего с тем, как устроен человеческий мозг. Поэтому идет интенсивный поиск новых технических структур, которые будут способны лучше решать задачи, связанные с интеллектуальными процессами. Сюда относятся исследования по нейроподобным искусственным сетям, попытки построить молекулярные машины, работы в области голографических систем и многое другое.

Существует несколько основных проблем, изучаемых в искусственном интеллекте:

1. *Представление знаний* – разработка методов и приемов для формализации и последующего ввода в память интеллектуальной системы знаний из различных проблемных областей, обобщение и классификация накопленных знаний, применение знаний при решении задач.

2. *Моделирование рассуждений* – изучение и формализация различных схем человеческих умозаключений, используемых в процессе решения разнообразных задач, создание эффективных программ для реализации этих схем в вычислительных машинах.

3. *Диалоговые процедуры общения на естественном языке*, обеспечивающие контакт между интеллектуальной системой и человеком-специалистом в процессе решения задач.

4. *Планирование целесообразной деятельности* – разработка методов построения программ сложной деятельности на основании тех знаний о проблемной области, которые хранятся в интеллектуальной системе.

5. *Обучение интеллектуальных систем* в процессе их деятельности, создание комплекса средств для накопления и обобщения умений и навыков, накапливаемых в таких системах.

Кроме этих проблем исследуются многие другие, составляющие тот задел, на который будут опираться специалисты на следующем витке развития теории искусственного интеллекта.

В практику человеческой деятельности интеллектуальные системы уже внедряются. Это наиболее известные широкому кругу специалистов экспертные системы, передающие опыт более подготовленных специалистов менее подготовленным. Сюда относятся и интеллектуальные информационные системы (например, системы машинного перевода), и интеллектуальные роботы, другие системы, имеющие полное право называться интеллектуальными. Без таких систем современный научно-технический прогресс уже невозможен.

САПР – это комплекс средств автоматизации проектирования, взаимосвязанных с проектными организациями (пользователями системы). САПР включает технические средства, математическое и программное обеспечение, информационное обеспечение, лингвистическое обеспечение (специальные языки, проблемно-ориентированные).

Актуальность и необходимость применения САПР. Ускорение темпов развития науки и техники привели к следующим особенностям при проектировании радиоэлектронной аппаратуры (РЭА):

1) непрерывному росту тактико-технических требований (масса, надежность, стоимость, электрические показатели и др.);

2) резкому сокращению сроков морального старения РЭА;

3) увеличению стоимости разработок;

4) сокращению сроков, отводимых на разработку новых изделий.

Эффективно решать эти противоречивые проблемы возможно лишь, применяя в процессе проектирования различные САПР, что в частности позволит:

– проанализировать большое количество вариантов, различных решений;

– создавать конструкции, оптимально учитывающие предъявляемые к ним требования;

– использовать более точные методы расчета и проектирования, сводящие к минимуму подстроечно-регулировочные операции;

– сократить сроки и снизить стоимость разработки аппаратуры.

При создании САПР учитываются принципы:

• *системного единства,* т.е. целостность системы, взаимосвязь между подсистемами и ее элементами;

• *совместимости,* т.е. обеспечиваются совместное функционирование составных частей САПР и сохранность открытости системы в целом;

• *типизации* – ориентирует на преимущественное создание и использование типовых и унифицированных элементов САПР с последующей их модернизацией;

• *развития* – способствует совершенствованию и обновлению составных частей САПР, а также взаимодействию и расширению взаимосвязи с автоматизированными системами различного уровня и функционального назначения;

• *иерархичности* – проектирование по уровням структуры САПР [6].

1.2. Постановка задачи повышения интеллектуальности подсистем проектирования

Повышение интеллектуальности подсистем проектирования осуществляется путем использования эвристического программирования, экспертных систем, путем перехода от режима диалога к пакетному режиму более высокого уровня.

Одно из требований автоматизированного проектирования в режиме диалога – это максимальное освобождение технолога-проектировщика от рутинных работ, требующих каких-либо вычислений или количественных оценок проектных ситуаций. В процессе проектирования технолог-проектировщик задает информацию о полезности следствий. Эта информация обрабатывается компьютером с целью сокращения числа целесообразных альтернатив и отбрасывания неприемлемых. Проектировщик, принимающий решение, анализирует результаты расчета на компьютере и отбирает рациональные (с его точки зрения) альтернативы, а если надо, то осуществляет дальнейшую детализацию альтернатив и возникающих из них следствий.

Под полезностью понимают обобщенную оценку альтернативы, описывающую ее пригодность для дальнейшего проектирования и легкость реализации. Эту оценку дает технолог-проектировщик. Полезность следствия обозначают через $П_{ij}$, где i – условный номер альтернативы; j – номер следствия данной i-й альтернативы. Для освобождения технолога-проектировщика от количественной оценки альтернатив (следствий) применяют несколько способов задания оценок полезностей. Например, производят простое ранжирование следствий или полезности альтернатив, сравнивают их между собой качественно, используя отношения типа «больше – меньше», «хуже – лучше», которые технолог-проектировщик может задавать знаками «>», «<».

На этапе выбора модели автомата для изготовления деталей может сложиться следующая ситуация [М]: для обработки втулки используют либо пруток, либо толстостенную трубу. Заготовку можно обработать на трех разных моделях автоматов. Следовательно, имеются две альтернативы и по три следствия из каждой. Для первой альтернативы необходимо получить полезность следствия $П_{11}$, $П_{12}$, $П_{13}$, для второй – $П_{21}$, $П_{22}$, $П_{23}$. Задача состоит в нахождении доверительного интервала для каждого значения $П_{ij}$.

Если ввести условие:

$$П = \sum_{j=1}^{n} П_i \qquad (1.1)$$

где n – число следствий из i-й альтернативы, то нахождение доверительного интервала сводится к нахождению верхней и нижней границ

оценки полезности следствий (альтернатив), т. е. к нахождению Π_{ij}^{+} и Π_{ij}^{-} соответственно.

Допустим, что технолог-проектировщик задал отношение между следствиями в следующем виде: $П_{11}<П_{12}$; $П_{13}<П_{12}<П_{23}$; $П_{23}<П_{11}$; $П_{22}<П_{11}$.

Вводят ограничения и преобразуют эти выражения:

$П_{11}–П_{12}<0$; $П_{13}–П_{12}<0$; $П_{12}–П_{23}<0$; $П_{23}–П_{11}<0$; $П_{22}–П_{11}>0$;
$П_{11}+П_{12}+П_{13}=1$; $П_{21}+П_{22}+П_{23}=1$.

Подобную задачу сводят к задаче линейного программирования:

$$Z = g^1x_1 + g^2x_{21} + \cdots + g^jx_j + \cdots + g^nx_n \to min;$$
$$a_1^1x_1 + a_2^2x_2 + \ldots + a_i^jx_j + \ldots + a_m^nx_n \to b_i \text{ при } i = \overline{1,s}; \quad (1.2)$$
$$a_1^1x_1 + a_2^2x_2 + \ldots + a_i^jx_j + \ldots + a_m^nx_n = b_i \text{ при } i = s + \overline{1,m};$$

где при всех $i = 1, 2, 3,\ldots, m$; $j = 1, 2, 3,\ldots, n$ – вещественные числа;

g^jb_i и $a^j{}_i$ – заданы, а неизвестные x_j подлежат определению.

Для перехода от режима диалога к пакетному режиму более высокого уровня формируют обучающие выборки. Составляют матрицу A «признак-значение». Элемент матрицы A_{ij} соответствует i-му значению j-го признака, $i = \overline{1,m}$; $j = \overline{1,n}$ (где m – максимальное число значений, принимаемых j-м признаком; n – число признаков). Тогда

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} & \ldots A_{1j} \ldots & A_{1n} \\ A_{21} & A_{22} & A_{23} & \ldots A_{2j} \ldots & A_{2n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ A_{i1} & A_{i2} & A_{i3} & \ldots A_{ij} \ldots & A_{in} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ A_{m1} & A_{m2} & A_{m3} & \ldots A_{mj} \ldots & A_{mn} \end{pmatrix} \quad (1.3)$$

С помощью матрицы A можно описывать любые объекты или ситуации. Разница будет состоять в числе признаков, описывающих объект (ситуацию), и в количествах значений каждого признака.

Матрицу S, полученную добавлением матрицы A к матрице «нуль-единичного» столбца, называют полной формой понятия:

$$S = (A) \dots \begin{pmatrix} F_1 \\ F_2 \\ F_i \\ \dots \\ F_m \end{pmatrix}, \tag{1.4}$$

где $F_i = 1$ – объект (ситуация), описываемый i-й строкой матрицы A, является положительным событием;

$F_i = 0$ – в противном случае.

Чтобы заполнить матрицу S, следует рассмотреть большое количество вариантов комбинаций значений N и классифицировать полученные комбинации $N = \prod_{j=1}^{n} m_j$. Матрицу S строят на основе обучающей выборки.

Задачей построения обучающей выборки является определение существенных признаков, описывающих объект, и нахождение количества значений, которые может принимать каждый признак объекта.

Как показали исследования, многие технологи-проектировщики при опросе не могли четко сформулировать причины выбора одной альтернативы из некоторого количества предлагаемых. Причем признаки, по которым определялся объект (например, комплекс элементарных обрабатываемых поверхностей), для разных технологов-проектировщиков были неодинаковыми. Поэтому задачу выделения существенных признаков, описывающих объект, следует возлагать на технолога-проектировщика, работающего с использованием методов САПР.

Однако на многих этапах проектирования возникает задача отнесения объекта не к одному из двух классов, а из нескольких (например, при отнесении комбинаций поверхностей к одному из шести комплексов элементарных обрабатываемых поверхностей). В данном случае к матрице A добавляется не «нуль-единичный» столбец, а столбец, в котором F_i принимает значения $F_i = \overline{1, k}$, где k – номер класса объекта, к которому относится классифицируемый объект, или $F_i = 0$, если объект не был отнесен ни к одному из классов. Значения признаков, оценивающих объект в процессе диалога технолога-

проектировщика с компьютером, накапливаются на магнитном диске. Каждому набору *i*-го значения признаков ставится в соответствие F_i, относящее этот объект к классу, определяемому технологом-проектировщиком.

Области распределения положительных и отрицательных объектов не должны пересекаться, иначе могут возникать ошибки. Влияние подобных ошибок на качество проектируемой наладки снижается за счет одновременного проектирования нескольких наладок и ведет к увеличению их числа.

Для оценки степени обученности системы используется экзамен на контролируемой группе объектов, который можно применить для постепенного перехода от режима диалога к новому уровню пакетного режима. Степень обученности системы оценивалась следующими показателями:

$p = \frac{n^+ + n^-}{N}$ – частота ошибок при оценке степени обученности системы;

$p^+ = \frac{n^+}{N^+}$; $p^- = \frac{n^-}{N^-}$ – частота ошибок при распознавании положительных и отрицательных объектов, где N – число контрольных объектов, используемых для оценки степени обученности системы; N^+, N^- – число положительных и отрицательных объектов; n^+, n^- – число ошибок при распознавании положительных и отрицательных объектов.

Экспериментальное определение величины обучающей выборки проводили для этапа расчленения поверхности детали на комплексы элементарных обрабатываемых поверхностей. Нужно было сформировать понятие «комплекс поверхностей, который можно обработать проходными резцами».

Были выделены следующие признаки, описывающие подобный комплекс поверхностей:

- вид поверхностей, вошедших в комплекс;
- последовательность диаметров поверхностей начиная с левой стороны;
- положение поверхностей, вошедших в комплекс;
- допустимость обработки этих поверхностей;
- наличие требования «притупить острые кромки»;
- вид заготовки.

Первый признак мог принимать восемь значений, второй – три, четвертый – два, пятый – два, и шестой – три значения.

Понятие, которое необходимо было сформировать с помощью программы, имело следующий вид:

$$(I^1 \vee 6^1) \wedge 2^2 \wedge (3^3 \vee (1^3 \wedge I^4)) \wedge 2^5 \wedge (1^6 \vee 2^6 \vee 3^6),$$

где верхний индекс обозначает номер признака,

а переменная с индексом – номер значения признака.

Составляли пять обучающих выборок, которые различались по числу входящих в них объектов и по соотношению входящих в них положительных и отрицательных объектов. Выборки составляли так, что первая была произвольной, а последующие формировались добавлением нескольких описаний объектов к предыдущей выборке, т. е. осуществлялся постепенный рост обучающей выборки.

При обработке обучающих выборок на компьютере фиксировались показатели степени обученности и были получены зависимости этих показателей от величины обучающей выборки. Эти зависимости имеют монотонно убывающий характер (рис. 1.1).

Таким образом, при переходе от диалогового режима проектирования к режиму пакетному более высокого уровня степень обученности системы следует оценивать с помощью экзаменующей выборки непосредственно во время процесса проектирования. Обучающие выборки следует накапливать на внешних носителях информации и использовать по мере роста этих выборок.

При достижении показателя степени обученности системы значений, удовлетворяющих технолога-проектировщика (p = 5...10 %), следует переходить к пакетному режиму более высокого уровня.

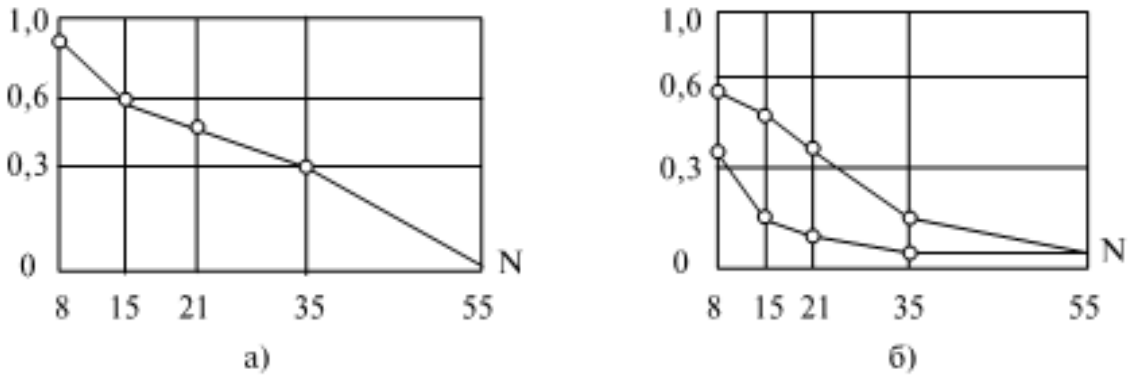


Рис. 1.1. Зависимость частоты ошибок от величины обучающей выборки N:
а – при оценке степени обученности системы;
б – при распознавании положительных и отрицательных объектов

1.3. Классификация САПР. Стадии проектирования электронных средств

Существуют следующие виды обеспечения САПР (рис. 1.2):

• ***Техническое обеспечение*** (ТО) – совокупность связанных и взаимодействующих технических средств (ЭВМ, периферийные устройства, сетевое оборудование, линии связи, измерительные средства).

Рис. 1.2. Виды обеспечения САПР

• ***Математическое обеспечение*** (МО), объединяющее математические методы, модели и алгоритмы, используемые для решения задач автоматизированного проектирования. По назначению и способам реализации делят на две части:

– математические методы и построенные на них математические модели;

– формализованное описание технологии автоматизированного проектирования.

• ***Программное обеспечение*** (ПО), которое подразделяется на общесистемное и прикладное:

– *прикладное ПО* реализует математическое обеспечение для непосредственного выполнения проектных процедур. Включает паке-

ты прикладных программ, предназначенные для обслуживания определенных этапов проектирования или решения групп однотипных задач внутри различных этапов (модуль проектирования трубопроводов, пакет схемотехнического моделирования, геометрический решатель САПР).

– *общесистемное* ПО предназначено для управления компонентами технического обеспечения и обеспечения функционирования прикладных программ. Примером компонента общесистемного ПО является операционная система.

- ***Информационное обеспечение*** (ИО) – совокупность сведений, необходимых для выполнения проектирования. Состоит из описания стандартных проектных процедур, типовых проектных решений, комплектующих изделий и их моделей, правил и норм проектирования. Основная часть ИО САПР – базы данных.
- ***Лингвистическое обеспечение*** (ЛО) – совокупность языков, используемых в САПР для представления информации о проектируемых объектах, процессе и средствах проектирования, а также для осуществления диалога «проектировщик – ЭВМ» и обмена данными между техническими средствами САПР. Включает термины, определения, правила формализации естественного языка, методы сжатия и развертывания. В лингвистическом обеспечении выделяют класс различного типа языков проектирования и моделирования (VHDL, VERILOG, UML, GPSS).
- ***Методическое обеспечение*** (МетО) – описание технологии функционирования САПР, методов выбора и применения пользователями технологических приемов для получения конкретных результатов. Включает в себя теорию процессов, происходящих в проектируемых объектах, методы анализа, синтеза систем и их составных частей, различные методики проектирования. Иногда к МетО относят также МО и ЛО.
- ***Организационное обеспечение*** (ОО) – совокупность документов, определяющих состав проектной организации, связь между подразделениями, организационную структуру объекта и системы автоматизации, деятельность в условиях функционирования системы, форму представления результатов проектирования. В ОО входят штатные расписания, должностные инструкции, правила эксплуатации, приказы, положения и т. п.

В САПР как проектируемой системе выделяют также эргономическое и правовое обеспечение.

• ***Эргономическое обеспечение*** объединяет взаимосвязанные требования, направленные на согласование психологических, психофизиологических, антропометрических характеристик и возможностей человека с техническими характеристиками средств автоматизации и параметрами рабочей среды на рабочем месте.

• ***Правовое обеспечение*** состоит из правовых норм, регламентирующих правоотношения при функционировании САПР, и юридический статус результатов её функционирования.

В области классификации САПР (рис. 1.3) используется ряд устоявшихся англоязычных терминов, применяемых для классификации программных приложений и средств автоматизации САПР по отраслевому и целевому назначению.

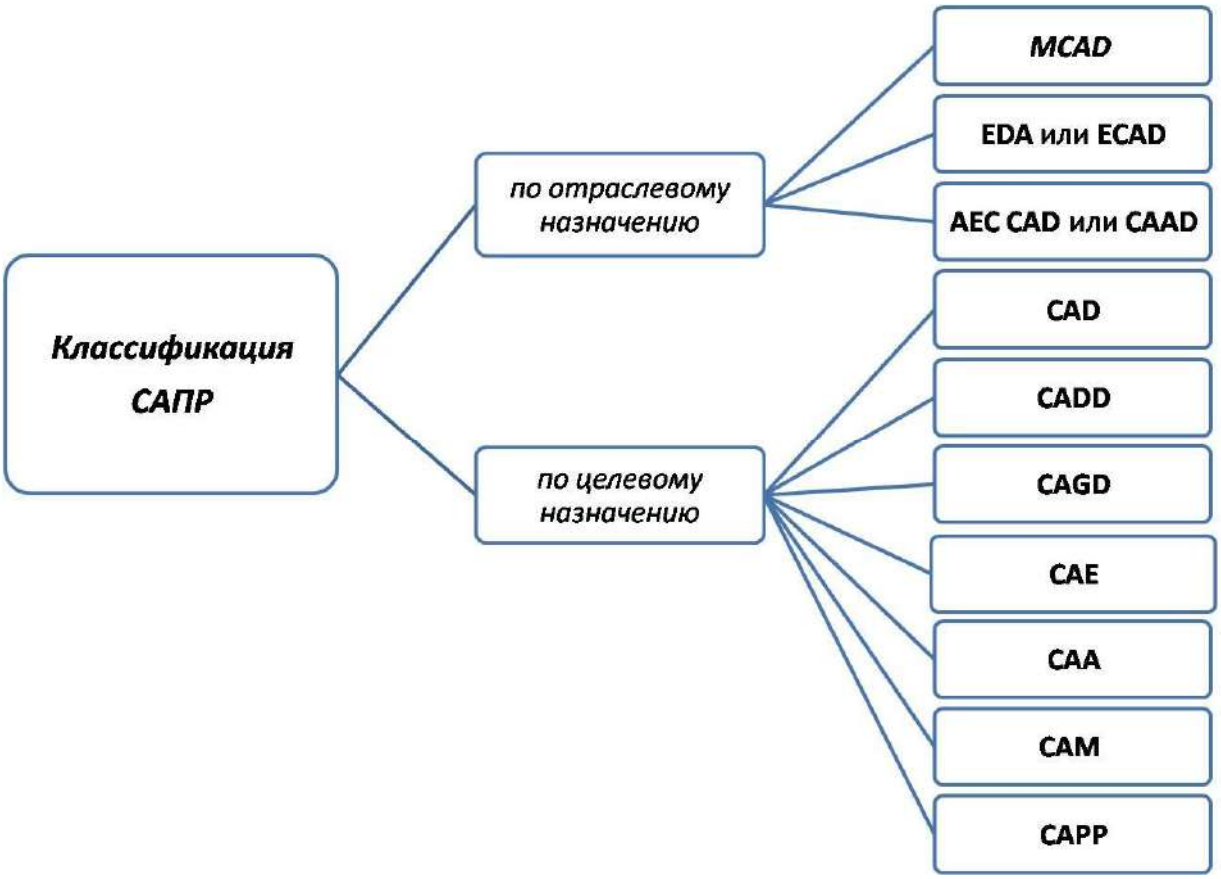


Рис. 1.3. Классификация САПР

– По отраслевому назначению:

MCAD – автоматизированное проектирование механических устройств. Это машиностроительные САПР, применяются в автомобиле- и судостроении, авиакосмической промышленности, производстве товаров народного потребления, включают в себя разработку деталей и сборок (механизмов) с использованием параметрического проекти-

рования на основе конструктивных элементов, технологий поверхностного и объемного моделирования (SolidWorks, Autodesk Inventor, КОМПАС, CATIA);

EDA или **ECAD** – САПР электронных устройств, радиоэлектронных средств, интегральных схем, печатных плат и т.п. (Altium Designer, OrCAD);

AEC CAD или **CAAD** – САПР в области архитектуры и строительства. Используются для проектирования зданий, промышленных объектов, дорог, мостов и пр. (Autodesk Architectural Desktop, AutoCAD Revit Architecture Suite, Piranesi, ArchiCAD).

– *По целевому назначению* различают САПР или подсистемы САПР, которые обеспечивают различные аспекты проектирования:

CAD – средства автоматизированного проектирования, в контексте указанной классификации термин обозначает средства САПР, предназначенные для автоматизации двумерного и/или трехмерного геометрического проектирования, создания конструкторской и/или технологической документации, и САПР общего назначения.

CADD – проектирование и создание чертежей.

CAGD – геометрическое моделирование.

CAE – средства автоматизации инженерных расчётов, анализа и симуляции физических процессов, осуществляют динамическое моделирование, проверку и оптимизацию изделий.

CAA – подкласс средств CAE, используемых для компьютерного анализа.

CAM – средства технологической подготовки производства изделий, обеспечивают автоматизацию программирования и управления оборудования с ЧПУ или ГАПС (гибкие автоматизированные производственные системы). Русским аналогом термина является АСТПП – автоматизированная система технологической подготовки производства.

CAPP – средства автоматизации планирования технологических процессов, применяемые на стыке систем CAD и CAM.

Многие системы автоматизированного проектирования совмещают в себе решение задач, относящихся к различным аспектам проектирования CAD/CAM, CAD/CAE, CAD/CAE/CAM. Такие системы называют комплексными или интегрированными.

С помощью CAD-средств создаётся геометрическая модель изделия, которая используется в качестве входных данных в системах CAM и на основе которой в системах CAE формируется требуемая для инженерного анализа модель исследуемого процесса.

Стадии проектирования – это проектные исследования, технические задания, техническое предложение эскизного, технического и рабочего проектов, испытаний и внедрения.

Стадия научно-исследовательской работы определяет назначение, принципы построения (создания) ЭВМ и формирует техническое задание на его проектирование.

На стадии эскизного проекта проверяется корректность и реализуемость основных принципов и положений, определяющих функционирование будущей ЭВМ, и создается эскизный проект.

На стадии технического проекта ведется всесторонняя проработка всех частей проекта, конкретизация и детализация технических решений.

На стадиях рабочего проекта, испытаний и внедрений формируется вся необходимая документация для изготовления изделия. Далее создается и испытывается опытный образец или пробная партия изделий, по результатам испытаний вносятся необходимые коррективы в проектную документацию, затем – внедрение в производство.

Способ организации процесса проектирования ЭС заключается в создании модели процесса проектирования, основанной на концепции управления.

Первый вариант модели – это схема процесса проектирования. Она включает в себя:

• цель проектирования, которая неизменна;

• знания технологии определенного типа для создания проекта;

• информацию (проект), которая может быть документирована и использована для производства тем или иным способом в процессе проектирования.

Второй вариант – это модель процесса производства. Если цель не достигнута, то проектные решения корректируются. Данные об отклонении предварительного проекта от спецификации передаются к операции синтеза. В среде проектирования находятся вычислительные средства, методическое обеспечение, сам проектировщик.

Проектные процедуры подразделяются на задачи анализа и синтеза (рис. 1.4). Синтез заключается в создании описания вычислительной системы, а анализ – в определении свойств и исследовании работоспособности объекта по его описанию, т.е. при синтезе создаются, а при анализе оцениваются проекты ВС.

Процедуры анализа могут быть одно- и многовариантные. *Одновариантный анализ* предполагает задание значений внутренних и внешних параметров и определение значений выходных параметров объекта. Задача анализа с одним вариантом сводится к однократному решению уравнений, составляющих математическую модель. *Многовариантный анализ* заключается в исследовании свойств ВС в некоторой области пространства внутренних параметров. Такой анализ требует многократного решения систем уравнений.

Рис. 1.4. Классификация проектных процедур

Процедуры синтеза – параметрические и структурные. Целью *структурного синтеза* является определение структуры ВС перечня типов элементов, составляющих ВС, и способа связи элементов (оборудования) между собой в составе ВС. *Параметрический синтез* заключается в определении числовых значений параметров элементов при заданных значениях структуры и условиях работоспособности выходных параметров объекта, т.е. при параметрическом синтезе необходимо определить точку или область в пространстве внутренних параметров, в которых выполняются те или иные условия [8].

1.4. Основные требования к математическим моделям объектов проектирования электронных средств. Методика составления математических моделей

При построении математических моделей (ММ) используют различные математические средства описания объекта: теорию множеств, теорию графов, теорию вероятностей, математическую логику,

математическое программирование, дифференциальные или интегральные уравнения и т. д.

Выполнение проектных операций и процедур в САПР основано на оперировании ММ. С их помощью прогнозируются характеристики и оцениваются возможности предложенных вариантов схем и конструкций, проверяется их соответствие предъявляемым требованиям, проводится оптимизация параметров, разрабатывается техническая документация.

В САПР для каждого иерархического уровня сформулированы основные положения математического моделирования – выбран и развит соответствующий математический аппарат, получены типовые ММ элементов проектируемых объектов, формализованы методы получения и анализа математических моделей систем. Сложность задач проектирования и противоречивость требований высокой точности, полноты и малой трудоемкости анализа обусловливают целесообразность компромиссного удовлетворения этих требований с помощью соответствующего выбора моделей. Это обстоятельство приводит к расширению множества используемых моделей и развитию алгоритмов адаптивного моделирования.

Основными требованиями, предъявляемыми к математическим моделям, являются требования адекватности, универсальности и экономичности.

Модель считается *адекватной*, если отражает заданные свойства объекта с приемлемой точностью. *Точность* определяется как степень совпадения значений выходных параметров модели и объекта.

Пусть ε_j – относительная погрешность модели по j-му выходному параметру:

$$\varepsilon_j = \frac{\tilde{y} - y_j}{y_j}, \qquad (1.5)$$

где $\tilde{y}$ – j-й выходной параметр, рассчитанный с помощью модели;

y_j – тот же выходной параметр, существующий в моделируемом объекте.

Погрешность модели ε_j по совокупности учитываемых выходных параметров оценивается одной из норм вектора $\varepsilon_j = (\varepsilon_1, \varepsilon_2, \ldots \varepsilon_m)$.

Точность модели различна в разных условиях функционирования объекта. Эти условия характеризуются внешними параметрами. Если задаться предельной допустимой погрешностью *\varepsilon _{пред}*,

то можно в пространстве внешних параметров выделить область, в которой выполняется условие

$$\varepsilon_M < \varepsilon_{пред.} \qquad (1.6)$$

Эту область называют *областью адекватности* (ОА) модели. Возможно введение индивидуальных предельных значений $\varepsilon_{пред}$ для каждого выходного параметра и определение ОА как области, в которой одновременно выполняются все m условий вида $|\varepsilon_j| \le \varepsilon_{предj}$.

Определение областей адекватности для конкретных моделей – сложная процедура, требующая больших вычислительных затрат. Эти затраты и трудности представления ОА быстро растут с увеличением размерности пространства внешних параметров. Определение ОА – более трудная задача, чем, например, задача параметрической оптимизации. Для моделей унифицированных элементов расчет областей адекватности становится оправданным в связи с однократностью определения ОА и многократностью их использования при проектировании различных систем. Знание ОА позволяет правильно выбирать модели элементов из числа имеющихся и тем самым повышать достоверность результатов машинных расчетов.

В библиотеку моделей элементов, наряду с алгоритмом, реализующим модель, и номинальными значениями параметров, должны включаться граничные значения внешних параметров q_k' и q_k'', задающие область адекватности.

При определении ОА необходимо выбрать совокупность внешних параметров и совокупность выходных параметров y_j, отражающих учитываемые в модели свойства. Типичными внешними параметрами при этом являются параметры нагрузки и внешних воздействий (электрических, механических, тепловых, радиационных и т. п.). Увеличение числа учитываемых внешних факторов расширяет применимость модели, но существенно удорожает работу по определению ОА. Выбор совокупности выходных параметров также неоднозначен, однако для большинства объектов число и перечень учитываемых свойств и соответствующих им выходных параметров сравнительно невелики, достаточно стабильны и составляют типовой набор выходных параметров. Например, для макромоделей логических элементов БИС такими выходными параметрами являются уровни

выходного напряжения в состояниях логических «0» и «1», запасы помехоустойчивости, задержка распространения сигнала, рассеиваемая мощность.

Если адекватность характеризуется положением и размерами ОА, то универсальность модели определяется числом и составом учитываемых в модели внешних и выходных параметров.

Экономичность модели характеризуется затратами вычислительных ресурсов для ее реализации, а именно затратами машинного времени T_M и памяти $П_M$. Общие затраты T_M и $П_M$ на выполнение в САПР какой-либо проектной процедуры зависят как от особенностей выбранных моделей, так и от методов решения.

В большинстве случаев при реализации численного метода происходят многократные обращения к модели элемента, входящего в состав моделируемого объекта. Тогда удобно экономичность модели элемента характеризовать затратами машинного времени при обращении к модели, а число обращений к модели должно учитываться при оценке экономичности метода решения.

Экономичность модели по затратам памяти оценивается объемом оперативной памяти, необходимой для реализации модели.

Требования широких областей адекватности, высокой степени универсальности, с одной стороны, и высокой экономичности – с другой, являются противоречивыми. Наилучшее компромиссное удовлетворение этих требований оказывается неодинаковым в различных применениях. Это обстоятельство обусловливает использование в САПР многих моделей для объектов одного и того же типа – разного рода макромоделей, многоуровневых, смешанных моделей и т. п.

Методика составления математических моделей заключается в следующем:

1. Выбор свойств ВС, которые подлежат отображению в ММ. Этот выбор основан на анализе возможных применений модели и определяет степень универсальности ММ.

2. Сбор исходной информации о выбранных свойствах объекта (опыт и знания проектировщика, научно-техническая и справочная литература, описание прототипов).

3. Синтез структуры ММ. Структура ММ – общий вид математических соотношений модели без конкретизации числовых значений параметров (в виде графа, схемы, формул).

4. Расчет числовых значений параметров ММ (минимизация погрешностей модели заданной структуры):

$$\min \varepsilon_{M(Y)} \text{ при } Y \in Y_D, \qquad (1.7)$$

где Y – вектор параметров модели;

Y_D – область варьирования параметров;

$\varepsilon_{M(Y)}$ – погрешность ММ.

Оценка адекватности ММ – сравнение расчетных значений с реальными (экспериментальными) данными.

Рассмотрим классификацию моделей структурного синтеза:

1) перебор вариантов готовых законченных структур;

2) последовательный синтез;

3) трансформация описания разных аспектов.

1. Перебор вариантов. Такие структуры создаются заранее и хранятся в базе данных. Выбор варианта основан либо на случайной выборке, либо на эвристических способностях человека в диалоговом режиме, либо на установлении корреляции параметров, характеризующих структуру. Затем выполняется оценка варианта структуры с помощью процедуры параметрического анализа и синтеза, и принимается решение на основе результатов оценки о выборе лучшей структуры из числа рассмотренных структур. Для такого сравнения вариантов предусматриваются некоторые критерии, объединяющие частные показатели.

Если задачу структурного синтеза удается сформулировать как задачу дискретного математического программирования, то она определяется выражением

$$\text{extr } F(Y) \text{ при } Y \in Y_D, \qquad (1.8)$$

где Y_D – дискретное множество;

$F(Y)$ – целевая функция;

$Y = \{y_i\}$, y_i – элементы некоторого типа в структуре.

2. Последовательный синтез характеризуется поэтапным решением задачи синтеза с возможностями оценки получающихся промежуточных структур. Отмечают два способа последовательного синтеза:

– *наращивание,* т.е. поочередное добавление элементов к исходной структуре (алгоритмы компоновки и размещения, где оценкой является количество межблочных связей, и предпочтение отдается тем промежуточным вариантам, при которых большое число связей оказывается сконцентрированным в пределах одного блока);

– *выделение,* т.е. из некоторой обобщенной структуры постепенно удаляются лишние элементы (обобщенные технологические маршруты обработки деталей, обработки печатных плат, куда включены операции, которые могут встретиться при различных сочетаниях конструктивных особенностей). Сопоставление чертежа конкретной платы и обобщенного маршрута позволяет убрать лишние операции и сформировать конкретный технологический маршрут.

3. Трансформация описаний разных аспектов, например, изготовление конструкторской документации, где формализовано преобразование результатов конструкторского проектирования в графическое изображение, выполняемое по правилу проекционного черчения.

Кроме указанных процедур синтеза существуют комбинированные, т.е. сочетание этих процедур. Например, диалоговые, в которых процедуры оценки выполняет компьютер, а принятие решения остается за человеком. Назначение компьютера – подсказать варианты; назначение человека – модифицировать структуру. Возможно применение экспертных систем для генерации вариантов структуры и для связи пользователя с САПР в режиме диалога (экспертные системы воспринимают от высококвалифицированных специалистов знания, а затем используют их при решении задач структурного синтеза).

Контрольные вопросы

1. Перечислите пути повышения интеллектуальности подсистем проектирования.
2. Что понимают под полезностью альтернативы?
3. Какие существуют способы оценки полезности?
4. Как формируют обучающие выборки?
5. Что является основой для развития искусственного интеллекта?
6. Поясните понятие «интеллект».
7. Какова основная задача создания искусственного интеллекта?
8. Как работает лабиринтная модель?
9. Поясните работу ассоциативной модели.
10. Почему используется модельная гипотеза?
11. Каковы цели и задачи искусственного интеллекта?
12. Какие современные проблемы решаются в искусственном интеллекте?

13. Поясните два направления исследований в области искусственного интеллекта.

14. Что называют машинным интеллектом?

15. Что входит в понятие «искусственный разум»?

16. Поясните понятие «САПР».

17. Перечислите основные принципы создания САПР.

18. Каковы основные виды обеспечения САПР?

19. Поясните классификацию по отраслевому и целевому назначению. Приведите примеры.

20. Что входит в понятие «стадии проектирования»?

21. В чем основное отличие процедур анализа от процедур синтеза?

22. Какая математическая модель считается адекватной? Что такое область адекватности?

23. Как характеризуются универсальность и экономичность математической модели?

Тестовые задания

1. При каком принципе создания САПР обеспечиваются совместное функционирование составных частей САПР и сохранность открытости системы в целом?

а) системного единства;

б) совместимости;

в) типизации;

г) развития.

2. При каком принципе создания САПР обеспечивается проектирование по уровням структуры САПР?

а) системного единства;

б) совместимости;

в) иерархичности;

г) развития.

3. Совокупность языков, используемых в САПР для представления информации о проектируемых объектах, процессе и средствах проектирования, относится к такому виду обеспечения САПР, как:

а) лингвистическое;

б) методическое;

в) информационное;

г) техническое.

4. Описание технологии функционирования САПР, методов выбора и применения пользователями технологических приемов для получения конкретных результатов относится к такому виду обеспечения САПР, как:

а) лингвистическое;

б) техническое;

в) информационное;

г) методическое.

5. Из описания стандартных проектных процедур, типовых проектных решений, комплектующих изделий и их моделей, правил и норм проектирования состоит такой вид обеспечения САПР, как:

а) лингвистическое;

б) техническое;

в) информационное;

г) методическое.

6. САПР для проектирования электронных устройств, радиоэлектронных средств, интегральных схем, печатных плат относится к:

а) CAE;

б) CAM;

в) EDA или ECAD;

г) CAGD.

7. Средства технологической подготовки производства изделий, которые обеспечивают автоматизацию программирования и управления оборудования с ЧПУ, относятся к:

а) CAE;

б) CAM;

в) EDA или ECAD;

г) CAGD.

8. В определении числовых значений параметров элементов при заданных значениях структуры и условиях работоспособности выходных параметров объекта заключается:

а) параметрический синтез;

б) многовариантный анализ;

в) структурный синтез;

г) одновариантный анализ.

9. В исследовании свойств ВС в некоторой области пространства внутренних параметров заключается:

а) параметрический синтез;

б) многовариантный анализ;

в) структурный синтез;

г) одновариантный анализ.

10. Расположите этапы составления математических моделей в правильном порядке:

	Расчет числовых значений параметров математической модели.
	Выбор свойств ВС, которые подлежат отображению в математической модели.
	Синтез структуры математической модели.
	Сбор исходной информации о выбранных свойствах объекта.

2. СИСТЕМНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ ЭЛЕКТРОННЫХ СРЕДСТВ

2.1. Общая характеристика проблемы

Основные идеи и принципы проектирования сложных систем выражены в системном подходе. Для специалиста в области системотехники они являются очевидными и естественными, тем не менее их соблюдение и реализация зачастую сопряжены с определенными трудностями, обусловливаемыми особенностями проектирования. Однако интуитивный подход без применения правил системного анализа может оказаться недостаточным для решения все более усложняющихся задач инженерной деятельности. Общий принцип системного подхода заключается в рассмотрении частей явления или сложной системы с учетом их взаимодействия.

Системный подход включает в себя выявление структуры системы, типизацию связей, определение атрибутов, анализ влияния внешней среды. Системный подход рассматривают как направление научного познания и социальной политики.

Теория систем – дисциплина, в которой конкретизируются положения системного подхода; она посвящена исследованию и проектированию сложных экономических, социальных, технических систем, чаще всего слабоструктурированных. Характерными примерами таких систем являются производственные системы. При проектировании систем цели достигаются в многошаговых процессах принятия решений. Методы принятия решений часто выделяют в самостоятельную дисциплину, называемую «Теория принятия решений».

В технике дисциплину, аналогичную теории систем, в которой исследуются технические системы, их проектирование, чаще называют системотехникой. Предметом системотехники являются, во-первых, организация процесса создания, использования и развития технических систем, во-вторых, методы, принципы их проектирования и исследования. В системотехнике важно уметь сформулировать цели системы и организовать ее рассмотрение с позиций поставленных целей. Тогда можно отбросить лишние и малозначимые части при проектировании и моделировании, перейти к постановке оптимизационных задач.

Системы автоматизированного проектирования и управления относятся к числу сложных современных искусственных систем.

