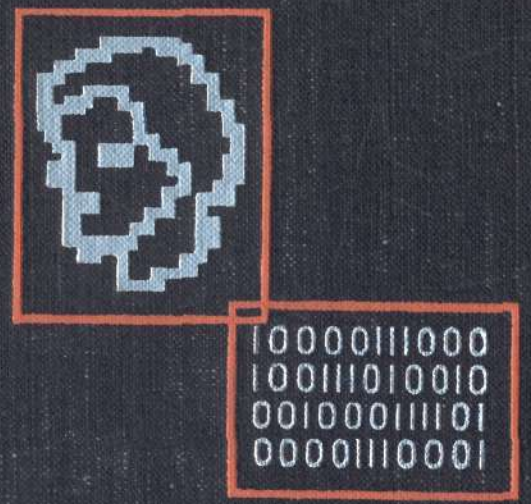
ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ и МЫШЛЕНИЕ



COMPUTERS AND THOUGHT

A collection of articles

Edited by
EDWARD A. FEIGENBAUM
and
JULIAN FELDMAN
University of California
Berkeley

McGRAW-HILL BOOK Co.
NEW YORK—SAN FRANCISCO—TORONTO—LONDON
1963

ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ И МЫШЛЕНИЕ

ПОД РЕДАКЦИЕЙ
Э. ФЕЙГЕНБАУМА
и
Дж. ФЕЛЬДМАНА

Перевод с английского
под редакцией
Э. М БРАВЕРМАНА,
А. В. НАПАЛКОВА и
Ю. В. ОРФЕЕВА

ИЗДАТЕЛЬСТВО «МИР»
МОСКВА 1967

УДК 681.142/155

Книга посвящена одному из самых интересных и перспективных направлений современной кибернетики — так называемому «эвристическому программированию», моделирующему сложные формы психической деятельности человека на уровне «информационных» процессов.

В книге собраны статьи известных американских исследователей, сообщающие о конкретных разработках и результатах экспериментов в этой области.

Часть I содержит работы по созданию «разумных машин» (машины, играющие в шахматы и шашки, доказывающие логические и геометрические теоремы, вычисляющие интегралы, рационализирующие производственные процессы, распознающие зрительные образы, отвечающие на вопросы, заданные на человеческом языке).

В части II собраны работы по исследованию и моделированию психологии «разумного поведения» («Общий решатель проблем», моделирующий процессы мышления человека при решении задач, модели вербального обучения и формирования понятий, моделирование поведения человека в условиях двоичного выбора, моделирование работы банковского служащего по размещению капиталов, моделирование социальных взаимоотношений людей в малой группе)

Обзорная статья Минского дает представление о современном состоянии работ по созданию «искусственного разума».

Самостоятельную ценность представляет помещенная в конце книги обширная библиография, снабженная оригинальным тематическим указателем.

Книга представляет большой интерес для широких кругов научных работников, инженеров и исследователей в области кибернетики, автоматики и вычислительной техники, для математиков, экономистов, нейрофизиологов, психологов и философов. Отсутствие сложного математического аппарата и понятность изложения делают ее доступной широкому кругу читателей.

Редакция литературы по новой технике

Инд. 3-3-14

ВСТУПИТЕЛЬНАЯ СТАТЬЯ

Настоящая книга посвящена актуальным проблемам современной кибернетики, связанным с исследованием и моделированием сложных форм психической деятельности. Она отражает одно из новых направлений, которое обычно определяется как «эвристическое программирование».

Книга охватывает все основные разделы этой области науки: решение задач, доказательство теорем, формирование понятий, выдвижение гипотез, игру в интеллектуальные игры — шахматы и шашки, а также создание программ, способных обучаться и отвечать на вопросы.

Эвристическое программирование как научное направление сформировалось относительно недавно. Однако уже в 1958 г. на симпозиуме по «механизации процессов мышления», который состоялся по инициативе Национальной физической лаборатории в Теддингтоне (Англия), в докладах М. Минского и Д. Маккея был поставлен вопрос о необходимости изыскания новых путей для исследования и моделирования творческих форм работы мозга [1].

Хотя кибернетика к этому времени достигла значительных результатов, связанных с созданием ряда разделов математической теории переработки информации, сохранялся большой разрыв между теоретической кибернетикой и исследованиями психических процессов. При этом возникли серьезные сомнения в том, что этот разрыв можно ликвидировать на путях развития «классической» кибернетики.

При решении этой проблемы большую роль, конечно, должна была сыграть корректная постановка задачи. Необходимо было более четко ответить на вопрос, что такое мышление и в чем именно заключается специфика «творческих форм работы мозга»

[1] В отечественной литературе эти вопросы рассматривались, например, Л. М. Веккером; см. Веккер Л. М., «Об осязательном образе как регуляторе движения руки», *Ученые записки ЛГУ*, № 203 (1956); Веккер Л. М., Ломов Б. Ф., «О чувственном образе как изображении», *Вопр. филос.*, № 4 (1961); Веккер Л. М., Восприятие и основы его моделирования, Изд-во ЛГУ, 1964.

по сравнению с известными в кибернетике процессами переработки информации. Эта важная задача была частично разрешена в докладах М. Минского и Д. Маккея и в последующей работе М. Минского «На пути к созданию искусственного разума». К этому же времени А. Ньюэллом, Г. Саймоном и Дж. Шоу была начата разработка основных теоретических положений нового направления кибернетики и созданы программы, моделирующие некоторые сложные формы работы мозга.

Эти теоретические положения изложены во вступительных статьях к частям и разделам настоящей книги. Наиболее существенным являлось утверждение о возможности разложения всех сложных форм так называемой «творческой» деятельности мозга на «элементарные информационные процессы». При этом решающее значение придавалось рассмотрению той структуры, в которую объединялись элементы.

Было показано, что одни и те же элементарные процессы, объединенные в различные комплексы, могут обеспечить появление таких различных форм работы мозга, как, например, распознавание образов или решение задач.

Другим важным положением являлось четкое определение понятия «эвристика». Строгое определение понятия «алгоритм», существовавшее в математике, предполагало, что открытие алгоритма означает нахождение метода точного решения задач определенного класса. При этом из поля зрения исследователей ускользнула большая категория явлений природы, связанных с решением задач в условиях, когда было невозможно заранее найти эффективные методы. В этих случаях невозможно гарантировать также оптимальность решения, однако можно предложить такую тактику поиска, которая оказывается значительно более эффективной, чем простой перебор. Было показано, что именно такие «эвристические тактики» наиболее характерны для работы мозга и именно они лежат в основе творческих способностей человека.

Открытие этой области явлений природы и возможность точного ее описания при помощи информационных языков программирования имеют очень большое значение в науке и, в частности, для моделирования сложных форм работы мозга.

Разработка этих положений привела к тому, что оказалось возможным перейти к моделированию таких сложных форм работы мозга, как решение проблем, выдвижение гипотез, самообучение и т. д., что было невозможно на основе созданной ранее теории кибернетики.

Эвристическое программирование открыло новые пути для решения ряда центральных проблем кибернетики. Например, до последнего времени большое внимание уделялось «самоорганизующимся системам управления». Этот вопрос встал в связи с тем, что теория автоматов не могла объяснить способности живых организмов к быстрому формированию целесообразного поведения в новых условиях. Решение этой проблемы искали на путях более полного исследования и моделирования нейронов. Высказывалось предположение, что в нервных сетях, состоящих из таких «пластических» нейронов, проявятся еще не известные в кибернетике свойства и способности. При взаимодействии с внешним миром они сами смогут целесообразно изменять свою структуру и создавать такие новые системы связей между элементами, которые обеспечат целесообразное поведение.

Это направление натолкнулось, однако, на большие трудности, связанные с отсутствием математического аппарата для описания работы нейронных сетей из пластических нейронов, а также с тем, что не существовало точных критериев оценки тех свойств, которыми должны были обладать эти нейронные сети. В том случае, если их работа связывалась с физиологическими понятиями (условный рефлекс, торможение и т. д.), моделирование было элементарно простым, однако полученные при этом общие результаты не представляли интереса для кибернетики. При попытках оперировать психологическими понятиями препятствием становилась неопределенность таких понятий, как «сознание», «мышление» и т. п.

Эвристическое программирование рассматривает «самоорганизацию» как результат работы эвристических программ различных уровней. При этом программа высшего уровня, не отвечая за конкретное поведение, содержит ряд правил переработки информации, которые обеспечивают формирование нового поведения в новых условиях. Таким образом, центр тяжести исследования переносится на выявление структуры программ и описание эвристик. Делается вывод о том, что явление самоорганизации невозможно объяснить только на основе свойств нейрона, что для этого необходимо использование сложного комплекса правил переработки информации, поступающей извне, и определенной тактики поисковых действий, направленных на исследование внешней ситуации.

В качестве материальной основы работы таких программ исследуются списковая структура организации памяти и некоторые другие проблемы, тесно связанные с теорией программирования.

Проблема самоорганизующихся систем тесно связана с *самообучением*. Изучение и моделирование самообучения привлекло большое внимание к кибернетике. Были созданы многочисленные модели стохастического обучения. Большое количество моделей было связано с воспроизведением способности живых организмов к выработке условнорефлекторных реакций. В связи с этим

значительный интерес представляет также теория самообучающихся матриц. Большая серия работ была связана со статистической теорией обучения распознаванию образов.

Эвристическое программирование, однако, внесло ряд изменений в постановку задачи исследования самообучения. Стало ясно, что до последнего времени кибернетика имела дело только с простейшими формами самообучения, эффективными в простейших информационных ситуациях. Теоретически и экспериментально было показано, что вскрытые ранее простые принципы не могут оказаться эффективными в более сложных условиях среды и что для исследования механизмов самообучения необходимо рассмотрение сложных эвристических программ различных уровней. Программы должны включать подпрограммы, определяющие условия формирования новых, «подкрепляющих» раздражителей разных типов, и блоки программ, обеспечивающие осуществление специальной тактики использования такого «многостепенного» подкрепления при отборе внешней информации и формировании новых систем условных рефлексов.

Самообучение требует специальной организации памяти, состоящей из нескольких отделов, способной обеспечить запись древовидных структур, вписывание в эти структуры новых звеньев в любые пункты без изменения записи в соседних ячейках памяти, а также обеспечить перегруппировку структур.