Их проектирование и сопровождение невозможны без системного подхода. Поэтому идеи и положения системотехники входят составной частью в дисциплины, посвященные изучению современных автоматизированных систем и технологий их применения. Интерпретация и конкретизация системного подхода имеют место в ряде известных подходов с другими названиями, которые также можно рассматривать как компоненты системотехники. Таковы структурный, блочно-иерархический, объектно-ориентированный подходы.

Математический аппарат, используемый в рассматриваемой большой системе, очень разнообразен из-за множества функций и процессов, имеющих место в подсистемах.

Теория множеств и теория графов используются при проектировании конструкций аппаратуры.

Теория подобия и моделирования применяется для создания моделей и исследования разрабатываемых конструкций и технологических процессов.

Теория вероятностей и математическая статистика широко используются при построении математических моделей из-за множества случайных воздействий при производстве и эксплуатации.

Методы пассивного и активного экспериментов применяются для построения математических моделей процессов.

Методы оптимизации позволяют отрабатывать на моделях качество объектов и технологических процессов при тех или иных ограничениях.

Теория случайных процессов дает возможность описания и исследования самых разнообразных процессов, происходящих в конструкциях аппаратуры при производстве и в эксплуатации.

Теория массового обслуживания позволяет решать многочисленные задачи обслуживания объектов при производстве и эксплуатации.

Теория надежности используется для исследования объектов и повышения надежности на всех этапах их «жизни».

Все многообразие разработанных и используемых для исследования сложных стохастических процессов математических методов можно разбить на две группы [33]:

1) вероятностно-статистические методы, включающие использование общих идей теории вероятностей, выборочного метода и проверку статистических гипотез, дисперсионного и регрессионного анализов, статистического планирования эксперимента;

2) методы исследования операций, включающие линейное, нелинейное и динамическое программирование, теорию массового обслуживания, теорию игр, теорию графов и сетей.

Первые методы успешно применяются для решения технологических задач, таких как:

– статистический анализ накопленных данных о техпроцессах и свойствах изделий для обобщения информации, для изучения влияния производственных факторов на показатели изделий;

– разработка математико-статистических моделей для принятия оптимальных технических и экономических решений, а также для управления процессом или отдельными операциями с использованием ЭВМ.

Общая схема решения технологических задач математико-статистическими методами приведена на рис. 2.1 [33].

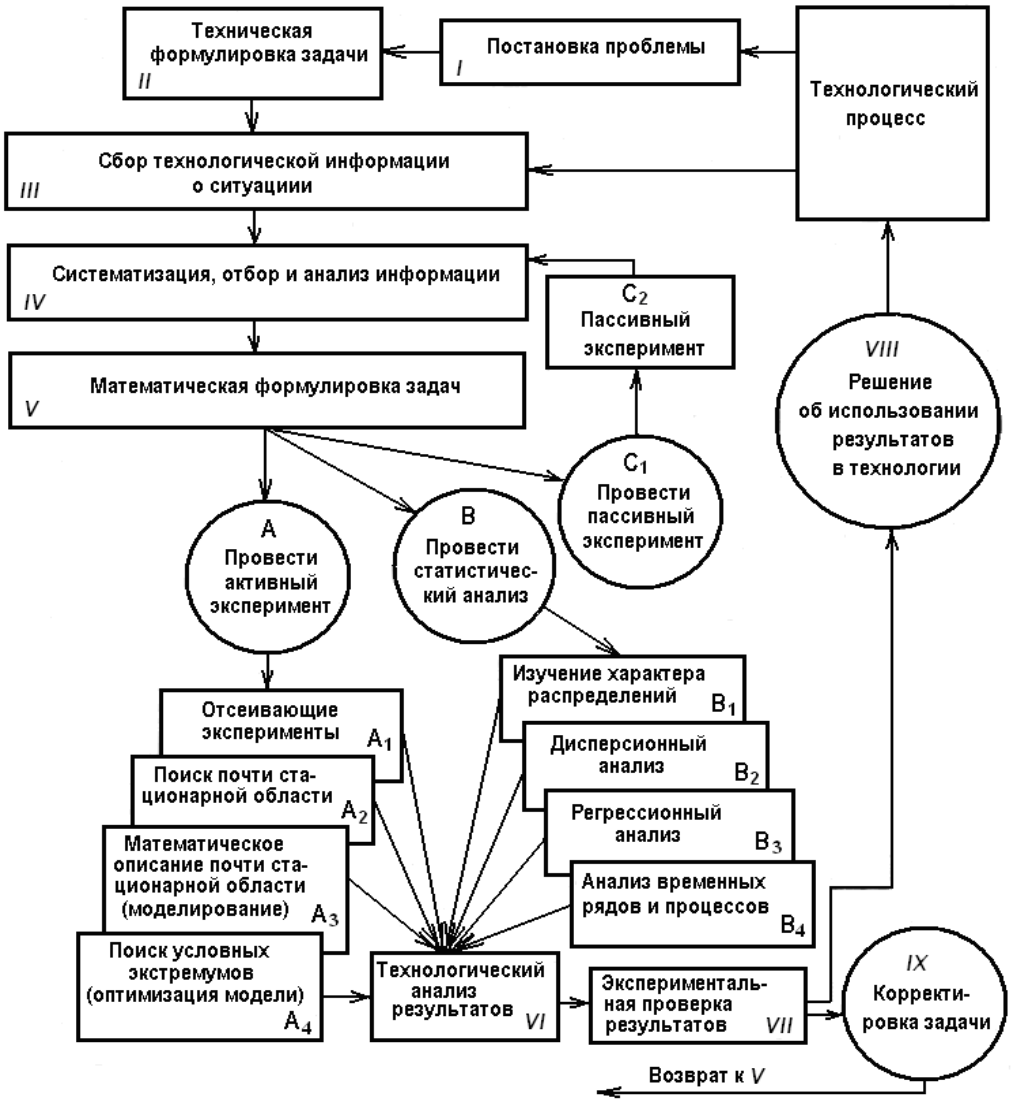


Рис. 2.1. Общая схема решения технологических задач математико-статистическими методами

Вторые методы используются для решения задач проектирования и обслуживания.

Если говорить о моделях, то существуют аналитические и статистические модели. Аналитические модели более грубы, учитывают меньшее число факторов, всегда требуют каких-то допущений и упрощений. Зато результаты расчета по ним легче обозримы, отчетливее отражают присущие явлению основные закономерности. А главное, аналитические модели более приспособлены для поиска оптимальных решений.

Статистические модели, по сравнению с аналитическими, более точны и подробны, не требуют грубых допущений, позволяют учесть большое число факторов. Но им присущи следующие недостатки: громоздкость, плохая обозримость, большой расход машинного времени, а главное, крайняя трудность поиска оптимальных решений, которые приходится искать «на ощупь», путем догадок и проб.

2.2. Сущность структурного подхода к проектированию электронных средств

Сущность структурного подхода к разработке информационной системы (ИС), описывающей работу технической системы, заключается в ее декомпозиции (разбиении) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и т.д. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны. При разработке системы «снизу вверх» – от отдельных задач ко всей системе – целостность теряется, возникают проблемы при информационной стыковке отдельных компонентов.

Все наиболее распространенные методологии структурного подхода [23] базируются на ряде общих принципов [13]. В качестве базовых используются два принципа:

- «разделяй и властвуй» – принцип решения сложных проблем путем их разбиения на множество меньших независимых задач, легких для понимания и решения;
- иерархического упорядочивания – принцип организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.

Выделение двух базовых принципов не означает, что остальные принципы являются второстепенными, поскольку игнорирование любого из них может привести к непредсказуемым последствиям (в том числе и к провалу всего проекта). К основным из них относятся принципы:

- абстрагирования – заключается в выделении существенных аспектов системы и отвлечении от несущественных;
- формализации – состоит в необходимости строгого методического подхода к решению проблемы;
- непротиворечивости – заключается в обоснованности и согласованности элементов;
- структурирования данных – данные должны быть структурированы и иерархически организованы.

В структурном анализе используются в основном две группы средств, иллюстрирующих функции, выполняемые системой и отношения между данными. Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными среди которых являются следующие:

- *SADT* (Structured Analysis and Design Technique) – модели и соответствующие функциональные диаграммы;
- *DFD* (Data Flow Diagrams) – диаграммы потоков данных;
- *ERD* (Entity-Relationship Diagrams) – диаграммы «сущность–связь».

На стадии проектирования ИС модели расширяются, уточняются и дополняются диаграммами, отражающими структуру программного обеспечения: архитектуру программного обеспечения, структурные схемы программ и диаграммы экранных форм. Перечисленные модели в совокупности дают полное описание ИС независимо от того, является ли она существующей или вновь разрабатываемой. Состав диаграмм в каждом конкретном случае зависит от необходимой полноты описания системы.

2.3. Методология функционального моделирования SADT при проектировании сложных систем

Методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области. Функциональная модель SADT отображает функциональную структуру объекта,

т.е. производимые им действия и связи между этими действиями. Основные элементы методологии основываются на следующих концепциях:

• Графическое представление блочного моделирования. Графика блоков и дуг SADT-диаграммы отображает функцию в виде блока, а интерфейсы входа/выхода представляются дугами, соответственно входящими в блок и выходящими из него. Взаимодействие блоков друг с другом описывается посредством интерфейсных дуг, выражающих «ограничения», которые в свою очередь определяют, когда и каким образом функции выполняются и управляются;

• Строгость и точность. Выполнение правил SADT требует достаточной строгости и точности, не накладывая в то же время чрезмерных ограничений на действия аналитика.

Правила SADT включают:

– ограничение количества блоков на каждом уровне декомпозиции (правило трех – шести блоков);

– связность диаграмм (номера блоков);

– уникальность меток и наименований (отсутствие повторяющихся имен);

– синтаксические правила для графики (блоков и дуг);

– разделение входов и управлений (правило определения роли данных);

– отделение организации от функции, т.е. исключение влияния организационной структуры на функциональную модель.

2.4. Состав функциональной модели

Результатом применения методологии SADT является модель, которая состоит из диаграмм, фрагментов текстов и глоссария, имеющих ссылки друг на друга. Диаграммы – главные компоненты модели, все функции ИС и интерфейсы на них представлены как блоки и дуги. Место соединения дуги с блоком определяет тип интерфейса. Управляющая информация входит в блок сверху, в то время как информация, которая подвергается обработке, показана с левой стороны блока, а результаты выхода – с правой стороны. Механизм (человек или автоматизированная система), который осуществляет операцию, представляется дугой, входящей в блок снизу (рис. 2.2).

Одной из наиболее важных особенностей методологии SADT является постепенное введение все бóльших уровней детализации по мере создания диаграмм, отображающих модель.

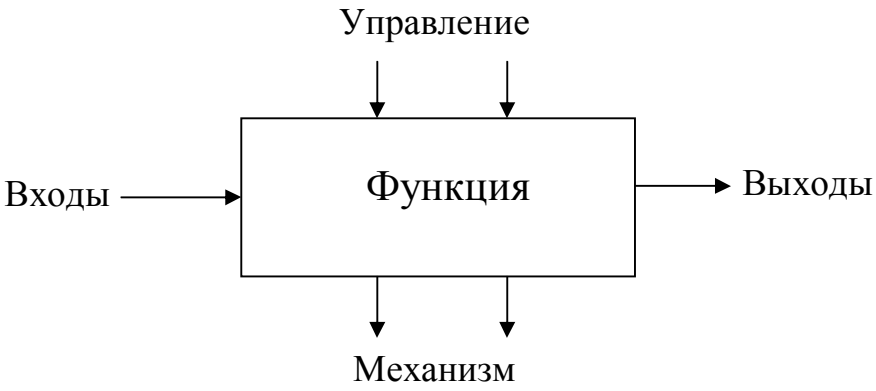


Рис. 2.2. Функциональный блок и интерфейсные дуги

Иерархия диаграмм. Построение SADT-модели начинается с представления всей системы в виде простейшего компонента – одного блока и дуг, изображающих интерфейсы с функциями вне системы. Поскольку единственный блок представляет всю систему как единое целое, имя, указанное в блоке, является общим. Это верно и для интерфейсных дуг – они также представляют полный набор внешних интерфейсов системы в целом.

Затем блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Эти блоки представляют основные подфункции исходной функции. Данная декомпозиция выявляет полный набор подфункций, каждая из которых представлена как блок, границы последнего определены интерфейсными дугами. Каждая из этих подфункций может быть декомпозирована подобным образом для более детального представления.

Во всех случаях каждая подфункция может содержать только те элементы, которые входят в исходную функцию. Кроме того, модель не может опустить какие-либо элементы, т.е. родительский блок и его интерфейсы обеспечивают контекст. К нему нельзя ничего добавить, и из него не может быть ничего удалено.

Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые представлены в виде блоков (рис. 2.3). Детали каждого из основных блоков показаны в виде блоков на других диаграммах.

Каждая детальная диаграмма является декомпозицией блока из более общей диаграммы. На каждом шаге декомпозиции более общая диаграмма называется родительской для более детальной диаграммы.

Дуги, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются теми же самыми, что и дуги, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма представляют одну и ту же часть системы.

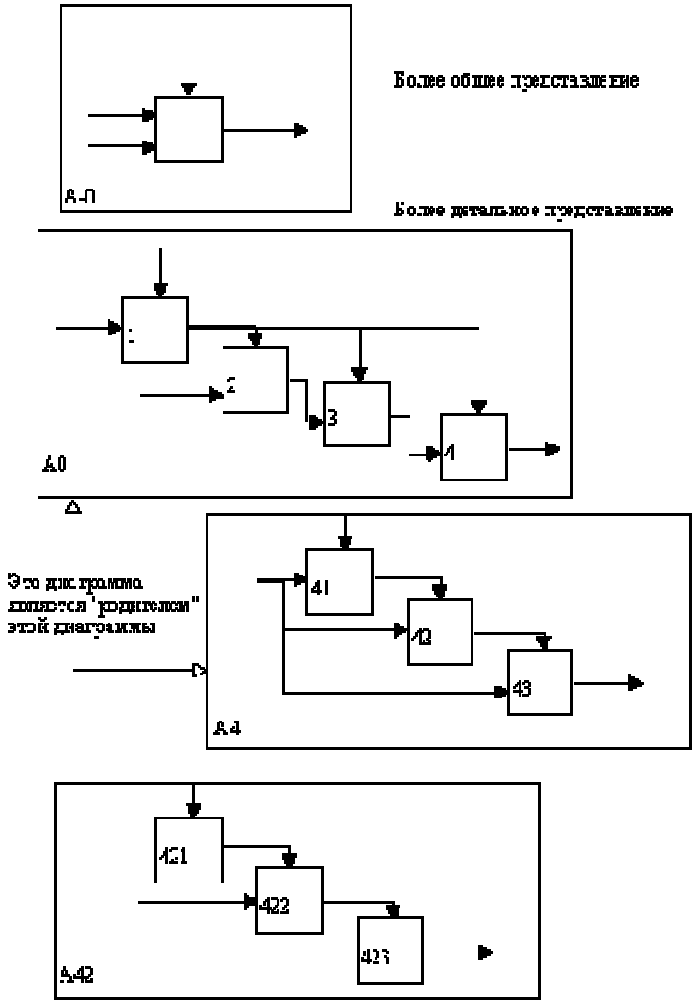


Рис. 2.3. Структура SADT-модели. Декомпозиция диаграмм

2.5. Типы связей между функциями в SADT-моделях

Системный подход позволяет найти оптимальное, в широком смысле, решение задачи проектирования за счет всестороннего, целостного рассмотрения как проектируемого изделия, так и самого процесса проектирования, и способен привести к подлинно творческим новаторским решениям, включая крупные изобретения и научные открытия.

Методическим средством реализации системного подхода к исследованию, проектированию или управлению сложным процессом служит системный анализ, под которым понимается совокупность приемов и методов исследования объектов (процессов) посредством представления их в виде систем и их последующего анализа.

Одним из важнейших моментов при моделировании бизнес-процессов организации с помощью метода SADT является точная согласованность типов связей между функциями. Различают по крайней мере связи семи типов (в порядке возрастания их относительной значимости):

– случайная;
– логическая;
– временная;
– процедурная;
– коммуникационная;
– последовательная;
– функциональная.

1) *Тип случайной связности* – наименее желательный. Случайная связность возникает, когда конкретная связь между функциями мала или полностью отсутствует. Это относится к ситуации, когда имена данных на SADT-дугах в одной диаграмме имеют малую связь друг с другом.

2) *Тип логической связности.* Логическое связывание происходит тогда, когда данные и функции собираются вместе вследствие того, что они попадают в общий класс или набор элементов, но необходимых функциональных отношений между ними не обнаруживается.

3) *Тип временной связности.* Связанные по времени элементы возникают вследствие того, что они представляют функции, связанные во времени, когда данные используются одновременно или функции включаются параллельно, а не последовательно.

4) *Тип процедурной связности.* Процедурно-связанные элементы появляются сгруппированными вместе вследствие того, что они выполняются в течение одной и той же части цикла или процесса.

5) *Тип коммуникационной связности*. Диаграммы демонстрируют коммуникационные связи, когда блоки группируются вследствие того, что они используют одни и те же входные данные и/или производят одни и те же выходные данные.

6) *Тип последовательной связности.* На диаграммах, имеющих последовательные связи, выход одной функции служит входными

данными для следующей функции. Связь между элементами на диаграмме является более тесной, чем на рассмотренных выше уровнях связок, поскольку моделируются причинно-следственные зависимости.

7) *Тип функциональной связности.* Диаграмма отражает полную функциональную связность при наличии полной зависимости одной функции от другой. Диаграмма, которая является чисто функциональной, не содержит чужеродных элементов, относящихся к последовательному или более слабому типу связности. Одним из способов определения функционально-связанных диаграмм является рассмотрение двух блоков, связанных через управляющие дуги.

Важно отметить, что уровни 4 – 6 устанавливают типы связностей, которые разработчики считают важнейшими для получения диаграмм хорошего качества.

2.6. CASE-средства. Общая характеристика и классификация

Современные CASE-средства охватывают обширную область поддержки многочисленных технологий проектирования ИС: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл программного обеспечения (ПО).

Наиболее трудоемкими этапами разработки ИС являются этапы анализа и проектирования, в процессе которых CASE-средства обеспечивают качество принимаемых технических решений и подготовку проектной документации. При этом большую роль играют методы визуального представления информации. Это предполагает построение структурных или иных диаграмм в реальном масштабе времени, использование многообразной цветовой палитры, сквозную проверку синтаксических правил. Графические средства моделирования предметной области позволяют разработчикам в наглядном виде изучать существующую ИС, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями.

В разряд CASE-средств попадают как относительно дешевые системы для персональных компьютеров с весьма ограниченными возможностями, так и дорогостоящие системы для неоднородных вычислительных платформ и операционных сред. Так, современный рынок программных средств насчитывает около 300 различных CASE-средств, наиболее мощные из которых так или иначе используются практически всеми ведущими западными фирмами.

К CASE-средствам относят, как правило, любое программное средство, автоматизирующее ту или иную совокупность процессов жизненного цикла ПО и обладающее такими характерными особенностями, как:

- мощные графические средства для описания и документирования ИС, обеспечивающие удобный интерфейс с разработчиком и развивающие его творческие возможности;
- интеграция отдельных компонентов CASE-средств, обеспечивающая управляемость процессом разработки ИС;
- использование специальным образом организованного хранилища проектных метаданных (репозитория).

Интегрированное CASE-средство (или комплекс средств, поддерживающих полный жизненный цикл ПО) содержит следующие компоненты;

- репозиторий, являющийся основой CASE-средства. Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость;
- графические средства анализа и проектирования, обеспечивающие создание и редактирование иерархически связанных диаграмм (DFD, ERD и др.), образующих модели ИС;
- средства разработки приложений, включая языки 4GL и генераторы кодов;
- средства конфигурационного управления;
- средства документирования;
- средства тестирования;
- средства управления проектом;
- средства реинжиниринга.

Все современные CASE-средства могут быть классифицированы в основном по типам и категориям. Классификация по типам отражает функциональную ориентацию CASE-средств на те или иные процессы ЖЦ. Классификация по категориям определяет степень интегрированности по выполняемым функциям и включает отдельные локальные средства, решающие небольшие автономные задачи (tools), набор частично интегрированных средств, охватывающих большинство этапов жизненного цикла ИС (toolkit) и полностью интегрированные средства, поддерживающие весь ЖЦ ИС и связанные общим репозиторием.

Помимо этого CASE-средства можно классифицировать по следующим признакам:

- применяемым методологиям и моделям систем и БД;
- степени интегрированности с СУБД;
- доступным платформам.

Классификация по типам в основном совпадает с компонентным составом CASE-средств и включает такие типы, как:

- средства анализа (Upper CASE), предназначенные для построения и анализа моделей предметной области [Design/IDEF (Meta Software), BPwin (Logic Works)];
- средства анализа и проектирования (Middle CASE), поддерживающие наиболее распространенные методологии проектирования и использующиеся для создания проектных спецификаций [Vantage Team Builder (Cayenne), Designer/2000 (ORACLE), Silverrun (CSA), PRO-IV (McDonnell Douglas), CASE.Аналитик (МакроПроджект)]. Выходом таких средств являются спецификации компонентов и интерфейсов системы, архитектуры системы, алгоритмов и структур данных;
- средства проектирования баз данных, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL) для наиболее распространенных СУБД. К ним относятся ERwin (Logic Works), S-Designor (SDP) и DataBase Designer (ORACLE). Средства проектирования баз данных имеются также в составе CASE-средств Vantage Team Builder, Designer/2000, Silverrun и PRO-IV;
- средства разработки приложений, такие как: средства 4GL [Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi (Borland) и др.] и генераторы кодов, входящие в состав Vantage Team Builder, PRO-IV и частично – в Silverrun;
- средства реинжиниринга, обеспечивающие анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций. Средства анализа схем БД и формирования ERD входят в состав Vantage Team Builder, PRO-IV, Silverrun, Designer/2000, ERwin и S-Designor. В области анализа программных кодов наибольшее распространение получают объектно-ориентированные CASE-средства, обеспечивающие реинжиниринг программ на языке C++ [Rational Rose (Rational Software), Object Team (Cayenne)].

К вспомогательным типам относятся:

- средства планирования и управления проектом (SE Companion, Microsoft Project и др.);
- средства конфигурационного управления [PVCS (Intersolv)];
- средства тестирования [Quality Works (Segue Software)];
- средства документирования [SoDA (Rational Software)].

На сегодняшний день российский рынок программного обеспечения располагает такими наиболее развитыми CASE-средствами, как:

- Vantage Team Builder (Westmount I-CASE);
- Designer/2000;
- Silverrun;
- ERwin+BPwin;
- S-Designor;
- CASE.Аналитик.

2.7. Технология внедрения CASE-средств и определение потребностей в них

Технология внедрения CASE-средств базируется в основном на стандартах IEEE (IEEE – Institute of Electrical and Electronics Engineers – Институт инженеров по электротехнике и электронике) [7, 23]. Термин «внедрение» используется в широком смысле и включает все действия – от оценки первоначальных потребностей до полномасштабного использования CASE-средств в различных подразделениях организации-пользователя. Процесс внедрения CASE-средств состоит из следующих этапов [7]:

- определение потребностей в CASE-средствах;
- оценка и выбор CASE-средств;
- выполнение пилотного проекта;
- практическое внедрение CASE-средств.

Процесс успешного внедрения CASE-средств не ограничивается их использованием. На самом деле он охватывает планирование и реализацию множества технических, организационных, структурных процессов, изменений в общей культуре организации и основан на четком понимании возможностей CASE-средств.

На способ внедрения CASE-средств может повлиять специфика конкретной ситуации. Например, если заказчик предпочитает конкретное средство или оно оговаривается требованиями контракта, этапы внедрения должны соответствовать такому предопределенному

выбору. В иных ситуациях относительная простота или сложность средства, степень согласованности или конфликтности с существующими в организации процессами, требуемая степень интеграции с другими средствами, опыт и квалификация пользователей могут привести к внесению соответствующих корректив в процесс внедрения.

Определение потребностей в CASE-средствах. Данный этап включает достижение понимания потребностей организации и технологии последующего процесса внедрения CASE-средств (рис. 2.4). Он должен привести к выделению тех областей деятельности организации, в которых применение CASE-средств может принести реальную пользу. Результатом данного этапа является документ, определяющий стратегию внедрения CASE-средств.

Рис. 2.4. Определение потребностей в CASE-средствах

Для того чтобы принять взвешенное решение относительно инвестиций в CASE-технологию, пользователи вынуждены производить оценку отдельных CASE-средств, опираясь на неполные и противо-

речивые данные. Эта проблема зачастую усугубляется недостаточным знанием всех возможных «подводных камней» использования CASE-средств.

Среди наиболее важных проблем выделяются следующие:

- достоверная оценка отдачи от инвестиций в CASE-средства затруднительна ввиду отсутствия приемлемых метрик и данных по проектам и процессам разработки ПО;
- внедрение CASE-средств может представлять собой достаточно длительный процесс и может не принести немедленной отдачи. Возможно даже краткосрочное снижение продуктивности в результате усилий, затрачиваемых на внедрение. Вследствие этого руководство организации-пользователя может утратить интерес к CASE-средствам и прекратить поддержку их внедрения;
- отсутствие полного соответствия между теми процессами и методами, которые поддерживаются CASE-средствами, и теми, которые используются в данной организации, может привести к дополнительным трудностям;
- CASE-средства зачастую трудно использовать в комплексе с другими подобными средствами. Это объясняется как различными парадигмами, поддерживаемыми разными средствами, так и проблемами передачи данных и управления от одного средства к другому;
- некоторые CASE-средства требуют слишком много усилий для того, чтобы оправдать их использование в небольшом проекте, при этом тем не менее можно извлечь выгоду из той дисциплины, к которой обязывает их применение;
- негативное отношение персонала к внедрению новой CASE-технологии может быть главной причиной провала проекта.

Пользователи CASE-средств должны быть готовы к необходимости долгосрочных затрат на эксплуатацию, частому появлению новых версий и возможному быстрому моральному старению средств, а также постоянным затратам на обучение и повышение квалификации персонала.

Несмотря на все высказанные предостережения и некоторый пессимизм, грамотный и разумный подход к использованию CASE-средств может преодолеть все перечисленные трудности. Успешное внедрение CASE-средств должно обеспечить такие выгоды, как:

- высокий уровень технологической поддержки процессов разработки и сопровождения ПО;

• положительное воздействие на некоторые или все из перечисленных факторов: производительность, качество продукции, соблюдение стандартов, документирование;

• приемлемый уровень отдачи от инвестиций в CASE-средства.

Оценка и выбор CASE-средств. Процессы оценки и выбора тесно взаимосвязаны. По результатам оценки цели выбора и/или критерии выбора и их веса могут потребовать модификации. В таких случаях может понадобиться повторная оценка. Когда анализируются окончательные результаты оценки и к ним применяются критерии выбора, может быть рекомендовано приобретение CASE-средства или набора CASE-средств. Альтернативой может быть отсутствие адекватных CASE-средств. В этом случае рекомендуется разработать новое CASE-средство, модифицировать существующее или отказаться от внедрения.

Процесс выбора включает в себя следующие действия:

– формулировку задач выбора, включая цели, предположения и ограничения;

– выполнение всех необходимых действий по выбору, включая определение и ранжирование критериев, определение средств-кандидатов, сбор необходимых данных и применение ранжированных критериев к результатам оценки для определения средств с наилучшими показателями. Для многих пользователей важным критерием выбора является интегрируемость CASE-средства с существующей средой;

– выполнение необходимого количества итераций с тем, чтобы выбрать (или отвергнуть) средства, имеющие сходные показатели;

– подготовку отчета по результатам выбора.

В процессе выбора возможно получение двух результатов:

– рекомендаций по выбору конкретного CASE-средства;

– запроса на получение дополнительной информации, необходимой для оценки.

Масштаб выбора должен устанавливать требуемый уровень детализации, необходимые ресурсы, график и ожидаемые результаты. Существует ряд параметров, которые могут быть применены для определения масштаба, включая:

– предварительный отбор (например, отбор только средств, работающих на конкретной платформе);

– использование ранее полученных результатов оценки, результатов оценки из внешних источников или комбинации того и другого.

В том случае, если предыдущие оценки выполнялись с использованием различных наборов критериев или конкретных критериев, но разными способами, результаты оценок должны быть представлены в согласованной форме. После завершения данного шага оценка каждого CASE-средства должна быть представлена в рамках единого набора критериев и непосредственно сопоставима с другими оценками.

Алгоритмы, обычно используемые для выбора, могут быть основаны на масштабе или ранге. Алгоритмы, основанные на масштабе, вычисляют единственное значение для каждого CASE-средства путем умножения веса каждого критерия на его значение (с учетом масштаба) и сложения всех произведений. CASE-средство с наивысшим результатом получает первый ранг. Алгоритмы, основанные на рангах, используют ранжирование CASE-средств-кандидатов по отдельным критериям или группам критериев в соответствии со значениями критериев в заданном масштабе. Затем аналогично предыдущему ранги сводятся вместе и вычисляются общие значения рангов.

При анализе результатов выбора предполагается, что процесс выбора завершен, CASE-средство выбрано и рекомендовано к использованию. Тем не менее может потребоваться более точный анализ для определения степени зависимости значений ключевых критериев от различий в значениях характеристик CASE-средств-кандидатов. Такой анализ позволит установить, насколько результат ранжирования CASE-средств зависит от оптимальности выбора весовых коэффициентов критериев. Он также может использоваться для определения существенных различий между CASE-средствами с очень близкими значениями критериев или рангами.

Если ни одно CASE-средство не удовлетворяет минимальным критериям, выбор (возможно, вместе с оценкой) может быть повторен для других CASE-средств-кандидатов.

Если различия между самыми предпочтительными кандидатами несущественны, дополнительная информация может быть получена путем повторного выбора (возможно, вместе с оценкой) с использованием дополнительных или других критериев.

Рекомендации по выбору должны быть строго обоснованы. В случае отсутствия адекватных CASE-средств, как было отмечено выше, рекомендуется разработать новое CASE-средство, модифицировать существующее или отказаться от внедрения.

2.8. Применение CASE-технологий в проектировании электронных средств

CASE-технология (Computer-Aided Software / System Engineering) представляет собой совокупность методологий анализа, проектирования, разработки и сопровождения сложных систем программного обеспечения (ПО), поддержанную комплексом взаимосвязанных средств автоматизации. CASE предоставляет системным аналитикам, проектировщикам и программистам инструментарий для автоматизации проектирования и разработки ПО.

CASE не только позволяет получать корректные программные продукты, но и обеспечивает технологически правильный процесс их создания. Главная цель CASE состоит в том, чтобы отделить проектирование ПО от его кодирования и последующих этапов разработки, а также скрыть от разработчиков все детали среды разработки ПО. Основной акцент в процессе создания ПО приходится на этапы анализа и проектирования, в отличие от кодирования.

CASE-технологии широко применяются для многих типов систем ПО, но чаще всего они используются в следующих областях:

1. Разработка делового и коммерческого ПО. Широкое применение CASE-технологий обусловлено массовостью этой прикладной области, в которой CASE применяется не только для разработки ПО, но и для создания моделей систем, помогающих коммерческим структурам решать задачи стратегического планирования, управления финансами, определения политики фирм, обучения персонала (это направление получило собственное название – бизнес-анализ).

2. Создание системного и управляющего ПО. Использование CASE-технологии в этой отрасли вызвано высокой сложностью данного вида работ и необходимостью повышения их производительности.

Помимо автоматизации структурных методологий и возможности применения современных методов системной и программной инженерии, CASE-средства имеют следующие преимущества:

• повышают качество создаваемого ПО благодаря использованию средств автоматического контроля, в частности контроля проекта;

• поддерживают создание прототипа будущей системы, что позволяет на ранних этапах оценить ожидаемый результат;

• ускоряют процесс проектирования и разработки;

• освобождают разработчика от рутинной работы, предоставляя ему возможность сосредоточиться на творческой части разработки;

• поддерживают развитие и сопровождение разработки;

• обеспечивают технологии повторного использования компонентов разработки.

Контрольные вопросы

1. Каков основной общий принцип системного подхода к проектированию электронных средств?

2. В чем заключается сущность структурного подхода к разработке ЭС?

3. Охарактеризуйте принципы методологий структурного подхода, выделите из них базовые.

4. В чем различие SADT-, DFD-, ERD-диаграмм?

5. В чем заключаются основные концепции методология SADT?

6. Что является результатом применения методологии SADT?

7. Перечислите и дайте краткую характеристику типов связей между функциями в SADT-моделях.

8. Поясните понятие «CASE-средство».

9. Назовите основные типы CASE-средств.

10. Из каких этапов состоит процесс внедрения CASE-средств?

11. В каких областях применяются CASE-технологии?

12. В чем заключаются преимущества CASE-средств?

Тестовые задания

1. Среди принципов методологии структурного подхода укажите базовые:

а) «разделяй и властвуй» и иерархического упорядочивания;

б) абстрагирования и формализации;

в) непротиворечивости и структурирования данных;

г) структурирования данных и иерархического упорядочивания.

2. Диаграммы потоков данных соответствуют методологии моделирования:

а) SADT; б) DFD; в) ERD; г) CAD.

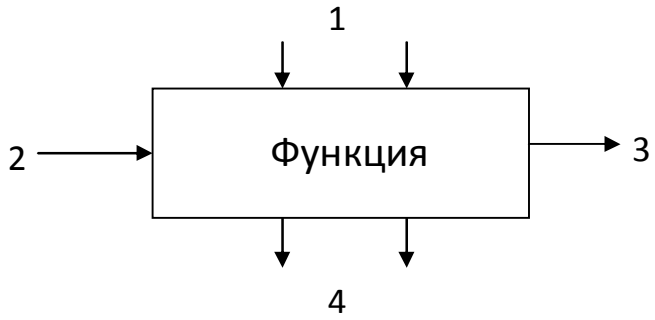
3. Расположите типы связей между функциями в SADT-моделях в порядке возрастания их относительной значимости:

а) процедурная;

б) логическая;
в) последовательная;
г) случайная;
д) функциональная.
е) временная;
ж) коммуникационная.

1	2	3	4	5	6	7

4. Для SADT-модели установите соответствия



а) механизм; б) выходы; в) входы; г) управление.

1	2	3	4

5. Серия диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые имеют вид блоков, представляет собой:

а) математическую модель;
б) модель DFD;
в) модель ERD;
г) модель SADT.

3. МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ИНТЕЛЛЕКТУАЛЬНОГО ПРОЕКТИРОВАНИЯ

3.1 Имитационные методы моделирования. Проблемы применения имитационного моделирования

Имитационное моделирование (ситуационное моделирование) – метод, позволяющий строить модели, описывающие процессы так, как они проходили бы в действительности. Такую модель можно «проиграть» во времени как для одного испытания, так и заданного их множества. При этом результаты будут определяться случайным характером процессов. По этим данным можно получить достаточно устойчивую статистику.

Имитационное моделирование – это метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему, с которой проводятся эксперименты с целью получения информации об этой системе. Экспериментирование с моделью называют *имитацией* (имитация – это постижение сути явления не прибегая к экспериментам на реальном объекте).

Имитационное моделирование является частным случаем математического моделирования. Существует класс объектов, для которых по различным причинам не разработаны аналитические модели либо не разработаны методы решения полученной модели. В этом случае аналитическая модель заменяется имитатором или имитационной моделью.

Имитационным моделированием иногда называют получение частных численных решений сформулированной задачи на основе аналитических решений или с помощью численных методов.

Имитационная модель – логико-математическое описание объекта, которое может быть использовано для экспериментирования на компьютере в целях проектирования, анализа и оценки функционирования объекта.

К имитационному моделированию прибегают, когда:

– дорого или невозможно экспериментировать на реальном объекте;

– невозможно построить аналитическую модель: в системе есть время, причинные связи, последствие, нелинейности, стохастические (случайные) переменные;

– необходимо сымитировать поведение системы во времени.

Цель имитационного моделирования состоит в воспроизведении поведения исследуемой системы на основе результатов анализа наиболее существенных взаимосвязей между её элементами или, другими словами, разработке симулятора (англ. *simulation modeling*) исследуемой предметной области для проведения различных экспериментов.

3.2. Имитационное моделирование в терминах SADT-технологий: основные понятия и аналитические методы моделирования

SADT (акроним от англ. Structured Analysis and Design Technique) – методология структурного анализа и проектирования, интегрирующая процесс моделирования, управление конфигурацией проекта, использование дополнительных языковых средств и руководство проектом со своим графическим языком.

Процесс моделирования может быть разделен на несколько этапов:

– опрос экспертов;

– создание диаграмм и моделей;

– распространение документации;

– оценка адекватности моделей и принятие их для дальнейшего использования.

Этот процесс хорошо отлажен, потому что при разработке проекта специалисты выполняют конкретные обязанности, а библиотекарь обеспечивает своевременный обмен информацией.

SADT возникла в конце 60-х годов XX века в ходе революции, вызванной структурным программированием. Когда большинство специалистов билось над созданием программного обеспечения, немногие старались разрешить более сложную задачу создания крупномасштабных систем, включающих как людей и машины, так и программное обеспечение, аналогичных системам, применяемым в телефонной связи, промышленности, управлении и контроле за вооружением. В то время специалисты, традиционно занимавшиеся созданием крупномасштабных систем, стали осознавать необходимость большей упорядоченности. Таким образом, разработчики решили формализовать процесс создания системы, разбив его на следующие фазы:

- анализ – определение того, что система будет делать;

- проектирование – определение подсистем и их взаимодействие;
- реализация – разработка подсистем по отдельности, объединение – соединение подсистем в единое целое;
- тестирование – проверка работы системы;
- установка – введение системы в действие;
- эксплуатация – использование системы.

Особенностью *непрерывно-детерминированного подхода* является применение в качестве математических моделей дифференциальных уравнений. Дифференциальными уравнениями называются такие уравнения, в которых неизвестными являются функции одной или нескольких переменных, причем в уравнение входят не только функции, но и их производные различных порядков. Если неизвестные – функции многих переменных, то уравнения называются уравнениями в частных производных, в противном случае при рассмотрении функции только одной независимой переменной уравнения называются обыкновенными дифференциальными уравнениями.

Обычно в таких математических моделях в качестве независимой переменной, от которой зависят неизвестные искомые функции, служит время *t*. Тогда математическое соотношение для детерминированных систем в общем виде будет

$$\vec{y}' = \vec{f}(\vec{y}, t);\ \vec{y}(t_0) = \vec{y}_0\,, \qquad (3.1)$$

где $\vec{y}' = {d\vec{y}}/{dt}$, $\vec{y} = (y_1, y_2, \dots, y_n)$ и $\vec{f} = (f_1, f_2, \dots, f_n)$ – *n*-мерные векторы;

$\boldsymbol{f}(\vec{y}, \vec{t})$ – вектор-функция, которая определена на некотором $(n+1)$-мерном $(\vec{y}, t)$ множестве и является непрерывной.

Математические схемы такого вида отражают динамику изучаемой системы, т.е. ее поведение во времени, и называются *D-схемами* (англ. *dynamic*).

При *дискретно-стохастическом подходе* определяют дискретные преобразования информации с памятью с помощью *P-схем,* называемых вероятностными автоматами.

Вероятностный автомат (ВА) [англ. *probabilistic automat*) – это дискретный потактный преобразователь информации с памятью,

функционирование которого в каждом такте зависит только от состояния памяти нем и может быть описано статистически.

Схемы вероятностных автоматов (Р-схем) применяются:

– в проектировании дискретных систем, проявляющих статистически закономерное случайное поведение;

– определении алгоритмических возможностей систем;

– обосновании границ целесообразности их использования;

– решении задач синтеза по выбранному критерию дискретных стохастических систем, удовлетворяющих заданным ограничениям.

Математическое понятие Р-автомата формируется на понятиях, введенных для F'-автомата.

В рамках *дискретно-детерминированного подхода* переменные входного воздействия $x(t)$, влияния внешней среды $v(t)$ и собственные параметры $h(t)$ объекта считаются детерминированными и меняющимися в дискретном времени t. Этот подход положен в основу математического аппарата теории автоматов. На основе этой теории объект представляется в виде автомата, перерабатывающего дискретную информацию и меняющего свои внутренние состояния лишь в допустимые моменты времени. *Конечным* называется автомат, у которого множество внутренних состояний, входных и выходных сигналов являются конечными множествами. Формально конечный автомат, можно представить как математическую схему (*F-схему*), которая характеризуется шестью элементами: конечным множеством входных сигналов (входным алфавитом) X; конечным множеством выходных сигналов (выходным алфавитом) Y; конечным множеством внутренних состояний (внутренним алфавитом или алфавитов состояний) Z; начальным состоянием $z_0 \in Z$; функцией переходов $\varphi(z, x)$, функцией выходов $\psi(z, x)$.

Автомат, задаваемый F-схемой: $F = \langle Z, X, Y, \varphi, \psi, z_0 \rangle$, функционирует в дискретном автоматном времени, моментами которого являются такты, по следующей схеме: в каждом t такте на вход автомата, находящегося в состоянии $z(t)$, подается некоторый сигнал $x(t)$, на который он реагирует переходом $y(t+1)$ в такте в новое состояние $z(t+1)$ с выдачей некоторого выходного сигнала.

При *непрерывно-стохастическом подходе* в качестве типовых математических схем применяются системы массового обслуживания (англ. queueing system), которые будем называть Q-схемами. Системы массового обслуживания представляют собой класс математических

схем, разработанных в теории массового обслуживания и различных приложениях для формализации процессов функционирования систем, которые по своей сути являются процессами обслуживания.

В качестве процесса обслуживания могут быть представлены различные по своей физической природе процессы функционирования экономических, производственных, технических и других систем, например: потоки поставок продукции некоторому предприятию, потоки деталей и комплектующих изделий на сборочном конвейере цеха, заявки на обработку информации компьютером от удаленных терминалов и т. д.

При этом характерным для работы таких объектов является случайное появление заявок (требований) на обслуживание и завершение обслуживания в случайные моменты времени, т. е. стохастический характер процесса их функционирования. Остановимся на основных понятиях массового обслуживания, необходимых для использования Q-схем как при аналитическом, так и при имитационном.

В любом элементарном акте обслуживания можно выделить две основные составляющие:

- ожидание обслуживания заявки;
- собственно обслуживание заявки.

В дискретном имитационном моделировании сформировалось несколько основных методологических подходов, в рамках которых описывается структура и поведение реальной системы: событийно-ориентированный подход, процессно-ориентированный, объектно-ориентированный и логический подход.

Процессно-ориентированный подход заключается в выделении в системе нескольких компонентов и описании функционирования системы с помощью нескольких последовательностей событий. Каждая последовательность событий (процесс) представляет изменения состояния системы с точки зрения одного компонента (изменения, в которых этот компонент принимает участие).

Общая последовательность событий, соответствующих *событийному подходу*, получается путем комбинации последовательностей событий, соответствующих отдельным процессам. В результате модель представляется как совокупность взаимодействующих квазипараллельных процессов, что более адекватным образом (чем совокупность событий) отражает структуру и поведение реальной системы. Квазипараллельность заключается в том, что программа, соответствующая процессу, составляется из последовательности программ со-

бытий независимо от других процессов (если отсутствуют явные указания взаимодействия), а исполняется с прерываниями, во время которых исполняются другие процессы.

Это вызвано тем, что события в модельном времени происходят «мгновенно» и могут выполняться при событийно-ориентированном подходе последовательно в соответствии с упорядоченностью моментов времени наступления события, а процессы обладают «протяженностью» в модельном времени и не могут исполняться последовательно, так как момент времени наступления события одного процесса может оказаться между моментами времени последовательных событий другого процесса. Таким образом, в каждый момент исполнения модели выполняется только один процесс, называемый активным, остальные процессы находятся в приостановленном состоянии.

Структура программы, соответствующая событию, проста – это подпрограмма, к которой обращается управляющая программа модели в момент модельного времени, на который запланировано событие. По окончании исполнения подпрограммы, соответствующей событию, управление возвращается управляющей программе модели.

Программа, соответствующая процессу, имеет более сложную структуру. С каждым приостановленным процессом связывается точка реактивации – место в программе процесса, с которого возобновится выполнение процесса, когда процесс снова станет активным.

Одно из неудобств событийно-ориентированного подхода заключается в «дроблении» логики модели на отдельные события. При процессно-ориентированном подходе, наоборот, логика каждого процесса сосредотачивается в одной программе, что облегчает описание динамики системы.

3.3. Методы интеллектуализации САПР

Существуют следующие основные поколения САПР:

• моделирующие (системы автоматизированного моделирования);

• синтезирующие (синтез выполняется не через многократное моделирование или оптимизацию, а путем создания сразу работоспособного варианта – генерационный синтез);

• САПР с формальными языками, где средства общения с пользователем ограничены определенными конструкциями, не подлежащими изменениям;

• САПР с неформальными языками, где имеется переменная (перестраиваемая) лексика со свободной грамматикой и синтаксисом.

Существующие САПР представлены поколениями, где описание объекта выполнено формальными языками. Создание синтезирующих САПР с неформальными языками зависит от использования методов ИИ. Поэтому все САПР можно разделить на интеллектуальные и неинтеллектуальные.

Обычно САПР работала в жестком режиме, по строго заданным алгоритмам, не используя опыт проектировщика, не обеспечивая взаимодействия пользователя с компьютером на языке, близком к естественному. Существует три метода интеллектуализации САПР:

1. *Внешняя универсальная* интеллектуализация с помощью инструментальных систем (ИСИИ), где используются различные оболочки экспертных, диалоговых, обучающих систем. Главное достоинство – высокая скорость разработки и малые финансовые затраты. Недостаток – низкое качество проектирования (скорость работы, требуемая память, надежность, мощность, учет всех особенностей предметной области). ИСИИ удобно пользоваться на начальных этапах разработки компьютеров.

2. *Внешняя специализированная* интеллектуализация с помощью специализированных программных приставок, работающих на принципах ИИ. Этот метод как развитие предыдущего, но с более высоким качеством работы. Здесь все сводится к улучшению сервисных характеристик САПР, что обеспечивает возможность формулировки типовой задачи проектирования на предметно-ориентированном языке, организации обучения пользователя и т.д. Совокупность средств общения пользователя с САПР представляет собой интеллектуальный интерфейс.

3. *Внутренняя интеллектуализация* со встроенными в САПР алгоритмами и методами ИИ (увеличены возможности синтеза, адаптации, самоорганизации, работы с нечеткой формулировкой задачи).

Как известно, в ИИ можно выделить бессловесный ИИ (низкий уровень, где нет интеллектуального интерфейса, пользователь работает в терминах предметной области, но присутствуют процедуры, имитирующие целесообразное поведение, например: адаптацию, самоорганизацию), словесный ИИ (высокий уровень, где есть интеллектуальный интерфейс с языком предметной области, используются понятия для представления знаний на любых символах и знаках с явно заданной семантикой) и искусственный разум (технически не реализован) [1, 6].

Системы, в которых существенную роль играют понятия, миры слов, т. е. семиотические (знаковые) с их явно заданной семантикой, называют *понятийными.*

Традиционные САПР, дополненные адаптационными процедурами приспособления к оперативной проектной обстановке, самонастройке на особенности постановки задачи, поиска и обоснованного выбора цели среди множества вариантов, т.е. процедурами, имитирующими целенаправленное поведение, следует считать ИСАПР нижнего уровня. Понятийные САПР, способные работать с различными понятиями, смысл которых определяется либо пользователем на входном языке, либо автоматически внутри программы, имеют уровень интеллекта выше.

Традиционные САПР работают по строгому алгоритму, понятийные САПР порождают иной алгоритм решения. Это позволяет пользователю применять эвристические, словесные, нематематические правила. Понятийные САПР (развитие традиционных САПР, алгоритмических) используют информацию не в количественном, а в качественном виде, например: «если *X* имеет свойство *а,* то Л имеет свойство *в*». Эти высказывания (предикаты) можно рассматривать как качественные аналоги количественных формул.

Наглядным примером понятийной формулы является *фрейм.* Фрейм-прототип можно рассматривать как понятийный аналог функции $Y = F(X_1, X_2, ..., X_n)$, где F – имя фрейма-прототипа (сложного понятия), а $X_1, X_2, ..., X_n$ – имена его незаполненных слотов (простых понятий). Заполняя слоты различными значениями имен (понятийных аргументов), получаем разные значения имени фрейма, т. е. понятийной функции.

3.4. Архитектура интеллектуальных САПР

Традиционная САПР характеризуется:

• описанием объекта и задач проектирования с помощью формализованных, фиксированных, символьно-цифровых языковых конструкций;

• ориентацией работы САПР на жесткую, формализованную постановку задачи проектирования;

• использованием процедуры моделирования как основы процесса проектирования;

• использованием алгоритмов, неадаптируемых и не учитывающих опыта пользователя;

• фиксированием функциональной структуры САПР;

• применением в качестве основы математического аппарата теории численных методов, теории множеств, имитационного моделирования;

• представлением информации в численном виде;

• использованием традиционных языков: Паскаль, Си и др.;

• хранением больших объемов информации в виде БД;

• представлением выходной информации в виде таблиц, графиков, чертежей.

Архитектура традиционных САПР отражает черты административно-командной системы, перенесенные в технику: чрезмерная централизация управления, штампы в языковых средствах, негибкость алгоритмов и принципов работы, представление информации в неосмысленном виде, сложность модификации, невозможность учета человеческого фактора, т. е. опыта и его психологии.

Основные концепции интеллектуальных САПР состоят в следующем:

1. Входная информация представляется в виде фраз на ограниченном естественном языке или на предметно-ориентированном языке, допускающем описание пользователем не только объекта, но самого алгоритма проектирования. Тем самым достигается высокая функциональная гибкость САПР. Постановка задачи может быть нечеткой или некорректной, тогда ИСАПР путем диалога с пользователем устраняет нечеткость и некорректность. Это заключается в распознавании задачи и отнесении ее к типовой при определении необходимых уточнений. В этом случае ИСАПР имеет архив типовых постановок задач [5] с соответствующими наборами корректной входной информации.

2. Информация представляется не в виде данных (чисел), а в виде знаний, т. е. характеризуется независимостью от пользователя, активностью и ситуативными связями. Это влечет за собой:

• представление и обработку информации не только в числовом, но и в символьном виде;

• переход к языкам программирования, удобным для работы с символьной информацией (Си, Лисп, Пролог);

• разработку и применение в ИСАПР способов представления информации в виде знаний (правил, фреймов, семантических сетей);

• хранение информации в виде баз знаний, частью которых могут быть БД.

3. Активность знаний выражается в том, что ИСАПР функционируют под управлением алгоритмов (процедур) и данных, т.е. диспетчирование работ подсистем в ИСАПР выполняется не с помощью внешней управляющей программы и автоматически по факту наличия или отсутствия необходимых для работы этих подсистем данных или знаний, а с помощью процедуры контроля необходимой информации. Как только эта информация появляется, то автоматически срабатывают процедуры ее обработки. Таким образом, информация побуждает к действию.

4. Представление информации в виде знаний, а не чисел позволяет вести обработку данных не численными, а логическими методами. Основным математическим аппаратом ИСАПР является аппарат алгебры логики в условиях детерминированной, нечеткой или вероятностной информации. Тогда применяют язык Пролог.

5. ИСАПР ориентированы не на процедуры моделирования и анализа, а на процедуры синтеза (переход от общего к частному: декомпозиция, поиск знаний по образцу, составление документации), т.е. дедуктивного или индуктивного вывода (композиция, агрегирование, обобщение (целое из частей) формализация эвристических знаний, построение понятий).

6. Входная и выходная информация представляется с использованием принципов когнитивной психологии [приемы визуального отображения информации (иерархические меню. многооконный графический интерфейс, различные способы образного отображения действий пользователя – мультипликация)]. Это примеры графического интеллектуального интерфейса.

7. ИСАПР имеют ЭС для консультации пользователя, методики проектирования. Это и диагностика ошибок, сбоев работы ИСАПР, выдача советов, обучение.

8. ИСАПР представляют объект проектирования многоаспектно, даны связи с другими объектами, правила модификации свойств объекта, правила взаимодействия с другими объектами с целью его эквивалентного представления. Такое описание значительно облегчает многоуровневое проектирование объекта.

Количественные и качественные характеристики интеллектуальных САПР

Количественные характеристики имеют следующие величины:

• число входных, внутренних и выходных языков;

• число слов (понятий) в каждом из языков;

• глубина «дерева» диалога ИСАПР – пользователь;

• среднее время реакции ИСАПР на вопрос пользователя;

• число уровней иерархии представления объекта;

• число ЭС (или число проблем, по которым пользователь может получить консультацию);

• объем базы знаний (число правил, фреймов).

Качественные характеристики содержат следующие показатели:

– функциональные возможности (моделирование, расчет, оптимизация, параметрический и структурный синтез);

– тип структуры ИСАПР: с внешней инструментальной интеллектуализацией (рис. 3.1, *а*); с внешней специализированной интеллектуализацией (рис. 3.1, *б*); с внутренней локализованной интеллектуализацией (рис. 3.1, *в*); с внутренней распределенной интеллектуализацией (рис. 3.1, *г*);

– тип интеллектуального интерфейса: языковой (ограниченный естественный язык, предметно-ориентированный); графический (иерархическое или простое меню, графический ввод, графический вывод, многооконный интерфейс);

– способ представления знаний (правила продукций, фреймы, семантические сети, предикаты, парадигмы);

– тип логического вывода (прямой или обратный логический вывод, с поиском в глубину или ширину, комбинированный);

– аппарат логической обработки знаний (метод резолюции, логический вывод);

– тип обрабатываемых знаний (четкие, нечеткие, вероятностные).

Наряду со специфическими характеристиками ИСАПР имеются и традиционные: тип языков программирования, объем памяти, тип ОС, тип технических средств.

Рассмотрим более подробно типовые структуры ИСАПР. При *внешней интеллектуализации* к САПР присоединяется готовая инструментальная интеллектуальная система, наполняемая знаниями. Достоинствами такой системы являются высокая скорость и простота разработки. Недостаток состоит в низкой эффективности из-за избыточности памяти и нерациональной организации работы системы (рис. 3.1, *а*). Внешняя специализированная интеллектуализация заключается в том, что ЭС специально создается под данную САПР. Такую систему называют *гибридной.*

Развитием предыдущей системы будет *внутренняя локализованная интеллектуализация,* где имеются хорошо отработанные блоки

лингвистического транслятора, логического вывода, представления знаний. Характеристики традиционной САПР значительно улучшаются при использовании по необходимости отдельных блоков, а не всего ИИ (рис. 3.1, *в*).

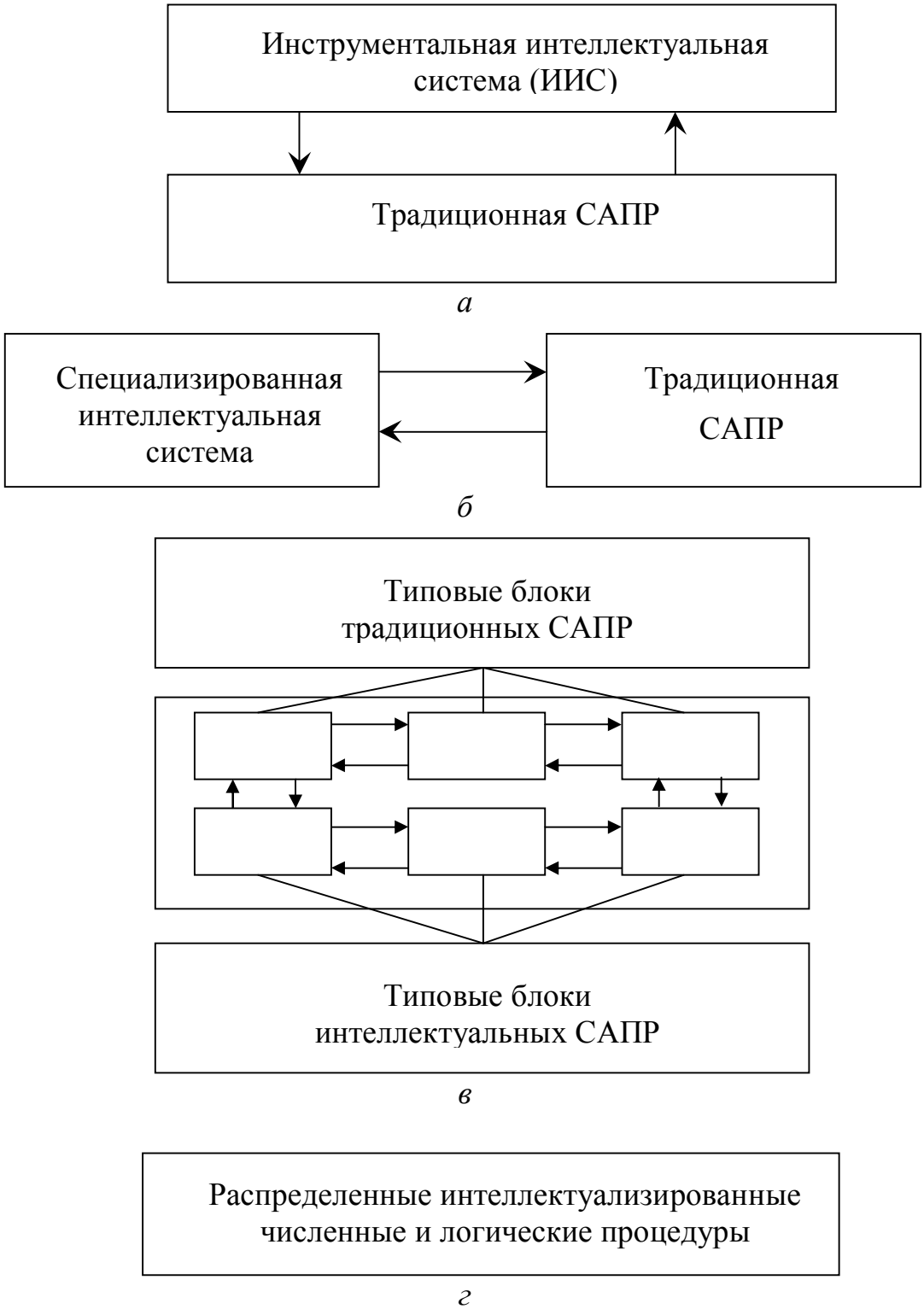


Рис. 3.1. Типы структур САПР:
а – внешняя интеллектуализация; *б* – внешняя специализированная интеллектуализация; *в* – внутренняя локализованная интеллектуализация; *г* – внутренняя распределенная интеллектуализация

При *внутренней распределенной интеллектуализации* каждый алгоритм САПР и принцип ее работы в целом основаны на использовании не данных, а знаний (рис. 3.1, *г)*. На основе знаний строится ИСАПР – моделирующая (интеллектуальные аналоги традиционных САПР) и синтезирующая.

Моделирующая интеллектуальная САПР выполняет описание объекта и основной задачи моделирования (задачи расчета, анализа, оптимизации) на предметно-ориентированном языке (рис. 3.2).

Рис. 3.2. Структура адаптивной моделирующей ИСАПР

Структура и начальные параметры считаются известными, задача состоит в выявлении конечных параметров и характеристик заданного объекта. Интеллект моделирующей ИСАПР заключается в адаптации работы ИСАПР к особенностям объекта: при составлении математической модели объекта база знаний, используя покомпонентное описание объекта, выдает те модели его компонента, которые наиболее соответствуют требованиям точности и скорости моделирования с учетом сложности всего объекта, а база знаний инициирует в соответствии с этими требованиями способ составления математической

модели объекта. Моделирование состоит в том, что база знаний на основе сведений, поступающих от «датчиков», анализирующих параметры и качество моделирующей процедуры (анализ, оптимизация, размещение, трассировка), управляет ходом моделирования – изменяет параметры моделирующей процедуры или меняет алгоритмы.

Основным отличием баз знаний в этой ИСАПР от БД в традиционной САПР является активный характер баз знаний, так как, кроме декларативных знаний (параметры компонентов, алгоритмов, оптимальные условия их применения), в этих базах хранятся и знания процедур – алгоритмы и методы, инициируемые знаниями, содержащимися в описании объекта и задачи, либо знаниями, извлекаемыми с помощью интеллектуальных датчиков из самого процесса моделирования.

В состав моделирующей адаптивной ИСАПР дополнительно включаются подсистемы диагностики, указывающие пользователю как на его ошибки (на семантическом уровне) при составлении описания объекта, так и на ошибки системы (на естественном языке).

Для проектирования вычислительных систем решается задача фрагментации объектов, т. е. моделирование по частям и агрегирование результатов моделирования.

Синтезирующая интеллектуальная САПР (рис. 3.3) решает задачи структурного и параметрического синтеза. Предполагается, что эти задачи можно решать отдельно и последовательно.

В блоке логического вывода может частично выполняться и параметризация синтезируемой структуры, если она имеет важное значение для функционирования структуры. Синтез в такой ИСАПР состоит в построении дерева декомпозиции задачи на подзадачи (в качестве задачи может выступать объект или процесс) в блоке *2* и поиске варианта декомпозиции в блоке логического вывода *3,* удовлетворяющего заданным требованиям (блок *4)*. В качестве дерева декомпозиции можно использовать И–ИЛИ дерево или дерево состояний и свойств. Вместо блоков *2* и *3* можно использовать любой эвристический прием синтеза.

Если на выбранном базисе терминальных элементов дерева декомпозиции структура не синтезируется, нужно сменить базис, а если синтезируется, то дальше выполняется ее параметризация. Параметризация может выполняться различными способами – либо эвристически, на основе знаний экспертов, либо формально, с помощью оптимизирующей системы, аналогичной адаптивной моделирующей

ИСАПР, но с использованием в качестве основной процедуры программы оптимизации. Описанная процедура синтеза может быть применена на каждом иерархическом уровне.

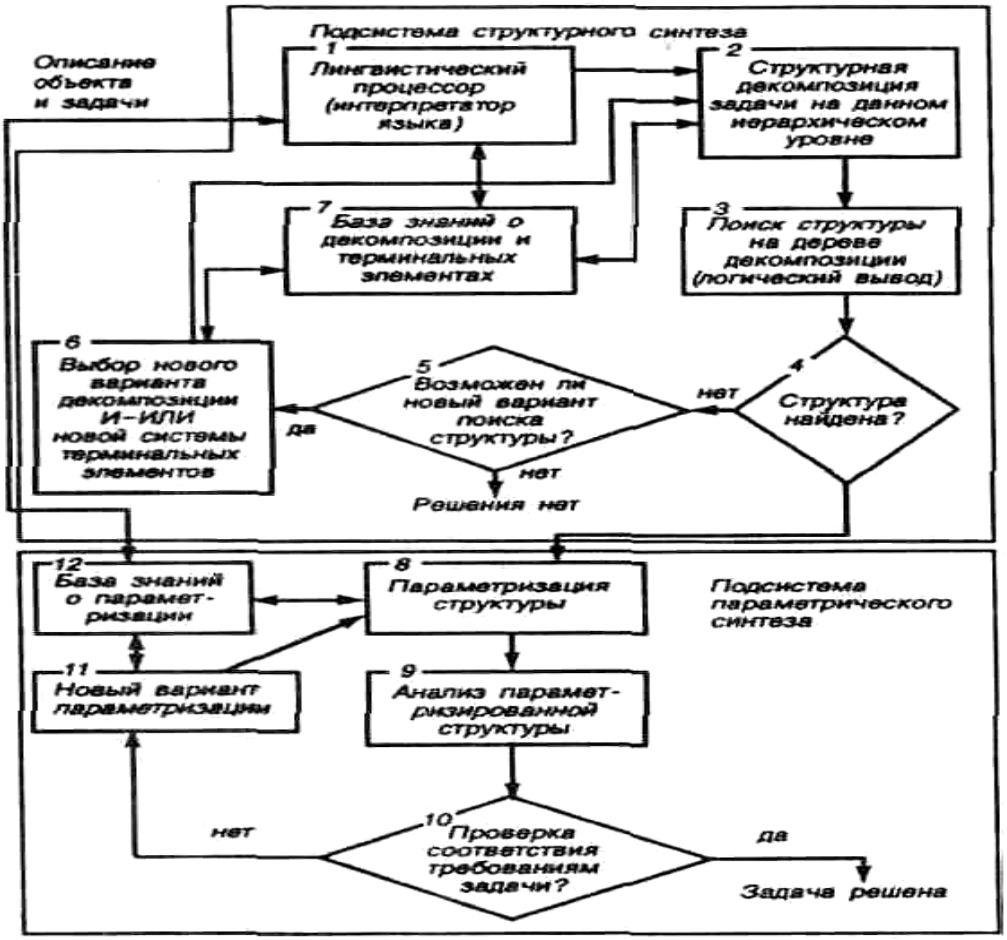


Рис. 3.3. Структура синтезирующей ИСАПР

3.5. Методы структурного и параметрического синтеза

Общая характеристика методов синтеза. В задачах синтеза наиболее эффективно используются методы ИИ. Особенностью этих задач является необходимость многократного принятия решений в процессе их реализации. Поскольку эти решения неоднозначны, результаты синтеза тоже неоднозначны. Существуют различные типы синтеза (рис. 3.4).

Оптимизационный синтез состоит в том, что сначала разработчик сам создает начальный вариант проектируемого объекта, т. е. задает его структуру и параметры. Если после расчета и анализа математической модели этого начального варианта окажется, что характе-

ристики объекта отличаются от требуемых, то изменяются сначала параметры, а если это не поможет, то и структура объекта в соответствии с каким-либо методом оптимизации.

Каждый шаг оптимизации требует повторного моделирования, расчета и анализа объекта, так как для оптимизации по нескольким переменным требуется обычно не менее ста шагов, а приемлемое время оптимизации составляет не более нескольких минут. Это требование ограничивает применение оптимизационного синтеза параметров и особенно структуры объектов. Обычно оптимизационный синтез используется при расчете параметров (классические непрерывные оптимизации). Задачи структурного синтеза решаются методами дискретной оптимизации (направленный перебор).

Рис. 3.4. Основные типы синтеза

Генерационный синтез, в отличие от оптимизационного, состоит не в последовательном улучшении первоначального варианта объекта, а в создании (генерации) сразу модели работоспособного объекта. Есть две разновидности генерационного синтеза. Первая – синтез по теоретически строго обоснованным соотношениям, опреде-

ляющим структуру или чаще параметры объекта проектирования. Эти соотношения отражают необходимые условия работоспособности объекта. Примером строгого теоретического синтеза является структурный синтез цифровых устройств.

Строгий синтез возможен при строгой формализации постановки задачи – получении заданной логической функции на выходе цифрового устройства. Но такие случаи редки, и формулировки задач синтеза трудны из-за разнообразия и сложности математизации, поэтому трудно иметь строгий общий теоретический аппарат и приходится прибегать к прошлому опыту, выраженному в виде методик, отношений, зависимостей, просто к здравому смыслу и творчеству.

Вторая разновидность генерационного синтеза – эвристический синтез. Большинство задач (технологических) вычислительных процессов решается именно этим способом.

В целом, если задача синтеза формулируется как задача поиска параметризированной структуры, то ее решение можно искать на путях теоретического синтеза (хотя результаты могут быть неточными). Например, преобразование множества X во множество Y или сигнала X в сигнал Y. Если таким образом задачу сформулировать нельзя, то прибегают к эвристическому синтезу.

Различие методов теоретического и эвристического синтеза:

• теоретический синтез реализуется с помощью алгоритмов, обладающих свойством определенности, т. е. его результат однозначен и не зависит от точности исполнителя;

• эвристический синтез допускает множество альтернативных решений, так как в нем важную роль играет выбор исполнителем того или иного способа синтеза, включая выбор типа элементов, из которых должен состоять объект, способа их соединения, способа расчета их параметров.

Методы структурного синтеза. Синтез ВС и вычислительных процессов (ВП) включает три этапа:

• выбор типовых элементов структуры (синтез базиса);

• построение структуры (структурный синтез);

• параметризация элементов структуры (параметрический синтез).

Выбор типовых элементов структуры выполняется эмпирически. Для формализации используют матрицу «элементы – свойства» (или, что то же самое, матрицу «средства – цели»). Строкам этой матрицы соответствуют различные альтернативные элементы, выполняющие одни и те же функции, а столбцам – свойства этих элементов, важные

для проектируемой ВС (ВП). В каждой клетке a_{ij} матрицы по балльной системе проставляется экспертная оценка i-го свойства у j-го элемента. Анализ этой матрицы проектировщиком позволит составить наглядное и достаточно полное представление о приемлемости того или иного элемента.

Структурный синтез редко бывает полностью независим от параметрического, обычно выбор структуры и определение значений параметров ее элементов тесно связано, составляя вместе структурно-параметрический синтез. Тем не менее структурный синтез можно выполнять независимо от параметрического в тех случаях, когда формирование определенной структуры объекта является необходимым условием его работоспособности, а параметры лишь обеспечивают более высокое качество функционирования. К таким случаям относятся, например, структурный синтез цифровых устройств, вычислительных процессов. Рассмотрим типовые приемы структурного синтеза:

1. *Выбор из готовых структур-прототипов.*

Этот прием предполагает наличие библиотеки готовых структур. Основной недостаток – необходимость прямого перебора всех структур. Если подобрать полностью подходящую структуру не удается, выбирают наиболее близкую и модифицируют ее под заданные требования путем удаления или добавления новых элементов, введения дополнительных или исключения ненужных связей и т. д.

2. *Построение частной структуры из общей.*

В этом случае сначала создают структуру с максимальной избыточностью, являющуюся обобщением всех известных структур объекта данного типа. Нужная структура синтезируется путем удаления лишних элементов и связей обобщенной структуры. Этот прием традиционно используется при синтезе вычислительных процессов, когда обобщенный процесс представляет собой цепочку всех вычислительных операций, применяемых для изготовления объекта данного класса, а синтез состоит в выборе из этой цепочки только тех операций, которые нужны в каждом конкретном случае. Так как выбор выполняется самим разработчиком на основе своего опыта, этот прием, как и предыдущий, относится к методам эмпирического синтеза.

3. *Направленный поиск по И–ИЛИ дереву.*

Этот прием можно рассматривать как специальный случай построения частной структуры из общей. В качестве общей структуры выступает заранее составленное И–ИЛИ «дерево», в котором каждая

группа путей от корневой вершины через вершины И, ИЛИ до терминальных вершин соответствует одной частной структуре (рис. 3.5).

Синтез по И–ИЛИ «дереву» удобно применить в тех случаях, когда объект легко декомпозируется на составляющие его части, которые декомпозируются на еще более мелкие и т. д., каждая декомпозиция порождает вершину типа И, а каждый уровень декомпозиции – ярус «дерева» из вершин И. Альтернативные варианты реализации каждой части объекта порождают вершину типа ИЛИ. Ветви «дерева», выходящие из вершин ИЛИ, – имена способов реализации этих частей или их свойств, играющих определенную роль при синтезе.

В качестве примера на рис. 3.5 приведено И–ИЛИ дерево для некоторого абстрактного объекта, который декомпозируется на части *a, b,* затем каждая из них соответственно на части *c, d* и *e, f, g, h.*

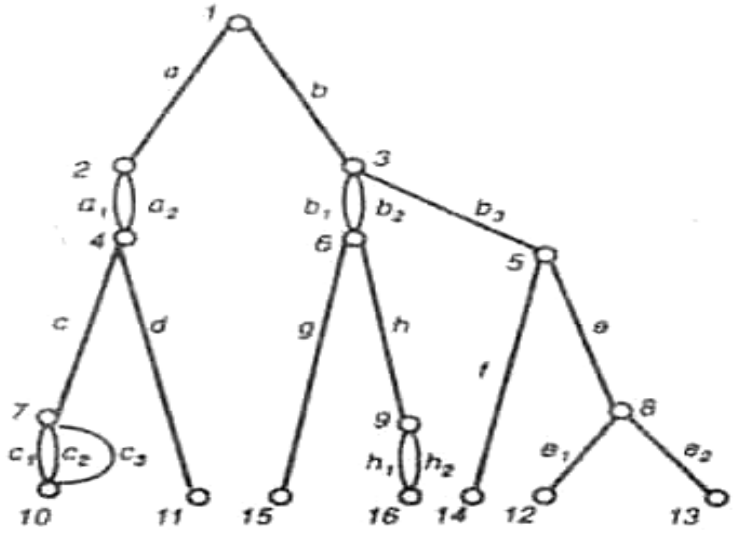


Рис. 3.5. Структура И-ИЛИ дерева

Если дерево состоит только из вершин И, то оно описывает одну структуру объекта. Альтернативный выбор структуры определяется вершинами ИЛИ. Обход дерева из вершин ИЛИ возможен либо в глубину, либо в ширину. При этом с каждой альтернативной реализацией (a_1, или a_2, c_1, c_2 или c_3 и т. д.) связывается оператор перехода, разрешающий переход в следующую вершину И только при выполнении определенных ограничений в вершине ИЛИ. Ограничения могут быть глобальными, относящимися ко всему объекту (общая масса или общая стоимость ВС, общее время вычислительного процесса и т. д.), и локальными, относящимися к данной реализуемой части объекта (условия стыковки с другими ЭВМ, ВС; ограничения на параметры части ВС).

Условием успешного окончания синтеза является прохождение по всем вершинам И до терминальных вершин. Практически строить И–ИЛИ дерево не обязательно, достаточно иметь дерево декомпозиции И, в каждой вершине которого нужно программно проверять альтернативные возможности реализации, задаваемые списком этих реализаций и списком ограничений. Проверка этих ограничений означает, что синтез носит структурно-параметрический характер, поскольку в процессе синтеза отбираются элементы с определенными значениями параметров.

4. Направленный поиск по «дереву» состояний и свойств.

Структурно-параметрический синтез можно выполнить не только на основе И–ИЛИ дерева, но и с помощью дерева состояний и свойств (рис. 3.6).

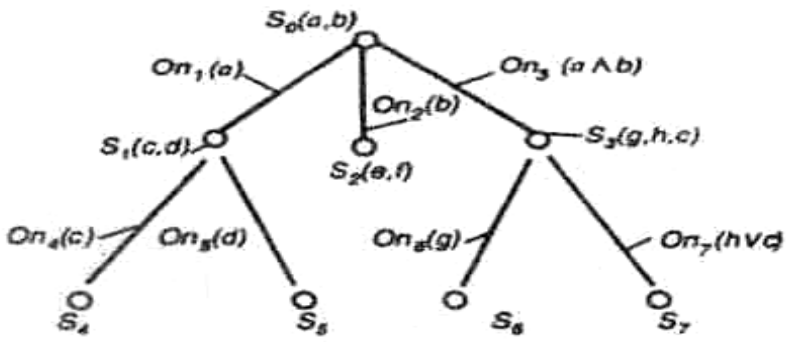


Рис. 3.6. Направленный поиск по дереву состояний и свойств

Каждая вершина $S_i(a_1 \ldots a_n)$ в этом дереве трактуется как задача, возможное решение которой обладает свойствами $a_1 \ldots a_n$, а каждый оператор перехода $On_i\{a)$ имеет смысл: *«Если* решение задачи S_{i-1} должно обладать свойством *a,* то перейти к решению задачи *S*». Таким образом, нужно решить задачу S_3. Пусть известно, что решение должно одновременно удовлетворять свойствам *a, b, h, c.* Нужно найти терминальную вершину, имеющую указанные свойства.

Приведенный пример позволяет иллюстрировать некоторые характерные особенности обработки знаний в ИИ при использовании дерева состояний:

1) *Наследование свойств.*

Из рис. 3.6 видно, что состояние $S_1(c, d)$ на самом деле, кроме свойств *c*, *d* должно также обладать и свойством *d* предыдущей вершины S_0, состояние $S_2(e, f)$ – свойством *b.* Обеспечиваемая последова-

тельным прохождением вершин при поиске по дереву передача свойств отцовской вершины к дочерним называется в ИИ *наследованием свойств.* Механизм автоматического наследования свойств позволяет упростить запись и реализацию операторов перехода, не учитывая в них свойств предыдущих состояний. Он по умолчанию обеспечивает присутствие в последующих состояниях свойств предыдущих.

Однако наследование свойств в дереве поиска необязательно, и его можно отменить, если условиться, что «отцовское» и «дочернее» состояние по своим свойствам друг с другом не связаны, например, относятся к понятиям с разной семантикой (S_0 – рабочий, S_1 – станок). Если же состояния в дереве относятся к одному классу, то механизм наследования свойств позволяет «наращивать» свойства состояний в процессе последовательного раскрытия вершин «дерева». Это особенно важно при решении задач классификации и диагностики.

2) *Наследование условий перехода.*

По аналогии с наследованием свойств состояний можно говорить о наследовании условий перехода в операторах перехода *Оп*. Это позволяет объяснить результат поиска. Например, переход к состоянию S_7 вызван наличием свойства $h \wedge c$ в S_3 и одновременно свойства $a \wedge b$ в S_0, что легко установить, анализируя условия перехода в $Оп_7$ и $Оп_3$. Обычно процесс последовательного применения операторов перехода к раскрытию вершин дерева состояний запоминается в программе поиска и может быть легко воспроизведен в виде трассы поиска, вследствие чего процедуру объяснения результатов часто называют *трассировкой поиска*.

3) *Поиск правил по образцу.*

Этот часто используемый в работах по ИИ термин означает, что левая часть правила «если S_{i-1} обладает свойством a, то переход к S_i» сравнивается с образцами, т.е. с перечнем фактически имеющихся имен состояний и их свойств. Совпадение левой части правила с каким-либо образцом вызывает активизацию правила, т.е. срабатывание его правой части.

Кроме описанных существует большое число других методов синтеза: метод морфологического ящика, весьма близкий к поиску по дереву состояний и И–ИЛИ дереву, методы «мозгового штурма» и контрольных вопросов, относящиеся к методам решения изобретательских задач, однако в САПР они не получили распространения из-за сложности их формализации.

Параметрический синтез. После синтеза структуры (если она не сопровождалась синтезом параметров) выполняется параметрический синтез. Вариант синтеза путем решения задачи оптимизации широко описан в литературе, поэтому рассмотрим генерационный параметрический синтез, т. е. расчет параметров по системе формул или уравнений без оптимизации, обеспечивающей получение работоспособного варианта объекта.

Задача ставится таким образом. Пусть функционирование ВС обеспечивается при выполнении системы соотношений (формул)

$$Y_i = f_i(x_1 \ldots x_m),\ i = 1, m,$$

где $x_1, \ldots, x_m$ – параметры ВС.

Необходимо определить последовательность применения формул расчета при задании некоторой части параметров, т. е. нужно синтезировать методику расчета по известным формулам.

Решение состоит в определении такой цепочки причинно-следственной связи в формулах (ранжирование формул), в которой расчет по первой формуле в этой цепочке позволил бы найти величины, необходимые для расчета по второй формуле и т. д. По существу данная задача состоит в планировании вычислений. Алгоритмы и программы, решающие эту задачу, в теории ИИ получили название *концептуальных решателей или планировщиков.*

Суть алгоритма планирования заключается в следующем. Найдем сначала формулу с минимальным числом неизвестных, например с одной неизвестной, и вычислим ее. Далее снова найдем новую формулу с новой неизвестной и также вычислим ее и т. д. Легко заметить, что если построить матрицу, столбцы которой соответствуют неизвестным m (часть из них задана), а строки – номерам формул, то при отыскании искомой последовательности расчета эта матрица будет иметь строго трапециевидную структуру:

а) расчет только по формулам;

б) расчет с решением системы уравнений относительно X_6, X_7. На рис. 3.7 показаны: *а* – случай переопределения (число формул больше неизвестных); *б* – случай недоопределения; 1, 7, 6, 4, 3 – номера формул, когда X_1 рассчитывается по формуле на основе известных X_2, X_5, а для расчета X_6, X_7 нужно решить систему из двух уравнений, после чего X_3, X_4 опять можно рассчитать по формулам.