При изучении механизмов, обеспечивающих самообучение такого вида, было показано, что эти механизмы не могут быть сведены к одной или нескольким формулам и принципам, и были выявлены конкретные структуры сложных эвристических программ. Важно отметить, что при этом было невозможно найти одну универсальную программу самообучения, применимую в любых условиях. Оказалось необходимым выделить различные классы информационных ситуаций, имеющих место при обучении, и для каждого из классов создавать свою собственную эвристическую программу. Наряду с этим было доказано существование программы более высокого уровня, которая, не обеспечивая самого процесса самообучения, производит специфический анализ ситуации, в которой происходит самообучение. Результатом такого анализа является выбор той или иной тактики самообучения (одной из программ более низкого уровня). Если применение этой программы не оказывается эффективным, то программа высшего уровня в связи с анализом результатов работы программы низшего уровня принимает решение о включении другой программы самообучения или об изменении той программы низшего уровня, которая использовалась ранее.

Таким образом, в целом процесс самообучения обеспечивается весьма сложной системой программ разных уровней, в ходе выполнения которых происходят перестройка и перегруппировка самих программ самообучения.

Подобные же изменения в методах исследования намечаются и в других областях кибернетики. Это может привести к большим практическим результатам, связанным с существенным расширением сферы применения вычислительной техники при управлении промышленностью и экономикой, а также значительно повысит эффективность управления. В ряде случаев трудности использования математических методов в управлении большими системами связаны с тем, что оказывается необходимым сбор и переработка очень большого количества информации. Часто это создает непреодолимые препятствия для автоматизации управления. Эвристические методы принятия решения могут в этих условиях оказаться незаменимыми.

Традиционные методы применения математического анализа и вычислительной техники обычно требовали на первом этапе предварительной «арифметизации» исследуемого процесса, другими словами, четкой математической постановки задачи. Второй этап обычно предполагает численный анализ сформулированной задачи. Применение вычислительных машин в этом случае ограничивается численной переработкой информации. Такой способ применения электронных вычислительных машин, однако, не всегда является плодотворным. Например, решение некоторых классов комбинаторных задач, которые характерны для ряда проблем оптимального экономического планирования и исследования операций, не удалось получить средствами современного математического анализа.

Трудности в создании содержательных математических моделей экономических и социальных процессов носят принципиальный характер и являются следствием неадекватности языка количественных (квантификационных) теорий тем исключительно сложным процессам переработки информации, которые лежат в основе организации целесообразного поведения человека в социально-экономических системах. В настоящее время становится очевидным, что арсенал средств (линейное, динамическое программирование, теория игр и т. п.), которые обычно используются при количественном анализе проблем оптимального планирования, явно недостаточен. Эти модели и инструменты анализа не в состоянии отразить активный, творческий характер экономического планирования. Ограниченность количественных методов анализа экономических процессов может быть в настоящее время преодолена средствами эвристического программирования.

Еще в 1958 г. А. Ньюэлл и Г. Саймон предложили разделить задачи исследования операций и экономического планирования

на два класса: задачи с *хорошей структурой* (well structure) и задачи с *плохой структурой* (ill structure). К задачам с хорошей структурой относят задачи, допускающие четкую математическую постановку и решение. К задачам же с плохой структурой относятся задачи, для которых отсутствуют четкая математическая постановка или эффективные методы численного решения.

Задачи с плохой структурой могут в настоящее время решаться средствами эвристического программирования, так как оно не предполагает создания какой-либо алгебраической или цифровой модели процесса, а модель процесса сразу же строится в форме программы нечисленной переработки информации.

В нашей стране вопрос о роли количественных математических методов служил предметом острых дискуссий. Центральным вопросом этих дискуссий фактически был вопрос о том, возможно ли перенести те традиционные подходы и методы, которые сложились при применении языка математики в физике и технике, на социально-экономические процессы. Подробно дискуссия экономистов и математиков изложена в книге «Экономисты и математики за круглым столом», изд-во «Экономика», 1965.

Эвристическое программирование интерпретирует экономическое и социальное планирование как творческий познавательный процесс, в котором план выступает как специфический инструмент организации процесса. Таким образом, в широком смысле слова проблемы оптимального планирования не могут рассматриваться как задачи нахождения экстремальных решений каких-либо уравнений [1].

Идеи и методы эвристического программирования имеют важное значение для развития экономических и организационных теорий. В экономических и организационных науках давно ощущался разрыв между теориями поведения человека, принятыми этими науками, и соответствующими теориями, выдвинутыми психологической наукой. Возможность, опираясь на методы эвристического программирования, создать информационные модели поведения человека в процессах принятия решений позволяет частично ликвидировать этот разрыв и поставить на службу экономической и организационной наукам весь комплекс наук, изучающих поведение человека.

Хорошим примером практического применения эвристических программ в организации производства является программа балансирования работы сборочного конвейера, разработанная в 1960 г. Тонгом и публикуемая в настоящем сборнике. Ценой некоторой потери оптимальности эвристическая программа позволила добиться выигрыша в простоте и времени решения

[1] См., например, выступление проф. А. Я. Боярского, опубликованное в указанной выше книге.

в несколько десятков раз по сравнению с математическими методами. Очевидно, что в случае управления сложными процессами, когда невозможно использование математических методов, эвристические программы окажутся очень полезными.

Большие перспективы открывает эвристическое программирование и для автоматизации процесса творческой деятельности, и прежде всего научно-исследовательских работ. Здесь речь идет, конечно, не о замене научных работников, а о радикальном повышении эффективности и производительности их труда. Известно, что исследовательские работы наших дней требуют больших капиталовложений, причем эффективность работ во многом зависит от правильного планирования хода экспериментов. Имеются большие возможности для оптимизации этого процесса и для приближения сроков научных открытий на основе использования эвристических программ выдвижения гипотез, формирования понятий и планирования экспериментов.

Большое значение имеет эвристическое программирование и для исследования мозга: оно вносит радикальные изменения в методы исследования, что оказывается в свою очередь важным для медицины, психологии, социологии, педагогики, для развития правовых наук и программированного обучения.

Основные трудности при исследовании механизмов работы мозга до последнего времени были связаны с отсутствием технических средств, которые могли бы обеспечить использование метода *экспериментального синтеза*. Как известно, при исследовании механизмов биологических процессов решающую роль играет расчленение сложных процессов на элементы с последующим применением метода экспериментального синтеза. При этом путем искусственного создания из элементов определенных сложных структур и воспроизведения в них исследуемых процессов последовательно вскрываются механизмы явлений. Например, при исследовании методом экспериментального синтеза явления фотосинтеза в растениях пытаются воспроизвести это сложное явление в системе, отличной по своей природе от живых организмов.

Но, рассматривая возможности использования аналогичных методов при исследовании мозга, следует особо подчеркнуть, что в этом случае предметом исследования являются не физические или химические процессы, а процессы переработки информации. В качестве «элементов» здесь следует рассматривать элементарные правила переработки информации, такие, как сравнение двух символов, запись символов в памяти и т. д. «Экспериментальный синтез» в этом случае подразумевает построение сложных программ, состоящих из особым образом организованных структур элементарных правил.

Отсутствие эффективных методов исследования механизмов психических процессов приводило вплоть до недавнего прошлого к тому, что при изучении нервной системы наблюдался глубокий разрыв между исследованиями бихевиористов и нейрофизиологов, с одной стороны, и представителей гештальт-психологии — с другой. В первом случае исследование ограничивалось анализом элементарных процессов, скажем исследованием корреляции между изменениями биоэлектрической активности в различных отделах центральной нервной системы при воздействии элементарных сигналов или корреляции между поступлением сигналов и ответами нервных клеток и т. п. Во втором случае сложные психические процессы исследовались как некое целостное, неразложимое явление, что исключало возможность выделения в нем элементарных процессов и использования методов экспериментального синтеза.

Такое состояние научных исследований, которое можно, пожалуй, в какой-то мере сравнить с периодом алхимии в истории развития химии, не создавало предпосылок для вскрытия механизмов психических процессов и их моделирования.

Уже на первых этапах развития эвристического программирования было высказано утверждение о возможности выделить новый «уровень исследования» мозга — *уровень информационных процессов*. На этом уровне исследования ученых не интересовал вопрос о структуре нервных сетей головного мозга и о том, какие именно отделы мозга ответственны за ту или иную форму переработки информации; ставилась цель лишь досконально и точно выяснить сами законы переработки информации, доводя это знание до программ, состоящих из элементарных информационных процессов. При этом делался вывод о том, что сходство в работе мозга и машины может быть доказано только на этом уровне.

Такой подход отражен и в настоящей книге.

Однако дальнейшее развитие науки показало, что эвристическое программирование имеет большое значение не только на информационном уровне исследования, но и для таких вопросов физиологии, как выяснение функций различных отделов мозга, структуры мозга, взаимосвязи его отделов и т. д.

При разработке эвристических программ выясняются сложные взаимодействия блоков и сложная картина структуры потоков и переработки информации. Методами теоретической кибернетики эта сложная структура может быть проанализирована с точки зрения ее оптимальности и может быть дана сравнительная оценка различных возможных в этом случае вариантов. Результатом такого рассмотрения может явиться создание двух или трех теоретически обоснованных гипотез о структуре информационных процессов головного мозга. Эти гипотезы оказываются значительно более сложными, чем существующие в физиологии представления о замыкании связей, иррадиации и концентрации нервных процессов.

Экспериментальное исследование после этого уже может быть организовано не как слепое выявление взаимодействия каких-либо отделов мозга, а как проверка или опровержение применимости к работе мозга того или иного принципа организации структуры, например подтверждение или опровержение идеи о раздельном существовании блока выработки новых подкрепляющих раздражителей и блока, обеспечивающего стратегию использования этих подкрепляющих сигналов при самообучении. Использование таких гипотез коренным образом меняет всю схему и методы исследования головного мозга. Оно обеспечивает эффективное взаимодействие кибернетической теории и физиологического опыта.

Таким образом, эвристическое программирование способно ликвидировать тот разрыв, который создался в последнее время между психологией и физиологией и между теоретической кибернетикой и исследованием мозга. Эти методы, видимо, уже в ближайшие годы окажут существенное влияние на исследование мозга, поскольку до последнего времени использование новых методических возможностей, связанных, например, с регистрацией биотоков отдельных нейронов и т. д., основывалось только на очень примитивных представлениях об информационных процессах.

Вопрос о роли эвристического программирования в исследовании мозга еще не получил должного освещения в литературе. Большой интерес в этом отношении представляет книга Д. И. Миллера, Е. Галантера и К. Прибрама «Планы и структура поведения», изд-во «Прогресс», 1964. Следует, однако, отметить, что в этой книге не дается полного анализа проблемы. Большой интерес в этой связи представляет также книга В. Рейтмана (Reitman W. R., Cognition and Thought. An Information-Processing Approach, Wiley, New York — London — Sydney, 1965), в которой делается попытка создания теории работы мозга на основе представления о *списковой структуре* организации памяти и переработки информации.

В настоящее время актуальное значение приобретает дальнейшая переработка методологии исследования мозга методами эвристического программирования. Хотя уже очевидна неэффективность попыток понять работу мозга, опираясь только на физические и химические методы исследования, еще не сформулированы четко методы, основанные на науке о переработке

информации, которые могут оказаться эффективным средством для вскрытия механизмов работы мозга.

Очевидно, что при определении функции отделов мозга неправомерно сводить их к торможению и возбуждению или пытаться связывать отдельные правила переработки информации типа «принципа доминанты» непосредственно со структурами мозга.

Однако в то же время не разработана и точная методика проверки приложимости тех сложных структур переработки информации, которые возникают при разработке эвристических программ, к работе мозга. Разработка всех этих вопросов, видимо, окажет решающее влияние и на исследование высшей нервной деятельности.

Наряду с большими успехами в моделировании сложных форм работы мозга следует, однако, признать, что эвристическое программирование находится только на первых этапах своего развития. В настоящее время приходится больше говорить о принципиальном значении этих методов, чем о полном решении проблемы моделирования способности мозга к решению задач, доказательству теорем или построению гипотез. Во всех этих случаях созданы программы только для узкого класса информационных задач, подпадающих под общие понятия «решения задач» или «формирования гипотез». Трудности заключаются в том, что в настоящее время невозможно точно определить как сами эти понятия, взятые часто из психологической терминологии, так и те классы информационных задач, которые решаются или не решаются уже созданными эвристическими программами.

Во всяком случае, программы, приводящиеся в настоящем сборнике, отражают только относительно простые случаи формирования понятий, решения задач и распознавания образов.

В этой связи большой интерес представляет помещенная в книге статья М. Минского, в которой дается глубокая и содержательная оценка путей развития кибернетики и ее современного состояния. Значительный интерес в ней, в частности, представляет сопоставление задач и методов эвристического программирования с методами других направлений кибернетики.

Большой интерес представляет также статья Э. Фейгенбаума, связанная с анализом процесса обучения. Обучение и распознавание — области, в которых оказалось наиболее трудно использовать методы эвристического программирования. Это связано с тем, что данные формы деятельности мозга относятся к сфере интуитивных процессов, о характере которых человек обычно не может дать словесного отчета и за которые в мозге ответственны «программы» более высоких уровней.

В то время как такие формы работы мозга, как, например, работа банковского служащего или игра в шахматы, имеющие более специализированный характер и формирующиеся в процессе индивидуальной жизни человека, сравнительно удобны для моделирования, обучение как более общее свойство мозга, присущее всем людям, является значительно более трудным объектом. Однако именно эти формы работы мозга в связи с их общностью представляют и наибольший интерес В этой связи следует также специально обратить внимание на методы проверки тождественности программы и работы мозга, которые, в частности, изложены в статье Дж. Фельдмана и которые вызывают в настоящее время большие споры среди психологов.

В настоящее время отчетливо выявляются трудности, стоящие на путях развития эвристического программирования.

Первая категория трудностей связана с *экспериментальным выявлением* эвристик и алгоритмов работы мозга. Как уже говорилось, во многих случаях работа мозга носит интуитивный характер и человек не может дать о ней словесного отчета. При объективном же исследовании поведения исследователь сталкивается с очень большим многообразием, в котором, казалось бы, невозможно установить более общие законы переработки информации.

Большие трудности возникают и в связи с тем, что в эксперименте с реальными предметами (например, в опытах на обезьянах с палками, ящиками и т. п.), по общеизвестной методике В. Келлера, даже при самых точных способах записи экспериментальных данных выявляются только частные поведенческие акты. Поведение всецело определяется предыдущим опытом, характером предметов и т. д. Результаты опытов получаются в форме большого количества совершенно различных по характеру протоколов, из которых очень трудно вывести данные для написания эвристических программ (последние, очевидно, должны обладать свойствами массовости и детерминированности).

Намечаются два пути преодоления этих трудностей. Первый связан с разработкой методов *математической обработки* результатов (протоколов опытов) на цифровых вычислительных машинах с целью выявления инвариантных участков и отыскания общих для всех случаев принципов переработки информации. Второй путь связан с созданием *абстрактных моделей* информационных ситуаций и проведением экспериментов на основе использования этих абстрактных моделей. Типичным примером использования такой методики является приведенная в настоящем сборнике работа А. Ньюэлла, Г. Саймона и Дж. Шоу, связанная с созданием программы «общего решателя проблем» (GPS). Оказалось, что некоторый класс информационных задач,

связанных с решением проблем человеком, можно формально описать при помощи языка, близкого к языку теории ассоциативных исчислений. Опыты, проведенные по этой методике, позволяют выявить принципы, применимые к целому классу информационных задач (свойство массовости) и общие для всех испытуемых. Кроме того, они проявляются в четкой форме, удобной для составления эвристических программ (свойство детерминированности).

В настоящее время, однако, такое формальное описание удалось создать только для относительно простых форм работы мозга. Для того чтобы иметь возможность формализовать более сложные информационные ситуации, необходимо прежде всего разработать соответствующие математические языки. Современные средства математики, видимо, недостаточны для решения этой проблемы. Следует также отметить, что для проведения более сложных опытов необходима разработка специализированных вычислительных устройств, так как эти опыты уже невозможно проводить вручную.

Другая категория трудностей связана с необходимостью более строгой и естественной *классификации* информационных задач, решаемых головным мозгом. Классификация, очевидно, должна быть построена на основе четкого определения систем ограничений, на которых базируется работа той или иной эвристической программы. Отсутствие классификации приводит в настоящее время к тому, что часто программу, пригодную для решения ограниченного круга информационных задач, начинают применять для других целей или сравнивать с работой мозга при совершенно других условиях. На основе таких неправильных сравнений часто делаются выводы о том, что эвристические программы не могут рассматриваться как модели работы мозга, или же выводы о недостаточной эффективности самого метода эвристического программирования.

Существенно также то, что без четкого определения границ применимости каждой программы очень трудно правильно поставить задачу для создания новых эвристических программ. Только после того, как удастся точно определить область применимости уже созданных в настоящее время программ (например, программы построения гипотез) и установить те системы ограничений, которые лежат в основе их работы, можно будет описать такие классы информационных ситуаций, которые типичны для работы мозга, но не охвачены ранее созданными программами. На этой основе можно четко поставить задачу для разработки новых эвристических программ.

На основе классификации и точного определения ограничений можно будет для каждой программы определить тот класс информационных задач, которые она может решать.

«Эвристическое программирование» как направление возникло и развивалось благодаря работам главным образом американских ученых. Однако в Советском Союзе проводились в течение длительного периода времени исследования такого же типа, хотя во многих случаях применялась другая терминология.

Одной из таких важных отраслей кибернетики был комплекс идей, связанных с понятием *алгоритмического описания* работы мозга.

Мы уже говорили о существенном отличии строгого математического понятия «алгоритм» от понятия эвристической программы. Для того чтобы иметь возможность исследовать более широкий класс явлений природы, некоторые советские ученые пошли не по пути введения нового понятия, а по пути расширения понятия алгоритма и применения его при исследовании работы мозга. Более широкое определение, предложенное, например, В. М. Глушковым, предусматривало возможность использования оператора случайного перебора и оператора поиска информации.

В Московском университете были разработаны специальные методы исследования мозга с целью его алгоритмического описания, были выявлены соответствующие принципы, созданы программы и проведено моделирование. В более позднее время были разработаны новые средства формального описания информационных ситуаций и создана соответствующая система классификации информационных задач.

В нашей стране были разработаны также эвристические программы игры в шахматы и карты, ряд инженерных программ по автоматизации умственного труда диспетчеров, проектировщиков, а также ряд программ по разработке календарных планов загрузки оборудования и составлению расписаний движения поездов, судов и автобусов. В связи с этим следует упомянуть работы Б. дел-Рио по автоматизации диспетчерского управления на железнодорожном транспорте[1], С. А. Думмлера по календарному планированию[2], работы по программированию игр М. Л. Брудно[3] и А. С. Кронрода[4] и многие другие работы, в которых в той или иной степени использовались эвристические приемы оптимизации.

[1] дел-Рио Б., Автоматизация диспетчерского управления, изд-во «Наукова думка», Киев, 1965.

[2] Думмлер С. А, Автоматизация системы управления промышленным предприятием, Москва, 1966.

[3] Брудно М. Л., «Грани и оценки», Проблемы кибернетики, № 10, 1963.

[4] Кронрод А. С., Беседы о программировании, изд-во «Просвещение», 1967.

В связи с этим необходимо отметить также интересную работу В. Н. Пушкина[1].

В настоящее время имеются серьезные основания утверждать, что развитие методов эвристического программирования уже в ближайшем будущем значительно расширит сферы применения электронных вычислительных машин — в первую очередь в сфере управления и при исследовании тех процессов, алгоритмизация которых средствами классической математики обычно мало эффективна. Эта точка зрения в настоящее время широко распространена в США. Например, Г. Саймон, один из создателей эвристического программирования, в недавно выпущенной книге «Пути автоматизации умственного труда человека и процессы управления»[2] утверждает, что к 1985 г. электронные вычислительные машины с эвристическими программами будут широко применяться в управлении промышленностью и экономикой, а психологические и социологические теории достигнут такого же уровня разработки, как современные теоретические представления химии и биологии.