Для автоматизации планирования можно каждую формулу заменить фреймовой системой инструкций. Например, для формулы $X_1 = f_1(X_2, X_5)$ можно записать:

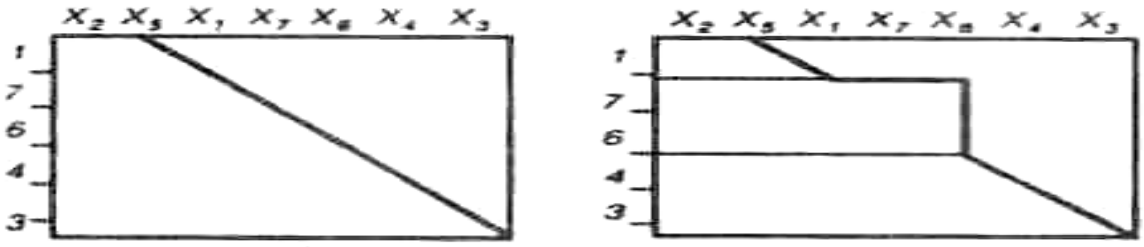
– если известно X_5, X_2, то известно X_1;

– если известно X_1, X_2, то известно X_5;

– если известно X_5, X_1, то известно X_2;

– если известно X_5, X_2, X_1, то неизвестных нет;

– если известно X_1, то неизвестны X_2, X_5 и т. д.



а *б*

Рис. 3.7. Построение матрицы по алгоритму планирования

Представляя каждую формулу в указанном виде (этот процесс легко автоматизировать, записывая и вводя в программу через дисплей лишь исходную формулу) и последовательно многократно просматривая их, можно определять порядок активизации каждой формулы, т. е. последовательность их использования в синтезируемой методике расчета. Но данный формализм правил соответствует формированию семантической вычислительной сети. Однако возможны и другие алгоритмы прямого определения структуры матриц, не связанные с построением семантических сетей (рис. 3.8).

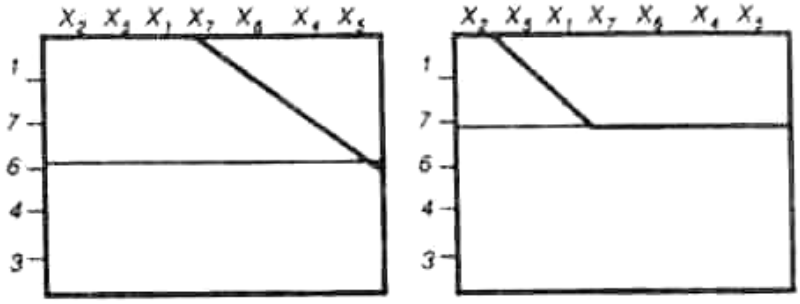


Рис. 3.8. Алгоритм прямого определения структуры матрицы

Контрольные вопросы

1. Какие процессы включаются в понятие «жизненный цикл изделий»?
2. Какие из функций реализованы в интеллектуальной системе?
3. Интеллектуальные САПР в проектировании электронных средств.
4. Понятие структурного синтеза. Параметрический синтез.
5. Синтезирующая интеллектуальная САПР.
6. В чем суть имитационного моделирования?
7. В каких случаях необходимо применять имитационное моделирование?
8. Каковы особенности непрерывно-детерминированного подхода к моделированию?
9. В чем особенности дискретно-стохастического подхода к моделированию?
10. В каких случаях применяют схемы вероятностных автоматов?
11. Каковы особенности дискретно-детерминированного подхода к моделированию?
12. В чем особенности непрерывно-стохастического подхода к моделированию?
13. Назовите особенности процессно-ориентированного подхода к моделированию.
14. Охарактеризуйте существующие методы интеллектуализации САПР.
15. В чем состоят основные концепции интеллектуальных САПР?
16. Каковы основные количественные и качественные характеристики интеллектуальных САПР?
17. В чем отличие внешней интеллектуализации САПР от внутренней?
18. Проведите сравнительный анализ оптимизационного и генерационного синтеза.
19. В чем различие методов теоретического и эвристического синтеза?
20. Перечислите типовые приемы структурного синтеза.

4. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ САПР

4.1. Представление знаний. Данные и знания

Одной из ключевых проблем создания ИИ является проблема представления и использования знаний. Ее разработка осуществляется различными направлениями ИИ.

Проблемы представления и использования знаний о мире, в котором функционирует ИИ, являются основными при построении его системы управления. В чем же сущность этих проблем и какова их специфика?

В области ИИ понятие о знаниях сформировалось в ходе исследований по созданию принципов и техники работы с большими объемами данных и по построению баз данных (БД). Эффективность БД во многом зависит от того, каким именно способом организовываются, структурируются данные в памяти компьютера. До недавнего времени основную роль в этом играли формальные характеристики данных: принадлежность их некоторой табличной рубрике, вхождение в одну тематическую группу и т. д.

Однако эффективность БД может быть существенно повышена, если связывать хранящуюся информацию не за счет форм тех или иных документов (таблиц, списков), а за счет тех отношений, которые существуют между фактами в объекте управления или в естественной среде. И отношения эти должны быть не случайными, ситуативными, а отражать существенные связи объекта, его природу, т. е. возникла необходимость отображения в БД знаний об объекте. Такие БД стали называть *интеллектуальными базами данных* или *базами* (системами) *знаний* (БЗ).

Итак, каждая БЗ является математической моделью некоторой области прикладного, неформализованного знания. Система понятий и отношений такой модели отображает систему понятий и отношений прикладного знания, а зависимости, существующие в модели, аппроксимируют соответствующие зависимости в нем. Разработанные модели должны быть зафиксированы в памяти компьютера и использоваться для решения прикладных задач. Создание БЗ предполагает решение следующих взаимосвязанных проблем. Прежде всего необходимо формализовать соответствующую область прикладного знания. Это трудная задача, поскольку решается она вручную и требует

совместной работы специалистов-прикладников и математиков. Для проведения формализации требуется выбрать или построить концептуальную схему модели. Разработка методологии всех этих операций и составляет содержание первой проблемы – *проблемы формализации.*

Вторая проблема – *проблема представления знаний* – связана с разработкой формального аппарата для описания способов их фиксации в памяти ЭВМ.

Разработка теории вычислений и других преобразований, проводимых в построенных моделях, составляет третью проблему – *проблему использования знаний.*

И, наконец, четвертая, технологическая проблема, решением которой занимаются системные программисты, – это *проблема разработки средств программной поддержки моделей,* т. е. создания баз знаний и систем управления ими.

Основное внимание в ИИ уделяется второй и третьей из перечисленных проблем, причем ведущая роль отводится проблеме представления. На практике ее разрабатывают совместно с вопросами построения концептуальных схем моделей знаний, и многие полагают, что именно эта проблема является основной для современного ИИ.

К настоящему времени в области разработки БЗ достигнуты значительные успехи. Полученные результаты послужили толчком к созданию полезных и интересных систем нового класса, имеющих широкое практическое применение, – экспертных систем, которые могут быть использованы в качестве советчиков и консультантов в самых разных сферах человеческой деятельности.

При изучении интеллектуальных систем традиционно возникает вопрос, что же такое знания и чем они отличаются от обычных данных, десятилетиями обрабатываемых ЭВМ. Можно предложить несколько рабочих определений, в рамках которых это становится очевидным.

Данные – это отдельные факты, характеризующие объекты, процессы и явления предметной области, а также их свойства [1].

При обработке на компьютере данные трансформируются, условно проходя следующие этапы:

1. D_1 – данные как результат измерений и наблюдений;

2. D_2 – данные на материальных носителях информации (таблицы, протоколы, справочники);

3. D_3 – модели (структуры) данных в виде диаграмм, графиков, функций;

4. D_4 – данные в компьютере на языке описания данных;

5. D_5 – базы данных на машинных носителях информации.

Знания основаны на данных, полученных эмпирическим путем. Они представляют собой результат мыслительной деятельности человека, направленной на обобщение его опыта, полученного в результате практической деятельности.

Знания – это закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области [1].

При обработке на компьютере знания трансформируются аналогично данным:

1. Z_1 – знания в памяти человека как результат мышления;

2. Z_2 – материальные носители знаний (учебники, методические пособия);

3. Z_3 – поле знаний – условное описание основных объектов предметной области, их атрибутов и закономерностей, их связывающих;

4. Z_4 – знания, описанные на языках представления знаний (продукционные языки, семантические сети, фреймы);

5. Z_5 – база знаний на машинных носителях информации.

Часто используется следующее определение знаний: *знания* – это хорошо структурированные данные, или данные о данных, или метаданные.

Для хранения данных используются базы данных (для них характерны большой объем и относительно небольшая удельная стоимость информации), для хранения знаний – базы знаний (небольшой объем, но исключительно дорогие информационные массивы). *База знаний* – основа любой интеллектуальной системы.

Знания могут быть классифицированы по следующим категориям:

• *поверхностные* – знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области;

• *глубинные* – абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов.

Пример. Поверхностные знания: «Если нажать на кнопку звонка, раздастся звук. Если, болит голова, то следует принять аспирин».

Глубинные знания: «Принципиальная электрическая схема звонка и проводки. Знания физиологов и врачей высокой квалификации о причинах, видах головных болей и методах их лечения».

Современные экспертные системы работают в основном с поверхностными знаниями. Это связано с тем, что на данный момент нет универсальных методик, позволяющих выявлять глубинные структуры знаний и работать с ними.

Кроме того, в учебниках по ИИ знания традиционно делят на *процедурные* и *декларативные.* Исторически первичными были процедурные знания, т.е. знания, «растворенные» в алгоритмах. Они управляли данными. Для их изменения требовалось изменять программы. Однако с развитием искусственного интеллекта приоритет данных постепенно изменялся, и все бóльшая часть знаний сосредоточивалась в структурах данных (таблицы, списки, абстрактные типы данных), т.е. увеличивалась роль декларативных знаний.

Сегодня знания приобрели чисто декларативную форму, т.е. знаниями считаются предложения, записанные на языках представления знаний, приближенных к естественному и понятных неспециалистам.

Представление знаний (ПЗ) – это соглашение о том, как описывать реальный мир. Основная цель ПЗ – получить математическую модель реального мира и его частей с целью принятия на ее основе необходимых решений.

Инженерия знаний – наука о компьютерном представлении знаний и их обработке. Она рассматривает средства, позволяющие описывать знания с помощью языка ПЗ, организовать хранение знаний в системе и т.д.

Знания можно разделить на факты и правила.

Модели представления знаний. Существуют десятки моделей (или языков) представления знаний для различных предметных областей. Большинство из них может быть сведено к следующим классам:

- продукционные модели;
- семантические сети;
- фреймы;
- формальные логические модели.

Продукционная модель или модель, основанная на правилах, позволяет представить знания в виде предложений типа «Если (условие), то (действие)».

Под условием *(антецедентом)* понимается некоторое предложение-образец, по которому осуществляется поиск в базе знаний, а под действием *(консекветом)* – действия, выполняемые при успешном исходе поиска (они могут быть промежуточными, выступающими далее как условия, и терминальными или целевыми, завершающими работу системы) [3].

Продукционные модели называют еще продукциями. Под продукцией понимается выражение A & B (если A, то B).

Чаще всего вывод на такой базе знаний бывает *прямой* (от данных к поиску цели) или *обратный* (от цели для ее подтверждения – к данным). Данные – это исходные факты, хранящиеся в базе фактов, на основании которых запускается машина вывода или интерпретатор правил, перебирающий правила из продукционной базы знаний.

Пример. Имеется фрагмент базы знаний из двух правил:

П1: Если «отдых – летом» и «человек – активный», то «ехать в горы».

П2: Если «любит солнце», то «отдых летом».

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

Прямой вывод – исходя из данных, получить ответ.

1-й проход.

Шаг 1. Пробуем $П_1$, не работает (не хватает данных «отдых летом»).

Шаг 2. Пробуем $П_2$, работает, в базу поступает факт «отдых летом».

2-й проход.

Шаг 3. Пробуем $П_1$, работает, активируется цель «ехать в горы», которая и выступает как совет, который дает ЭС.

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

1-й проход.

Шаг 1. Цель – «ехать в горы»: пробуем $П_1$ – данных «отдых летом» нет, они становятся новой целью, и ищется правило, где она в правой части.

Шаг 2. Цель «отдых летом»: правило $П_2$ подтверждает цель и активирует ее.

2-й проход.

Шаг 3. Пробуем $П_1$, подтверждается искомая цель.

Если в памяти системы хранится некоторый набор продукций, то они образуют систему продукций. В системе продукций должны быть заданы специальные процедуры управления продукциями, с помощью которых происходит актуализация продукций и выбор для выполнения той или иной продукции из числа актуализированных.

Системы продукций широко распространены в экспертных системах. Популярность продукционных моделей определяется несколькими факторами:

1. Подавляющая часть человеческих знаний может быть записана в виде продукций.

2. Системы продукций являются модульными. За небольшим исключением удаление или добавление продукций не приводит к изменениям в остальных продукциях.

3. При необходимости системы продукций могут реализовать любые алгоритмы и, следовательно, способны отражать любое процедурное знание, доступное компьютеру.

4. Наличие в продукциях указателей на сферу применения продукции позволяет эффективно организовать память, сократив время поиска в ней необходимой информации. Классификация сфер может быть многоуровневой, что еще более повышает эффективность поиска знаний, так как позволяет наследовать информацию в базе знаний.

5. При объединении систем продукций и сетевых представлений получаются средства, обладающие большой вычислительной мощностью.

6. Естественный параллелизм в системе продукций, асинхронность их реализации делают продукционные системы удобной моделью вычислений для компьютера новой архитектуры, в которой идея параллельности и асинхронности является центральной.

Продукционные модели имеют, по крайней мере, два недостатка. При большом числе продукции становится сложной проверка непротиворечивости системы продукций. Это заставляет при добавлении новых продукций тратить много времени на проверку непротиворечивости новой системы. Из-за присущей системе недетерминированности (неоднозначный выбор выполняемой продукции из фронта активизированных продукций) возникают принципиальные трудности при проверке корректности работы системы. Считается, что если в ИС число продукций достигает тысячи, то мало шансов, что система продукций во всех случаях будет правильно функционировать. Именно поэтому число продукций, с которым, как правило, работают современные ИС, не превышает тысячи.

Логические модели. В основе моделей такого типа лежит понятие формальной системы. Постановка и решение любой задачи всегда связаны с ее «погружением» в подходящую предметную область. Так, решая задачу составления расписания обработки деталей на металлорежущих станках, вовлекаем в предметную область такие объекты, как конкретные станки, детали, интервалы времени, и общие понятия «станок», «деталь», «тип станка» и т. п. Все предметы и события, которые составляют основу общего понимания необходимой для решения задачи информации, называются *предметной областью.* Мысленно предметная область представляется состоящей из реальных или абстрактных объектов, называемых *сущностями.*

Сущности предметной области находятся в определенных *отношениях* друг к другу (ассоциациях), которые также можно рассматривать как сущности и включать в предметную область. Между сущностями наблюдаются различные отношения подобия. Совокупность подобных сущностей составляет *класс сущностей,* являющийся новой сущностью предметной области.

Отношения между сущностями выражаются с помощью суждений. *Суждение* – это мысленно возможная ситуация, которая может иметь место для предъявляемых сущностей или не иметь места. В языке (формальном или естественном) суждениям отвечают *предложения.* Суждения и предложения также можно рассматривать как сущности и включать в предметную область.

Семантические сети. Термин «семантическая» означает смысловая, а сама семантика – это наука, устанавливающая отношения между символами и объектами, которые они обозначают, т.е. наука, определяющая смысл знаков [5].

Семантическая сеть – это ориентированный граф, вершины которого – понятия, а дуги – отношения между ними.

Понятиями обычно выступают абстрактные или конкретные объекты, а *отношения* – это связи типа: «это» («is»), «имеет частью» («has part»), «принадлежит», «любит». Характерной особенностью семантических сетей является обязательное наличие трех типов отношений:

- класс – элемент класса (цветок – роза);
- свойство – значение (цвет – желтый);
- пример элемента класса (роза – чайная).

Можно ввести несколько классификаций семантических сетей:

– по количеству типов отношений:

• однородные (с единственным типом отношений);

• неоднородные (с различными типами отношений);

– по типам отношений:

• бинарные (в которых отношения связывают два объекта);

• N-арные (в которых есть специальные отношения, связывающие более двух понятий).

Наиболее часто в семантических сетях используются следующие отношения:

• связи типа *«часть – целое» («класс – подкласс», «элемент – множество»* и т.п.);

• функциональные связи (определяемые обычно глаголами «производит», «влияет»...);

• количественные (больше, меньше, равно...);

• пространственные (далеко от, близко от, за, под, над...);

• временные (раньше, позже, в течение...);

• атрибутивные связи (иметь свойство, иметь значение...);

• логические связи (и, или, не) и др.

Проблема поиска решения в базе знаний типа семантической сети сводится к задаче поиска фрагмента сети, соответствующего некоторой подсети, отвечающей поставленному вопросу.

*Пример***.** На рис. 4.1 изображена семантическая сеть. В качестве вершин – понятия: Человек, Иванов, Волга, Автомобиль, Вид транспорта. Двигатель.

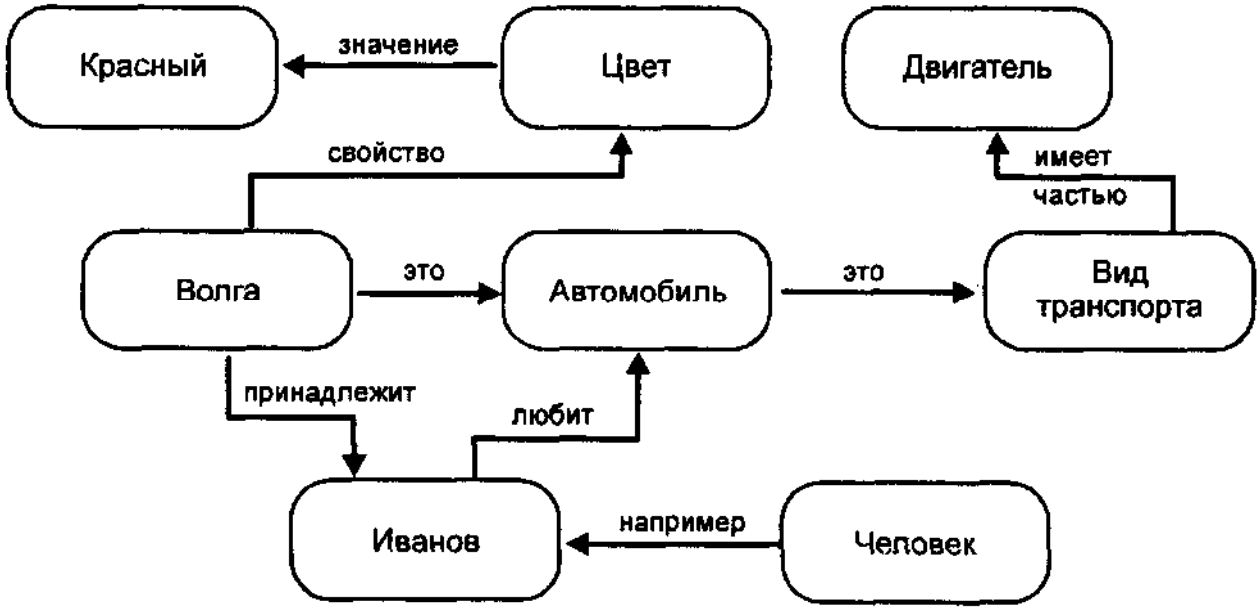


Рис. 4.1. Семантическая сеть

Основное преимущество этой модели – в соответствии современным представлениям об организации долговременной памяти человека. Недостаток модели – сложность поиска вывода на семантической сети.

Для реализации семантических сетей существуют специальные сетевые языки, например NET и др. Широко известны экспертные системы, использующие семантические сети в качестве языка представления знаний, – PROSPECTOR, CASNET, TORUS.

Фреймовые модели. Термин «фрейм» (от англ. *frame* –каркас или рамка) был предложен Марвином Минским в 70-е годы XX века для обозначения структуры знаний для восприятия пространственных сцен. Эта модель, как и семантическая сеть, имеет глубокое психологическое обоснование.

Фрейм – это абстрактный образ для представления некоего стереотипа восприятия [2]. В психологии и философии известно понятие абстрактного образа. Например, произнесение вслух слова «комната» порождает у слушающих образ комнаты: «жилое помещение с четырьмя стенами, полом, потолком, окнами и дверью, площадью 6–20 м^2». Из этого описания ничего нельзя убрать (например, убрав окна, мы получим уже чулан, а не комнату), но в нем есть «дырки», или слоты, – это незаполненные значения некоторых атрибутов, например: количество окон, цвет стен, высота потолка, покрытие пола и др.

В теории фреймов такой образ комнаты называется фреймом комнаты. Фреймом также называется и формализованная модель для отображения образа.

Различают *фреймы-образцы,* или *прототипы,* хранящиеся в базе знаний, и *фреймы-экземпляры,* которые создаются для отображения реальных фактических ситуаций на основе поступающих данных. Модель фрейма является достаточно универсальной, поскольку позволяет отобразить все многообразие знаний о мире через:

• *фреймы-структуры,* использующиеся для обозначения объектов и понятий (заем, залог, вексель);

• *фреймы-роли* (менеджер, кассир, клиент);

• *фреймы-сценарии* (банкротство, собрание акционеров, празднование именин);

• *фреймы-ситуации* (тревога, авария, рабочий режим устройства) и др.

Традиционно структура фрейма может быть представлена как список свойств:

(ИМЯ ФРЕЙМА:

(имя 1-го слота: значение 1-го слота), (имя 2-го слота: значение 2-го слота),

(имя N-го слота: значение N-го слота).

Ту же запись можно представить в виде таблицы, дополнив ее двумя столбцами:

Имя фрейма			
имя слота	тип слота	значение слота	присоединенная процедура

В таблице дополнительные столбцы предназначены для описания типа слота и возможного присоединения к тому или иному слоту специальных процедур, что допускается в теории фреймов. В качестве значения слота может выступать имя другого фрейма; так образуют сети фреймов.

Важнейшим свойством теории фреймов является заимствованное из теории семантических сетей наследование свойств. И во фреймах, и в семантических сетях наследование происходит по *АКО-связям (A-Kind-Of)*. Слот АКО указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, т.е. переносятся, значения аналогичных слотов.

Пример. Например, в сети фреймов на рис. 4.2 понятие «ученик» наследует свойства фреймов «ребенок» и «человек», которые находятся на более высоком уровне иерархии. Так, на вопрос: «Любят ли ученики сладкое?» следует ответ: «Да», так как этим свойством обладают все дети, что указано во фрейме «ребенок». Наследование свойств может быть частичным. Так, возраст для учеников не наследуется из фрейма «ребенок», поскольку указан явно в своем собственном фрейме.

Основным преимуществом фреймов как модели представления знаний является способность отражать концептуальную основу организации памяти человека, а также ее гибкость и наглядность.

Специальные языки представления знаний в сетях фреймов FRL (Frame Representation Language) и другие позволяют эффективно

строить промышленные экспертные системы (ЭС). Широко известны такие фрейм-ориентированные экспертные системы, как ANALYST, МОДИС.

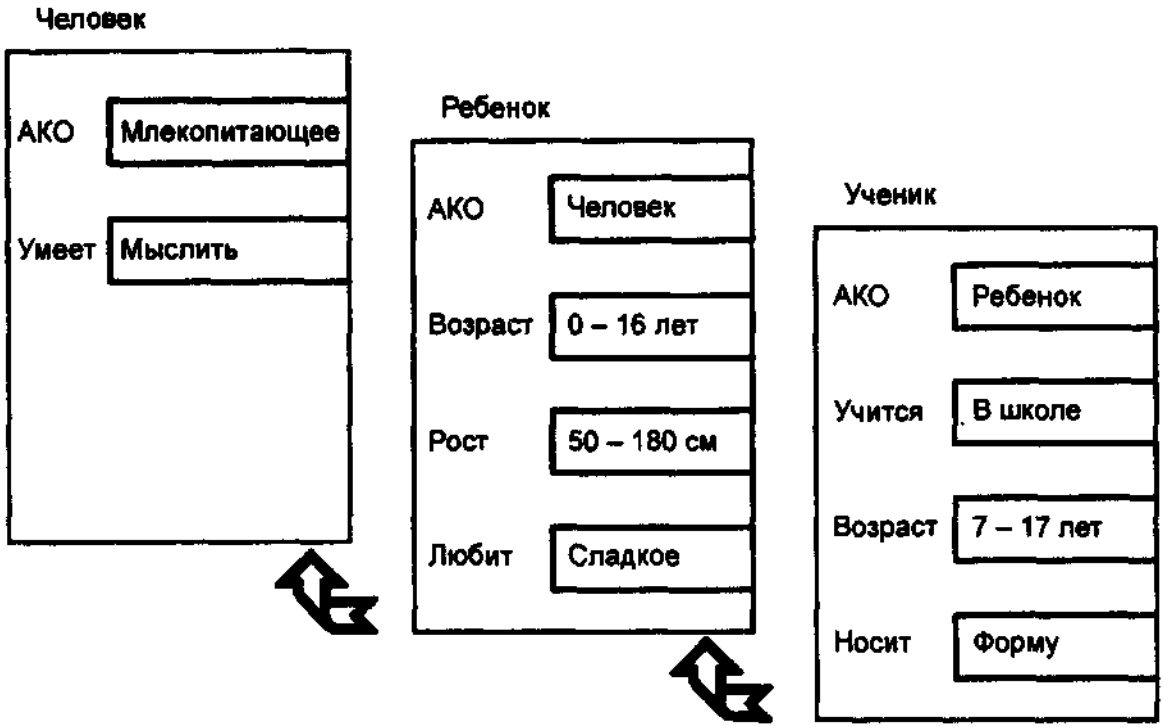


Рис. 4.2. Сеть фреймов

4.2. Вывод на знаниях

Несмотря на все недостатки, наибольшее распространение получила продукционная модель представления знаний. При использовании продукционной модели база знаний состоит из набора правил. Программа, управляющая перебором правил, называется машиной вывода.

Машина вывода (интерпретатор правил) выполняет две функции: во-первых, просмотр существующих фактов из рабочей памяти (базы данных) и правил из базы знаний и добавление (по мере возможности) в рабочую память новых фактов и, во-вторых, определение порядка просмотра и применения правил. Этот механизм управляет процессом консультации, сохраняя для пользователя информацию о полученных заключениях, и запрашивает у него информацию, когда для срабатывания очередного правила в рабочей памяти оказывается недостаточно данных.

В подавляющем большинстве систем, основанных на знаниях, механизм вывода представляет собой небольшую по объему программу и включает два компонента: один реализует собственно вывод, другой управляет этим процессом. Действие *компонента вывода* основано на применении правила, называемого *modus ponens*.

Правило modus ponens: если известно, что истинно утверждение А и существует правило вида «ЕСЛИ А, ТО В», тогда утверждение В также истинно.

Правила срабатывают, когда находятся факты, удовлетворяющие их левой части: если истинна посылка, то должно быть истинно и заключение. Компонент вывода должен функционировать даже при недостатке информации. Полученное решение может и не быть точным, однако система не должна останавливаться из-за того, что отсутствует какая-либо часть входной информации.

Управляющий компонент определяет порядок применения правил и выполняет четыре функции:

1. *Сопоставление* – образец правила сопоставляется с имеющимися фактами.

2. *Выбор* – если в конкретной ситуации может быть применено сразу несколько правил, то из них выбирается одно, наиболее подходящее по заданному критерию (разрешение конфликта).

3. *Срабатывание* – если образец правила при сопоставлении совпал с какими-либо фактами из рабочей памяти, то правило срабатывает.

4. *Действие* – рабочая память подвергается изменению путем добавления в нее заключения сработавшего правила. Если в правой части правила содержится указание на какое-либо действие, то оно выполняется (как, например, в системах обеспечения безопасности информации).

Интерпретатор продукций работает циклически. В каждом цикле он просматривает все правила, чтобы выявить те посылки, которые совпадают с известными на данный момент фактами из рабочей памяти. После выбора правило срабатывает, его заключение заносится в рабочую память, и затем цикл повторяется сначала.

В одном цикле может сработать только одно правило. Если несколько правил успешно сопоставлены с фактами, то интерпретатор производит выбор по определенному критерию единственного правила, которое срабатывает в данном цикле. Цикл работы интерпретатора схематически представлен на рис. 4.3.

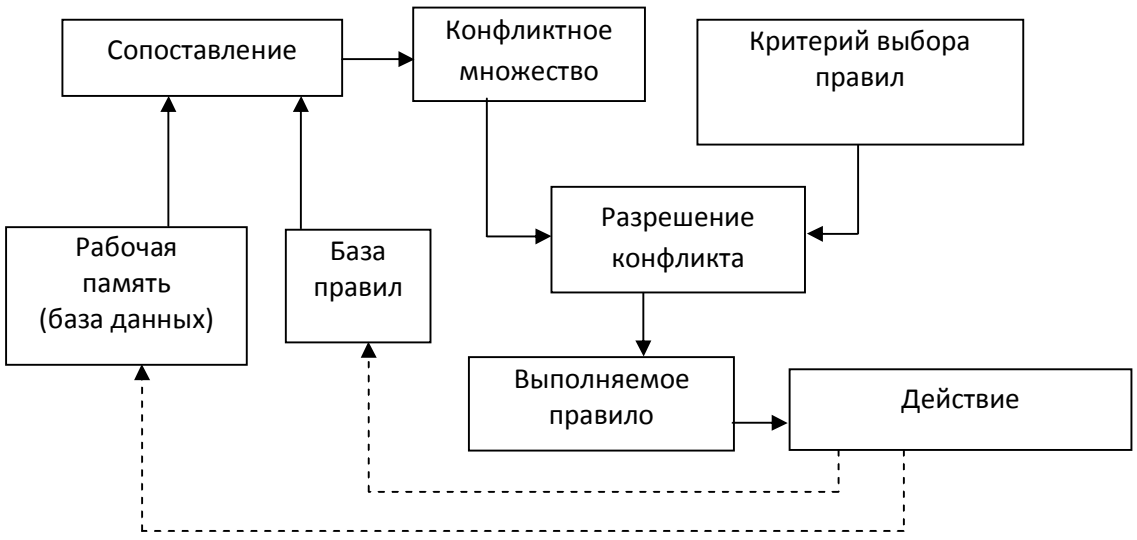


Рис. 4.3. Цикл работы интерпретатора

Информация из рабочей памяти последовательно сопоставляется с посылками правил для выявления успешного сопоставления. Совокупность отобранных правил составляет так называемое *конфликтное множество.* Для разрешения конфликта интерпретатор имеет критерий, с помощью которого он выбирает единственное правило, после чего оно срабатывает. Это выражается в занесении фактов, образующих заключение правила, в рабочую память или в изменении критерия выбора конфликтующих правил. Если же в заключение правила входит название какого-нибудь действия, то оно выполняется.

Работа машины вывода зависит только от состояния рабочей памяти и от состава базы знаний. На практике обычно учитывается история работы, т.е. поведение механизма вывода в предшествующих циклах. Информация о поведении механизма вывода запоминается в памяти состояний. Обычно память состояний содержит протокол системы.

Стратегии управления выводом. От выбранного метода поиска, т.е. стратегии вывода, будет зависеть порядок применения и срабатывания правил. Процедура выбора сводится к определению направления поиска и способа его осуществления. Процедуры, реализующие поиск, обычно «зашиты» в механизм вывода, поэтому в большинстве систем инженеры знаний не имеют к ним доступа и, следовательно, не могут в них ничего изменять по своему желанию.

При разработке стратегии управления выводом важно определить два вопроса:

1. Какую точку в пространстве состояний принять в качестве исходной? От выбора этой точки зависит и метод осуществления поиска – в прямом или обратном направлении.

2. Какими методами можно повысить эффективность поиска решения? Эти методы определяются выбранной стратегией перебора – в глубину, в ширину, по подзадачам или иначе.

Прямой и обратный вывод. При обратном порядке вывода вначале выдвигается некоторая гипотеза, а затем механизм вывода как бы возвращается назад, переходя к фактам, пытаясь найти те, которые подтверждают гипотезу (рис. 4.4, правая часть). Если она оказалась правильной, то выбирается следующая гипотеза, детализирующая первую и являющаяся по отношению к ней подцелью. Далее отыскиваются факты, подтверждающие истинность подчиненной гипотезы. Вывод такого типа называется управляемым целями или управляемым консеквентами. Обратный поиск применяется в тех случаях, когда цели известны и их сравнительно немного.

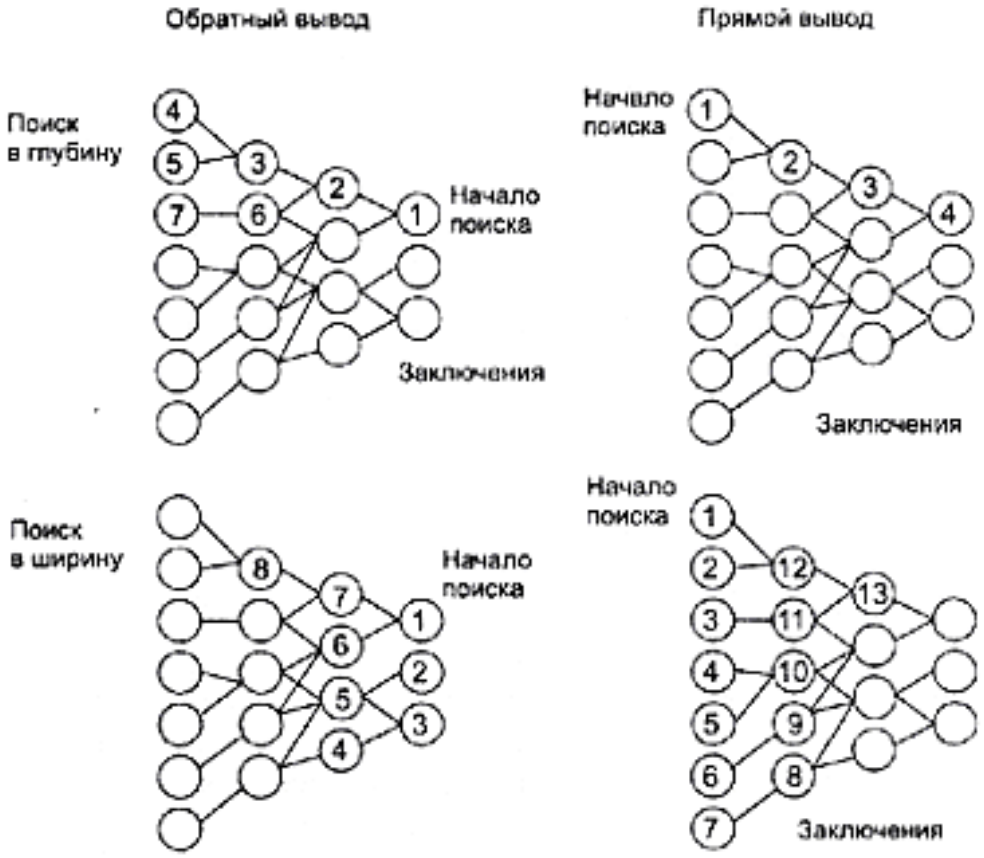


Рис. 4.4. Стратегии вывода

В системах с прямым выводом по известным фактам отыскивается заключение, которое из этих фактов следует (рис. 4.4, левая часть). Если такое заключение удается найти, то оно заносится в рабочую память. Прямой вывод часто называют выводом, управляемым данными, или выводом, управляемым антецедентами.

Существуют системы, в которых вывод основывается на сочетании упомянутых выше методов – обратного и ограниченного прямого. Такой комбинированный метод получил название циклического (см. пример продукционной модели).

Методы поиска в глубину и ширину. В системах, база знаний которых насчитывает сотни правил, желательным является использование стратегии управления выводом, позволяющей минимизировать время поиска решения и тем самым повысить эффективность вывода. К числу таких стратегий относятся: поиск в глубину, поиск в ширину, разбиение на подзадачи и альфа-бета алгоритм.

При ***поиске в глубину*** в качестве очередной подцели выбирается та, которая соответствует следующему, более детальному уровню описания задачи. Например, диагностирующая система, сделав на основе известных симптомов предположение о наличии определенного заболевания, будет продолжать запрашивать уточняющие признаки и симптомы этой болезни до тех пор, пока полностью не опровергнет выдвинутую гипотезу.

При ***поиске в ширину***, напротив, система вначале проанализирует все симптомы, находящиеся на одном уровне пространства состояний, даже если они относятся к разным заболеваниям, и лишь затем перейдет к симптомам следующего уровня детальности.

Разбиение на подзадачи подразумевает выделение подзадач, решение которых рассматривается как достижение промежуточных целей на пути к конечной цели. Примером, подтверждающим эффективность разбиения на подзадачи, является поиск неисправностей в компьютере – сначала выявляется отказавшая подсистема (питание, память и т. д.), что значительно сужает пространство поиска. Если удается правильно понять сущность задачи и оптимально разбить ее на систему иерархически связанных целей-подцелей, то можно добиться того, что путь к ее решению в пространстве поиска будет минимален.

Альфа-бета алгоритм позволяет уменьшить пространство состояний путем удаления ветвей, неперспективных для успешного поиска. Поэтому просматриваются только те вершины, в которые можно попасть в результате следующего шага, после чего неперспективные направления исключаются. Альфа-бета алгоритм нашел широкое применение в основном в системах, ориентированных на различные игры, например в шахматных программах.

Графы И/ИЛИ. Различают два основных типа стратегий управления: *безвозвратный* и *пробный.*

В ***безвозвратном режиме*** управления выбирается применимое правило и используется необратимо, без возможности пересмотра в дальнейшем.

В ***пробном режиме*** управления выбирается применимое правило (либо произвольно, либо на каком-то разумном основании). Это правило используется, но резервируется возможность впоследствии заново вернуться к этой ситуации, чтобы применить другое правило.

Далее различают два типа пробных режимов управления: *с возвращением* и *поиском на графе.*

В ***режиме с возвращением*** при выборе правила определяется некоторая *точка возврата.* Если последующие вычисления приведут к трудностям в построении решения, то процесс вычисления переходит к предыдущей точке возврата, где применяется другое правило, и процесс продолжается.

Во втором типе пробного режима, который называют *управление с поиском на графе,* предусмотрено запоминание результатов применения одновременно нескольких последовательностей правил. Здесь используются различные виды графовых структур и процедур поиска на графе.

Рассмотрим систему с исходной базой данных (C, B, Z), правила продукций которой основаны на следующих правилах переписывания:

П1: C Є (D, L); П2: C Є (B, M); ПЗ: B Є (M, M); П4: Z Є (B, B, M), а терминальное условие состоит в том, что эта база данных должна содержать только символ M.

Исходная база данных разлагается на составляющие C, B и Z. Правила продукций применимы независимо к каждой составляющей (возможно, параллельно). Результаты этих операций также подвергаются декомпозиции и т.д., пока каждый компонент базы данных не будет содержать только символы M.

Как и в случае обычных графов, граф типа И/ИЛИ состоит из вершин, помеченных глобальными базами данных. Вершины, помеченные составными базами данных, имеют множество вершин-преемников, каждая из которых помечена одной из составляющих. Эти вершины-преемники называются *вершинами типа И,* так как для полной обработки составной базы данных должны быть обработаны полностью все ее составляющие.

Множество вершин типа И (рис. 4.5) обозначаются дугой, объединяющей входящие в них дуги графа, которая называется *k*-связкой (*k* – число вершин-преемников).