В заключение следует сказать, что при переводе настоящей книги на русский язык были сделаны небольшие отклонения от оригинала: были опущены статьи Тьюринга «Может ли машина мыслить?», Армера «Отношение к разумным машинам» и Селфриджа и Нейсера «Машинное распознавание образов»; первая и вторая из этих работ уже переведены на русский язык, а третья к настоящему времени в значительной своей части устарела. Кроме того, в тексте некоторых статей произведены незначительные сокращения. Наряду с этим в связи с большим значением информационных языков была дополнительно включена статья Ньюэлла и Тонга, содержащая краткое описание языка IPL-V. Большую и самостоятельную ценность представляет помещенная в книге обширная библиография, снабженная тематическим указателем, которая существенно дополнена при переводе.

А. В. Напалков
Ю. В. Орфеев

---

[1] Пушкин В. Н., Оперативное мышление больших систем, изд-во «Энергия», Москва, 1965

[2] Simon H. The Shape of Automation for Men and Management, New York, 1965.

ПРЕДИСЛОВИЕ К АНГЛИЙСКОМУ ИЗДАНИЮ

Работы, включенные в настоящий сборник, посвящены исследованию тех процессов переработки информации, которые лежат в основе «разумного» поведения как человека, так и вычислительных машин. Три причины побудили нас взяться за подготовку такого сборника.

Во-первых, данная тема вызывает всеобщий интерес, является дискуссионной и имеет потенциально большое научное и социальное значение. Поэтому нам хотелось дать широкому кругу читателей возможность познакомиться с рядом важных научных работ в этой области. Бо́льшая часть этих работ была опубликована в технических журналах, доступ к которым относительно затруднен. Здесь они перепечатываются без изменений.

Во-вторых, мы считаем важным, чтобы эти статьи легко можно было использовать в курсах лекций для студентов и аспирантов, читаемых (или планируемых) в различных колледжах и университетах.

В-третьих, нам хотелось дать удобное справочное пособие для тех, кто работает или начинает работать в области искусственного разума и моделирования процессов мышления. Хорошая библиография — необходимая составная часть такого справочника. Мы думаем, что составленная Минским библиография и указатели к ней окажутся чрезвычайно полезным рабочим инструментом.

Жребий редактора сборника нельзя назвать счастливым. Редактора ругают как за включение в сборник тех или иных работ, так и за невключение других. Если же сборник является первым по данной тематике и посвящен такой неопределенной теме, как наша, которой мы дали название «Вычислительные машины и мышление», задача отбора становится еще более сложной.

Во введении к ч. I, которая посвящена искусственному разуму, мы изложили нашу интерпретацию и наше понимание цели этих исследований. Мы отобрали работы, в которых, как нам кажется, удалось продвинуться дальше всего на пути к этой цели. Такой критерий отдает большое предпочтение одному

частному направлению исследований, которое не очень точно обозначено как «познавательные модели». Противоположное направление, обозначаемое иногда как «нейрокибернетика» или «самоорганизующиеся системы», обладает своей особой привлекательностью, и оно уже выдвинуло ряд конкретных проектов исследований. Подход нейрокибернетиков к задаче создания «разумных» машин заключается в следующем. Большое число элементов, каждый из которых осуществляет очень простое преобразование информации, соединяется в случайную или определенным образом организованную сеть, и задаются правила их возбуждения или торможения. Для создателей познавательных моделей характерен гораздо более макроскопический подход, при котором в качестве исходного пункта берутся весьма сложные механизмы для переработки информации. Исследователи этого направления считают, что создание машины с разумным поведением — слишком трудная цель, чтобы можно было надеяться достигнуть ее начиная «с нуля». Поэтому они стремятся сразу заложить в свои модели как можно более сложные процессы переработки информации, требуя лишь, чтобы они оставались понятными и допускали перевод на машинный язык (при существующих методах программирования).

Такой подход привел к ощутимому прогрессу на пути к намеченной цели в области работ по искусственному разуму, тогда как прогресс нейрокибернетического направления к настоящему времени едва заметен. Поэтому наше пристрастие к познавательным моделям кажется нам оправданным, хотя, разумеется, существуют и другие критерии для оценки обоих направлений.

Мы попытались сосредоточить внимание на статьях, в которых приведены *результаты*. Сюда относятся статьи, содержащие описания фактически работающих программ для вычислительных машин, обнаруживающих интересное поведение. Ввиду ограниченного объема сборника мы решили не включать в него работы более спекулятивного характера, несмотря на стимулирующее значение их для исследований по созданию «разумных» машин. По этой причине в сборнике не представлены работы некоторых известных теоретиков в данной области.

Смежные области, как, например, машинный перевод, автоматический поиск информации и теория автоматов, здесь не затрагиваются, так как каждая из них представляет собой отдельную дисциплину и сама по себе заслуживает подробного рассмотрения.

Имеется много статей по психологии и физиологии, тесно связанных с проблемами переработки информации в человеческом мозге. Мы не включили их в настоящий сборник, так как нам хотелось сосредоточить внимание на методах, связанных

с использованием вычислительных машин. По той же причине мы не включили сюда и работы, посвященные такой важной теме, как социальные аспекты проблемы создания разумных машин.

Собранные здесь статьи использовались как основа лекционного курса «Искусственный разум и машинное моделирование познавательных процессов», который мы читали перед смешанной аудиторией, включающей студентов и аспирантов различных специальностей: административного управления, психологии, лингвистики, философии, биологии, физики, биофизики и электроники. Для понимания излагаемого материала не требуется специальной математической подготовки, хотя некоторые познания в математике окажутся полезными (например, при чтении работы Слэйджла, в которой описана программа, вычисляющая неопределенные интегралы). Полезно было бы также знание основ психологии, хотя это и не обязательно. Мы требовали от наших слушателей лишь некоторых элементарных знаний в области вычислительных машин и умения программировать, предпочтительно на одном из списковых языков, например IPL-V, FLPL, COMIT или LISP. Студентам, не имеющим такой подготовки, мы преподали основные сведения по программированию на языке IPL-V [1].

Мы рекомендуем следующий порядок чтения книги.

Для читателей, не имеющих специальной подготовки: Введения к ч. I (искусственный разум) и ч. II (моделирование процессов познания), статья Тьюринга, последующие статьи (их можно читать в том порядке, который соответствует интересам читателя и степени его подготовленности) и наконец обзорные статьи Армера и Минского, которые могут служить и отправной точкой при чтении настоящего сборника [2].

Для работающих в области вычислительных машин и управления производством: Оба введения, затем обзор Минского. Наибольший интерес для этой группы читателей могут представить обучающаяся программа Сэмюэля, программа для балансировки конвейера (Тонг) и исследования, касающиеся программ для доказательства теорем (Ньюэлл, Шоу и Саймон; Гелернтер).

Для психологов и философов: Введение к ч. II, статьи, посвященные решению задач, вербальному обучению, поведению

[1] В русском издании книги из тех же соображений в конце помещена статья Ньюэлла и Тонга о языке IPL-V. — *Прим. ред.*

[2] Статьи Тьюринга и Армера исключены из русского издания книги, так как они уже имеются в русском переводе. См. Тьюринг А., Может ли машина мыслить? Физматгиз, 1960; Таубе М., Вычислительные машины и здравый смысл (Приложение), изд-во «Прогресс», 1964. — *Прим. ред.*

в условиях двоичного выбора, формированию понятий, социальному поведению и принятию решений (их можно читать в любом порядке в зависимости от интересов читателя), а в конце — работы по искусственному разуму.

Мы хотели бы выразить благодарность авторам, которые любезно разрешили нам перепечатать их статьи или специально переработали статьи для этого сборника, а также всем, кто помогал нам своими советами при отборе материала для сборника. Мы благодарим Артура Сэмюэля за обработку для печати записи шашечной партии между Нили и его программой и особенно выражаем признательность Марвину Минскому за время и энергию, которые он потратил на обновление библиографии. Мы чувствуем себя в большом долгу перед А. Ньюэллом и Г. Саймоном за руководство и содействие на протяжении ряда лет. Наконец, мы выражаем признательность миссис Памеле Теллефсен за неустанную заботу и внимание, которые она посвящала рукописи книги в течение многих месяцев, и Рите Фельдман, составившей для книги алфавитный указатель.

Эдвард Фейгенбаум
Джулиан Фельдман

ЧАСТЬ I

Искусственный разум

ВВЕДЕНИЕ

Цель настоящего сборника — дать неспециалистам представление о современных исследованиях по выработке «разумного поведения» у вычислительных машин. При этом составители полагают, что эта цель наилучшим образом может быть достигнута не путем обсуждения и переложения чужих работ, а объединением в одной книге важнейших научных работ, которые в совокупности отражают достигнутый к настоящему времени прогресс в этой области. Каждая из помещенных в этом сборнике работ существенна для правильного понимания исследований по искусственному разуму. В настоящем введении мы надеемся сообщить читателю некоторые основные предварительные сведения, которые помогут ему осмысленно подойти к последующему материалу.

Что такое вычислительная машина? Только ли это «машина, вырабатывающая числа»?

По распространенному мнению, вычислительная машина — это очень быстро работающая счетная машина. Однако такое мнение справедливо лишь отчасти. На самом деле цифровая вычислительная машина — это универсальное устройство для обработки символов, способное осуществить любой достаточно определенный процесс обработки и преобразования информации.

В принципе все универсальные цифровые вычислительные машины подобны друг другу. Они содержат следующие основные узлы:

1. Одно или несколько входных устройств, которые преобразуют представленную некими символами внешнюю информацию в доступную машине форму. Эту «внутреннюю» форму образуют *символы*, которыми оперирует машина. Устройство считывания

с перфокарт — один из примеров такого входного преобразователя.

2. Одно или несколько выходных устройств для вывода машинных символов снова во внешнюю среду. В качестве примера можно привести печатающее устройство вычислительной машины.

3. Одно или несколько «запоминающих» устройств («память») для хранения символов до, во время и после их обработки.

4. «Арифметическое устройство». Одна из возможных интерпретаций, которую можно дать машинным символам, является их интерпретация как *чисел*. Арифметическое устройство — это электронный блок, выполняющий различные операции над этими числами, в результате чего на выходе устройства появляются (в случае числовой интерпретации) суммы, разности, произведения и т. д. Однако большинство вычислений, описанных в этом сборнике, не являются числовыми. Например, шахматные фигуры в машине для игры в шахматы Ньюэлла, Шоу и Саймона представлены и обрабатываются в виде символов, а не чисел.

5. «Устройство управления». Это устройство обеспечивает организацию работы вычислительной машины. Оно предназначено для расшифровки и выполнения некоторого набора инструкций (или *команд*), вводящих по мере надобности в действие другие блоки машины. Обычно команды означают довольно элементарные процессы, например: извлечь символ из определенной ячейки запоминающего устройства, записать некоторый символ в определенную ячейку памяти, сдвинуть символ в «оперативной памяти» на некоторое число ячеек влево или вправо.

Очень важная команда «сравнить и передать управление» дает машине возможность осуществлять простой выбор из двух альтернатив, т. е. в зависимости от информации, хранящейся в некоторой ячейке памяти, выполнить одну из двух определенных последовательностей операций. Сложные решения могут быть получены путем последовательного (каскадного) выполнения такого простого выбора.

Более сложные информационные процессы, чем «встроенные» в вычислительную машину, могут быть выполнены с помощью последовательности элементарных команд, называемой *программой*. Программа — это точная формулировка информационных процессов, выполнения которых человек требует от машины. Так же как исходные данные и другая необходимая информация, программа вычислительной машины хранится в памяти. Одна часть программы может вызвать другую из памяти в оперативное запоминающее устройство и видоизменить ее. Универсальная цифровая вычислительная машина может выполнить любую задачу по переработке информации, для которой может быть написана программа. Та же машина, которая только что вычисляла платежные ведомости какого-то предприятия, может в следующий момент быть использована для расчета авиационных конструкций или страховых премий. Любая программа для универсальной вычислительной машины превращает ее в специализированную машину, рассчитанную на выполнение той задачи, для которой написана программа.

Может ли машина мыслить?

Нет — если определять мышление как нечто, свойственное только и исключительно *человеку*. В этом случае любое поведение машины, напоминающее мышление, следует называть лишь «мыслеподобным» поведением.

Нет — если постулировать, что в самой сущности мышления есть что-то *непостижимое, таинственное, мистическое*.

Да — если признать, что этот вопрос должен быть решен путем *эксперимента и наблюдения*, путем сравнения поведения машины с поведением человека, с которым мы обычно связываем понятие «мышление».

Мы склонны рассматривать оба отрицательных ответа как ненаучные и догматические. Правомерность положительной, т. е. эмпирической, точки зрения убедительно показана в работе Тьюринга [1085]. Армер [35] указывает, что существует целый континуум разумных поведений, и вопрос о том, насколько далеко можно расширить возможности машины в этом континууме, решается путем исследования, а не на основании догм. Мы хотели бы добавить следующее: чтобы утверждать, что думающие машины возможны, не обязательно доказывать, что они уже существуют (или будут существовать в ближайшем будущем). Читателю предлагается составить себе собственное мнение по этому вопросу. Помещенные в этом сборнике работы, как нам кажется, содержат лучшее из всех имеющихся изложений этого круга проблем.

Какова же в таком случае цель исследований искусственного разума? По нашему мнению, эта цель состоит *в создании таких программ для вычислительных машин, поведение которых мы бы назвали «разумным», если бы обнаружили его у людей*.

Так как эта область в настоящее время еще находится в стадии развития, исследования ведутся в нескольких различных направлениях. Поэтому наше определение цели может одним исследователям показаться слишком смелым, а другим — недостаточно смелым (главным образом из-за его зависимости от поведения человека).

Многие исследовательские работы, описанные в ч. I настоящего сборника, стремятся достигнуть этой цели в рамках своего специфического круга проблем. Вопрос же о том, следует или не следует называть такое поведение машин «разумным», пожалуй, дело вкуса. По нашему мнению, вопрос этот едва ли представляет важную и плодотворную тему для дискуссии.

Но не делает ли машина лишь в точности то и не более того, что предусмотрено заложенной в нее программой?

Обсуждая этот привычный вопрос, известный исследователь в этой области, Саймон, пишет: *«Утверждение, что вычислительные машины могут делать только то, что нами в них запрограммировано, интуитивно очевидно и несомненно правильно, но не доказывает справедливости ни одного из тех выводов, которые обычно из этого делают.*

Человек может мыслить, учиться и творить благодаря тому, что программа, обусловленная его биологической природой, а также те изменения в этой программе, которые вызваны взаимодействием организма с окружающей средой после его рождения, делают его способным мыслить, учиться и творить. Если окажется, что машина мыслит, учится и творит, то этими качествами ее наделила ее программа. Ясно, что эта программа не в большей степени, чем человеческая будет требовать существенно стереотипного и повторяющегося поведения, не зависящего от стимулов из окружающей среды и выполняемой задачи. Это будет программа, делающая поведение системы в сильной степени зависимым от внешней среды, от целей поставленной задачи и от сигналов о том, насколько система продвинулась на пути к достижению этой цели. Такая программа будет каким-то способом анализировать собственное поведение, обнаруживать свои ошибки и вносить необходимые изменения, которые позволят в будущем повысить ее эффективность» ([1011], стр. 25).

Подобным же образом едва ли правильно было бы заключать, что машина не может быть более разумной, чем ее программист, и что этот проницательный джентльмен может точно предсказать поведение своей программы. Такие выводы игнорируют огромную сложность процессов переработки информации, возможных в обучающихся машинах. В этих выводах содержится предположение, что так как программист может записать (в виде программ) общие свойства адаптивного поведения таких механизмов, он сможет предсказать и те отдаленные последствия, к которым оно приведет после выполнения этими механизмами миллионов операций по переработке информации и взаимодействия с соответствующей задаче средой. И, что более важно, при этом предполагается, что программист может выполнить эти же сложные процессы обработки информации столь же хорошо с помощью устройства, заложенного в его мозге.

Верно ли, что машина может стать чемпионом по шахматам, потому что она настолько быстро работает, что может проанализировать все возможные ходы и их последствия?

Такой взгляд на способность вычислительных машин решать задачи основан на предположении, что машина в силу большой скорости выполнения операций может «обдумать все». Однако такой метод вычислений следовало бы назвать методом «грубой силы». Обычно такие программы, основанные на «грубой силе», обладают простой структурой и в них используются исчерпывающий перебор всех возможностей и исчерпывающий поиск. Но является ли такой метод общим методом решения задач, имеющих некоторое «интеллектуальное содержание»?

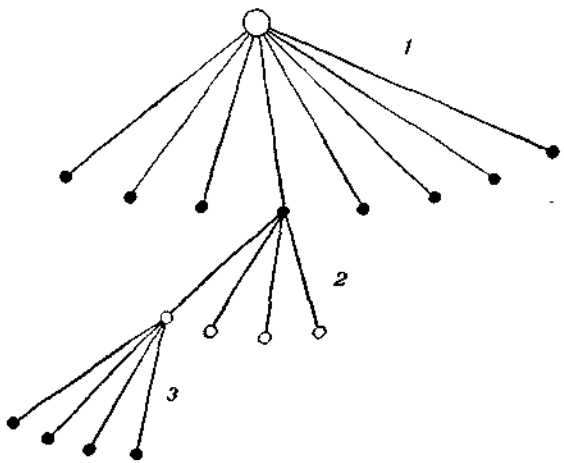
Прежде чем ответить на этот вопрос, необходимо рассмотреть, что представляет собой «задача». Задача возникает, когда вычислительная машина сталкивается с необходимостью выбора одной из множества связанных с данной задачей альтернатив. Перед «решающей задачи» машиной нет проблемы, если окружающая среда предлагает ей лишь одну возможность. В таком случае машина должна следовать этой возможности. Трудности при выборе альтернатив возникают не столько из-за их большого числа, сколько из-за их следствий. Альтернативы обычно имеют сложные и разветвленные последствия, которые должны быть оценены до того, как будет произведен выбор альтернативы. Строгая формулировка этих понятий приводит к так называемой «лабиринтной модели задачи».

Рассмотрим в качестве примера задачу выбора хода в некоторой шахматной позиции. Если позиция дает игроку возможность сделать лишь один определенный ход, то нет никакой проблемы — этот ход *вынужденный*. Если же есть много альтернатив, то решение должно быть принято с учетом непосредственных и отдаленных следствий, возможных ходов противника, ответов на них и т. д. На фиг. 1 изображено такое «дерево» возможных ходов.

В принципе это дерево можно детально проанализировать. Все конечные точки можно классифицировать на победы, поражения, ничьи и принять некоторую стратегию для выбора наилучшей из альтернатив в вершине дерева.

Почему же шахматы все еще остаются интересной игрой, если процедуру выбора стратегии принципиально можно запрограммировать на современной быстродействующей вычислительной машине?

Ответ очень прост: размеры шахматного лабиринта колоссальны. По имеющимся оценкам, он содержит приблизительно (с точностью до нескольких порядков) 10^{120} различных путей.



Фиг. 1.

1 — ближайшие возможные ходы белых; *2* — возможные ходы черных; *3* — возможные ответные ходы белых.

Даже при наиболее благоприятных предположениях относительно возможностей современных и будущих вычислительных машин весьма неправдоподобно, что машина когда-нибудь сможет «оптимально» играть в шахматы, используя упомянутую выше стратегию исчерпывающего перебора.

За исключением некоторых наиболее тривиальных примеров, метод грубого перебора всех путей в лабиринте задачи едва ли следует считать приемлемым. Решение задач этим методом выходит за пределы практических возможностей.

Как же мы тогда надеемся создать разумное устройство для решения задач?

По-видимому, ключевым в проблеме разумного поведения человека или машины в значительной степени является *выборочный поиск*, т. е. существенное сокращение дерева подлежащих рассмотрению возможностей. *Для того чтобы программа вычислительной машины обладала разумным поведением, необходимо наличие высоко селективного поиска в лабиринте задачи путем исследования лишь относительно эффективных путей.*

Что такое эвристическая программа?

Эвристика (эвристическое правило, эвристический метод) — это основанные на опыте правило, стратегия, ловкий прием, упрощение или иное средство, существенно ограничивающее поиск решения сложных задач. Эвристика отнюдь не гарантирует оптимальность решений. В действительности она даже вообще не гарантирует достижения решения: *все, что можно сказать о полезной эвристике, это то, что она предлагает решения, которые большей частью оказываются достаточно хорошими. Эвристическая программа* — это программа, пользующаяся при решении сложных задач эвристикой.