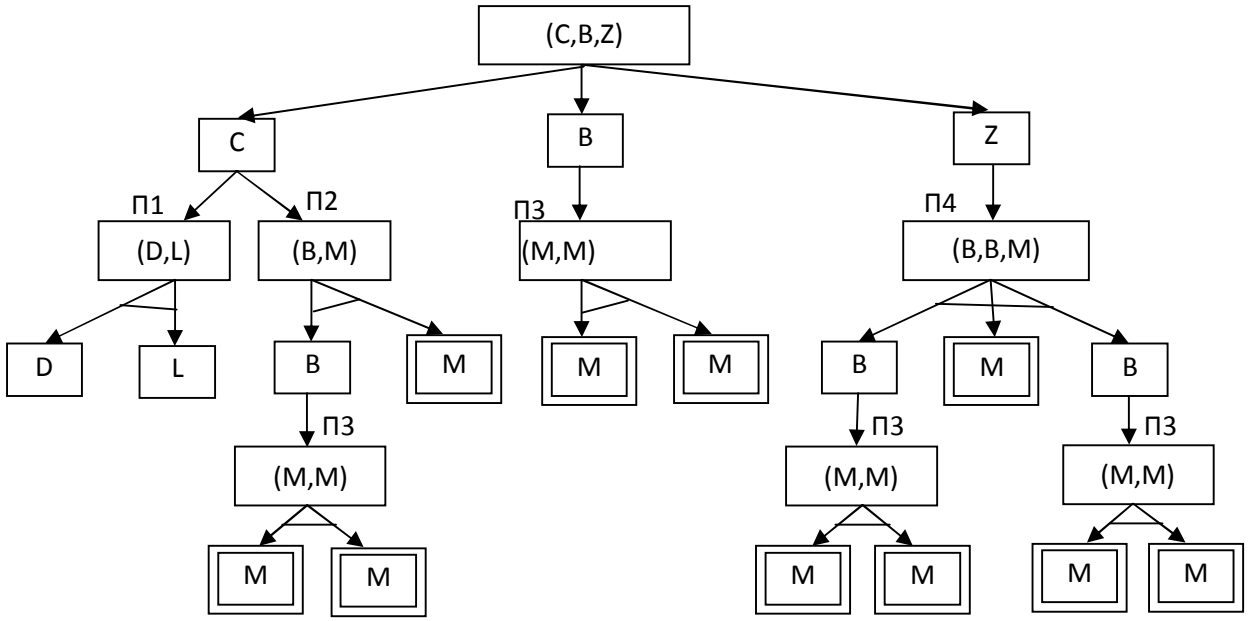


Рис. 4.5. Граф типа И/ИЛИ

К составляющим базам данных можно применять правила. Вершины, помеченные этими составляющими базами данных, имеют вершины-преемники, которые помечены базами, полученными в результате применения правил. Эти вершины-преемники называют *вершинами типа ИЛИ,* поскольку для полной обработки составляющей базы данных полностью должна быть обработана база данных, порожденная в результате применения лишь одного из правил.

На рис. 4.5 все вершины, соответствующие какой-либо составляющей базе данных, удовлетворяющей терминальному условию, заключены в двойную рамку. Такие вершины называются терминальными.

4.3. Прикладные интеллектуальные системы

Основой интеллектуальных технологий сегодня является обработка знаний. Системы, ядром которых является база знаний или модель предметной области, описанная на языке сверхвысокого уровня, приближенном к естественному, называют интеллектуальными. Будем называть такой язык сверхвысокого уровня *языком представления знаний (ЯПЗ).* Чаще всего интеллектуальные системы применя-

ются для решения сложных задач, где основная сложность решения связана с использованием слабоформализованных знаний специалистов-практиков и где логическая (или смысловая) обработка информации превалирует над вычислительной. Например: понимание естественного языка, поддержка принятия решения в сложных ситуациях, постановка диагноза и рекомендации по методам лечения, анализ визуальной информации, управление диспетчерскими пультами и др.

Фактически сейчас прикладные интеллектуальные системы используются в десятках тысяч приложений. Наиболее распространенным видом ИС являются экспертные системы.

Экспертные системы – это наиболее распространенный класс ИС, ориентированный на тиражирование опыта высококвалифицированных специалистов в областях, где качество принятия решений традиционно зависит от уровня экспертизы, например: медицина, юриспруденция, геология, экономика, военное дело и др.

ЭС эффективны лишь в специфических экспертных областях, где важен эмпирический опыт специалистов.

Ежегодно крупные фирмы разрабатывают десятки ЭС типа «in-house» для внутреннего пользования. Эти системы интегрируют опыт специалистов компании по ключевым и стратегически важным технологиям.

Современные ЭС – это сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных областях и распространяющие этот эмпирический опыт для консультирования менее квалифицированных пользователей.

Разработка экспертных систем, как активно развивающаяся ветвь информатики, направлена на использование компьютера для обработки информации в тех областях науки и техники, где традиционные математические методы моделирования малопригодны. В этих областях важна смысловая и логическая обработка информации, важен опыт экспертов.

Приведем некоторые условия, которые могут свидетельствовать о необходимости разработки и внедрения экспертных систем:

• нехватка специалистов, затрачивающих значительное время для оказания помощи другим;

• выполнение небольшой задачи требует многочисленного коллектива специалистов, поскольку ни один из них не обладает достаточным знанием;

• сниженная производительность, поскольку задача требует полного анализа сложного набора условий, а обычный специалист не в состоянии просмотреть (за отведенное время) все эти условия;

• большое расхождение между решениями самых хороших и самых плохих исполнителей;

• наличие конкурентов, имеющих преимущество в силу того, что они лучше справляются с поставленной задачей.

Подходящие задачи имеют следующие характеристики:

• являются узкоспециализированными;

• не зависят в значительной степени от общечеловеческих знаний или соображений здравого смысла;

• не являются для эксперта ни слишком легкими, ни слишком сложными (время, необходимое эксперту для решения проблемы, может составлять от трех часов до трех недель).

Экспертные системы достаточно молоды – первые системы такого рода, MYCIN и DENDRAL, появились в США в середине 70-х годов. В настоящее время в мире насчитывается несколько тысяч промышленных ЭС, которые дают советы:

• при управлении сложными диспетчерскими пультами, например сети распределения электроэнергии;

• постановке медицинских диагнозов;

• поиске неисправностей в электронных приборах, диагностике отказов контрольно-измерительного оборудования;

• по проектированию интегральных микросхем;

• управлению перевозками;

• прогнозу военных действий;

• формированию портфеля инвестиций, оценке финансовых рисков, налогообложению и т. д.

Современное состояние разработок в области ЭС в России можно охарактеризовать как стадию все возрастающего интереса среди широких слоев специалистов – финансистов, топ-менеджеров, преподавателей, инженеров, медиков, психологов, программистов, лингвистов. В последние годы этот интерес имеет пока достаточно слабое материальное подкрепление – явная нехватка учебников и специальной литературы, отсутствие символьных процессоров и рабочих станций, ограниченное финансирование исследований в этой области, слабый отечественный рынок программных продуктов для разработки ЭС.

Поэтому появляется возможность распространения «подделок» под экспертные системы в виде многочисленных диалоговых систем и интерактивных пакетов прикладных программ, которые дискредитируют в глазах пользователей это чрезвычайно перспективное направление. Процесс создания экспертной системы требует участия высококвалифицированных специалистов в области искусственного интеллекта, которых пока выпускает небольшое количество высших учебных заведений страны.

Наибольшие трудности в разработке ЭС вызывает сегодня не процесс машинной реализации систем, а домашинный этап анализа знаний и проектирования базы знаний. Этим занимается специальная наука – инженерия знаний.

4.3.1. Введение в экспертные системы. Определение и структура

Экспертные системы – это сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных областях и тиражирующие этот эмпирический опыт для консультаций менее квалифицированных пользователей.

В целом процесс функционирования ЭС можно представить следующим образом: пользователь, желающий получить необходимую информацию, через пользовательский интерфейс посылает запрос к ЭС; решатель, пользуясь базой знаний, генерирует и выдает пользователю подходящую рекомендацию, объясняя ход своих рассуждений при помощи подсистемы объяснений.

Пользователь – специалист предметной области, для которого предназначена система. Обычно его квалификация недостаточно высока, и поэтому он нуждается в помощи и поддержке своей деятельности со стороны ЭС.

Инженер по знаниям – специалист в области искусственного интеллекта, выступающий в роли промежуточного буфера между экспертом и базой знаний.

Интерфейс пользователя – комплекс программ, реализующих диалог пользователя с ЭС как на стадии ввода информации, так и при получении результатов.

База знаний – ядро ЭС, совокупность знаний предметной области, записанная на машинный носитель в форме, понятной эксперту и пользователю (обычно на некотором языке, приближенном к естественному).

Решатель – программа, моделирующая ход рассуждений эксперта на основании знаний, имеющихся в БЗ. Синонимы: *дедуктивная машина, машина вывода, блок логического вывода.*

Подсистема объяснений – программа, позволяющая пользователю получить ответы на вопросы: «Как была получена та или иная рекомендация?» и «Почему система приняла такое решение?». Ответ на вопрос «как» – это трассировка всего процесса получения решения с указанием использованных фрагментов БЗ, т.е. всех шагов цепи умозаключений. Ответ на вопрос «почему» – ссылка на умозаключение, непосредственно предшествовавшее полученному решению, т.е. отход на один шаг назад. Развитые подсистемы объяснений поддерживают и другие типы вопросов.

Интеллектуальный редактор БЗ – программа, предоставляющая инженеру по знаниям возможность создавать БЗ в диалоговом режиме. Включает в себя систему вложенных меню, шаблонов языка представления знаний, подсказок и других сервисных средств, облегчающих работу с базой.

Следует подчеркнуть, что представленная структура является минимальной, что означает обязательное присутствие указанных на ней блоков. Промышленные прикладные ЭС могут быть существенно сложнее и дополнительно включать базы данных, интерфейсы обмена данными с различными пакетами прикладных программ, электронными библиотеками и т. д.

Классификация систем, основанных на знаниях. Класс ЭС сегодня объединяет несколько тысяч различных программных комплексов, которые можно классифицировать по различным критериям.

1. Классификация по решаемой задаче:

• *Интерпретация данных.* Это одна из традиционных задач для экспертных систем. Под интерпретацией понимается процесс определения смысла данных, результаты которого должны быть согласованными и корректными. Обычно предусматривается многовариантный анализ данных (например: обнаружение и идентификация различных типов океанских судов по результатам аэрокосмического сканирования – SIAP; определение основных свойств личности по результатам психодиагностического тестирования).

• *Диагностика.* Под диагностикой понимается процесс соотнесения объекта с некоторым классом объектов и/или обнаружение неисправности в некоторой системе. Неисправность – это отклонение от нормы. Такая трактовка позволяет с единых теоретических позиций

рассматривать и неисправность оборудования в технических системах, и заболевания живых организмов, и все возможные природные аномалии. Важной спецификой является здесь необходимость понимания функциональной структуры («анатомии») диагностирующей системы (например: диагностика и терапия сужения коронарных сосудов – ANGY; диагностика ошибок в аппаратуре и математическом обеспечении ЭВМ).

• *Мониторинг.* Основная задача мониторинга – непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или иных параметров за допустимые пределы. Главные проблемы – «пропуск» тревожной ситуации и инверсная задача «ложного» срабатывания. Сложность этих проблем в размытости симптомов тревожных ситуаций и необходимости учета временного контекста (например: контроль за работой электростанций СПРИНТ, помощь диспетчерам атомного реактора – REACTOR; контроль аварийных датчиков на химическом заводе – FALCON и др.).

• *Проектирование.* Проектирование состоит в подготовке спецификаций на создание объектов с заранее определенными свойствами. Под спецификацией понимается весь набор необходимых документов – чертеж, пояснительная записка и т. д. Основные проблемы здесь – получение четкого структурного описания знаний об объекте и проблема «следа». Для организации эффективного проектирования и в еще большей степени перепроектирования необходимо формировать не только сами проектные решения, но и мотивы их принятия. Таким образом, в задачах проектирования тесно связываются два основных процесса, выполняемых в рамках соответствующей ЭС, – *процесс вывода решения и процесс объяснения* (например: проектирование конфигураций ЭВМ; синтез электрических цепей – SYN и др.).

• *Прогнозирование.* Прогнозирование позволяет предсказывать последствия некоторых событий или явлений на основании анализа имеющихся данных. Прогнозирующие системы логически выводят вероятные следствия из заданных ситуаций. В прогнозирующей системе обычно используется параметрическая динамическая модель, в которой значения параметров «подгоняются» под заданную ситуацию. Выводимые из этой модели следствия составляют основу для прогнозов с вероятностными оценками (например: предсказание погоды – система WILLARD; оценки будущего урожая – PLANT; прогнозы в экономике – ECON и др.).

• *Планирование.* Под планированием понимается нахождение планов действий, относящихся к объектам, способным выполнять некоторые функции. В таких ЭС используются модели поведения реальных объектов с тем, чтобы логически вывести последствия планируемой деятельности (например: планирование поведения робота – STRIPS; планирование промышленных заказов – ISIS; планирование эксперимента – MOLGEN и др.).

• *Обучение.* Под обучением понимается использование компьютера для обучения какой-то дисциплине или предмету. Системы обучения диагностируют ошибки при изучении какой-либо дисциплины с помощью ЭВМ и подсказывают правильные решения. Они аккумулируют знания о гипотетическом «ученике» и его характерных ошибках, затем в работе они способны диагностировать слабости в познаниях обучаемых и находить соответствующие средства для их ликвидации. Кроме того, они планируют акт общения с учеником в зависимости от успехов последнего с целью передачи знаний (например: обучение языку программирования ЛИСП в системе «Учитель ЛИСПа»; система PROUST – обучение языку Паскаль и др.

• *Управление.* Под управлением понимается функция организованной системы, поддерживающая определенный режим деятельности. Такого рода ЭС осуществляют управление поведением сложных систем в соответствии с заданными спецификациями (например: помощь в управлении газовой котельной – GAS; управление системой календарного планирования Project Assistant и др.).

• *Поддержка принятия решений.* Поддержка принятия решения – это совокупность процедур, обеспечивающая лицо, принимающее решения, необходимой информацией и рекомендациями, облегчающими процесс принятия решения. Эти ЭС помогают специалистам выбрать и/или сформировать нужную альтернативу среди множества выборов при принятии ответственных решений (например: выбор стратегии выхода фирмы из кризисной ситуации – CRYSIS; помощь в выборе страховой компании или инвестора – CHOICE и др.).

2. Классификация по связи с реальным временем:

• *Статические ЭС* разрабатываются в предметных областях, в которых база знаний и интерпретируемые данные не меняются во времени. Они стабильны, например, диагностика неисправностей в автомобиле.

• *Квазидинамические ЭС* интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени (например,

микробиологические ЭС, в которых снимаются лабораторные измерения с технологического процесса один раз в 4-5 часов и анализируется динамика полученных показателей по отношению к предыдущему измерению).

• *Динамические ЭС* работают в сопряжении с датчиками объектов в режиме реального времени с непрерывной интерпретацией поступающих в систему данных (например: управление гибкими производственными комплексами, мониторинг в реанимационных палатах).

4.3.2. Стадии разработки экспертных систем

Стадия существования характеризует степень проработанности и отлаженности ЭС. Обычно выделяют следующие стадии: демонстрационный прототип; исследовательский прототип; действующий прототип; промышленная система; коммерческая система.

Демонстрационным прототипом называют ЭС, которая решает часть требуемых задач, демонстрируя жизнеспособность метода инженерии знаний. При наличии развитых инструментальных средств (ИС) для разработки демонстрационного прототипа требуется в среднем примерно один-два месяца, а при отсутствии – 12 – 18 месяцев. Демонстрационный прототип работает, имея в БЗ 50–100 правил. Развитие демонстрационного прототипа приводит к исследовательскому прототипу.

Исследовательским прототипом называют систему, которая решает все требуемые задачи, но неустойчива в работе и не полностью проверена. На доведение системы до стадии исследовательского прототипа уходит три – шесть месяцев. Исследовательский прототип обычно имеет в БЗ 200–500 правил, описывающих проблемную область.

Действующий прототип надежно решает все задачи, но для решения сложных задач может потребоваться чрезмерно много времени и (или) памяти. Для доведения системы до стадии действующего прототипа требуется 6–12 мес., при этом количество правил в БЗ увеличивается до 500–1000.

Экспертная система, достигшая *промышленной стадии,* обеспечивает высокое качество решений всех задач при минимуме времени и памяти. Обычно процесс преобразования действующего прототипа в промышленную систему состоит в расширении БЗ (до 1000–1500 правил) и переписывании программ с использованием более

эффективных ИС, например в перепрограммировании на языках низкого уровня. Для доведения ЭС от начала разработки до стадии промышленной системы требуется 1–1,5 года.

Обобщение задач, решаемых ЭС на стадии промышленной системы, позволяет перейти к стадии *коммерческой системы* – системы, пригодной не только для собственного использования, но и для продажи различным потребителям. Для доведения системы до коммерческой стадии требуется 1,5–3 года. При этом в БЗ системы 1500–3000 правил.

Трудности разработки экспертных систем. При разработке ЭС разработчиков поджидают различные трудности:

– глобальные, имеющие отношение ко всему процессу разработки или к нескольким этапам разработки;

– локальные, проявляющиеся в основном на некотором этапе разработки.

Глобальные трудности. Это общие проблемы, с которыми сталкивается коллектив, пытающийся применить технологию ЭС для решения своих задач. Источником глобальных трудностей являются, по крайней мере, следующие факторы: недостаток ресурсов; ограничения существующих ЭС; длительность разработки ЭС.

Недостаток ресурсов для создания ЭС выражается в нехватке человеческих, программных и аппаратных ресурсов. В связи с относительной новизной направления практически отсутствуют инженеры по знаниям, очень мало специалистов по разработке инструментальных средств (ИС) для систем ИИ и ЭС.

Ограничения программных ресурсов выражаются в слабом развитии ИС, что приводит к чрезмерной длительности и трудоемкости разработки ЭС.

Ограничения существующих ЭС можно разделить на ограничения, присущие существующим ИС, и на ограничения, присущие собственно ЭС. К последним ограничениям следует отнести следующие: крайне слабые возможности по работе с динамическими обьектами, требующими представления времени и (или) пространства; недостаточные возможности ЭС по генерации умозаключений, основанных на «здравом смысле»; ЭС плохо распознают ситуации, когда пользователь выходит за рамки их возможностей (что приводит к непредвиденным неудачам); как правило, ЭС не способны определить несогласованность (противоречивость) знаний, объяснить причину несогласованности и устранить ее. К ограничениям ИС можно отнести

следующее: они могут оказать очень малую помощь в процессе приобретения и корректировки знаний в наиболее трудоемком аспекте разработки ЭС; существующие отечественные инструментальные средства, как правило, не позволяют использовать смешанное представление (например, правила и фреймы).

Процесс разработки ЭС довольно длительный, и это является существенным ограничением при выборе данного способа реализации задачи.

Локальные трудности. Инженер по знаниям на разных этапах создания ЭС может столкнуться с локальными трудностями.

На *этапе идентификации* основные трудности возникают при решении следующих проблем:

1. Определение пригодности методов инженерии знаний для решения предлагаемой задачи.

При решении данной проблемы могут возникнуть следующие препятствия:

– задача слишком сложна и не может быть решена доступными ресурсами в пределах заданных ограничений (время, тип ЭВМ и т.п.);

– задача может быть решена ЭС, но пользователь не получит от ее решения существенной пользы;

– задача может быть решена, но либо требуется ввести слишком много правил и объектов (что чрезмерно затянет процесс разработки), либо результирующая система работает слишком медленно.

Чтобы выявить сложность задачи, инженер должен разработать небольшой прототип ЭС, работа которого позволит надежно ответить на вопрос о приемлемости задачи. В простых случаях ответ на этот вопрос можно получить, имея опыт разработки ЭС, при анализе описания задачи. Для выявления полезности ЭС следует до ее разработки ответить на следующий вопрос: если ЭС будет работать хорошо, то будет ли это давать тот эффект, которого хочет достичь пользователь?

Другими словами, надо понять, не окажется ли, что для удовлетворения потребностей пользователя надо решать еще какие-то задачи, не предусмотренные в данной ЭС. Для преодоления последнего из перечисленных препятствий необходимо ограничить задачу или область ее приложения. Надежным средством выявления наличия или отсутствия данной трудности является разработка прототипа.

2. Определение необходимых ресурсов.

При решении этой проблемы возникают следующие препятствия: срок разработки ЭС назначается на основании анализа, выполненного

инженером по знаниям, а не потребностей пользователей (интерактивная природа процесса приобретения знаний существенно ограничивает эффективность ускорения разработки ЭС за счет увеличения числа разработчиков); попытка разработать ЭС без инженеров по знаниям, а только с помощью программистов приводит либо к чрезмерному увеличению сроков разработки, либо к ее полной неудаче.

3. Выбор эксперта.

При работе с экспертом может оказаться, что процесс извлечения знаний протекает чрезвычайно неэффективно. Наиболее вероятной причиной этого обычно является неправильный выбор эксперта. К последнему следует предъявлять следующие требования: высокая компетентность в данной проблемной области; способность ясно, просто и доходчиво выражать свои идеи и методы; наличие у эксперта заинтересованности в разработке ЭС; эксперт должен верить в полезность использования вычислительной техники при решении данной задачи (чем больше эксперт знает о компьютере и программировании, тем лучше); эксперт должен хотеть и быть способен уделять разработке ЭС примерно половину своего рабочего времени в первые полгода разработки и примерно одну треть времени в последующее время.

На *этапе концептуализации* могут встретиться следующие трудности:

1) К работе подключено излишне много экспертов. Инженер по знаниям, общаясь со всеми, не имеет возможности глубоко исследовать данную область. Инженеру рекомендуется работать только с одним или двумя экспертами. После построения прототипа для его тестирования и модификации целесообразно подключать дополнительных экспертов.

2) Инженер по знаниям выявляет правила, наблюдая за тем, как эксперт решает конкретные задачи, что может привести к образованию специфических правил. Для преодоления этой трудности инженер должен пытаться выявить общие принципы, используемые экспертом. Часто оказывается полезным введение понятий высокого уровня, которые эксперт фактически использует, но не называет, так как они им не вербализованы (в научной литературе такие понятия не введены). Введение этих понятий может позволить заменить несколько специфических правил одним общим.

3) После нескольких месяцев взаимодействия с экспертом инженер по знаниям выделил несколько сотен правил и представил их

в выбранном языке. Однако начало проверки прототипа показывает, что многие фундаментальные понятия в приобретенных правилах отсутствуют (выявлены ошибки, допущенные на этапе концептуализации). Чтобы описанная выше ситуация не возникла, опытный инженер по знаниям осуществляет тестирование принятых решений в ходе всех этапов разработки, а не только по окончании этапа выполнения.

На *этапе формализации* основные проблемы относятся к выбору и использованию инструментальных средств. Можно выделить следующие трудности:

1) При использовании выбранного ИС инженеры по знаниям (разработчики) приходят к выводу, что в нем трудно представлять как понятия области экспертизы, так и управляющие структуры, необходимые для решения задачи. Выбор ИС и определение его пригодности для данного приложения весьма сложны и неформальны. Поэтому разработчики, как правило, могут оценить сделанный выбор только после разработки в нем небольшого прототипа. Если разработчики приходят к выводу, что выбранное инструментальное средство не подходит для данной проблемы, то необходимо либо выбрать другое ИС, либо разрабатывать новое (в этом случае практически дело сводится к откладыванию разработки ЭС на неопределенный срок).

2) Инженер по знаниям выбирает то ИС, которым он хорошо владеет, а не то, которое наиболее соответствует данной проблемной области. При этом, если ИС явно плохо подходит, инженер прилагает усилия к изменению исходной задачи (что может снизить или устранить практическую значимость создаваемой ЭС), чтобы она подходила под выбранное им ИС. Для устранения подобных затруднений руководитель проекта по разработке ЭС должен при выборе ИС проконсультироваться с несколькими инженерами по знаниям, чтобы убедиться в обоснованности сделанного выбора. Кроме того, сам инженер по знаниям должен осознавать наличие подобной ловушки и тщательно подходить к проблеме выбора инструментального средства.

3) В качестве ИС выбирается (на начальных этапах разработки ЭС) один из общецелевых языков программирования (например, Паскаль, Си, Фортран), что приводит к чрезмерному увеличению времени разработки ЭС или к полной неудаче всего проекта. Ошибочность подобного выбора состоит в том, что для выполнения таких наиболее трудоемких работ, как наполнение экспертной системы знаниями, тестирование и корректировка БЗ, общецелевые языки

плохо приспособлены. Более правильно разработать ЭС, используя инструментальные средства, ориентированные на разработку экспертных систем, и затем, если ЭС будет работать медленно, переписать ее, используя общецелевые языки программирования.

4) Выбранное ИС содержит ошибки, препятствующие использованию многих из его свойств. Для предотвращения указанной трудности рекомендуется: избегать ИС, не находившихся в употреблении; не использовать инструментальные средства, которые более не сопровождаются разработчиком; выбирать то ИС, которое сопровождается разработчиком и было успешно использовано в других случаях.

В ходе *этапа выполнения* могут встретиться такие трудности, как:

1. Эксперт теряет интерес к разработке ЭС. Время, которое он выделяет на работу с инженером по знаниям, постоянно сокращается. В подобных ситуациях рекомендуется: поддерживать интерес эксперта к работе, обеспечивая дружественную обстановку; объяснять, что для успеха необходимы регулярные контакты (не реже 2-3 раза в неделю); помнить, что эксперт не специалист в программировании и ЭС, и не требовать от эксперта того, что он не умеет делать; предоставить эксперту возможность непосредственного наблюдения за результатами предлагаемых им изменений БЗ.

2. Эксперт при корректировке БЗ прототипа не совсем понимает, как система интерпретирует правила. Чтобы избежать указанного затруднения, рекомендуется: использовать в правилах ту же терминологию, которую применяет эксперт; хранить в системе определения всех используемых понятий и иметь возможность выдавать их по требованию пользователя; если правила во внутреннем представлении «нечитаемы», то иметь средства для преобразования правил в вид, понятный пользователю.

3. Правила, введенные экспертом, в сложных ситуациях не позволяют прототипу ЭС получить качественные решения. Для преодоления данного затруднения рекомендуется: ориентировать эксперта на решение реальных, а не «игрушечных» задач; наблюдать за работой эксперта, стремясь выделить все типы возможных задач (подзадач); привлекать для оценки работы прототипа других экспертов.

4. Для построения ЭС было выбрано ИС без объяснительных способностей. После отработки ЭС разработчики пытаются дополнить ЭС этим механизмом, однако обычно сделать это чрезвычайно трудно. Чтобы не сталкиваться с подобными трудностями, разработчики

должны выбирать ИС с объяснительными способностями. Объяснения ускоряют отладку, тестирование, модификацию прототипа и обеспечивают удобство для конечного пользователя и эксперта.

5. В процессе разработки ЭС знания об области экспертизы переплетаются с общими знаниями о решении задач (проблемно-независимыми знаниями), что затрудняет (делает невозможным) проведение модификации и развитие ЭС. По сути дела, нарушается основной принцип разработки ЭС – отделение знаний эксперта от остальных знаний и представление знаний эксперта в явном виде.

На этапе тестирования и опытной эксплуатации могут встретиться следующие трудности:

1) Тестирование не выполняется в ходе разработки, а впервые осуществляется после этапа выполнения. При подобном подходе вероятность отрицательной оценки ЭС очень велика. Кроме того, весьма вероятно, что перепроектирование ЭС придется начать с ранних этапов (идентификации и концептуализации). Общая рекомендация при разработке экспертных систем состоит в проведении тестирования и оценки как можно раньше. Поэтому на начальных этапах разработчики ЭС должны составить методику оценки пригодности и полезности конечного продукта, что позволит по ходу разработки (а не в ее конце) оценивать состояние проекта и прогнозировать срок его успешного завершения.

2) Пользователи (эксперты) находят процесс взаимодействия с ЭС трудным и неудобным: им непонятны сообщения системы, время реакции ее велико, возможности неочевидны и трудноиспользуемы и т. п. Для устранения подобных ситуаций целесообразно освободить пользователей от любой технической работы, непосредственно не связанной с областью экспертизы (работа с операционной системой, сопровождение словарей, взаимодействие с «недружественным» редактором). Недопустимо использовать аббревиатуры.

3) Если правил больше нескольких сотен, возникают затруднения с дальнейшей корректировкой и добавлением их, так как внесение одних изменений порождает цепь других, процесс перестает быть управляемым и кажется бесконечным. Выход из подобных ситуаций – разработка системы сопровождения, запоминающей, какие задачи решены, какие получены результаты, какие правила использованы, когда и какие изменения внесены и т. п.

4.3.3. Пример разработки экспертной системы

Рассмотрим процесс разработки ЭС на примере «Экспертной системы для определения уровня яркости электросветосигнальных огней взлетно-посадочных полос аэродрома, а также значения вероятности взлета и посадки самолетов при плохих метеорологических условиях» [4].

Определим состав задач, решаемых разработанной экспертной системой (рис. 4.6):

1. Определение значения величины яркости электросветосигнальных огней (ЭССО) аэропорта для всепогодного взлета и посадки самолетов в сложных метеорологических условиях.

2. Определение значения вероятности взлета или посадки самолетов при заданных метеоусловиях.

3. Отображение зависимости этих величин от погодных условий в виде графиков.

Для решения этих задач экспертной системе необходимо:

– проанализировать основные метеорологические элементы и явления погоды, определяющие условия взлета и посадки самолетов;

– установить диалог с экипажем самолета на предмет видимости взлетно-посадочной полосы и уровня яркости ЭССО;

– динамически отслеживать изменения метеоусловий и корректировать принятые решения.

Проанализировав метеорологические элементы, явления погоды и согласовав их с экипажем самолета, экспертная система формирует следующие результаты:

– значение величины яркости ЭССО;

– указания по проведению посадки/взлета (например, уход на второй круг, отказ от посадки и т.д.);

– значение величины вероятности взлета и посадки самолетов в заданных метеорологических условиях.

При создании базы знаний и механизма логического вывода используется байесовский подход.

Диалог экспертной системы с экипажем самолета. Ввод данных в ЭС предлагаем осуществлять в диалоге экспертной системы с диспетчером и с экипажем. Числовые значения основных метеорологических элементов поступают непосредственно от приборов измерения. Диалог ЭС с диспетчером будем осуществлять с помощью письменной речи. Это необходимо для анализа текущих метеорологических условий.

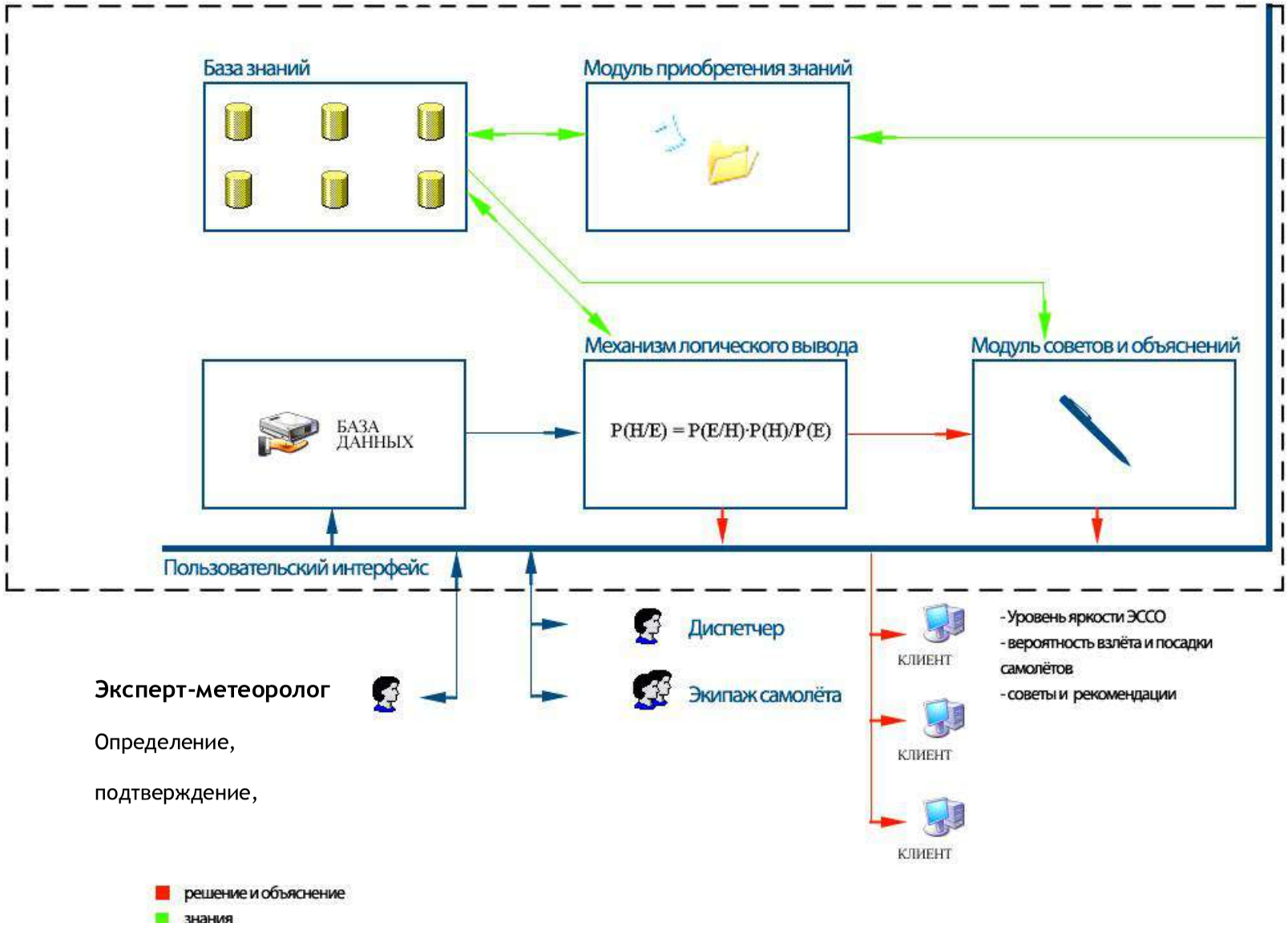


Рис. 4.6. Структура экспертной системы

Диалог ЭС с экипажем необходим для определения значения нормального уровня яркости ЭССО. В случае тусклого освещения взлетно-посадочной полосы уровень яркости увеличивается, в обратном случае – уменьшается. В качестве средства общения экспертной системы с экипажем автор считает целесообразным использовать устную речь. Для диалога предлагается использовать в экспертной системе речевую технологию общения.

Эта технология подразумевает под собой разработку системы речевого общения как составной части экспертной системы. Основой для разработки современных систем речевого общения являются лингвоакустическая и информационная теории речеобразования и восприятия речи. Связь с экипажем ЭС устанавливает по системе связи.

Преимущества системы речевого общения заключаются в следующем:

– удобство, простота и естественность процедуры общения ЭС с экипажем самолета;

– отсутствие ручных манипуляций, что для пилота самолета очень важно;

– большая скорость ввода информации.

Предлагаем систему речевого общения строить на базе специализированных речевых процессоров двух основных типов – анализаторов и синтезаторов (рис. 4.7, 4.8).

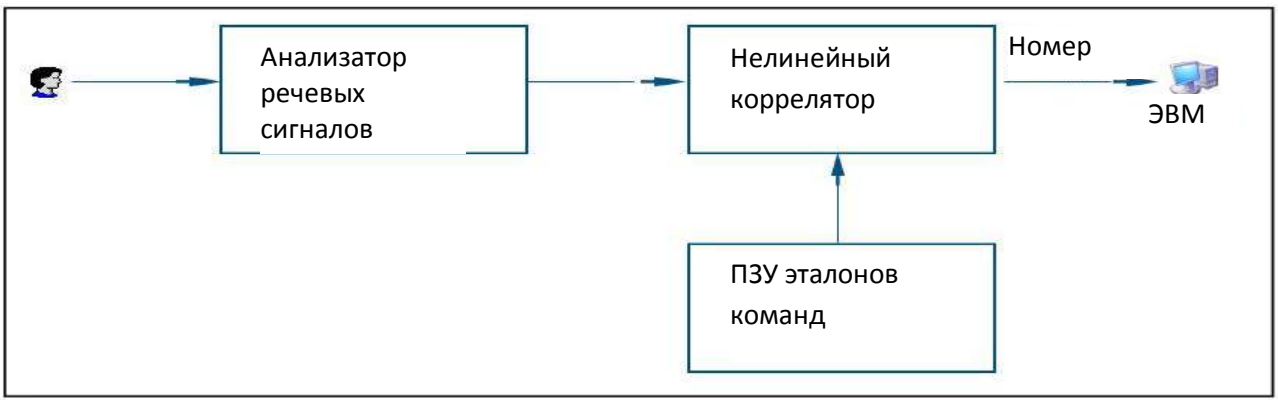


Рис. 4.7. Схема анализатора речевых сообщений

Анализатор предназначен для преобразования речевых сигналов с микрофона в последовательность цифровых кодов с обязательным сохранением передачи смыслового компонента речи. В анализаторе речевых сообщений осуществляется сжатие информационного потока за счет введения операции распознавания смысловых элементов речи.

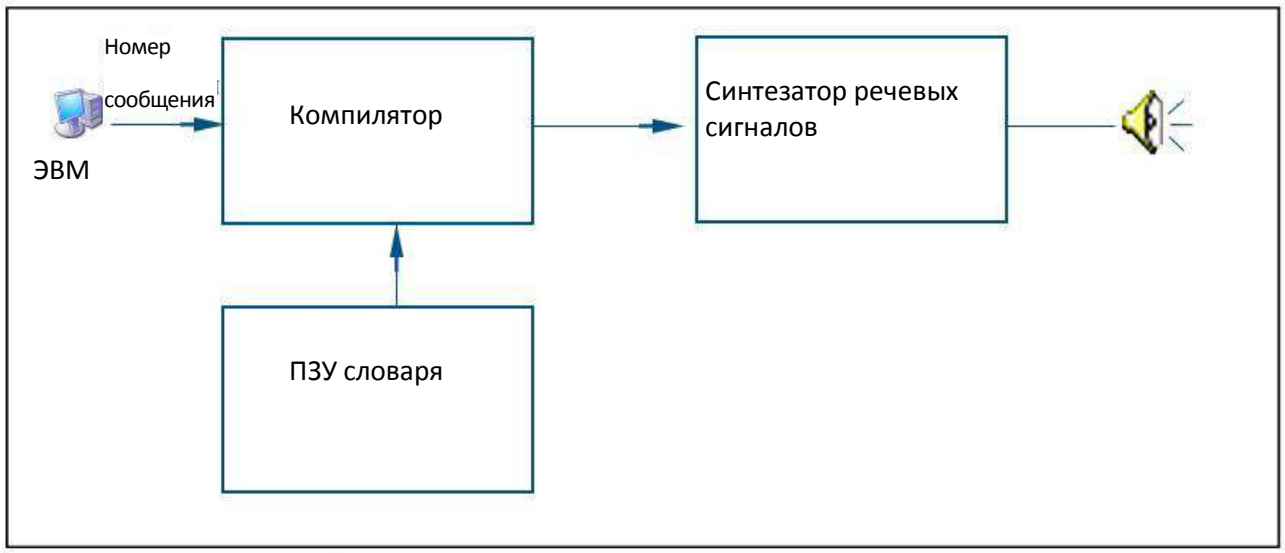


Рис. 4.8. Схема синтезатора речевых сообщений

Синтезатор предназначен для преобразования кодовой последовательности, поступающей от ЭВМ, в непрерывный речевой сигнал.