Одно время эвристические методы поиска решений весьма настойчиво противопоставляли *алгоритмическим методам*, и достаточное количество «интеллектуальной крови» было пролито на этом поле брани. Не вдаваясь в подробности этих распрей, можно указать, что в математике и логике термин «алгоритм» используется в довольно различном смысле. По одному из общепринятых определений алгоритмы являются решающими процедурами, которые при наличии достаточного времени *гарантируют* достижение искомого решения. В этом смысле алгоритмом для игры в шахматы является и описанная выше программа грубого перебора. Однако лишь для очень небольшого числа интересных задач, для которых мы хотели бы иметь машинное решение, найдены алгоритмы (в указанном смысле). По-видимому, эвристические методы имеют гораздо более широкую сферу приложений.

Преимущества использования эвристики заключаются в значительном ограничении поиска, что обусловливает практичность метода. Хотя и не всегда, но часто за это приходится платить определенную цену: из-за сильного ограничения поиска можно упустить наилучшее решение (или даже *все* решения).

Существуют два варианта эвристического метода: специализированный и универсальный. Рассмотрим несколько примеров:

1. Машина, имитирующая шахматного игрока, могла бы применять следующее эмпирическое правило: прекращать исследование любой последовательности ходов, которая приводит к непосредственной угрозе потери ферзя. Это пример специализированной эвристики для шахматной игры; такое правило полезно, поскольку исключает один вид опасности. Однако, следуя такому грубому ограничению поиска, машина никогда не найдет

тех восхитительных комбинаций с жертвой ферзя, которые в руководствах по шахматам неизменно сопровождаются двумя восклицательными знаками.

2. При доказательстве теорем математик обычно идет «обратным путем» — от теоремы, которую он стремится доказать, к известным теоремам или аксиомам, вместо того чтобы прямо продвигаться с помощью правил математического вывода от известных выражений к доказываемой теореме. При известных условиях такой «обратный путь» является эффективным общим эвристическим методом использования содержащейся в задаче информации для поиска решения.

3. В большинстве случаев при решении новых задач люди используют следующее эмпирическое правило: применить к данной задаче те методы, с помощью которых в прошлом решались подобные проблемы. Критерии «подобия» поэтому сами по себе имеют эвристический характер. Этот метод может оказаться очень полезным в том случае, если окружающая среда постоянна по отношению к типам задач. Однако, если среда требует существенно нового решения задачи, этот эвристический метод будет скорее препятствовать, чем способствовать отысканию такого решения.

4. В человеческих рассуждениях обычно используются два универсальных эвристических метода решения задач: *метод анализа целей и средств* (means-ends analysis) и *метод планирования* (planning). Метод анализа целей и средств состоит в выборе и выполнении над исходным состоянием задачи таких операций, которые шаг за шагом уменьшают разницу между исходным и конечным состояниями. В методе планирования вырабатывается некоторая упрощенная формулировка исходной задачи. Эта новая, более простая задача решается методом анализа целей и средств. В результате получается некоторое множество прогнозов (каждый прогноз соответствует одной из возможных последовательностей операций). По предположению один из этих вариантов будет «работать», т. е. даст решение исходной задачи. Метод анализа целей и средств подробно разбирается в работе, посвященной системе GPS (General Problem Solver, см. стр. 283).

Каковы некоторые нерешенные проблемы исследования искусственного разума?

Неудивительно, что в такой новой области исследования, как проблема искусственного разума, большинство проблем пока еще остается нерешенным, а к изучению других еще не приступали. На данной стадии трудно даже сформулировать эти проблемы, за исключением самых общих. Ниже приведено несколько

примеров задач, которые, по нашему мнению, требуют интенсивных исследований:

1. *Обучение эвристикам.* Здесь возникает трудный и очень интересный вопрос: как вычислительные машины могли бы обучиться (и как это делает человек) новым специализированным и универсальным эвристическим методам и правилам? В настоящее время наши знания относительно механизмов обучения для программ по решению задач находятся еще в зачаточном состоянии. Любой существенный успех в этой области позволил бы надеяться на создание гораздо более эффективных программ.

2. *Использование метода индукции.* На данном этапе искусственный разум в значительной мере основан на использовании дедукции и лишь в очень слабой степени — на индукции. Однако в повседневной сознательной жизни человека индукция, несомненно, занимает гораздо более значительное место. С одной стороны, проблема состоит в том, что нужны программы, которые в некотором смысле индуктивно создавали и запоминали бы «модели» окружающих сред — модели, на основании которых эти программы могли бы делать существенные и полезные выводы о будущих состояниях среды. С другой стороны, это задача формирования машиной гипотез или общая проблема распознавания образов. В настоящее время мы знаем еще очень мало относительно этой коренной проблемы.

3. *Понимание машиной естественного языка.* Эта очень интересная как с теоретической, так и с практической точки зрения проблема состоит в конструировании программы для общения человека с машиной на человеческом языке. Попросту говоря, хотелось бы иметь возможность вступать в такой диалог с машиной, в котором последняя поддерживала бы разговор, проявляя приспособляемость, разумность и понятливость. Хорошим началом является изучение программ, отвечающих на вопросы (например, программа BASEBALL, опубликованная в данном сборнике). Многое в эту область можно перенести из исследований по машинному переводу, поиску информации, моделям ассоциативной памяти человека и из других разделов науки об информации. Эта проблема требует интенсивных исследований средствами различных дисциплин.

Каковы границы исследований искусственного разума?

На этот вопрос в настоящее время едва ли можно дать ответ. Пожалуй, вопрос этот более интригующий, чем практически важный. Если рассматривать континуум разумных поведений, предложенный Армером, то программы вычислительных машин,

которые мы оказались способными создавать к настоящему времени, находятся все еще на самом низком уровне. Важно подчеркнуть, что развитие этой области идет по пути приближения к возможностям человеческого разума. Есть ли какие-либо основания полагать, что этот уровень никогда не будет достигнут? Такие основания отсутствуют. Нет никаких данных, ни одного логического аргумента или доказательства в пользу существования непреодолимых препятствий на пути такого развития в пределах указанного континуума.

Несмотря на всю недостаточность наших нынешних познаний, мы сегодня можем указать на существующий биологический пример — мыслящий мозг — в том же смысле, в каком ученые несколько столетий назад указывали на птиц как на естественный пример летающих аппаратов тяжелее воздуха.

РАЗДЕЛ I

ИГРАЮЩИЕ МАШИНЫ

Создание играющих машин интересно с многих точек зрения. В самом деле, имея такие машины, можно устроить «очное» соревнование между умом человека и «умом» машины. Говоря более серьезно, игровая ситуация — это интересный пример среды, которая требует от субъекта принятия некоторого решения. Эта среда сравнительно высоко организована и хорошо определена, однако найти требуемое решение достаточно сложно, так что разум и умение оперировать символами играют в этом поиске решающую роль. Короче говоря, игровая среда очень полезна для изучения природы и структуры процессов решения сложных задач.

Игра в шахматы — одно из любимых развлечений человека. Для нее было создано много программ. Исторический обзор этих работ дается в статье Ньюэлла, Шоу и Саймона (стр 33), бо́льшая часть которой посвящена детальному описанию разработанной ими шахматной программы (известной под названием «Шахматист НШС»). В конце статьи там приводится партия, сыгранная этой программой и прокомментированная известными шахматными мастерами.

Программа Ньюэлла, Шоу и Саймона относится к тем исследованиям, которые лежат в пограничной области между проблемой искусственного интеллекта и моделированием процесса решения задач человеком. Строго говоря, она вовсе не задумана как некая модель решения задач человеком в шахматной ситуации. Однако, разрабатывая свою программу, авторы исходили

из убеждения, что человеческие методы решения задач, характеризующиеся высокой степенью адаптивности и избирательности, должны оказаться более эффективными при решении шахматных задач, чем другие предложенные вычислительные схемы. Работа их шахматной программы как будто подтверждает правильность этого убеждения.

Характерное для этой работы Ньюэлла, Шоу и Саймона внимание к методам, используемым человеком при решении задач, лежит в основе всех совместных исследований этих авторов. В этой связи интересно указать на работы [1005] и [1008], в которых, хотя они написаны свыше 10 лет назад, уже в значительной степени содержится основная схема механизма принятия решений. Эта схема была затем воплощена в программах «Шахматист» и «Логик-теоретик», а также в программе Кларксона, выбирающей портфель ценных бумаг (см. стр. 358 настоящего сборника).

Программа Сэмюэля для игры в шашки (стр. 71) целиком лежит в русле работ по искусственному интеллекту. В своем основном механизме, особенно в схеме оценки положения в игре, она не пользуется человеческими способами решения задач. Сэмюэль считает, что эффективный путь прогресса в создании искусственного разума лежит, по-видимому, не в имитации и приспосабливании к машине человеческих методов решения задач. Довольно высокий уровень игры, достигнутый программой Сэмюэля, убеждает в правомерности такого подхода.

Поведение этой программы настолько высокоорганизовано, что ее можно рассматривать как одну из главных вех на пути исследований по искусственному разуму. В конце статьи Сэмюэля приводится реальная партия, в которой программа выиграла у мастера по шашкам.

В этой программе особенно интересны алгоритмы обучения, которые улучшают игру программы, позволяя ей накапливать опыт с увеличением числа сыгранных партий. Пока это единственная успешная попытка обучения машины решению сложных задач.

1. ПРОГРАММА ДЛЯ ИГРЫ В ШАХМАТЫ И ПРОБЛЕМА СЛОЖНОСТИ

А. Ньюэлл, Дж. Шоу и Г. Саймон

Введение

Человек может решать задачи, не зная, как он это делает. Именно это обстоятельство создает основу для попыток понять и формализовать процесс принятия человеком решений

в сложных ситуациях и процесс решения задач. Давайте согласимся на том, что если нам известно, как выполнить автоматически то, что человек делает с помощью своего мозга, то эти знания полезны как для понимания мыслительного процесса, так и для увеличения арсенала средств, с помощью которых человек может управлять и манипулировать окружающей его средой. Мы попробуем оценить прогресс в понимании и автоматизации интеллектуальных достижений человека, рассматривая лишь одну узкую область: попытки создать цифровую вычислительную машину, играющую в шахматы.