Формирование базы знаний экспертной системы и механизма логического вывода. База знаний экспертной системы реализует функции представления знаний и управления ими. Для формализации знаний определим ключевые понятия, отношения и характеристики, которыми оперирует эксперт в процессе решения задачи. Для принятия решения эксперту необходимо проанализировать состояние атмосферы и процессы, происходящие в ней. Определим основные метеорологические элементы и явления погоды, влияющие на условия взлета и посадки самолетов: атмосферное давление, температура воздуха, видимость, влажность воздуха, облака, осадки, ветер, туман, метель, пыльная буря, град, шквал.

Таким образом, в результате анализа предметной области очевидно, что информация, участвующая в принятии решений экспертной системой, большого объема, разнопланова и разнородна. Часть данных – это точные числовые значения, часть – словесные описания, наблюдения. Источниками этой информации являются приборы, наблюдения метеорологов, данные самолетов – разведчиков погоды и т.д. Чтобы работать с таким количеством информации, необходимо понимать её структуру.

Основной целью создания ЭС является простота и удобство представления знаний. Для организации знаний в разработанной автором экспертной системе предлагаем использовать продукционные модели.

При формировании механизма логического вывода будем использовать байесовский подход. Этот метод основан на применении теории вероятностей, он позволяет вычислить относительное правдоподобие конкурирующих гипотез, исходя из значений вероятности свидетельства.

Модуль приобретения знаний. Модуль приобретения знаний является важной частью ЭС. С его помощью знания в БЗ пополняются, уточняются, корректируются и обновляются. Все операции со знаниями выполняет эксперт. В нашем случае это специалист метеорологической службы. Для корректной работы разработанной ЭС необходим опытный эксперт в этой области.

Для работы с базой знаний автор разработал модуль приобретения знаний, предоставляющий возможность дистанционного манипулирования знаниями.

Эта составляющая экспертной системы позволяет автоматизировать процесс управления знаниями в БЗ и решить кадровый вопрос.

Сбор, корректировку и обновление знаний в базе знаний экспертной системы предлагаем осуществлять с помощью сервера. Опишем принципы организации модуля приобретения знаний. Специалист-эксперт, ответственный за сбор, отправку и пополнение знаний, на своей локальной машине, подключенной к сети Интернет, запускает браузер и в строке адреса вводит адрес web-сервера ЭС.

На сервере ЭС устанавливаем специализированное программное обеспечение: программный web-сервер Apache, работающий по принципам клиент-серверной технологии; .php – интерпретатор; базу данных MySQL; PhpMyAdmin – систему управления базами данных MySQL.

Сервер на запрос клиентской машины отсылает ей код для формирования в браузере клиента форм ввода статистической информации для последующей отсылки и обработки информации на веб-сервере.

После отображения в браузере клиента форм ввода информации специалист-эксперт заносит требуемые сведения и нажимает кнопку «Добавить запись». Информация уходит на сервер, проверяет права удалённого пользователя с помощью специального скрипта

do_authuser.php на предмет того, предоставлены ли данному пользователю полномочия по работе с базой данных MySQL, имеет ли он право вносить свою информацию, видоизменять базу данных, выполнять запросы и т.п.

В случае успешной идентификации и аутентификации клиента ЭС (если присланные логин и пароль имеются в соответствующей данному пользователю строке таблицы базы MySQL в кэшированном, зашифрованном виде) присланная информация заносится в соответствующие таблицы и разделы определённого сегмента базы данных MySQL посредством специального скрипта do_addrecord.php. Далее осуществляется анализ собранной данной системой информации. Модуль сбора знаний ЭС является по своей сути средством мониторинга.

Для анализа собранной информации может использоваться язык запросов к базе данных MySQL или внешняя статистическая обработка программой SPSS. Во втором случае база данных должна быть отконвертирована в соответствующий формат. Операция конвертации, а также все манипуляции с информационными базами осуществляются посредством системы управления базами данных PHPMyAdmin.

4.3.4. Особенности экспертных систем

Экспертная система – наиболее известный и распространенный вид интеллектуальных систем. Хотя этот термин употребляется весьма широко, но его точного определения пока нет. Можно лишь указать ряд особенностей, которые присущи только экспертным системам.

Первая особенность экспертных систем состоит в том, что они предназначены для пользователей, сфера деятельности которых далека от искусственного интеллекта, программирования, математики, логики. Для таких пользователей экспертная система выступает как некая система, помогающая им в повседневной работе. Общение с экспертными системами, работа с ними должны быть так же просты, как просты, например, управление телевизором, стиральной машиной или автомобилем.

Специфическим именно для экспертных систем является наличие блока объяснений. Дело в том, что после консультации с экспертной системой решение, полученное пользователем, может показаться ему

либо неприемлемым, либо не лучшим. Происходит это потому, что часть рассуждений экспертная система делает самостоятельно, используя свой сценарий и те знания, которыми она располагает. Пользователю же кажется, что в логике получения решения имеются «провалы», «перескоки», не обоснованные шагами диалога.

Есть и еще один класс систем, не имеющих собственного названия и поэтому часто называемых экспертными. В отличие от классических экспертных систем они рассчитаны не на пользователя, являющегося новичком или средним специалистом в некоторой области деятельности, а на самих экспертов-профессионалов. Для таких специалистов нужна не консультирующая или советующая система, а система, способная помочь им в научной работе. Системы такого рода называют системами автоматизации научных исследований.

ЭС обладает способностями к накоплению знаний, выдаче рекомендаций и объяснению полученных результатов, возможностями модификации правил, подсказки пропущенных экспертом условий, управления целью или данными. ЭС отличают следующие характеристики:

- интеллектуальность;
- простота общения с компьютером;
- возможность наращивания модулей;
- интеграция неоднородных данных;
- способность разрешения многокритериальных задач при учете предпочтений лиц, принимающих решения (ЛПР);
- работа в реальном времени;
- документальность;
- конфиденциальность;
- унифицированная форма знаний;
- независимость механизма логического вывода;
- способность объяснения результатов.

Выделяют следующие характеристики ЭС: назначение, проблемная область, глубина анализа проблемной области, тип используемых методов и знаний, класс системы, стадия существования, инструментальные средства.

Назначение определяется совокупностью параметров: цель создания экспертной системы – для обучения специалистов, для решения задач, для автоматизации рутинных работ, для тиражирования знаний экспертов и т.п.; основной пользователь – неспециалист в области экспертизы, специалист, учащийся.

Проблемная область может быть определена совокупностью параметров предметной области и задач, решаемых в ней. Каждый из параметров можно рассматривать с точки зрения как конечного пользователя, так и разработчика экспертной системы.

С точки зрения пользователя предметную область можно характеризовать ее описанием в терминах пользователя, включающим наименование области, перечень и взаимоотношения подобластей и т.п., а задачи, решаемые существующими экспертными системами, – их типом. Обычно выделяют следующие типы задач:

- интерпретация символов или сигналов – составление смыслового описания по входным данным;
- диагностика – определение неисправностей;
- предсказание – определение последствий наблюдаемых ситуаций;
- конструирование – разработка объекта с заданными свойствами при соблюдении установленных ограничений;
- планирование – определение последовательности действий, приводящих к желаемому состоянию объекта;
- слежение – наблюдение за изменяющимся состоянием объекта и сравнение его показателей с установленными или желаемыми;
- управление – воздействие на объект для достижения желаемого поведения.

4.3.5. Отличие ЭС от традиционных программ

Еще один способ определить ЭС – это сравнить их с обычными программами. Главное различие состоит в том, что ЭС манипулируют знаниями, тогда как обычные программы манипулируют данными. Эти различия описаны фирмой Teknowledge, которая занимается производством коммерческих экспертных систем (табл. 4.1) [19].

Таблица 4.1

Различия экспертных коммерческих систем

Обработка данных	Инженерия знаний
Представление и использование данных Алгоритмы Повторный прогон Эффективная обработка больших баз данных	Представление и использование знаний Эвристики Процесс логического вывода Эффективная обработка баз знаний

Специалисты в области ИИ имеют несколько более узкое (и более сложное) представление о том, что такое ЭС. Под экспертной системой понимается программа для ЭВМ, которая обладает рядом свойств (рис. 4.9).

Рис. 4.9. Особенности ЭС, отличающие ее от обычных программ

Компетентность ЭС. Экспертная система должна демонстрировать компетентность, т.е. достигать в конкретной предметной области того же уровня профессионализма, что и эксперты-люди. Но просто уметь находить хорошие решения еще недостаточно. Настоящие эксперты не только находят верные решения, но часто находят их очень быстро, тогда как новичкам для нахождения тех же решений, как правило, требуется намного больше времени. Следовательно, ЭС должна быть умелой – она должна применять знания для получения решений эффективно и быстро, используя приемы и ухищрения, какие применяют эксперты-люди, чтобы избежать громоздких или ненужных вычислений.

Для того чтобы по-настоящему подражать поведению эксперта-человека, ЭС должна обладать ***робастностью***. Это подразумевает не только глубокое, но и достаточно широкое понимание предмета.

А этого можно достичь, используя общие знания и методы нахождения решений проблем, чтобы уметь рассуждать, исходя из фундаментальных принципов в случае некорректных данных или неполных наборов правил. Это один из наименее разработанных методов в современных ЭС, но именно им успешно пользуются эксперты-люди.

Символьные рассуждения в ЭС. Эксперты, решая какие-то задачи (особенно такого типа, для решения которых применяются ЭС), обходятся без решения систем уравнений или других трудоемких математических вычислений. Вместо этого они с помощью символов представляют понятия предметной области и применяют различные стратегии и эвристики в процессе манипулирования этими понятиями. В ЭС знания тоже представляются в символьном виде, т. е. наборами символов, соответствующих понятиям предметной области. На языке ИИ *символ* – это строка знаков, соответствующая содержанию некоторого понятия реального мира.

Примеры символов:

- продукт;
- ответчик;
- 0.8.

Эти символы объединяют, чтобы выразить отношения между ними. Когда эти отношения представлены в программе ИИ, они называются ***символьными структурами***.

Примеры символьных структур:

(ДЕФЕКТНЫЙ продукт) (ВЫПУЩЕННЫЙ ответчиком продукт) (РАВНО (ОТВЕТСТВЕННОСТЬ ответчик) 0.8).

Эти структуры можно интерпретировать следующим образом: «продукт является дефектным», «продукт был выпущен в продажу ответчиком» и «ответственность ответчика равна 0.8».

При решении задачи ЭС вместо выполнения стандартных математических вычислений манипулирует этими символами. Нельзя сказать, что ЭС вообще не производит математических расчетов – она их делает, но в основном она приспособлена для манипулирования символами. Вследствие подобного подхода *представление знаний* – выбор, форма и интерпретация используемых символов – становится очень важным.

Кроме того, эксперты могут получить задачу, сформулированную неким произвольным образом, и преобразовать ее к тому виду, который в наибольшей степени соответствует быстрому получению решения или гарантирует его максимальную эффективность. Эта спо-

собность переформулирования задачи – как раз то свойство, которое должно быть присуще ЭС для того, чтобы приблизить их мастерство к уровню экспертов-людей. К сожалению, большинство существующих в настоящее время ЭС не обладают этим умением [19].

Глубина ЭС. Экспертная система должна иметь глубокие знания; это значит, что она способна работать эффективно в узкой предметной области, содержащей трудные, нетривиальные задачи. Поэтому правила в ЭС с необходимостью должны быть сложными либо в смысле сложности каждого правила, либо в смысле их обилия. Экспертные системы, как правило, работают с предметными областями реального мира, а не с тем, что специалисты в области ИИ называют игрушечными предметными областями. В предметной области реального мира тот, кто решает задачу, применяет фактическую информацию к практической проблеме и находит решения, которые являются ценными с точки зрения некоторого критерия, определяющего соотношение стоимости и эффективности. В игрушечной предметной области либо задача подвергается чрезвычайному упрощению, либо производится нереалистическая адаптация некоторой сложной проблемы реального мира. Тот, кто решает такую проблему, обрабатывает искусственную информацию, которая в целях облегчения решения упрощена и порождает решения, имеющие чисто теоретический интерес.

В тех случаях, когда по отношению к сложной задаче или данным о ней сделаны существенные упрощения, полученное решение может оказаться неприменимым в масштабах, которые характерны для реальной проблемы. Рекомендации, методы представления знаний, организация знаний, необходимые для применения методов решения задач к этим знаниям, часто связаны с объемом и сложностью пространства поиска, т. е. множества возможных промежуточных и окончательных решений задачи. Если проблема сверхупрощена или нереалистична, то размерность пространства поиска будет, скорее всего, резко уменьшена, и не возникнет проблем с быстродействием и эффективностью, столь характерных для реальных задач. Эта проблема размерности возникает столь естественно и неуловимо, что даже искушенные в ИИ специалисты могут не оценить ее истинные масштабы.

Самосознание ЭС. Экспертные системы имеют знания, позволяющие им рассуждать об их собственных действиях, и структуру, упрощающую такие рассуждения. Например, если ЭС основана на

правилах, то ей легко просмотреть цепочки выводов, которые она порождает, чтобы прийти к решению задачи. Если заданы еще и специальные правила, из которых ясно, что можно сделать с этими цепочками выводов, то можно использовать эти знания для проверки точности, устойчивости и правдоподобия решений задачи и даже построить доводы, оправдывающие или объясняющие процесс рассуждения. Это знание системы о том, как она рассуждает, называется ***метазнанием***, что означает всего лишь знания о знаниях.

У большинства ныне существующих ЭС есть так называемый ***механизм объяснения***. Это знания, необходимые для объяснения того, каким образом система пришла к данным решениям. Большинство этих объяснений включают демонстрацию цепочек выводов и доводов, объясняющих, на каком основании было применено каждое правило в цепочке. Возможность проверять собственные процессы рассуждения и объяснять свои действия – это одно из самых новаторских и важных свойств ЭС. Но почему это свойство так важно?

Самосознание так важно для ЭС потому, что:

- пользователи начинают больше доверять результатам, испытывать бóльшую уверенность в системе;
- ускоряется развитие системы, так как систему легче отлаживать;
- предположения, положенные в основу работы системы, становятся явными, а не подразумеваемыми;
- легче предсказывать и выявлять влияние изменений на работу системы.

Умение объяснить – это всего лишь один из аспектов самосознания. В будущем самосознание позволит ЭС делать даже больше. Они сами смогут создавать обоснования отдельных правил путем рассуждения, исходящего из основных принципов. Они будут приспосабливать свои объяснения к требованиям пользователя. Они смогут изменять собственную внутреннюю структуру путем коррекции правил, реорганизации базы знаний и *реконфигурации системы*.

Первый шаг в этом направлении – выделить метазнания и сделать их явными, точно так же как знания о предметной области выделены и сделаны явными. Ниже приведен пример метазнания – знания о том, как использовать предметные знания.

ЕСЛИ: к данной ситуации применимо несколько правил,

ТО: использовать сначала правила, предложенные экспертами, прежде чем прибегнуть к правилам, предложенным новичками.

Это метаправило говорит ЭС, каким образом она должна выбирать те правила, которые надо выполнить. Специалисты по ИИ еще только начинают экспериментировать с формами представления метазнаний и их организацией в ЭС.

Ошибки экспертных систем. Существует еще одно очень важное отличие ЭС от традиционных программ. Традиционные программы разрабатываются таким образом, чтобы каждый раз порождать правильный результат, но ЭС заведомо создаются так, чтобы вести себя как эксперты, которые, как правило, дают правильные ответы, но иногда способны ошибаться.

На первый взгляд кажется, что в этом отношении традиционные программы имеют явное преимущество. Однако это преимущество мнимое. Традиционные программы для решения сложных задач, напоминающих те, которые подходят для ЭС, тоже могут делать ошибки. Но их ошибки чрезвычайно трудно исправлять, поскольку стратегии, эвристики и принципы, лежащие в основе этих программ, не сформулированы явно в их тексте. Следовательно, эти ошибки нелегко определить и исправить. Подобно людям, ЭС могут ошибаться. Но в отличие от обычных программ, они имеют потенциальную способность учиться на своих ошибках. С помощью компетентных пользователей можно заставить экспертные системы совершенствовать свое умение решать задачи в ходе практической работы.

4.4. Нейронные сети. Биологический нейрон

Нервная система и мозг человека состоят из нейронов, соединенных между собой нервными волокнами. Нервные волокна способны передавать электрические импульсы между нейронами. Все процессы передачи раздражений от кожи, ушей и глаз к мозгу, процессы мышления и управления действиями – все это реализовано в живом организме как передача электрических импульсов между нейронами.

Нейрон (нервная клетка) является особой биологической клеткой, которая обрабатывает информацию (рис. 4.10). Он состоит из тела (cell body), или сомы (soma), и отростков нервных волокон двух типов – дендритов (dendrites), по которым принимаются импульсы, и единственного аксона (axon), по которому нейрон может передавать импульс. Тело нейрона включает ядро (nucleus), которое содержит информацию о наследственных свойствах, и плазму, обладающую молекулярными средствами для производства необходимых нейрону

материалов. Нейрон получает сигналы (импульсы) от аксонов других нейронов через дендриты (приемники) и передает сигналы, сгенерированные телом клетки, вдоль своего аксона (передатчика), который в конце разветвляется на волокна (strands). На окончаниях этих волокон находятся специальные образования – синапсы (synapses), которые влияют на величину импульсов.

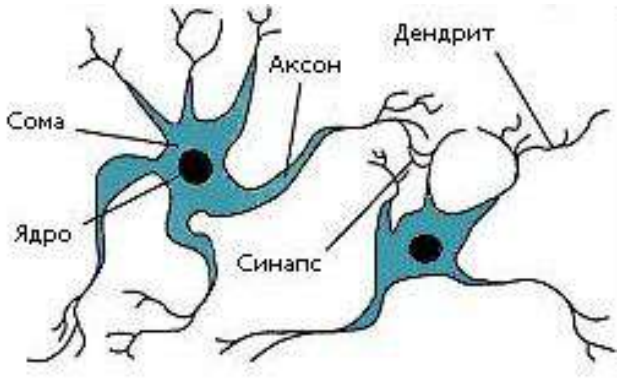


Рис. 4.10. Взаимосвязь биологических нейронов

Синапс является элементарной структурой и функциональным узлом между двумя нейронами (волокно аксона одного нейрона и дендрит другого). Когда импульс достигает синаптического окончания, высвобождаются химические вещества, называемые нейротрансмиттерами. Нейротрансмиттеры диффундируют через синаптическую щель, возбуждая или затормаживая, в зависимости от типа синапса, способность нейрона-приемника генерировать электрические импульсы. Результативность передачи импульса синапсом может настраиваться проходящими через него сигналами так, что синапсы могут обучаться в зависимости от активности процессов, в которых они участвуют. Эта зависимость от предыстории действует как память, которая, возможно, ответственна за память человека. Важно отметить, что веса синапсов могут изменяться со временем, а значит, меняется и поведение соответствующих нейронов.

Кора головного мозга человека содержит около 1011 нейронов и представляет собой протяженную поверхность толщиной от 2 до 3 мм с площадью около 2200 см^2. Каждый нейрон связан с 103-104 другими нейронами. В целом мозг человека содержит приблизительно от 1014 до 1015 взаимосвязей.

Нейроны взаимодействуют короткими сериями импульсов продолжительностью, как правило, несколько миллисекунд. Сообщение передается посредством частотно-импульсной модуляции. Частота

может изменяться от нескольких единиц до сотен герц, что в миллион раз медленнее, чем быстродействующие переключательные электронные схемы. Тем не менее сложные задачи распознавания человек решает за несколько сотен миллисекунд. Эти решения контролируются сетью нейронов, которые имеют скорость выполнения операций всего несколько миллисекунд. Это означает, что вычисления требуют не более 100 последовательных стадий. Другими словами, для таких сложных задач мозг «запускает» параллельные программы, содержащие около 100 шагов. Рассуждая аналогичным образом, можно обнаружить, что количество информации, посылаемое от одного нейрона другому, должно быть очень малым (несколько бит). Отсюда следует, что основная информация не передается непосредственно, а захватывается и распределяется в связях между нейронами [34].

Искусственный нейрон. Первая формальная модель искусственного нейрона была предложена в 1943 году известным американским физиологом Уорреном Мак-Каллоком и его учеником Вальтером Питтсом. Тогда же ими было сформулировано фундаментальное утверждение о том, что любая функция нервной системы, которая может быть логически описана с помощью конечного числа слов, может быть реализована сетью искусственных нейронов.

Модель искусственного нейрона представлена на рис. 4.11.

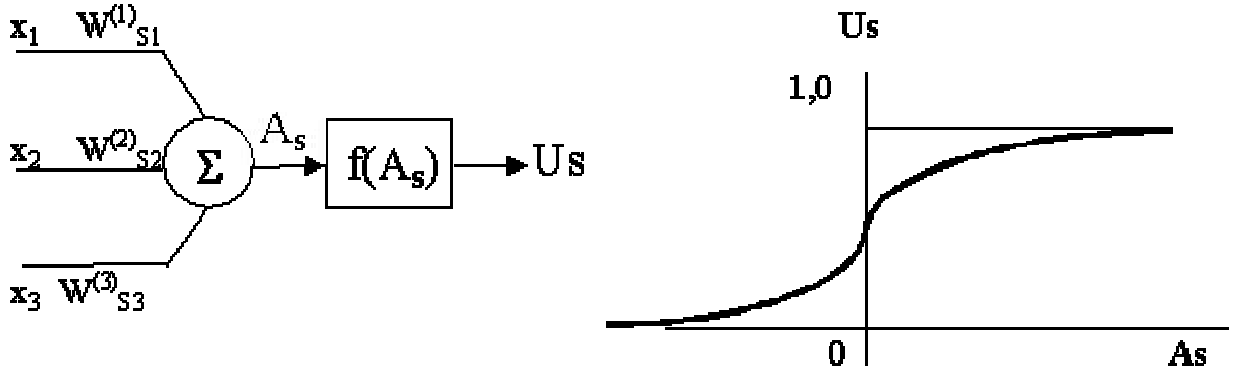


Рис. 4.11. Модель нейрона

Нейрон получает входные сигналы посредством входных связей (синапсов) с определенными весовыми коэффициентами Wi. Объект-сумматор (или net в некоторых литературных источниках) определяет взвешенную сумму входных величин As. F(As) – нелинейная функция, выбираемая в значительной степени произвольно.

Искусственные нейронные сети. В 50-60-е годы XX века группой ученых были созданы первые искусственные нейронные сети (ИНС) путем объединения биологических и физиологических подходов в процессе их моделирования. Реализованные первоначально как электронные сети, позднее они были перенесены и адаптированы в компьютерных средах моделирования, где активно исследуются и сегодня.

В этот период активно работают многие выдающиеся ученые в области ИНС: М. Минский, Ф. Розенблатт, Б. Уидроу и др. Ими изучаются и разрабатываются ИНС, состоящие из одного слоя искусственных нейронов с обучением, называемые перцептронами (персептронами). Их широкое применение позволило получить качественно новые результаты при решении таких задач, как предсказание погоды, анализ электрокардиограмм, искусственное зрение. Несколько наиболее активных ученых, таких как Т. Кохонен, С. Гроссберг, Дж. Андерсон, развили и продолжили исследования в 70 – 80-е годы.

Сегодня существует много практических примеров, демонстрирующих впечатляющие возможности ИНС: их научили превращать текст в фонетическое представление, которое затем превращается в речь; другие ИНС могут распознавать рукописные буквы, сжимать изображения и т.д.

Под ***искусственными нейронными сетями*** (ИНС) подразумеваются вычислительные структуры, которые моделируют простые биологические процессы, обычно ассоциируемые с процессами человеческого мозга. Они представляют собой распределенные и параллельные системы, способные к адаптивному обучению путем анализа положительных и отрицательных воздействий. Элементарным преобразователем в данных сетях является **искусственный нейрон,** или просто **нейрон**, названный так по аналогии с биологическим прототипом.

Искусственные нейронные сети строятся по принципам организации и функционирования их биологических аналогов. Они способны решать широкий круг задач распознавания образов, идентификации, прогнозирования, оптимизации, управления сложными объектами. Дальнейшее повышение производительности компьютеров все в большой мере связывают с ИНС, в частности с нейрокомпьютерами (НК), основу которых составляет искусственная нейронная сеть.

Нейронная сеть представляет собой совокупность нейроподобных элементов, определенным образом соединенных друг с другом

и с внешней средой с помощью связей, определяемых весовыми коэффициентами. В зависимости от функций, выполняемых нейронами в сети, можно выделить три их типа:

– входные нейроны, на которые подается вектор, кодирующий входное воздействие или образ внешней среды; в них обычно не осуществляется вычислительных процедур, а информация передается со входа на выход путем изменения их активации;

– выходные нейроны, выходные значения которых представляют выходы нейронной сети; преобразования в них осуществляются по выражениям:

$$S = \sum_{i=1}^{n} w_i x_i + b \ ; \tag{4.1}$$

$$y = f(s), \tag{4.2}$$

где w_i – вес (weight) синапса, $i = 1...n$;

b – значение смещения (bias);

s – результат суммирования (sum);

x_i – компонент входного вектора (входной сигнал);

$x_i = 1...n$; y – выходной сигнал нейрона;

n – число входов нейрона;

f – нелинейное преобразование (функция активации);

– промежуточные нейроны, составляющие основу нейронных сетей, преобразования в которых выполняются также по выражениям (4.1) и (4.2).

На входной сигнал *(s)* нелинейный преобразователь отвечает выходным сигналом *f(s)*, который представляет собой выход *y* нейрона.

Примеры активационных функций представлены в табл. 4.2 и на рис. 4.12.

Одной из наиболее распространенных является нелинейная функция активации с насыщением, так называемая логистическая функция, или сигмоид (функция *S*-образного вида): $f(s) = 1 / (1+e^{-as})$.

При уменьшении a сигмоид становится более пологим, в пределе при $a = 0$ вырождаясь в горизонтальную линию на уровне 0,5, при увеличении a сигмоид приближается к виду функции единичного скачка с порогом 0. Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне (0, 1). Одно из ценных

свойств сигмоидальной функции – простое выражение для ее производной, что важно при реализации процесса обучения нейронной сети: $f'(s) = a\,f(s)[1-f(s)]$.

Таблица 4.2

Функции активации нейронов

Название	Формула	Область значений
Линейная	$f(s)=k\,s$	$(-\infty, \infty)$
Полулинейная	$f(s)=\begin{cases} k\,s, & s>0, \\ 0, & s\le 0 \end{cases}$	$(0, \infty)$
Логистическая (сигмоидальная)	$f(s)=\dfrac{1}{1+e^{-as}}$	$(0, 1)$
Гиперболический тангенс (сигмоидальная)	$f(s)=\dfrac{e^{as}-e^{-as}}{e^{as}+e^{-as}}$	$(-1, 1)$
Экспоненциальная	$f(s)=e^{-as}$	$(0, \infty)$
Синусоидальная	$f(s)=\sin(s)$	$(-1, 1)$
Сигмоидальная (рациональная)	$f(s)=\dfrac{s}{a+\lvert s\rvert}$	$(-1, 1)$
Шаговая (линейная с насыщением)	$f(s)=\begin{cases} -1, & s\le -1, \\ s, & -1<s<1, \\ 1, & s\ge 1 \end{cases}$	$(-1, 1)$
Пороговая	$f(s)=\begin{cases} 0, & s<0, \\ 1, & s\ge 0 \end{cases}$	$(0, 1)$
Модульная	$f(s)=\lvert s\rvert$	$(0, \infty)$
Знаковая (сигнатурная)	$f(s)=\begin{cases} 1, & s>0, \\ -1, & s\le 0 \end{cases}$	$(-1, 1)$
Квадратичная	$f(s)=s^2$	$(0, \infty)$

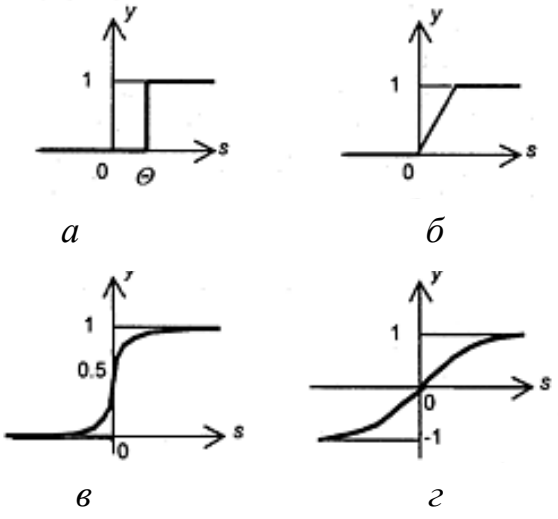


Рис. 4.12. Примеры активационных функций:
а – функция единичного скачка; *б* – линейный порог (гистерезис);
в – сигмоид (логистическая функция); *г* – сигмоид (гиперболический тангенс)

Следует отметить, что сигмоидальная функция дифференцируема на всей оси абсцисс, что используется в некоторых алгоритмах обучения. Кроме того, она обладает свойством усиливать слабые сигналы лучше, чем сильные, и предотвращает насыщение от сильных сигналов, так как они соответствуют областям аргументов, где сигмоид имеет пологий наклон [30].

Биологический нейрон и его математическая модель

Искусственный нейрон (математический нейрон Маккалока – Питтса, формальный нейрон) – узел искусственной нейронной сети, являющийся упрощённой моделью естественного нейрона. Математически искусственный нейрон обычно представляют как некоторую нелинейную функцию от единственного аргумента – линейной комбинации всех входных сигналов. Данную функцию называют функцией активации или функцией срабатывания, передаточной функцией. Полученный результат посылается на единственный выход. Такие искусственные нейроны объединяют в сети – соединяют выходы одних нейронов с входами других. Искусственные нейроны и сети являются основными элементами идеального нейрокомпьютера.

Математически нейрон представляет собой взвешенный сумматор, единственный выход которого определяется через его входы и матрицу весов следующим образом:

$$u = \sum_{i=1}^{n} w_i x_i + w_0 x_0 ,$$

где $y = f(u)$;

x_i и w_i – соответственно сигналы на входах нейрона и веса входов;

функция u – индуцированное локальное поле;

$f(u)$ – передаточная функция.

Возможные значения сигналов на входах нейрона считают заданными в интервале [0, 1]. Они могут быть либо дискретными (0 или 1), либо аналоговыми. Дополнительный вход x_0 и соответствующий ему вес w_0 используются для инициализации нейрона. Под инициализацией подразумевается смещение активационной функции нейрона по горизонтальной оси, т.е. формирование порога чувствительности нейрона [5]. Кроме того, иногда к выходу нейрона специально добавляют некую случайную величину, называемую сдвигом. Сдвиг можно рассматривать как сигнал на дополнительном, всегда нагруженном синапсе.

Множество математических моделей нейрона может быть построено на базе простой концепции строения нейрона. На рис. 4.13 показана наиболее общая схема. Так называемая суммирующая функция объединяет все входные сигналы X_i , которые поступают от нейронов-отправителей. Значением такого объединения является взвешенная сумма, где веса w_i представляют собой синаптические мощности. Возбуждающие синапсы имеют положительные веса, а тормозящие синапсы – отрицательные веса. Для выражения нижнего уровня активации нейрона к взвешенной сумме прибавляется компенсация (смещение) Q.

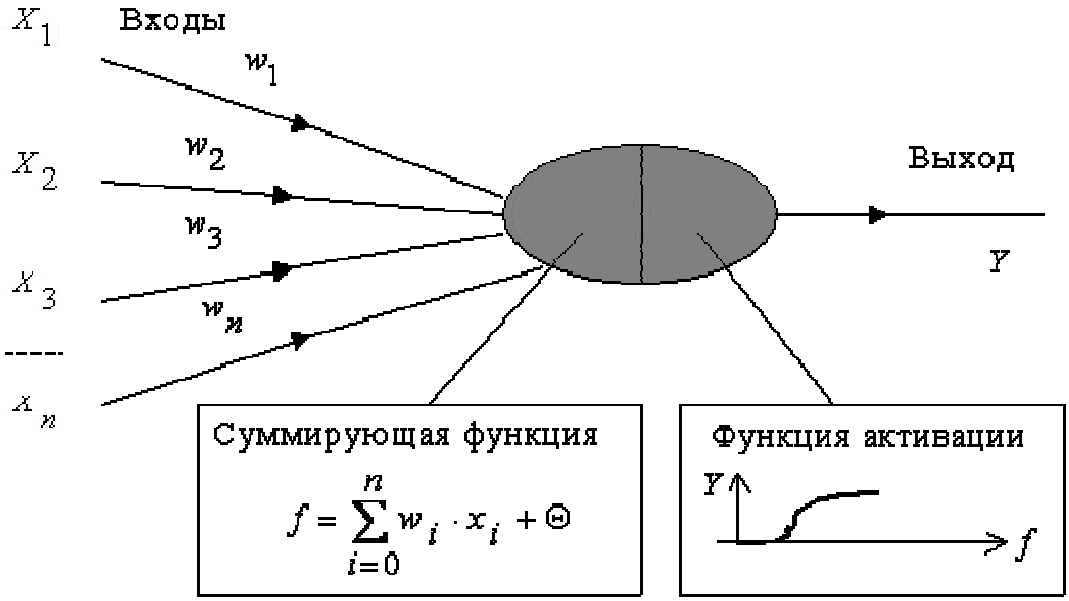
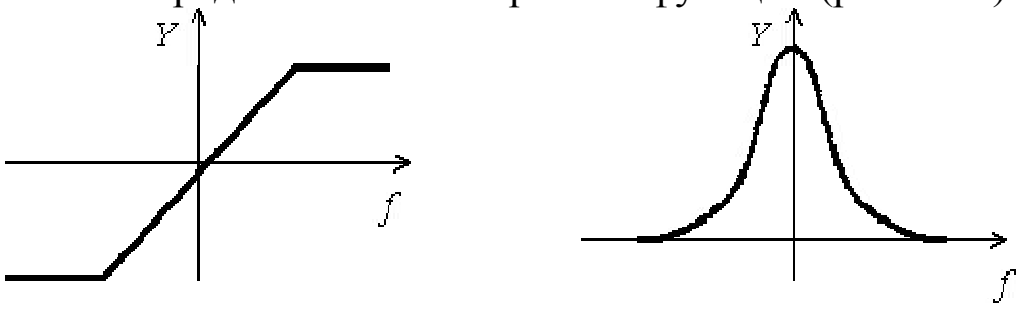


Рис. 4.13. Простая математическая модель нейрона

Так называемая функция активации рассчитывает выходной сигнал нейрона Y по уровню активности f. Функция активации обычно является сигмоидной, как показано в правой нижней рамке на рис. 4.13. Другими возможными видами функций активации являются линейная и радиально-симметричная функции (рис. 4.14).



а *б*

Рис. 4.14. Функции активации нейронов:
а – линейная; *б* – радиально-симметрическая

4.4.1. Персептрон Розенблатта. Теорема об обучении персептрона

Одной из первых искусственных сетей, способных к перцепции (восприятию) и формированию реакции на воспринятый стимул, явился PERCEPTRON Розенблатта (1957). Персептрон рассматривался его автором не как конкретное техническое вычислительное устройство, а как модель работы мозга. Нужно заметить, что после нескольких десятилетий исследований современные работы по искусственным нейронным сетям редко преследуют такую цель.

Простейший классический персептрон содержит нейроподобные элементы трех типов (рис. 4.15), назначение которых в целом соответствует нейронам рефлекторной нейронной сети. *S*-элементы формируют сетчатку сенсорных клеток, принимающих двоичные сигналы от внешнего мира. Далее сигналы поступают в слой ассоциативных или *A*-элементов (для упрощения изображения часть связей от входных *S*-клеток к *A*-клеткам не показана). Только ассоциативные элементы, представляющие собой формальные нейроны, выполняют нелинейную обработку информации и имеют изменяемые веса связей. *R*-элементы с фиксированными весами формируют сигнал реакции персептрона на входной стимул.

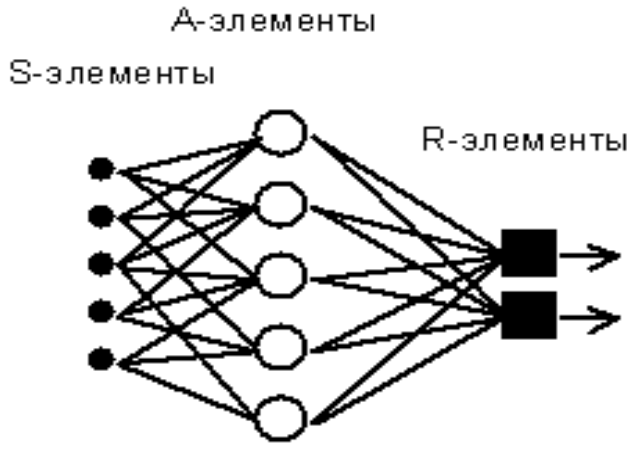


Рис. 4.15. Элементарный персептрон Розенблатта

Розенблатт называл такую нейронную сеть трехслойной, однако по современной терминологии представленная сеть обычно называется однослойной, так как имеет только один слой нейропроцессорных элементов. Однослойный персептрон характеризуется матрицей синаптических связей W от S- к A-элементам. Элемент матрицы W_{ij} отвечает связи, ведущей от i-го S-элемента к j-му A-элементу.

В Корнельской авиационной лаборатории была разработана электротехническая модель персептрона MARK-1, которая содержала 8 выходных *R*-элементов и 512 *A*-элементов, которые можно было соединять в различных комбинациях. На этом персептроне была проведена серия экспериментов по распознаванию букв алфавита и геометрических образов.

В работах Розенблатта был сделано заключение о том, что нейронная сеть рассмотренной архитектуры будет способна к воспроизведению любой логической функции, однако, как было показано позднее М. Минским и С. Пейпертом (1969), этот вывод оказался неточным. Были выявлены принципиальные неустранимые ограничения однослойных персептронов, и в последствии стал в основном рассматриваться многослойный вариант персептрона, в котором имеются несколько слоев процессорных элементов.

С сегодняшних позиций однослойный персептрон представляет скорее исторический интерес, однако на его примере могут быть изучены основные понятия и простые алгоритмы обучения нейронных сетей.

Теорема об обучении персептрона. Обучение сети состоит в подстройке весовых коэффициентов каждого нейрона. Пусть имеется набор пар векторов (x^i, y^i), $i = 1..p$, называемый *обучающей выборкой*. Будем называть нейронную сеть обученной на данной обучающей выборке, если при подаче на входы сети каждого вектора x^i на выходах всякий раз получается соответствующий вектор y^i.

Предложенный Ф. Розенблаттом метод обучения состоит в итерационной подстройке матрицы весов, последовательно уменьшающей ошибку в выходных векторах. Алгоритм включает несколько шагов:

Шаг 0. Начальные значения весов всех нейронов $W(t = 0)$ полагаются случайными.

Шаг 1. Сети предъявляется входной образ x^i, в результате формируется выходной образ $\mathbf{y}^\alpha \neq \mathbf{y}^\alpha$.

Шаг 2. Вычисляется вектор ошибки $\delta^\alpha = \left(\mathbf{y}^\alpha - \mathbf{y}^\alpha\right)$, делаемой сетью на выходе. Дальнейшая идея состоит в том, что изменение вектора весовых коэффициентов в области малых ошибок должно быть пропорционально ошибке на выходе и равно нулю, если ошибка равна нулю.

Шаг 3. Вектор весов модифицируется по следующей формуле: $W(t+\Delta t) = W(t) + \eta x^{\alpha} \cdot \left(\delta^{\alpha}\right)^{T}$.

Здесь $0 < \eta < 1$ – темп обучения.

Шаг 4. Шаги 1 – 3 повторяются для всех обучающих векторов. Один цикл последовательного предъявления всей выборки называется *эпохой*. Обучение завершается по истечении нескольких эпох: а) когда итерации сойдутся, т.е. вектор весов перестает изменяться, или б) когда полная просуммированная по всем векторам абсолютная ошибка станет меньше некоторого малого значения.