Шахматы — игра по существу своему интеллектуальная. Об этой игре можно без преувеличения говорить как о борьбе двух интеллектов в ситуации, которая настолько сложна, что ни один из них не в состоянии ее полностью охватить, но которая вместе с тем достаточно поддается анализу для того, чтобы каждый надеялся сделать это лучше противника. О глубине и тонкости игры свидетельствуют как количество профессиональных игроков, так и тот факт, что 200 лет глубокого изучения и интенсивного развития шахматной игры отнюдь не исчерпали ее и не сделали скучной. Все эти свойства делают шахматы естественной ареной для попыток автоматизации. Создание машины, успешно играющей в шахматы, позволило бы, вероятно, проникнуть в сущность интеллектуальной деятельности человека.

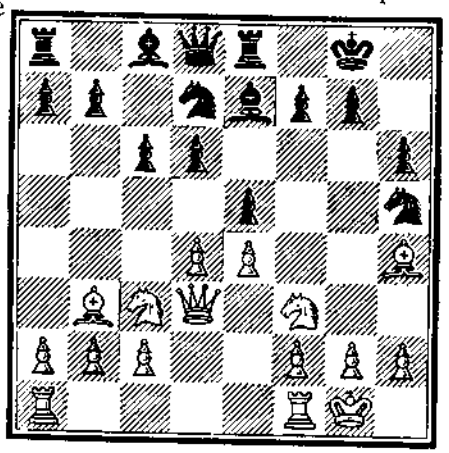
История создания шахматных программ — пример попыток научиться работать со сложными алгоритмами. Однако внезапно может оказаться, что будет открыто нечто, подобное колесу по сравнению с человеческой ногой: устройство, использующее совершенно иные, нежели человек, методы, но чрезвычайно эффективное и, возможно, очень простое. Это устройство могло бы отлично играть в шахматы, но было бы бессильно продвинуть вперед наши знания относительно умственной деятельности человека. Такая находка, безусловно, сама по себе была бы ценным открытием, однако ничего подобного на горизонте пока не видно.

Однако вернемся к нашему исходному положению: люди играют в шахматы и их действия в процессе игры представляются чрезвычайно сложными и запутанными, но успешными. Рассмотрим, например, отрывок, показывающий, как игрок (играющий белыми) описывает процесс анализа позиции, изображенной на фиг. 1.

Есть ли еще какие-нибудь угрозы? Черные могут пойти конем на f4, угрожая ферзю, и могут усилить также давление на королевском фланге, открыв своего белопольного слона после ухода коня с поля d7; однако это не непосредственная угроза. С другой стороны, пешка e5 угрожает моей пешке...

Заметим, что анализ этот качественный. Игрок «блуждает» от одного признака к другому, накапливая по частям информацию на протяжении всего анализа. Он дает оценки в терминах «давления» и «непосредственной угрозы» и постепенно приобретает некоторое упорядоченное представление о создавшейся ситуации.

Можно ли создать алгоритмы, сложность которых была бы сравнима со сложностью рассуждений шахматиста? Нет необходимости, чтобы они играли точно таким же образом: копирование действий человека вовсе не есть наша непосредственная задача. Но мы утверждаем, что сложность — это существенная характеристика умственной деятельности и что сложность удачной шахматной программы должна приближаться к сложности мыслительных процессов опытного шахматиста. Сложность решения диктуется сложностью задачи, а не какими-то особыми свойствами человеческого мозга.



Фиг. 1.

Существует тесная зависимость между сложностью систем и средствами для их описания. С одной стороны, сложность систем, которые мы можем описывать, зависит от языка, на котором мы должны их описывать. Человек обладает ограниченной способностью к переработке информации. Но, увеличив мощность языка, мы, несмотря на это ограничение, можем описывать более сложные системы.

Проиллюстрируем эту мысль на следующем примере. Никто не собирается строить шахматные машины в буквальном смысле этого слова: собирать из электромеханических деталей играющие в шахматы автоматы. Вместо этого мы придумываем шахматную программу: на языке, который называется машинным кодом, мы пишем инструкцию, как играть в шахматы, — инструкцию, которая может быть выполнена любой стандартной цифровой вычислительной машиной. Но, помимо желания избежать конструирования специализированного машинного оборудования, существует еще одна причина, по которой предпочтение отдается именно этому способу, — машинный код представляет собой

более мощный язык, чем блок-схемы радиоинженера. С помощью каждого символа машинного кода перерабатывается больше информации, чем с помощью символа блок-схемы. Даже умеренная по сложности программа становится безнадежно сложной, если описывать ее в терминах вентилей и импульсов.

Но есть и другая сторона связи между средствами описания и сложностью: мы не можем использовать любой удобный для нас язык. Нас должны понимать то лицо или та машина, к которым мы обращаемся. Шахматная программа, написанная на английском языке, не будет работать, ибо не существует машин, понимающих английский язык. Описание на английском языке может предназначаться только для другого человека, который затем должен будет создать соответствующую машину. Преимущество машинного кода заключается именно в том, что существуют понимающие его машины. Поэтому шахматная программа, записанная в машинном коде, функционально эквивалентна машине, играющей в шахматы.

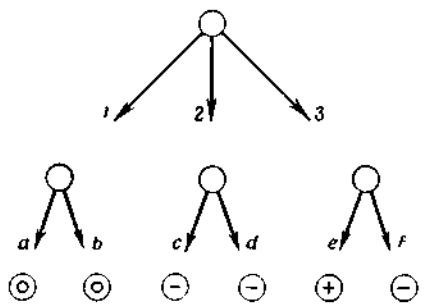
Если бы машина могла понимать еще более мощные языки, можно было бы применить их для написания шахматных программ и таким образом получить более сложные и «умные» программы, несмотря на ограниченную человеческую способность к переработке информации. Но возможности обмена информацией ограничиваются интеллектом наименее развитого участника такого обмена, а в настоящее время машина обладает лишь пассивными возможностями к обмену информацией. Язык, который она понимает, — это язык простых команд; поэтому приходится очень подробно описывать все, что мы требуем от машины.

Таким образом, возрастание эффективности коммуникации человека с машиной, по-видимому, связано с возрастанием машинного «интеллекта» так, чтобы человек мог, меньше думая, больше сказать. Но на данной стадии развития единственный способ получить более умные машины заключается в том, чтобы построить их, — мы еще не научились «выращивать» или «высиживать» их или же тренировать их теми методами, которые применяются для обучения людей. Ситуация здесь примерно та же, как в случае, когда имеется система с двумя устойчивыми состояниями, находящаяся не в том состоянии, в котором бы нам хотелось. Мы могли бы создать более умные машины, если бы умели лучше общаться с ними, и мы могли бы лучше с ними общаться, если бы они были более умными. Поэтому каждое продвижение в одном из этих двух направлений требует значительных усилий. Однако каждый успех позволяет, приложив соответствующее усилие, продвинуться немного и в другом направлении. На какой-то стадии процесс пойдет сам собой, и система перейдет в то желаемое для нас состояние, в котором мы будем обладать и умными машинами, и высокоэффективными средствами общения с ними.

Выяснив, таким образом, цель, ради которой мы собираемся работать, вернемся к основному содержанию статьи: разработке шахматных программ. Мы будем рассматривать их в исторической последовательности. Такое рассмотрение позволит нам наиболее отчетливо выявить прогресс, достигнутый в создании систем все возрастающей сложности и «разумности».

Проект Шеннона

История шахматных программ начинается со статьи Клода Шеннона, опубликованной в 1949 г. В этой статье он не предложил какой-либо детально разработанной шахматной программы, но рассмотрел основные возникающие при этом проблемы. Предложенная им схема была положена в основу большинства более поздних исследований в этом направлении.



Фиг. 2. Дерево игры и минимаксная процедура.

Шахматы, как было отмечено Шенноном, представляют собой конечную игру. В этой игре имеется лишь конечное число возможных позиций, причем в каждой позиции существует конечное число возможных ходов. Правила игры гарантируют, что за конечное число ходов будет достигнут выигрыш, проигрыш или ничья. Поэтому можно дать исчерпывающее описание шахмат с помощью «ветвящегося дерева» (фиг. 2). Вершины этого дерева соответствуют позициям, а ветви — возможным ходам в каждой позиции. Интуитивно ясно (и это легко доказать), что для игрока, который может обозревать все дерево целиком и видеть самые отдаленные последствия каждого принятого

решения, шахматы становятся простой игрой. Начав с конечных позиций, для которых однозначно определен исход партии, он может двигаться в обратном направлении, определяя в каждой вершине наилучшие для себя или для своего противника ветви, пока не дойдет до исходной вершины, из которой он должен сделать очередной ход.

Такая процедура, называемая в теории игр *минимаксной*, лежит в основе всех известных к настоящему времени попыток программирования игры в шахматы. Убедимся, что она нам понятна. На фиг. 2 изображена ситуация, в которой белые должны сделать очередной ход, выбрав одну из трех возможностей: *1*, *2* или *3*. За ходом белых последует ход черных: *a* или *b*, если сделан ход *1*, *c* или *d*, если сделан ход *2*, и *e* или *f*, если сделан ход *3*. Чтобы упростить пример, предположим, что все ходы черных ведут к позициям, для которых определен исход партии: (+) означает выигрыш белых, (0) — ничью и (—) — проигрыш белых. Какой из трех ходов предпочтительнее? Ясно, что независимо от дальнейшего поведения черных ход *1* приводит к ничьей. Аналогично ход *2* в любом случае приводит к проигрышу белых. Понятно поэтому, что белые должны предпочесть ход *1* ходу *2*. Но что можно сказать относительно хода *3*? Он дает возможность выигрыша, но содержит и возможность проигрыша; более того, исход партии в этом случае всецело зависит от черных. Если белые имеют дело с противником, который обладает некоторой способностью к анализу, они должны заключить, что ход *3* приведет к проигрышу, и, следовательно, ход *1* предпочтительнее. Выигрыш же после хода *3* практически невозможен. Таким образом, белые могут приписать позиции некоторую оценку (в данном случае это ничья), рассуждая с помощью обратного движения от позиций с известными оценками.