Используемая на шаге 3 формула учитывает следующие обстоятельства:

а) модифицируются только компоненты матрицы весов, отвечающие ненулевым значениям входов;

б) знак приращения веса соответствует знаку ошибки, т.е. положительная ошибка ($\delta > 0$, значение выхода меньше требуемого) приводит к усилению связи;

в) обучение каждого нейрона происходит независимо от обучения остальных нейронов, что соответствует важному с биологической точки зрения принципу локальности обучения.

Данный метод обучения был назван Ф. Розенблаттом «методом коррекции с обратной передачей сигнала ошибки». Представленный алгоритм относится к широкому классу алгоритмов обучения с учителем, поскольку известны как входные векторы, так и требуемые значения выходных векторов (имеется учитель, способный оценить правильность ответа ученика).

Доказанная Розенблаттом теорема о сходимости обучения по d-правилу говорит о том, что персептрон способен обучиться любому обучающему набору, который он способен представить.

4.4.2. Задача обучения нейронной сети на примерах. Классификация и категоризация

По своей организации и функциональному назначению искусственная нейронная сеть с несколькими входами и выходами выполняет некоторое преобразование входных стимулов – сенсорной информа-

ции о внешнем мире – в выходные управляющие сигналы. Число преобразуемых стимулов равно n-числу входов сети, а число выходных сигналов соответствует числу выходов m. Совокупность всевозможных входных векторов размерности n образует векторное пространство X, которое будем называть признаковым пространством. При рассмотрении соответствующих пространств предполагается использование обычных векторных операций сложения и умножения на скаляр. Аналогично выходные векторы также формируют признаковое пространство, которое будет обозначаться Y. Теперь нейронную сеть можно мыслить как некоторую многомерную функцию $F: X \rightarrow Y$, аргумент которой принадлежит признаковому пространству входов, а значение – выходному признаковому пространству.

При произвольном значении синаптических весовых коэффициентов нейронов сети функция, реализуемая сетью, также произвольна. Для получения требуемой функции необходим специфический выбор весов. Упорядоченная совокупность всех весовых коэффициентов всех нейронов может быть представлена как вектор W. Множество всех таких векторов также формирует векторное пространство, называемое пространством состояний или конфигурационным (фазовым) пространством W. Термин «фазовое пространство» пришел из статистической физики систем многих частиц, где под ним понимается совокупность координат и импульсов всех частиц, составляющих систему.

Задание вектора в конфигурационном пространстве полностью определяет все синаптические веса и тем самым состояние сети. Состояние, при котором нейронная сеть выполняет требуемую функцию, называют обученным состоянием сети W^*. Отметим, что для заданной функции обученное состояние может не существовать или быть не единственным. Задача обучения теперь формально эквивалентна построению процесса перехода в конфигурационном пространстве от некоторого произвольного состояния W^0 к обученному состоянию.

Требуемая функция однозначно описывается путем задания соответствия каждому вектору признакового пространства X некоторого вектора из пространства Y. В случае сети из одного нейрона в задаче детектирования границы полное описание требуемой функции достигается заданием всего четырех пар векторов. Однако в общем случае, как например, при работе с видеоизображением, признаковые пространства могут иметь высокую размерность, поэтому даже в случае

булевых векторов однозначное определение функции становится весьма громоздким (при условии, конечно, если функция не задана явно, например, формулой; однако для явно заданных функций обычно не возникает потребности представления их нейросетевыми моделями).

Во многих практических случаях значения требуемых функций для заданных значений аргумента получаются из эксперимента или наблюдений, и, следовательно, известны лишь для ограниченной совокупности векторов. Кроме того, известные значения функции могут содержать погрешности, а отдельные данные могут даже частично противоречить друг другу.

По этим причинам перед нейронной сетью обычно ставится задача приближенного представления функции по имеющимся примерам. Имеющиеся в распоряжении исследователя примеры соответствий между векторами, либо специально отобранные из всех примеров наиболее представительные данные называют обучающей выборкой. Обучающая выборка определяется обычно заданием пар векторов, причем в каждой паре один вектор соответствует стимулу, а второй – требуемой реакции. Обучение нейронной сети состоит в приведении всех векторов стимулов из обучающей выборки требуемым реакциям путем выбора весовых коэффициентов нейронов.

Общая проблема кибернетики, заключающаяся в построении искусственной системы с заданным функциональным поведением, в контексте нейроных сетей понимается как задача синтеза требуемой искусственной сети. Она может включать в себя следующие подзадачи:

1) выбор существенных для решаемой задачи признаков и формирование признаковых пространств;

2) выбор или разработка архитектуры нейронной сети, адекватной решаемой задаче;

3) получение обучающей выборки из наиболее представительных, по мнению эксперта, векторов признаковых пространств;

4) обучение нейронной сети на обучающей выборке.

Отметим, что подзадачи 1 – 3 во многом требуют экспертного опыта работы с нейронными сетями, и здесь нет исчерпывающих формальных рекомендаций.

Классификация и категоризация. В случае, когда выходное признаковое пространство представляет собой дискретный перечень из двух или более групп данных, задачей нейронной сети является отне-

сение входных векторов к одной из этих групп. В этом случае говорят, что нейросетевая система выполняет классификацию или категоризацию данных.

Эти две интеллектуальные задачи, по-видимому, следует отличать друг от друга. Термин «класс» можно определить как совокупность предметов или понятий (образов), выделенных и сгруппированных по определенным признакам или правилам. Под классификацией будем понимать отнесение некоторого образа к классу, выполняемое по этим формальным правилам по совокупности признаков. Категория же (если отвлечься от специфического философского характера этого понятия) определяет лишь некоторые общие свойства образов и связи между ними. Задача категоризации, т.е. определения отношения данного образа к некоторой категории, гораздо менее определена, чем задача отношения к классу. Границы различных категорий являются нечеткими, расплывчатыми, и обычно сама категория понимается не через формальное определение, а только в сравнении с другими категориями. Границы классов, напротив, определены достаточно точно – образ относится к данному классу, если известно, что он обладает необходимым числом признаков, характерных для этого класса.

Итак, задачей систем-классификаторов является установление принадлежности образа к одному из формально определенных классов. Примерами такой задачи являются задача классификации растений в ботанике, классификация химических веществ по их свойствам и типам возможных реакций, в которые они вступают, и др. Формальные признаки могут быть определены посредством правил типа «если... то», а системы, оперирующие с такими правилами, получили название экспертных систем. Традиционной областью применения классификаторов на нейронных сетях является экспериментальная физика высоких энергий, где одной из актуальных задач выступает выделение среди множества зарегистрированных в эксперименте событий с элементарными частицами событий, представляющих интерес для данного эксперимента.

Проблема категоризации находится на ступеньку выше по сложности в сравнении с классификацией. Ее особенность заключается в том, что помимо отнесения образа к какой-либо группе требуется определить сами эти группы, т.е. сформировать категории.

В случае обучения с учителем (например, в персептроне) формирование категорий происходит методом проб и ошибок на основе примеров с известными ответами, предоставляемыми экспертом. Формирование категорий весьма напоминает процесс обучения у жи-

вых организмов, поэтому обычно эксперта называют «супервизором» или учителем. Учитель управляет обучением при помощи изменения параметров связей и реже самой топологии сети.

Задачей системы-категоризатора является формирование обобщающих признаков в совокупности примеров. При увеличении числа примеров несущественные, случайные признаки сглаживаются, а часто встречающиеся – усиливаются, при этом происходит постепенное уточнение границ категорий. Хорошо обученная нейросетевая система способна извлекать признаки из новых примеров, ранее неизвестных системе, и принимать на их основе приемлемые решения.

Важно отметить различие в характере неявных «знаний», запомненных искусственной нейронной сетью, и явных, формальных «знаний», заложенных в экспертных системах (табл. 4.3).

Таблица 4.3

Сходства и различия ЭС и НС

Объект сравнения	Экспертные системы (ЭС)	Нейросетевые системы (НС)
Источник знаний	Формализованный опыт эксперта, выраженный в виде логических утверждений – правил и фактов, безусловно принимаемых системой	Совокупный опыт эксперта-учителя, отбирающего примеры для обучения + индивидуальный опыт обучающейся на этих примерах нейронной сети
Характер знаний	Формально-логическое «левополушарное» знание в виде правил	Ассоциативное «правополушарное» знание в виде связей между нейронами сети
Развитие знаний	В форме расширения совокупности правил и фактов (базы знаний)	В форме дообучения на дополнительной последовательности примеров, с уточнением границ категорий и формированием новых категорий
Роль эксперта	Задает на основе правил полный объем знаний экспертной системы	Отбирает характерные примеры, не формулируя специально обоснование своего выбора
Роль искусственной системы	Поиск цепочки фактов и правил для доказательства суждения	Формирование индивидуального опыта в форме категорий, получаемых на основе примеров, и категоризация образов

Различия в характере экспертных и нейросетевых систем обуславливают и различия в их сферах применения. Экспертные системы используются в узких предметных областях с хорошо структурированными знаниями, например в классификации неисправностей конкретного типа оборудования, фармокологии, анализе химсостава проб и т.д. Нейронные сети применяются, кроме перечисленных областей, и в задачах с плохо структурированной информацией, например при распознавании образов, рукописного текста, анализе речи и т.д.

4.4.4. Обучение нейронной сети с учителем как задача многофакторной оптимизации

Возможность применения теории оптимизации к обучению нейронных сетей крайне привлекательна, так как имеется множество хорошо опробованных методов оптимизации, доведенных до стандартных компьютерных программ. Сопоставление процесса обучения с процессом поиска некоторого оптимума также не лишено и биологических оснований, если рассматривать элементы адаптации организма к окружающим условиям в виде оптимального количества пищи, оптимального расходования энергии и т.п.

Функция одной действительной переменной $f(x)$ достигает локального минимума в некоторой точке x_0, если существует такая d-окрестность этой точки, что для всех x из этой окрестности, т.е. таких, что $| x - x_0 | < d$, имеет место $f(x) > f(x_0)$.

Без дополнительных предположений о свойствах гладкости функции выяснить, является ли некоторая точка достоверной точкой минимума, используя данное определение, невозможно, поскольку любая окрестность содержит континуум точек. При применении численных методов для приближенного поиска минимума исследователь может столкнуться с несколькими проблемами. Во-первых, минимум функции может быть не единственным. Во-вторых, на практике часто необходимо найти глобальный, а не локальный минимум, однако обычно неясно, нет ли у функции еще одного, более глубокого, чем найденный, минимума.

Математическое определение локального минимума функции в многомерном пространстве имеет тот же вид, если заменить точки x и x_0 на векторы, а вместо модуля использовать норму. Поиск минимума для функции многих переменных (многих факторов) является существенно более сложной задачей, чем для одной переменной. Это

связано прежде всего с тем, что локальное направление уменьшения значения функции может не соответствовать направлению движения к точке минимума. Кроме того, с ростом размерности быстро возрастают затраты на вычисление функции.

Решение задачи оптимизации во многом является искусством, общих, заведомо работающих и эффективных в любой ситуации методов нет. Среди часто используемых методов можно рекомендовать симплекс-метод Нелдера, некоторые градиентные методы, а также методы случайного поиска.

В случае, если независимые переменные являются дискретными и могут принимать одно значение из некоторого фиксированного набора, задача многомерной оптимизации несколько упрощается. При этом множество точек поиска становится конечным, а следовательно, задача может быть, хотя бы в принципе, решена методом полного перебора. Будем называть оптимизационные задачи с конечным множеством поиска задачами *комбинаторной оптимизации*.

Для комбинаторных задач также существуют методы поиска приближенного решения, предлагающие некоторую стратегию перебора точек, сокращающую объем вычислительной работы. Отметим, что имитация отжига и генетический алгоритм также применимы и к комбинаторной оптимизации.

Постановка задачи оптимизации при обучении нейронной сети. Пусть имеется нейронная сеть, выполняющая преобразование F: *X*®*Y* векторов *X* из признакового пространства входов *X* в вектора *Y* выходного пространства *Y*. Сеть находится в состоянии *W* из пространства состояний *W*. Пусть далее имеется обучающая выборка (*Xa, Ya*), а = 1..p. Рассмотрим полную ошибку *E*, делаемую сетью в состоянии *W*:

$$E = E(W) = \sum_{a} \left\| F(X^a; W) - Y^a \right\| = \sum_{a} \sum_{i} [F_i(X^a; W) - Y_i^a]^2$$

Отметим два свойства полной ошибки. Во-первых, ошибка *E = E(W)* является *функцией состояния W*, определенной на пространстве состояний. По определению, она принимает неотрицательные значения. Во-вторых, в некотором *обученном* состоянии W^*, в котором сеть не делает ошибок на обучающей выборке, данная функция принимает нулевое значение. Следовательно, обученные состояния являются *точками минимума* введенной функции *E(W)*.

Таким образом, задача обучения нейронной сети является задачей поиска минимума функции ошибки в пространстве состояний, и, сле-

довательно, для ее решения могут применяться стандартные методы теории оптимизации. Эта задача относится к классу многофакторных задач, так, например, для однослойного персептрона с N входами и M выходами речь идет о поиске минимума в NxM-мерном пространстве.

На практике могут использоваться нейронные сети в состояниях с некоторым малым значением ошибки, не являющихся в точности минимумами функции ошибки. Другими словами, в качестве решения принимается некоторое состояние из окрестности обученного состояния W^*. При этом допустимый уровень ошибки определяется особенностями конкретной прикладной задачи, а также приемлемым для пользователя объемом затрат на обучение.

4.4.5. Необходимость иерархической организации нейросетевых архитектур. Многослойный персептрон

Особенности строения биологических сетей подталкивают исследователя к использованию более сложных, в частности, иерархических архитектур. Идея относительно проста – на низших уровнях иерархии классы преобразуются таким образом, чтобы сформировать линейно разделимые множества, которые в свою очередь будут успешно распознаваться нейронами на следующих (высших) уровнях иерархии.

Однако основной проблемой, традиционно ограничивающей возможные сетевые топологии простейшими структурами, является проблема обучения. На этапе обучения сети предъявляются некоторые входные образы, называемые обучающей выборкой, и исследуются получаемые выходные реакции. Цель обучения состоит в приведении наблюдаемых реакций на заданной обучающей выборке к требуемым (адекватным) реакциям путем изменения состояний синаптических связей. Сеть считается обученной, если все реакции на заданном наборе стимулов являются адекватными. Данная классическая схема обучения с учителем требует явного знания ошибок при функционировании каждого нейрона, что, разумеется, затруднено для иерархических систем, где непосредственно контролируются только входы и выходы. Кроме того, необходимая избыточность в иерархических сетях приводит к тому, что состояние обучения может быть реализовано многими способами, что делает само понятие «ошибка, делаемая данным нейроном» весьма неопределенным.

Наличие таких серьезных трудностей в значительной мере сдерживало прогресс в области нейронных сетей вплоть до середины 80-х годов XX века, годов, когда были получены эффективные алгоритмы обучения иерархических сетей.

*Многослойный ПЕРСЕПТРОН***.** Рассмотрим иерархическую сетевую структуру, в которой связанные между собой нейроны (узлы сети) объединены в несколько слоев (рис. 4.16). На возможность построения таких архитектур указал еще Ф. Розенблатт, однако им не была решена проблема обучения. Межнейронные синаптические связи сети устроены таким образом, что каждый нейрон на данном уровне иерархии принимает и обрабатывает сигналы от каждого нейрона более низкого уровня. Таким образом, в данной сети имеется выделенное направление распространения нейроимпульсов – от входного слоя через один или несколько скрытых слоев к выходному слою нейронов. Нейросеть такой топологии будем называть обобщенным многослойным персептроном или просто персептроном.

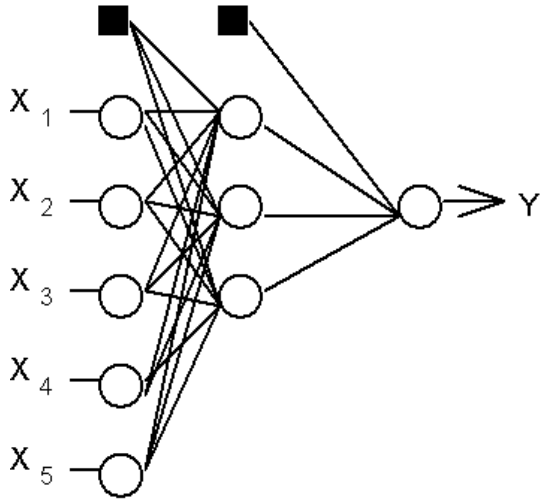


Рис. 4.16. Структура многослойного персептрона с пятью входами, тремя нейронами в скрытом слое и одним нейроном выходного слоя

Персептрон представляет собой сеть, состоящую из нескольких последовательно соединенных слоев формальных нейронов МакКаллока и Питтса. На низшем уровне иерархии находится входной слой, состоящий из сенсорных элементов, задачей которого является только прием и распространение по сети входной информации. Далее имеются один или, реже, несколько скрытых слоев. Каждый нейрон на скрытом слое имеет несколько входов, соединенных с выходами нейронов предыдущего слоя или непосредственно со входными сен-

сорами $X_1..X_n$, и один выход. Нейрон характеризуется уникальным вектором весовых коэффициентов w. Веса всех нейронов слоя формируют матрицу, которую будем обозначать V или W. Функция нейрона состоит в вычислении взвешенной суммы его входов с дальнейшим нелинейным преобразованием ее в выходной сигнал:

$$y = 1 \Big/ \left(1 + \exp(-[\sum_i W_i x_i - \Theta]) \right). \tag{4.3}$$

Выходы нейронов последнего, выходного слоя описывают результат классификации $Y = Y(X)$. Особенности работы персептрона состоят в следующем. Каждый нейрон суммирует поступающие к нему сигналы от нейронов предыдущего уровня иерархии с весами, определяемыми состояниями синапсов, и формирует ответный сигнал (переходит в возбужденное состояние), если полученная сумма выше порогового значения. Персептрон переводит входной образ, определяющий степени возбуждения нейронов самого нижнего уровня иерархии, в выходной образ, определяемый нейронами самого верхнего уровня. Число последних обычно сравнительно невелико. Состояние возбуждения нейрона на верхнем уровне говорит о принадлежности входного образа к той или иной категории.

Традиционно рассматривается аналоговая логика, при которой допустимые состояния синаптических связей определяются произвольными действительными числами, а степени активности нейронов – действительными числами между 0 и 1. Иногда исследуются также модели с дискретной арифметикой, в которой синапс характеризуется двумя булевыми переменными: активностью (0 или 1) и полярностью (-1 или +1), что соответствует трехзначной логике. Состояния нейронов могут при этом описываться одной булевой переменной. Данный дискретный подход делает конфигурационное пространство состояний нейронной сети конечным (не говоря уже о преимуществах при аппаратной реализации).

Для обучения многослойной сети в 1986 г. Руммельхартом и Хинтоном был предложен алгоритм обратного распространения ошибок (*error back propagation*). Многочисленные публикации о промышленных применениях многослойных сетей с этим алгоритмом обучения подтвердили его принципиальную работоспособность на практике.

В начале возникает резонный вопрос: а почему для обучения многослойного персептрона нельзя применить уже известное d-правило

Розенблатта? Ответ состоит в том, что для применения метода Розенблатта необходимо знать не только текущие выходы нейронов *y*, но и требуемые *правильные* значения *Y*. В случае многослойной сети эти правильные значения имеются только для нейронов *выходного* слоя. Требуемые значения выходов для нейронов скрытых слоев неизвестны, что и ограничивает применение d-правила.

Основная идея обратного распространения состоит в том, как получить оценку ошибки для нейронов скрытых слоев. Заметим, что *известные* ошибки, делаемые нейронами выходного слоя, возникают вследствие *неизвестных* пока ошибок нейронов скрытых слоев. Чем больше значение синаптической связи между нейроном скрытого слоя и выходным нейроном, тем сильнее ошибка первого влияет на ошибку второго. Следовательно, оценку ошибки элементов скрытых слоев можно получить как взвешенную сумму ошибок последующих слоев. При обучении информация распространяется от низших слоев иерархии к высшим, а оценки ошибок, делаемые сетью, – в обратном направлении, что и отражено в названии метода.

4.4.6. Программное обеспечение для работы с нейронными сетями

Программное обеспечение, имитирующее работу нейронной сети, называют *нейросимулятором* либо *нейропакетом*.

Большинство нейропакетов включают следующую последовательность действий:

- создание сети (выбор пользователем параметров либо одобрение установленных по умолчанию);
- обучение сети;
- выдача пользователю решения.

Существует огромное разнообразие нейропакетов, возможность использования нейросетей включена практически во все известные статистические пакеты.

Среди специализированных нейропакетов можно назвать BrainMaker, NeuroOffice, NeuroPro и др.

Критерии сравнения нейропакетов: простота применения, наглядность представляемой информации, возможность использовать различные структуры, скорость работы, наличие документации. Выбор определяется квалификацией и требованиями пользователя.

Пакет Matlab (The MathWorks) также предоставляет пользователям возможность работы с *нейронными сетями*. Входящий в стан-

дартную поставку Matlab «*Neural Network Toolbox*» предоставляет широкие возможности для работы с *нейронными сетями* всех типов.

Преимущество пакета Matlab состоит в том, что при его применении пользователь не ограничен моделями нейронных сетей и их параметрами, жестко заложенными в нейросимуляторе, а имеет возможность самостоятельно сконструировать ту сеть, которую считает оптимальной для решения поставленной задачи.

Рассмотрим пример конструирования нейронной сети в пакете Matlab.

Пусть имеется 15 независимых переменных – показателей деятельности фирмы и одна зависимая переменная – объем продаж. Имеем базу данных за прошедший год. Необходимо построить понедельный прогноз объемов продаж на месяц. Для решения задачи предлагается использовать трехслойную *сеть обратного распространения*.

Сформируем такую сеть, которая включает 15 нейронов во входом слое (по количеству входных переменных), 8 нейронов во втором слое и один нейрон в выходном слое (по количеству выходных переменных).

Для каждого слоя выберем передаточную функцию: первый слой – tansig, второй – logsig, третий – purelin.

В среде Matlab *синтаксис* такой *нейронной сети* выглядит следующим образом:

Net=netff(PR, [S1,S2, : , Sn],{TF1,TF2, : , TFn},btf, blf, pf),

где PR – *массив* минимальных и максимальных значений для R векторов входа;

Si – количество нейронов в *i*-м *слое*;

TFi – *функция активации слоя i*;

btf – обучающая *функция*, реализующая *метод обратного распространения*;

blf – *функция* настройки, реализующая *метод обратного распространения*;

pf – критерий качества обучения.

Активационной функцией может выступать любая дифференцируемая *функция*, например, tansig, logsig, purelin.

Net=netff(minmax (P), [n,m, l],{ tansig, logsig, purelin },trainpr),

где P – множество входных векторов;

n – количество входов НС;

m – количество нейронов в скрытом *слое*;

l – количество выходов НС.

Необходимо также установить метод расчета значения ошибки. Например, если выбран *метод наименьших квадратов*, то эта функция будет выглядеть так: *Net*.performFcn='*SSE*'.

Для установления максимального количества *эпох* равным 10 000 воспользуемся функцией: net.trainParam.epochs=10000.

Запустить процесс обучения можно таким образом:

[net,tr]=train(net,P,T).

После окончания обучения сети ее можно сохранить в файле, например, с именем nn1.*mat*.

Для этого необходимо выполнить команду:

save nn1 net.

Таким образом, в пакете возможно конструирование сети любой сложности, и нет необходимости привязываться к ограничениям, накладываемым нейросимуляторами. Однако для работы с нейронными сетями в пакете Matlab необходимо изучить как саму среду, так и большинство функций Neural Network Toolbox.

4.5. Нечеткая логика

Пусть X – произвольное непустое множество**. *Нечетким множеством*** (fuzzy set) A называется множество пар

$$A = \{x \mid \mu(x) = 1\},\ x \in X,$$

где x принадлежит X, значения $\mu(x)$ лежат на отрезке [0, 1].

Функция $\mu(x)$, которая отображает множество X на отрезок [0, 1], называется ***функцией принадлежности*** нечеткого множества A, а X – базовым множеством или базовой шкалой.

Возможные формы задания функции принадлежности: графический, аналитический, табличный (рис. 4.17).

Существует ряд методов построения по экспертным оценкам функции принадлежности нечеткого множества. Можно выделить две группы методов: прямые и косвенные.

Прямые методы определяются тем, что эксперт непосредственно задает правила определения значений функции принадлежности μ_A, характеризующей понятие A. Эти значения согласуются с его предпочтениями на множестве объектов U следующим образом:

– для любых u_1, $u_2 \in U$ $\mu A(u_1) < \mu A(u_2)$ тогда и только тогда, когда u_2 предпочтительнее u_1, т.е. в большей степени характеризуется понятием A;

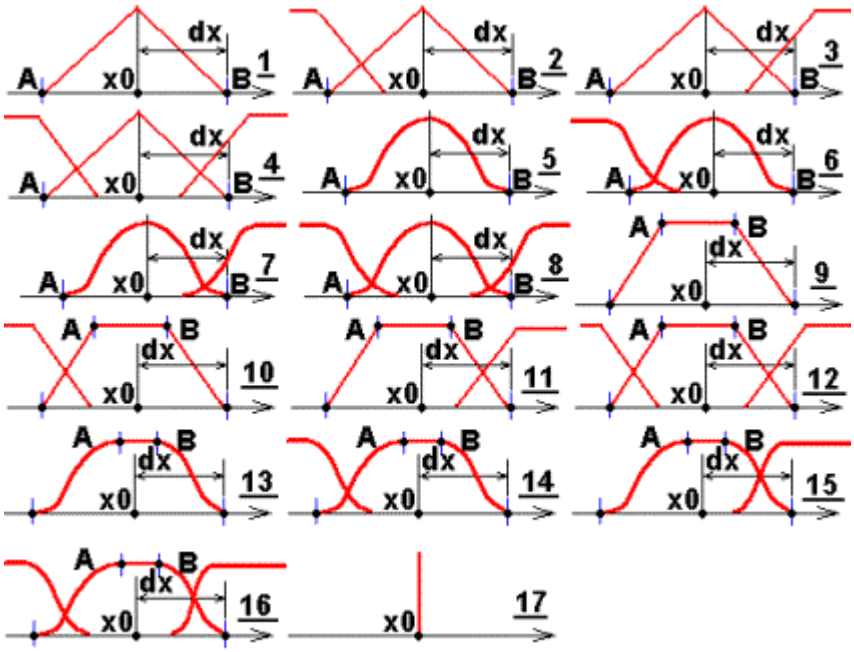


Рис. 4.17. Формы функций принадлежности:
1 – треугольная; 2 – треугольная Z типа; 3 – треугольная S типа;
4 – треугольная Z-S типа; 5 – колоколообразная; 6 – колоколообразная Z типа;
7 – колоколообразная S типа; 8 – колоколообразная Z-S типа; 9 – трапецеидальная;
10 – трапецеидальная Z типа; 11 – трапецеидальная S типа;
12 – трапецеидальная Z-S типа; 13 – трапецивидная-колоколообразная;
14 – трапецивидная-колоколообразная Z типа;
15 – трапецивидная-колоколообразная S типа;
16 – трапецивидная-колоколообразная Z-S типа; 17 – точечная

– для любых u_1, $u_2 \in U$ $\mu A(u_1)=\mu A(u_2)$ тогда и только тогда, когда u_1 и u_2 безразличны относительно понятия A.

Примеры прямых методов: непосредственное задание функции принадлежности таблицей, формулой, примером.

В косвенных методах значения функции принадлежности выбираются таким образом, чтобы удовлетворить заранее сформулированным условиям. Экспертная информация является только исходной информацией для дальнейшей обработки. Дополнительные условия могут налагаться как на вид получаемой информации, так и на процедуру обработки.

Однако функция принадлежности может отражать как мнение группы экспертов, так и мнение одного (уникального) эксперта, следовательно, возможны, по крайней мере, четыре группы методов: прямые и косвенные для одного эксперта, прямые и косвенные для группы экспертов. Кроме того, существуют методы построения

функций принадлежности терм-множеств. Классификация методов представлена на рис. 4.18.

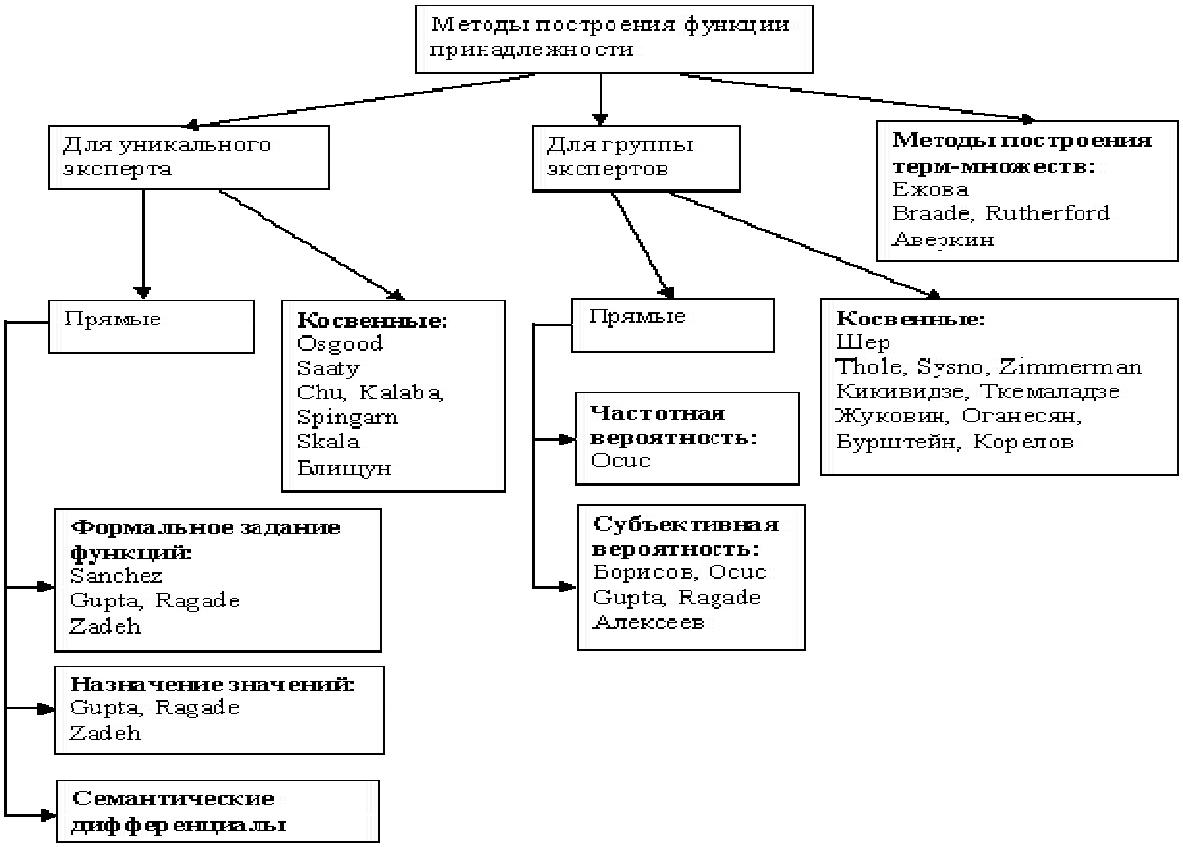


Рис. 4.18. Классификация методов построения функции принадлежности

Лингвистической переменной (linguistic variables) называется тройка <*b, T, X*>, где *b* – наименование лингвистической переменной; *T* – множество ее значений (термов), представляющих наименование нечетких переменных, областью определения каждой из которых является множество *X*. Например: <Стоимость, T, [0; 5000]>, где T={«малая», «средняя», «высокая»}.

Носителем *A* называется множество тех его элементов *x*, для которых $\mu(x)$ положительна:

Носитель $(A) = \{x \in X \mid \mu(x) > 0\}$.

Точка перехода *A* – это элемент *x* множества *A*, для которого $\mu(x)$=0,5.

α-срез нечеткого множества (A_α) – множество элементов *x*, для которых функция принадлежности $\mu(x)$ принимает значения не меньше заданного числа α ($0<\alpha<1$):

$$A_\alpha = \{x \in X \mid \mu(x) \geq \alpha\}.$$

Высота нечеткого множества A находится как точная верхняя грань (максимум) его функции принадлежности:

Высота $(A) = \sup_{x \in X} \mu(x)$.

Если высота нечеткого множества равна единице, то такое множество называется ***нормализованным***. В том случае, когда высота нечеткого множества A меньше единицы (такое множество называется ***субнормальным)***, можно осуществить переход к нормализованному множеству путем деления его функции принадлежности $\mu(x)$ на высоту $\sup_{x \in X} \mu(x)$

Если носитель нечеткого множества A состоит из единственной точки x, то такое множество называется ***одноточечным*** (singleton).

Лингвистические правила – инструкции, построенные по схеме логической *импликации ЕСЛИ-ТО.*

Нечеткий алгоритм (fuzzy algorithm) – упорядоченное множество нечетких инструкций (правил), в формулировке которых содержатся нечеткие указания (термы).

Процесс перехода от четкого (измеренного) значения к его нечеткой интерпретации называется ***фаззификацией*** (fuzzyfication).

Логическим выводом называется процесс получения нечеткого значения результирующих переменных на основе фактических значений входных лингвистических переменных с использованием нечеткого алгоритма. В процессе логического вывода определяются уровни активности правил исходя из значений функции принадлежности переменных в условной части правила.

Наиболее известные механизмы логического вывода:

– метод максимума-минимума (MAX-MIN-Inference);

– метод максимума-произведения (MAX-Product-Inference).

При использовании логического вывода результатом выполнения лингвистических правил является некоторое нечеткое множество, описываемое результирующей функцией принадлежности.

Переход от полученного нечеткого множества к единственному четкому значению, которое и признается затем в качестве решения поставленной задачи, называется ***дефаззификацией*** (defuzzyfication).

Некоторые из наиболее известных методов дефаззификации:

а) *Метод максимума* – выбирается тот элемент нечеткого множества, который имеет наивысшую степень принадлежности этому множеству.

Если такой элемент не является единственным, т.е. функция принадлежности $\mu_в(y)$ имеет несколько локальных максимумов $y_1, y_2,\ldots y_m$ со значениями $\mu_в(y_1)=\mu_в(y_2)=\ldots=\mu_в(y_m)$, или если имеется максимальное «плато» между y_1 и y_m, то выбор среди элементов, имеющих наивысшую степень принадлежности множеству, осуществляется на основе определенного критерия.

б) *Метод левого (правого) максимума* – выбирается наименьшее (наибольшее) из чисел $y_1, y_2,\ldots y_m$, имеющих наивысшую степень принадлежности нечеткому множеству.

в) *Метод среднего из максимумов* – в качестве искомого четкого значения y_0 принимается среднее арифметическое координат локальных максимумов:

$$y_0 = \frac{1}{m}\sum_{j=1}^{m} y_j .$$

г) *Метод центра тяжести* (Center-of-Area) – в качестве выходного значения y_0 выбирается абсцисса центра тяжести площади, расположенной под функцией принадлежности $\mu_в(y)$, $y \in Y$:

$$y_0 = \frac{\int_Y y\mu_B(y)dy}{\int_Y \mu_B(y)dy}.$$

При необходимости вычисления y_0 на компьютере в реальном времени, с учетом реальных вычислительных затрат, обычно операцию интегрирования заменяют суммированием.

Существует простая возможность использования для этих целей взвешенного среднего значения

$$y_0 \approx \frac{\sum_{i=1}^{n} \alpha_i y_i^*}{\sum_{i=1}^{n} \alpha_i},$$

где y_i^*, (i=1, 2, … n) – центральные значения нечетких подмножеств $B_i(y)$ выходной переменной y;

α_i – веса, учитывающие уровень выполнения условия «ЕСЛИ» i-го правила, называемые также уровнями активности соответствующих правил или α-уровнями;

n – число правил вывода.

д) *Модифицированный метод центра тяжести* – интегрирование производится только в тех областях, где $\mu_в(y)>\alpha$, $\alpha\in(0, 1)$, $y\in Y$. Параметр α используется здесь для подавления шумов, отсеивания влияния малосущественных для процедуры вывода факторов (на практике можно применять α=0,05÷0,1).

Функциональная схема интеллектуальной системы поддержки решений приведена на рис. 4.19.

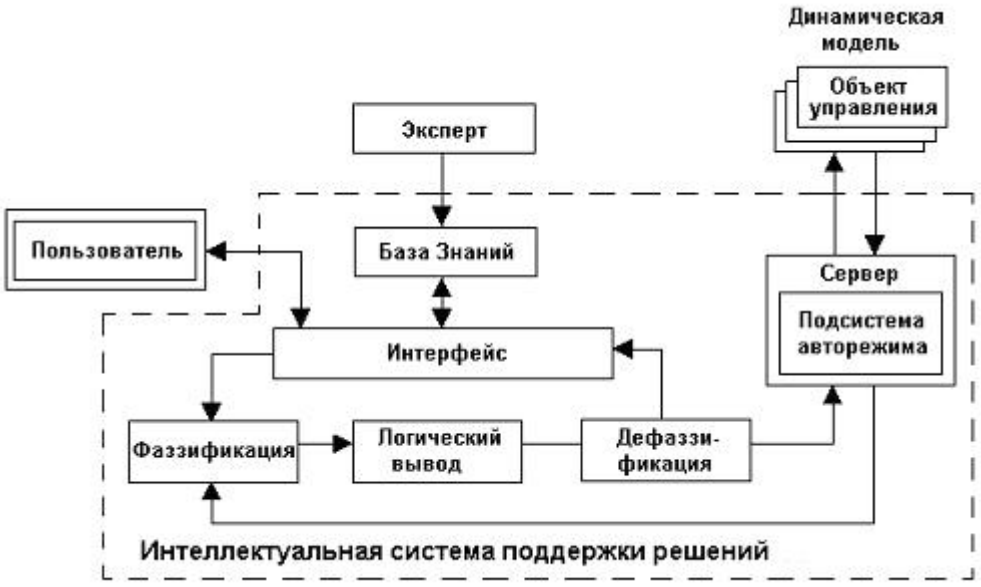


Рис. 4.19. Функциональная схема интеллектуальной системы поддержки решений на базе алгоритмов нечеткой логики

4.6. Генетические алгоритмы

4.6.1. Основные принципы генетических алгоритмов

Генетические алгоритмы (ГА) – адаптивные методы поиска, которые в последнее время часто используются для решения задач функциональной оптимизации. Они основаны на генетических процессах биологических организмов: биологические популяции развиваются в течение нескольких поколений, подчиняясь законам естественного отбора и по принципу «выживает наиболее приспособленный» (survival of the fittest), открытому Чарльзом Дарвином. Подражая этому процессу, генетические алгоритмы способны «развивать» решения реальных задач, если те соответствующим образом закодированы. Например, ГА могут использоваться, чтобы проектировать структуры моста, для поиска максимального отношения прочности/веса или оп-

ределять наименее расточительное размещение для нарезки форм из ткани. Они могут также использоваться для интерактивного управления процессом, например на химическом заводе, или балансирования загрузки на многопроцессорном компьютере. Вполне реальный пример: израильская компания Schema разработала программный продукт Channeling для оптимизации работы сотовой связи путем выбора оптимальной частоты, на которой будет вестись разговор. В основе этого программного продукта и используются генетические алгоритмы.

Основные принципы ГА были сформулированы Голландом (1975) и хорошо описаны во многих работах. В отличие от эволюции, происходящей в природе, ГА только моделируют те процессы в популяциях, которые являются существенными для развития. Точный ответ на вопрос: какие биологические процессы существенны для развития, и какие нет? – все еще открыт для исследователей.

В природе особи в популяции конкурируют друг с другом за различные ресурсы, такие, например, как пища или вода. Кроме того, члены популяции одного вида часто конкурируют за привлечение брачного партнера. Те особи, которые наиболее приспособлены к окружающим условиям, будут иметь относительно больше шансов воспроизвести потомков. Слабо приспособленные особи либо совсем не произведут потомства, либо их потомство будет очень немногочисленным. Это означает, что гены от высоко адаптированных или приспособленных особей будут распространяться в увеличивающемся количестве потомков на каждом последующем поколении. Комбинация хороших характеристик от различных родителей иногда может приводить к появлению «суперприспособленного» потомка, чья приспособленность больше, чем приспособленность любого из его родителей. Таким образом, вид развивается, лучше и лучше приспосабливаясь к среде обитания.

ГА используют прямую аналогию с таким механизмом. Они работают с совокупностью особей – популяцией, каждая из которых представляет возможное решение данной проблемы. Каждая особь оценивается мерой ее «приспособленности», согласно тому, насколько «хорошо» соответствующее ей решение задачи. Например, мерой приспособленности могло бы быть отношение силы/веса для данного проекта моста. (В природе это эквивалентно оценке того, насколько эффективен организм при конкуренции за ресурсы.) Наиболее приспособленные особи получают возможность «воспроизводить» по-

томство с помощью «перекрестного скрещивания» с другими особями популяции. Это приводит к появлению новых особей, которые сочетают в себе некоторые характеристики, наследуемые ими от родителей. Наименее приспособленные особи с меньшей вероятностью смогут воспроизвести потомков, так что те свойства, которыми они обладали, будут постепенно исчезать из популяции в процессе эволюции.

Так и воспроизводится вся новая популяция допустимых решений, выбирая лучших представителей предыдущего поколения, скрещивая их и получая множество новых особей. Это новое поколение содержит более высокое соотношение характеристик, которыми обладают хорошие члены предыдущего поколения. Таким образом, из поколения в поколение хорошие характеристики распространяются по всей популяции. Скрещивание наиболее приспособленных особей приводит к тому, что исследуются наиболее перспективные участки пространства поиска. В конечном счете популяция будет сходиться к оптимальному решению задачи.

В настоящее время под термином «генетические алгоритмы» скрывается не одна модель, а достаточно широкий класс алгоритмов, подчас мало похожих друг на друга. Исследователи экспериментировали с различными типами представлений, операторов кроссовера и мутации, специальных операторов и различных подходов к воспроизводству и отбору.

Хотя модель эволюционного развития, применяемая в ГА, сильно упрощена по сравнению со своим природным аналогом, тем не менее ГА является достаточно мощным средством и может с успехом применяться для широкого класса прикладных задач, включая те, которые трудно, а иногда и вовсе невозможно решить другими методами.

4.6.2. Пример работы простого генетического алгоритма

На рис. 4.20 приведен пример простого генетического алгоритма.

Работа ГА представляет собой итерационный процесс, который продолжается до тех пор, пока поколения не перестанут существенно отличаться друг от друга или не пройдет заданное количество поколений или заданное время. Для каждого поколения реализуются отбор, кроссовер (скрещивание) и мутация. Рассмотрим этот алгоритм.

Шаг 1: генерируется начальная популяция, состоящая из N особей со случайными наборами признаков.

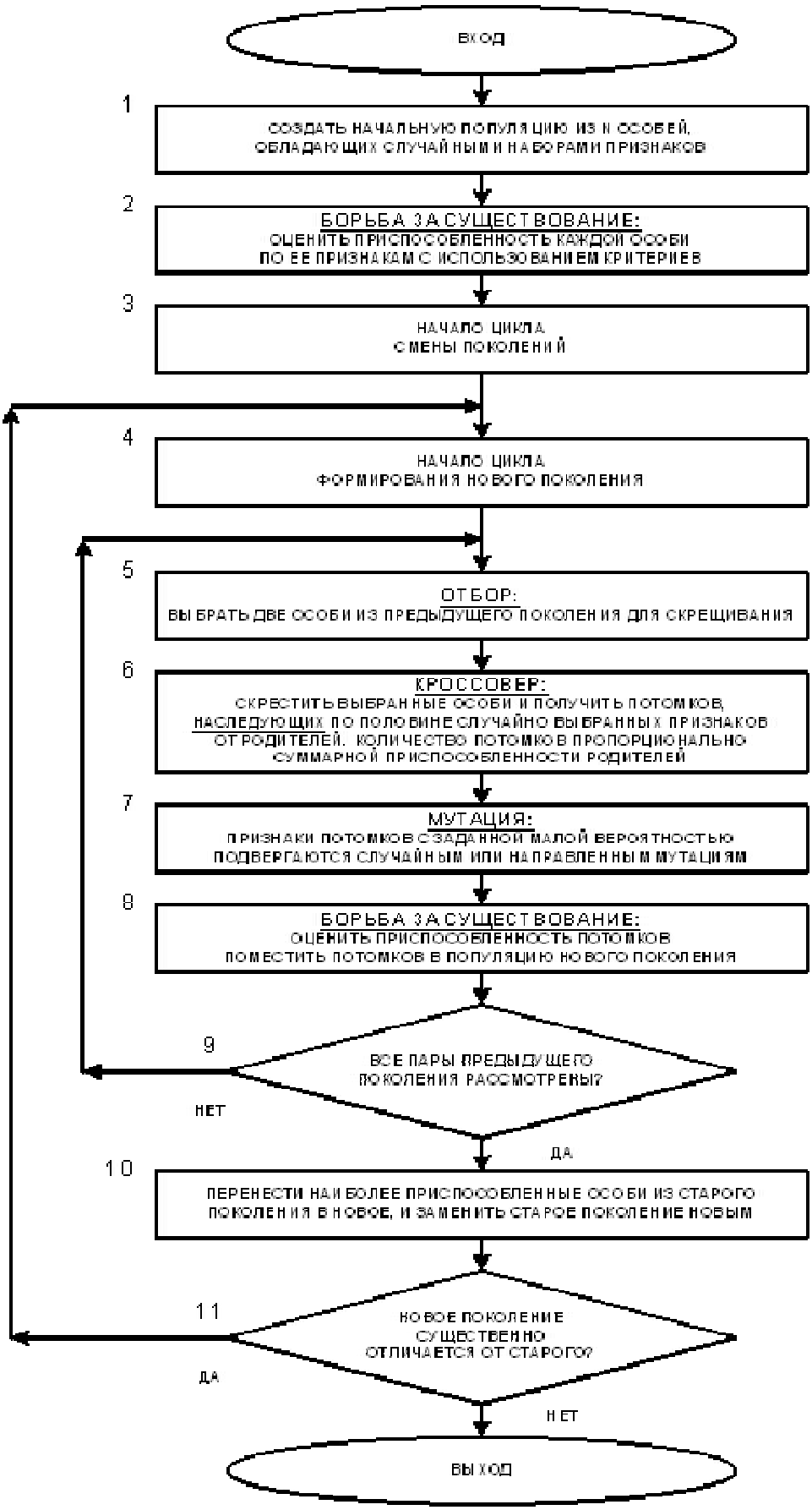


Рис. 4.20. Простой генетический алгоритм

Шаг 2 (борьба за существование): вычисляется абсолютная *приспособленность* каждой особи популяции к условиям среды *f(i)* и суммарная приспособленность особей популяции, характеризующая приспособленность всей популяции. Затем при *пропорциональном отборе* для каждой особи вычисляется ее *относительный вклад в* суммарную приспособленность популяции $P_s(i)$, т.е. отношение ее абсолютной приспособленности *f(i)* к суммарной приспособленности всех особей популяции:

$$P_s(i) = \frac{f(i)}{\sum_{i=1}^{N} f(i)}. \qquad (4.3)$$

В выражении (4.3) сразу обращает на себя внимание возможность сравнения абсолютной приспособленности *i-й* особи *f(i)* не с суммарной приспособленностью всех особей популяции, а со средней абсолютной приспособленностью особи популяции:

$$\bar{f} = \frac{1}{N}\sum_{i=1}^{N} f(i). \qquad (4.4)$$

Тогда получим:

$$P(i) = \frac{f(i)}{\bar{f}} = \frac{f(i)}{\frac{1}{N}\sum_{i=1}^{N} f(i)}. \qquad (4.5)$$

Если взять логарифм по основанию 2 от выражения, то получим *количество информации, содержащееся в признаках особи о том, что она выживет и даст потомство*:

$$I(i) = Log_2 \frac{f(i)}{\bar{f}}. \qquad (4.6)$$

Необходимо отметить, что эта формула совпадает с формулой для семантического количества информации Харкевича, если целью считать *индивидуальное выживание и продолжение рода*. Это значит, что даже чисто формально *приспособленность особи представляет собой количество информации, содержащееся в ее фенотипе о продолжении ее генотипа в последующих поколениях.*

Поскольку количество потомства особи пропорционально ее приспособленности, то естественно считать, что *если это количество информации:*

– *положительно,* то данная особь выживает и дает потомство, численность которого пропорциональна этому количеству информации;

– *равно нулю,* то особь доживает до половозрелого возраста, но потомства не дает (его численность равна нулю);

– *меньше нуля,* то особь погибает до достижения половозрелого возраста.

Таким образом, можно сделать фундаментальный вывод, имеющий даже мировоззренческое звучание, о том, что *естественный отбор представляет собой процесс генерации и накопления информации о выживании и продолжении рода в ряде поколений популяции как системы.*

Это накопление информации происходит на различных уровнях иерархии *популяции как системы*, включающей:

– *элементы системы*: отдельные особи;

– *взаимосвязи между элементами*: отношения между особями в популяции, обеспечивающие передачу последующим поколениям максимального количества информации об их выживании и продолжении рода (путем скрещивания наиболее приспособленных особей и наследования рациональных приобретений);

– *цель системы*: сохранение и развитие популяции, реализуется через цели особей: индивидуальное выживание и продолжение рода.

Фенотип соответствует генотипу и представляет собой его внешнее проявление в признаках особи. Особь взаимодействует с окружающей средой и другими особями в соответствии со своим фенотипом. В случае, если это взаимодействие удачно, то особь передает генетическую информацию, определяющую фенотип, последующим поколениям [28].

Шаг 3: начало цикла смены поколений.

Шаг 4: начало цикла формирования нового поколения.

Шаг 5 (отбор): осуществляется *пропорциональный отбор* особей, которые могут участвовать в продолжении рода. Отбираются только те особи популяции, у которых количество информации в фенотипе и генотипе о выживании и продолжении рода положительно, причем вероятность выбора пропорциональна этому количеству информации.

Шаг 6 (кроссовер): отобранные для продолжения рода на предыдущем шаге особи с заданной вероятностью P_c подвергаются *скрещиванию* или *кроссоверу* (*рекомбинации*).

Если кроссовер происходит, то потомки получают по половине случайным образом определенных признаков от каждого из родителей. Численность потомства пропорциональна суммарной приспособленности родителей. В некоторых вариантах ГА потомки после своего появления заменяют собой родителей и переходят к мутации.

Если кроссовер не происходит, то исходные особи – несостоявшиеся родители – переходят на стадию мутации.

Шаг 7 (мутация): выполняются операторы *мутации*. При этом признаки потомков с вероятностью P_m случайным образом изменяются на другие. Отметим, что использование механизма случайных мутаций роднит генетические алгоритмы с таким широко известным методом имитационного моделирования, как *метод Монте-Карло*.

Шаг 8 (борьба за существование): оценивается приспособленность потомков (по тому же алгоритму, что и на шаге 2).

Шаг 9: проверяется, все ли отобранные особи дали потомство.

Если нет, то происходит переход на шаг 5, и продолжается формирование нового поколения, иначе – переход на следующий шаг 10.

Шаг 10: происходит смена поколений:

– потомки помещаются в новое поколение;

– наиболее приспособленные особи из старого поколения переносятся в новое, причем для каждой из них это возможно не более заданного количества раз;

– полученная новая популяция замещает собой старую.

Шаг 11: проверяется выполнение условия останова генетического алгоритма. ***Выход*** из генетического алгоритма происходит либо тогда, когда новые поколения перестают существенно отличаться от предыдущих, т.е., как говорят, «алгоритм сходится», либо когда пройдено заданное количество поколений или заданное время работы алгоритма (чтобы не было «зацикливания» и динамического зависания в случае, когда решение не может быть найдено в заданное время).

Если ГА сошелся, то это означает, что решение найдено, т.е. получено поколение, идеально приспособленное к условиям данной фиксированной среды обитания.

Иначе – переход на шаг 4 – начало формирования нового по0коления.

В реальной биологической эволюции этим дело не ограничивается, так как любая популяция кроме освоения некоторой экологиче-

ской ниши пытается также выйти за ее пределы и освоить другие ниши, как правило, «смежные». Именно за счет этих процессов жизнь вышла из моря на сушу, проникла в воздушное пространство и поверхностный слой почвы, а сейчас осваивает космическое пространство.

Конечно, реальные генетические алгоритмы, на которых проводятся научные исследования, чаще всего мало похожи на приведенный пример. Исследователи экспериментируют с различными параметрами генетических алгоритмов, например: со способами отбора особей для скрещивания; с критериями приспособленности и жесткостью влияния факторов среды; со способами выбора признаков, передающихся от родителей потомкам (рецессивные и нерецессивные гены и т.д.); с интенсивностью, видом случайного распределения и направленностью мутаций; различными подходами к воспроизводству и отбору.

Поэтому под термином «генетические алгоритмы» по сути дела надо понимать не одну модель, а довольно широкий класс алгоритмов, подчас мало похожих друг на друга.

В настоящее время рассматривается много различных операторов отбора, кроссовера и мутации:

– турнирный отбор (Brindle, 1981; Goldberg и Deb, 1991) реализует n турниров, чтобы выбрать n особей, при этом каждый турнир построен на выборке k элементов из популяции и выборе лучшей особи среди них (наиболее распространен турнирный отбор с k=2);

– элитный отбор (De Jong, 1975) гарантирует, что при отборе обязательно будут выживать лучший или лучшие члены популяции совокупности (наиболее распространена процедура обязательного сохранения только одной лучшей особи, если она не прошла, как другие, через процесс отбора, кроссовера и мутации);

– двухточечный кроссовер (Cavicchio, 1970; Goldberg, 1989c) и равномерный кроссовер (Syswerda, 1989) отличаются способами наследования потомками признаков родителей.

4.6.3. Примеры применения генетических алгоритмов

В 1994 году Эндрю Кин из университета Саутгемптона использовал генетический алгоритм в дизайне космических кораблей. За основу была взята модель опоры космической станции, спроектированной

в NASA, из которой после смены 15 поколений, включавших 4500 вариантов дизайна, получилась модель, превосходящая по тестам тот вариант, что разработали люди.

Аналогичный генетический алгоритм был использован NASA при разработке антенны для спутника.

Джон Коза из Стэнфорда разработал технологию генетического программирования, в которой результатом эволюции становятся не отдельные числовые параметры особей, а целые имитационные программы, которые являются виртуальными аналогами реальных устройств. Эта технология позволила компании Genetic Programming повторить 15 человеческих изобретений, шесть из которых были запатентованы после 2000 года, т.е. представляют собой самые передовые достижения, а один из контроллеров, «выведенных» в GP, даже превосходит аналогичную человеческую разработку.

Сейчас плоды электронной эволюции можно найти в самых разных сферах: от двигателя самолета Boeing 777 до новых антибиотиков.

Генетические алгоритмы представляют собой компьютерное моделирование эволюции. Материальное воплощение сконструированных таким образом систем до сих пор была невозможна без участия человека. Однако интенсивно ведутся работы, результатом которых является уменьшение зависимости машинной эволюции от человека. Эти работы ведутся по двум основным направлениям:

1. Естественный отбор, моделируемый ГА, переносится из виртуального мира в реальный, например, проводятся эксперименты по реальным битвам роботов на выживание.

2. Интеллектуальные системы, основанные на ГА, конструируют роботов, которые в принципе могут быть изготовлены на автоматизированных заводах без участия человека.

Пример воплощения ГА в реальной битве роботов на выживание: в 2002 году в британском центре Magna открылся павильон Live Robots, где боролись за выживание 12 роботов двух видов: *гелиофаги*, способные добывать электроэнергию с использованием солнечных батарей; *хищники*, которые могли получать электроэнергию только от гелиофагов. Выжившие роботы загружали свои «гены» в погибших и таким образом образовывали новые поколения. Те хищники, которые забирали всю энергию у гелиофагов, теряли источник питания и погибали, не передавая свою тактику потомкам, поступавшие же

«более разумно» продолжили свой род. В результате возникла равновесная сбалансированная искусственная экосистема с двумя популяциями.

***Пример конструирования роботов роботами*:** в Brandeis University была создана программа Golem, которая сама конструировала роботов. В программу была включена база деталей, а также механизм мутаций и функция пригодности для «отсеивания» неудачников – тех, кто не научился двигаться. После 600 поколений за несколько дней программа получила модели трех ползающих роботов. Показательно, что роботы оказались симметричными, хотя симметрия никак не была явно прописана в правилах эволюции и исходных данных. Это означает, что она появилась в ходе моделирования машинной эволюции как полезная черта, позволяющая двигаться прямолинейно.

Генетические алгоритмы (ГА) – это стохастические, эвристические оптимизационные методы, впервые предложенные Джоном Холландом в 1975 году. Они основываются на идее эволюции с помощью естественного отбора. Кроме более быстрого нахождения экстремума, к положительным свойствам генетических алгоритмов можно отнести и нахождение «глобального» экстремума. В задачах, где целевая функция имеет значительное количество локальных экстремумов, в отличие от градиентного метода, генетические алгоритмы не «застревают» в точках локального экстремума, а позволяют найти «глобальный» минимум.

Генетические алгоритмы работают с совокупностью особей – популяцией, где каждая особь представляет возможное решение данной проблемы. Иногда происходят мутации, или спонтанные изменения в генах.

Таким образом, из поколения в поколение хорошие характеристики распространяются по всей популяции. Скрещивание наиболее приспособленных особей приводит к тому, что наследуются наиболее перспективные участки пространства поиска. В конечном итоге популяция будет сходиться к оптимальному решению задачи. Преимущество ГА состоит в том, что он находит приблизительные оптимальные решения за относительно короткое время [31].

ГА оперирует следующей терминологией:

- *Хромосома* – решение рассматриваемой проблемы, носитель наследственной информации. Совокупность хромосом (значений параметров целевой функции) характеризует особь. Хромосома состоит из генов.

• *Гены* – элементы кодирования наследственной информации (параметров целевой функции). В качестве генов чаще всего выступает битовое кодирование информации.

• *Особь* – набор хромосом (совокупность параметров, для которой ищется значение целевой функции).

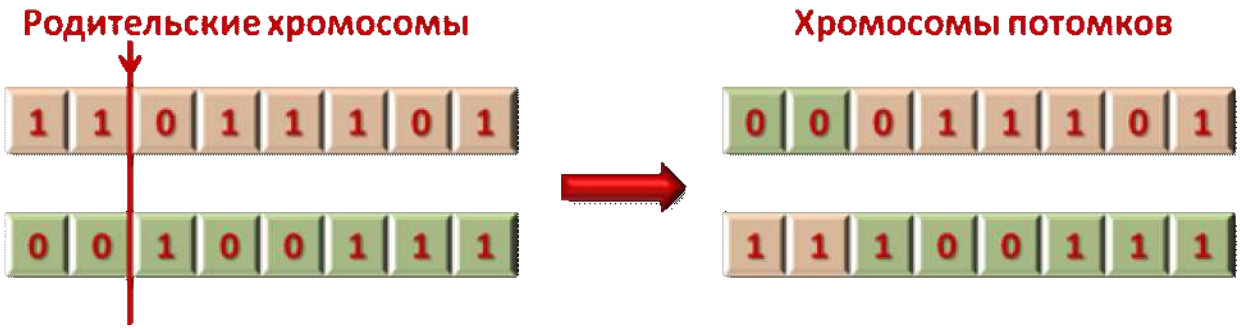
• *Приспособленность особи* – значение целевой функции для данного набора параметров по отношению к требуемому значению.

ГА производит над особями следующие действия:

• *генерация начальной популяции хромосом* – случайным образом выбираются значения параметров целевой функции, и для этих значений параметров находится значение целевой функции;

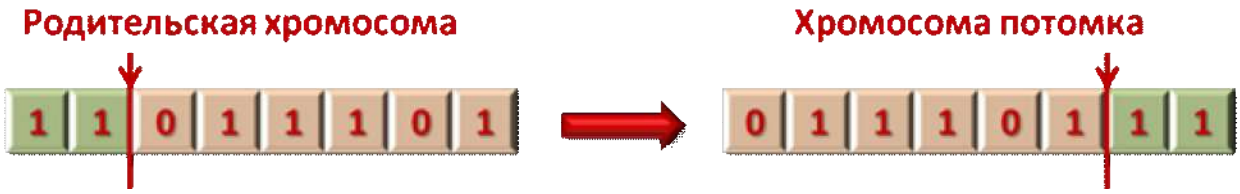
• *селекция* – выбор особей с наилучшей приспособленностью для воспроизводства (сортировка по значению целевой функции). Чем лучше приспособленность особи, тем выше ее шансы на скрещивание и наследование ее генов следующим поколением;

• *кроссовер* – скрещивание. Случайным образом выбирается точка разрыва – участок между соседними битами в строке. Обе родительские структуры разрываются на два сегмента по этой точке. Затем соответствующие сегменты различных родителей склеиваются, и получаются два генотипа потомков:



• *мутация* – случайное изменение генов. Случайным образом выбранный ген с некоторой вероятностью меняется на другой:

• *инверсия* – изменение порядка следования частей кода. Случайным образом выбирается точка разрыва – участок между соседними битами в строке. Обе части родительской структуры, разорванной по этой точке, меняются местами, после чего склеиваются:



Вначале ГА-функция генерирует определенное количество возможных решений (особей), а затем вычисляет для каждого приспособленность – близость к истине. Эти решения дают потомство (производится операция кроссовера). Более приспособленные решения имеют больший шанс к воспроизводству, а «слабые» особи постепенно «отмирают». Таким образом, происходит процесс эволюции.

На определенных этапах данного процесса происходят спонтанные изменения генов (мутации и инверсии). Полезные изменения, приводящие к увеличению приспособленности особи, дают свое потомство, в то время как «бесполезные» изменения «отмирают». После скрещивания, мутаций и инверсий снова определяется приспособленность особей нового поколения. Процесс повторяется до тех пор, пока не найдено решение или не получено достаточное к нему приближение.

В качестве *примера* применения генетического алгоритма рассмотрим задачу численного поиска решения.

Целевая функция будет иметь вид

$$f(x) \; . \quad in\left(\frac{\pi}{180}x\right) - \frac{1}{x} = 0$$

В качестве функции кроссовера будем использовать операцию нахождения среднего арифметического двух рассматриваемых точек. Для скрещивания выбираются несколько точек с наилучшим решением (со значением целевой функции, наиболее близким к нулю).

Мутацией будет являться операция генерации нового случайного числа рассматриваемой популяции. Инверсия будет изменять значение хромосомы на некоторую небольшую величину, таким образом осуществляя поиск в окрестностях точки с наилучшим решением.

Реализация на C++

```
    #define _USE_MATH_DEFINES
#include <iostream>
#include <cmath>
#include <ctime>
using namespace std;
double func(double x)
{
```

```
  return sin(M_PI * x / 180) - 1 / x;
}
double mutation(double x0, double x1)  // мутация: генерация случайной
величины
{
  const int NUM = 100000000;
  return fabs((double)((rand() * NUM) % (int)((x1 -
 x0)*NUM) + 1) / NUM) + x0;
}
double inversion(double x, double eps)  // инверсия: поиск в окрестно-
стях точки
{
  static int sign = 0;
  sign++;
  sign %= 2;
  if (sign == 0) return x - eps;
  else return x + eps;
}
void crossover(double *x, double eps, double x0, double x1)  // кроссовер:
среднее арифметическое
{
  int k = 99;
  for (int i = 0; i < 8; i++)
    for (int j = i + 1; j < 8; j++)
    {
      x[k] = (x[i] + x[j]) / 2;
      k--;
    }
  for (int i = 0; i < 8; i++)
  {
    x[k] = inversion(x[i], eps); k--;
    x[k] = inversion(x[i], eps); k--;
  }
  for (int i = 8; i < k; i++)
    x[i] = mutation(x0, x1);
}
void sort(double *x, double *y)  //сортировка
{
  for (int i = 0; i < 100; i++)
```

```
    for (int j = i + 1; j < 100; j++)
      if (fabs(y[j]) < fabs(y[i])) {
        double temp = y[i];
        y[i] = y[j];
        y[j] = temp;
        temp = x[i];
        x[i] = x[j];
        x[j] = temp;
      }
}
double genetic(double x0, double x1, double eps)  // поиск решения с ис-
пользованием ГА
{
  double population[100];
  double f[100];
  int iter = 0;
  for (int i = 0; i < 100; i++)   // Формирование начальной популяции
  {
    population[i] = mutation(x0, x1);
    f[i] = func(population[i]);
  }
  sort(population, f);
  do {
    iter++;
    crossover(population, eps, x0, x1);
    for (int i = 0; i < 100; i++)
      f[i] = func(population[i]);
    sort(population, f);
  } while (fabs(f[0]) > eps && iter<20000);
  cout << iter << " iterations" << endl;
  return population[0];
}
int main()
{
  srand(time(NULL));
  cout << genetic(1.0, 10.0, 0.000001);
  cin.get();
  return 0;
}
```

Результат выполнения

Применение генетических алгоритмов не всегда дает лучший результат по сравнению с другими методами. Однако этот метод имеет бесспорное преимущество при решении многомерных задач поиска глобального экстремума, содержащих значительное количество локальных экстремумов.

Генетический алгоритм – пример применения методики

Нужно найти с помощью генетического алгоритма, что кратчайший путь между двумя точками – это прямая.

Постановка задачи: заданы две точки на плоскости A и B. Между ними через N точек проложен маршрут. Нужно оптимизировать (минимизировать) путь, используя генетический алгоритм.

Решение: понадобятся начальное решение (одно или несколько), метод отбора и методы генерации новых потомков.

Начальное решение: случайно задаем на плоскости N точек маршрута, соединяющего точки A и B:

```
1    //массив точек, здесь N - кол-во промежуточных точек
2    var pts = [];
3    while (pts.length < N)
4      pts.push([Math.random(), Math.random()]);
```

Метод отбора*:* будем убирать из популяции самые длинные агенты (особи, маршруты). Но перед тем как убрать их оттуда, надо измерить длину пути:

```
1   //координаты точек A и B
2   var A = [0.2, 0.5];
3   var B = [0.8, 0.5];
4
5   //аккумулятор длины
6   var size = 0;
7   var lastP = A; //начинается маршрут с точки A
8   pts = ... //здесь у нас массив точек одного из агентов
9   for (var k=0; k < N; k++) {
10    //прибавляем отрезок пути – это гипотенуза = корень
11  из квадрата катетов
```

```
12    size += Math.sqrt(Math.pow(lastP[0] - pts[k][0], 2)
13          + Math.pow(lastP[1] - pts[k][1], 2));
14    lastP = pts[k];
15  }
16  //завершается маршрут в точке B
17  size += Math.sqrt(Math.pow(lastP[0] - B[0], 2) + Math.pow(lastP[1]
    - B[1], 2));
    /* теперь size содержит длину пути */
```

Генерация потомков. Простейшая мутация – это смещение одной или нескольких точек агента в произвольном направлении. Можно также реализовать кроссовер, скрещивание – будем брать часть одного маршрута и присоединять к нему часть другого маршрута.
addMutant – имитирует мутацию.

```
    // передаём номер агента в массиве, который является
    //отправной точкой для мутации
1   this.addMutant = function(agentIndex) {
2     //клонируем объект
3     var newAgent =
4   JSON.parse(JSON.stringify(this.agents[agentIndex]));
5     //выполняем изменение нескольких координат
6     for (k = 0; k < Math.floor(1 + Math.random() * Math.sqrt(this.N));
7   k++ ) {
8       var mutPosIndex = Math.floor(Math.random() * this.N);
9       var rMut = 0.10 * Math.random();
10      var alphaMut = 2 * Math.PI * Math.random();
11      //apply mutation
12      newAgent.p[mutPosIndex][0] += rMut * Math.sin(alphaMut);
13      newAgent.p[mutPosIndex][1] += rMut * Math.cos(alphaMut);
14    }
15    //добавляем агента в популяцию
16    this.agents.push(newAgent);
17  }
```

addCrossOver – имитирует скрещивание.

```
1
2
3
4   // нужны два агента для обмена генетической информацией
5   this.addCrossOver = function (agentIndex1, agentIndex2) {
```

```
    //узнаём, какую часть возьмем из первого предка
    var mutPosIndex = Math.floor(Math.random() * this.N);
    //клонирум агента
    var newAgent =
JSON.parse(JSON.stringify(this.agents[agentIndex1]));
    //дописываем гены из второго предка
    while (mutPosIndex < this.N) {
      newAgent.p[mutPosIndex][0] =
this.agents[agentIndex2].p[mutPosIndex][0];
       newAgent.p[mutPosIndex][1] =
    this.agents[agentIndex2].p[mutPosIndex][1];
     }
     //добавляем агента в популяцию
     this.agents.push(newAgent);
    }
```

Контрольные вопросы

1. Основные понятия, принципы и предпосылки генетических алгоритмов. Пример работы простого генетического алгоритма
2. Достоинства и недостатки генетических алгоритмов.
3. Дать определение однослойной нейронной сети.
4. Дать определение многослойной нейронной сети.
5. Дать определение следующих понятий «обучение нейронной сети», «обучающая выборка», «тестовая выборка», «перцептрон».
6. Дать определение понятия «знания».
7. Перечислите виды знаний, приведите примеры.
8. Укажите основные модели представления знаний, приведите примеры.
9. Какие модели представления знаний распространены в экспертных системах?
10. В основе какой модели представления знаний лежит понятие формальной системы?
11. Из чего состоит база знаний при использовании продукционной модели?
12. В чем различия прямого и обратного вывода на знаниях?
13. Каковы основные стратегии управления выводом, позволяющей минимизировать время поиска решения в базах знаний?

14. Дать определение понятия «язык представления знаний».

15. Каковы характерные признаки и особенности экспертных систем?

16. Перечислите основные стадии разработки экспертных систем.

17. Какие трудности возникают при разработке экспертных систем?

18. В чем основное отличие экспертных систем от традиционных программ?

19. Что такое искусственные нейронные сети?

20. Что представляет собой нейрон математически?

21. В чем сходства и различия экспертных и нейросетевых систем?

22. Какую нейросеть можно назвать обобщенным многослойным персептроном?

23. Приведите примеры программного обеспечения для работы с нейронными сетями.

24. Перечислите основные формы функций принадлежности.

25. Каковы наиболее известные методы дефаззификации?

26. В чем заключаются достоинства и недостатки генетических алгоритмов?

Ответы к тестовым заданиям

Раздел 1

1	2	3	4	5	6	7	8	9	10
б	в	а	г	в	в	б	а	б	4, 1, 3, 2

Раздел 2

1	2	3	4	5
а	б	1г, 2б, 3е, 4а, 5ж, 6в, 7д	1г, 2в, 3б, 4а	г

ЛИТЕРАТУРА

1. Евгенев, Г.Б. Интеллектуальные системы проектирования: учеб. пособие / Г.Б. Евгенев. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2009. – 334 с.

2. Евгенев, Г.Б. Онтология инженерных знаний / Г.Б. Евгенев // Информационные технологии. – 2001. – № 5. – С. 2 – 5.

3. Евгенев, Г.Б. Многоагентные САПР в машиностроении / Г.Б. Евгенев, А.С. Кобечев // Информационные технологии. – 2003. – № 11. – С. 19 – 24.

4. Емельянов, В.В. Теория и практика эволюционного моделирования / В.В. Емельянов, В.В. Курейчик, В. М. Курейчик. – М.: Физматлит, 2003.

5. Клир, Дж. Системология. Автоматизация решения системных задач: [пер. англ.] / Дж. Клир. – М.: Радио и связь, 1990.

6. Амиров, Ю.Д. Основы конструирования: творчество – стандартизация – экономика / Ю.Д. Амиров. – М.: Изд-во стандартов, 1991.

7. Колесов, Ю.Б. Объектно-ориентированное моделирование сложных динамических систем / Ю.Б. Колесов. – СПб.: Изд-во СПбГПУ, 2004.

8. Норенков, И.П. Основы автоматизированного проектирования / И.П. Норенков. – 3-е изд., перераб. и доп. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2006.

9. Гаврилова, Т.В. Базы знаний интеллектуальных систем / Т.В. Гаврилова, В.Ф. Хорошевский. – СПб.: Питер, 2000.

10. Вендров, А.М. Методы и средства моделирования бизнес-процессов (обзор) [Электронный ресурс] / А.М. Вендров. – Режим доступа: http://www.jetinfo.ru/2004/10/1/article1.10.2004.html.

11. Тарасов, В.Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика / В.Б. Тарасов. – М.: Эдиториал УРСС, 2002.

12. Ли, К. Основы САПР (CAD/CAM/CAE) / К. Ли. – СПб.: Питер, 2004.

13. Питерсон, Дж. Теория сетей Петри и моделирование систем: [пер. с англ.] / Дж. Питерсон. – М.: Мир, 1984.

14. Казиев, В.М. Введение в анализ, синтез и моделирование систем: учебное пособие / В.М. Казиев. – Интернет-ун-т информ. технологий.

15. Советов, Б.Я. Моделирование систем: учебник для студ. вузов, обуч. по специальностям «Информатика и вычислительная техника» и «Информационные системы» / Б.Я. Советов, С.А. Яковлев. – 5-е изд., стереотип. – М. : Высш. шк., 2007. – 343 с. : ил.

16. Основы компьютерного проектирования и моделирования радиоэлектронных средств в среде Micro-Cap: метод. указания к лабораторным работам / сост. А.Н. Копысов, Е.М. Зайцева. – Ижевск: Издательство ИжГТУ, 2012. – 66 с.

17. Прикладная математическая статистика [Электронный ресурс]: учебное пособие. – Электрон. текстовые данные. – Томск: Томский государственный университет систем управления и радиоэлектроники, 2016. – 113 с. – 2227-8397. – Режим доступа: http://www.iprbookshop.ru/72166.html

18. Норенков, И.П. Автоматизированные системы управления технологическими процессами [Электронный ресурс] / И.П. Норенков // Вестник МГТУ. Сер. Приборостроение. 2002. – № 1. – Режим доступа: https://lib-bkm.ru/load/19-1-0-196.

19. Представление знаний в информационных системах: учебник для студентов вузов / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – М.: Академия, 2011. – 144 с.

20. Колесов, Ю.Б. Моделирование систем. Практикум по компьютерному моделированию/ Ю.Б. Колесов. – БЧВ-Петербург, 2007. – 352 с.

21. Зограф, Ф.Г. Основы компьютерного проектирования и моделирования РЭС. Учебный практикум. [Электронный ресурс] / Ф.Г. Зограф. – Красноярск: Сиб. федерал. ун-т, 2011. – 120 с.

22. Лобанова, В.А. Информационные технологии проектирования электронных средств [Электронный ресурс]: учеб. пособие для вузов / В.А. Лобанова. – Орел: Изд-во ОрелГТУ, 2009. – 176 с. – Режим доступа: http://elib.oreluniver.ru/uchebniki-i-uch-posobiya/informacionnye-tekhnologii-proektirovani.html.

23. Корячко, В.П. Теоретические основы САПР / В.П. Корячко, В.М. Курейчик, И.П. Норенков. – М.: Энергоиздат, 1987. – 400 с.

24. Муромцев, Ю.Л. Задачи трассировки печатных плат: лабораторные работы / Ю.Л. Муромцев, В.В. Трейгер, В.Н. Грошев. – Тамбов: ТИХМ, 1990. – 32 с.

25. Муромцев, Ю.Л. Задачи размещения радиоэлектронной аппаратуры: лабораторные работы / Ю.Л. Муромцев, В.В. Трейгер, В.Н. Грошев. – Тамбов: ТИХМ, 1988. – 32 с.

26. Леоненков, А.В. Нечеткое моделирование в среде MATLAB fuzzy TECH / А.В. Леоненков. – СПб.: БХВ-Петербург, 2003. – 736 с.

27. Федорчук, В.Г. Искусственные нейронные сети [Электронный ресурс] / В.Г. Федорчук. – Режим доступа: http://bigor.bmstu.ru/?cnt/?doc=NN/base.cou.

28. Норенков, И.П.Эволюционные методы для решения задач проектирования и логистики [Электронный ресурс] / И.П.Норенков. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2006. – Режим доступа: http://bigor.bmstu.ru/?cnt/?doc=NN/base.cou.

29. Головицына, М.В. Интеллектуальные САПР для разработки современных конструкций и технологических процессов [Электронный ресурс] / М.В. Головицына. – 2-е изд. – М. : Интернет-университет информационных технологий (ИНТУИТ), 2016. – 249 с. – Режим доступа: http://www.iprbookshop.ru/73681.html.

30. Джонс, М.Т. Программирование искусственного интеллекта в приложениях [Электронный ресурс] / М.Т. Джонс. – М.: ДМК Пресс, 2011. – 312 с. – Режим доступа: http://e.lanbook.com/books/element.php?pl1_cid=25&pl1_id=1244 .

31. Матвеев, М.Г. Модели и методы искусственного интеллекта. Применение в экономике [Электронный ресурс] / М.Г. Матвеев, А.С. Свиридов, Н.А. Алейникова. – М.: Финансы и статистика, 2011. – 448 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=220187&sr=1

32. Применение искусственных нейронных сетей и системы остаточных классов в криптографии [Электронный ресурс] / Н.И. Червяков [и др.]. – М.: Физматлит, 2012. – 280 с. – Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=5300 .

33. Лобанова, В.А. Информационные технологии проектирования электронных средств: учебное пособие / В.А. Лобанова, О.А. Воронина. – Орел: ОГУ имени И.С. Тургенева, 2018. – 257 с.

Электронные ресурсы:

Электронная библиотека образовательных ресурсов (ЭБОР) <http://elib.oreluniver.ru/>

Электронная библиотечная система «Лань» <http://www.e.lanbook.com/>

Электронная библиотечная система «IPRbooks» <http://www.iprbookshop.ru/>.

Математика / Российское образование (федеральный портал). – http://window.edu.ru/library?p_rubr=2.2.74.12

Кибернетика, нейросети / Нехудожественная библиотека. – http://www.nehudlit.ru/books/subcat261.html
http://www.intuit.ru/studies/courses/651/507/lecture/11535
http://www.intuit.ru/studies/courses/651/507/lecture/11537
http://www.intuit.ru/studies/courses/651/507/lecture/11547
http://www.intuit.ru/studies/courses/651/507/lecture/11539
http://www.intuit.ru/studies/courses/651/507/lecture/11541
http://www.intuit.ru/studies/courses/651/507/lecture/11543
http://www.intuit.ru/studies/courses/651/507/lecture/11545
http://www.intuit.ru/studies/courses/651/507/lecture/11547
http://www.intuit.ru/studies/courses/651/507/lecture/11549
http://staratel.com/iso/ISO9000/Article/ConImp/ContImprove.htm.
http://www.cfin.ru/itm/bpr/idef0_bpr.shtml.
http://staratel.com/iso/ISO9000/Article/ConImp/ContImprove.htm.

Учебное издание

Лобанова Валентина Андреевна
Воронина Оксана Александровна

ПРИНЦИПЫ ОРГАНИЗАЦИИ САПР С ЭЛЕМЕНТАМИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Учебное пособие

Редактор Т.Д. Васильева
Технический редактор Т.П. Прокудина

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Орловский государственный университет имени И.С. Тургенева»

Подписано к печати 05.07.2021 г. Формат 60×90/16.
Усл. печ. л. 10,5. Тираж 100 экз.
Заказ №_____

Отпечатано с готового оригинал-макета
на полиграфической базе ОГУ имени И.С. Тургенева
302026, г. Орел, ул. Комсомольская, 95.