Повторяем: если можно полностью просмотреть все дерево игры, лучший ход легко определяется с помощью минимаксной процедуры. Этот метод мог бы сыграть в шахматах роль упомянутого выше «колеса» (игра оканчивалась бы, не успев начаться), если бы дерево игры не было столь большим по размерам, что даже современные вычислительные машины, потратив годы на вычисления, могут исследовать лишь ничтожную его часть. Число вариантов, подлежащих рассмотрению, по оценке Шеннона составляет приблизительно 10^{120}, в то время как столетие содержит менее 10^{16} микросекунд.

Шеннон предложил следующую схему шахматной программы Игра в шахматы состоит из рассмотрения возможных ходов, анализа этих ходов с целью получения некоторой оценки каждого из них (мы будем называть ее *рабочей оценкой*) и выбора наилучшего хода на основании этой оценки. Анализ, представляющий собой самую трудную задачу, производится в три этапа. Во-первых, исследуются возможные продолжения на некоторую глубину. Во-вторых, поскольку исследовать все продолжения вплоть до окончания партии невозможно, позиции, на которых заканчивается исследование, оцениваются исходя из расположения фигур на доске. Эти оценки мы будем называть *статическими*. Комбинируя затем эти статические оценки с помощью минимаксной процедуры, мы получаем рабочие оценки каждого возможного хода. В соответствии с этим выбирается ход с наибольшей рабочей оценкой.

Основанием для такого подхода может служить следующее рассуждение. Если оценочная функция задана, то чем больше глубина анализа, тем лучше будет играть программа. В пределе этот подход привел бы к идеальной игре, так как он означал бы нахождение конечных позиций для всех продолжений. Таким образом, имеется одномерный параметр, по которому можно сравнивать все программы, — *глубина анализа*.

Чтобы завершить описание схемы, необходимо еще указать способ статической оценки позиции, т. е. ее оценки, не производя дальнейших ходов Шеннон предложил измерять ценность позиции некоторым числом. Это число получается путем суммирования с соответствующими весами числовых оценок, характеризующих различные особенности позиций, которые знатоки шахмат считают важными. Такой подход внушает определенное доверие в связи с тем, что для шахматных позиций существует несколько естественных количественных характеристик, как, например, ценность фигур и подвижность пешек. Он оправдан, конечно, еще и потому, что в науке и технике для получения первого приближения часто пользуются предположением о линейности.

Резюмируем: Шеннон предложил при создании шахматных программ руководствоваться схемой, представляющей собой последовательность вопросов:

1. Возможные ходы.
   Какие из возможных ходов следует рассматривать?
2. Анализ.
   а) Какие продолжения исследовать и на какую глубину?
   б) Как находить статическую оценку позиций?
   в) Как объединить статические оценки в одну рабочую оценку для каждого возможного хода?
3. Окончательный выбор.
   С помощью какой процедуры выбирать наиболее предпочтительный ход?

Статью Шеннона вспоминают главным образом в связи с теми ответами, которые он предложил на эти вопросы: 1) рассматривать все возможные ходы; 2а) исследовать все продолжения

на фиксированную глубину n; 2б) давать численную оценку в виде суммы; 2в) для получения рабочих оценок использовать минимаксную процедуру и, наконец, 3) выбирать ход с максимальной оценкой. Шеннон, однако, в своей статье идет дальше и рассматривает возможность отбора лишь небольшого числа целесообразных ходов и продолжений. Кроме того, он обсуждает возможность анализа в терминах, употребляемых шахматистами, — «блокирование», «атака», «защита». Но в то время шахматные программы можно было представить себе лишь в виде чрезвычайно примитивных алгоритмов. Конкретные предложения Шеннона были постепенно осуществлены в различных программах, в то время как остальные его идеи в значительной степени игнорировались. Сейчас эти идеи вновь оказались в поле зрения исследователей, но уже в связи с иными задачами.

Программа Тьюринга

Шеннон не представил конкретной шахматной программы. Реализация его предложений потребовала бы очень большого объема вычислений даже при такой умеренной глубине анализа, как два или три хода. Воспользовавшись его схемой, А. М. Тьюринг [1085] составил программу, которая была настолько простой, что могла выполняться вручную, без помощи цифровой машины.

В табл. 1 дана характеристика программы Тьюринга по всем пунктам шенноновской схемы. Смысл других рубрик таблицы станет ясным из дальнейшего изложения. Подобные же характеристики даны и для остальных трех программ, рассмотренных в этой статье.

Программа Тьюринга в каждой позиции рассматривает все разрешенные правилами ходы. Чтобы ограничить объем необходимых вычислений, Тьюрингу пришлось позаботиться об уменьшении числа рассматриваемых вариантов. Он ввел понятие «мертвой» позиции, т. е. позиции, которая в некотором смысле стабильна и для которой, следовательно, целесообразно вычислять статическую оценку. Например, нет никакого смысла подсчитывать имеющийся на доске материал в тот момент, когда начавшийся размен ферзей еще не закончен. Каждый вариант следует продолжать до получения «мертвой» позиции, в которой следующий ход не может привести к уменьшению имеющегося на доске материала. Программа Тьюринга оценивает материал только в «мертвых» позициях. В его статической оценке доминирующей является ценность фигур, так что остальные особенности позиции принимаются во внимание лишь тогда, когда минимаксная процедура указывает на несколько возможных продолжений, равноценных с точки зрения материала. В этих случаях используется дополнительная аддитивная оценка позиции, включающая много разных факторов: подвижность фигур, отстающие пешки, защита фигур и т. п., — каждый с соответствующим весом.

Таким образом, программа Тьюринга представляет собой хороший, хотя и не очень сложный с точки зрения требуемого объема вычислений пример шахматной программы, играющей по методу Шеннона. Насколько нам известно, была опубликована только одна партия, которую сыграла эта программа. Ее игра оказалась довольно слабой: она проиграла плохому игроку (который, между прочим, не был знаком с программой), хотя программа и не может считаться вовсе несостоятельной.

Игра программы в общем была довольно бесцельной. Она допускала грубые ошибки, одна из которых стоила ей партии. Как можно было ожидать, это свело к нулю все преимущества сложной оценочной функции. Многочисленные факторы, учитываемые этой функцией, редко оказывали какое-либо влияние на выбор хода. Итак, программа Тьюринга не была хорошим шахматистом, она находилась всего лишь на уровне очень слабого игрока.

A priori нет никаких возражений против выполнения программы вручную, хотя опыт показывает, что для программы такой сложности это почти всегда приводит к ошибкам. Примером может служить партия, сыгранная Тьюрингом по его программе. Он не исследовал ходы, которые, как он «заранее знал», будут отклонены программой, и однажды ошибся. Главное возражение против выполнения программы вручную — количество требуемых для этого усилий. Только машина дает возможность исследовать поведение сложной программы. Потенциальные возможности схемы Шеннона можно реализовать, лишь программируя ее для машины.

Лос-Аламосская программа

В 1956 г. группой сотрудников из Лос-Аламосской лаборатории была создана шахматная программа для машины MANIAC-I [553][1]. Лос-Аламосская программа наиболее точно воплощает проект Шеннона. Как видно из табл. 1, программа

[1] За период с 1951 по 1956 г. были проведены еще два исследования: моделирование вручную, выполненное Ф. Мостеллером, и русская программа для БЭСМ, однако из-за недостатка информации мы их здесь не излагаем.

Таблица 1

Сравнительные характеристики шахматных программ

		Тьюринг	Кистер, Стейн, Юлэм, Вальден, Уэллс [552] (Лос-Аламос)	Бернстейн, Робертс, Арбукл, Бельский [120]	Ньюэлл, Шоу, Саймон [797]
Общие сведения	Год написания	1951	1956	1957	1958
	Размер доски	8×8	6×6	8×8	8×8
	Машина	Вручную	MANIAC-I; 11 000 *опер/сек*	IBM-704; 42 000 *опер/сек*	JOHNNIAC; 20 000 *опер/сек*
Шахматная программа	Число рассматриваемых в каждой позиции ходов	Все ходы	Все ходы	7 правдоподобных ходов; набор генераторов ходов	Переменное; набор генераторов ходов
	Глубина анализа	До «мертвой» позиции (только в отношении разменов)	2 хода; все продолжения	2 хода; в каждой позиции проверяется 7 ходов	До «мертвой» позиции. Каждая цель генерирует ходы
	Статическая оценка	Число; много факторов	Число; материал, подвижность	Число; материал, подвижность, контролируемая площадь, защита короля	Вектор оценок; пригодность с точки зрения каждой цели
	Объединение статических оценок	Минимаксная процедура	Минимаксная процедура (модифицированная)	Минимаксная процедура	Минимаксная процедура
	Окончательный выбор	Доминирует материальный перевес; в случае равенства — по наибольшей оценке	По наибольшей оценке	По наибольшей оценке	1. Первый приемлемый вариант 2. Ход, выполняющий двойную функцию
Программирование	Язык	—	Машинный код	Машинный код	IPL-IV (интерпретируемый код)
	Некоторые особенности программы	—	Играет на одной доске	Играет на одной доске; централизованное обращение к таблицам; использует повторные вычисления [1]	Играет на одной доске; децентрализована; списочная структура; использует повторные вычисления [1]
	Время	Минуты	12 *мин* на ход	8 *мин* на ход	1—10 *час* на ход (оценка)
	Размер программы	—	600 слов	7000 слов	Сейчас 6000 слов; в будущем предполагается 16 000 слов
Результаты	Игровой опыт	1 партия	3 партии (программа больше не существует)	2 партии	0 партий; несколько экспериментов с моделированием вручную
	Качество игры	Проигрывает слабому игроку; играет без цели; применение тонкой оценочной функции не приносит пользы	Выигрывает у слабого игрока; эквивалентна игроку с опытом игры в 20 партий	На уровне сносно играющего любителя; слепые пятна; игра позиционная	Сильна в дебюте Пока не имеет агрессивных целей

[1] Вместо того чтобы хранить в памяти большое количество разнообразной информации, часто бывает выгоднее заново вычислять те или иные данные по мере того, как в них возникает необходимость. — *Прим. перев.*

рассматривает все возможные ходы; все продолжения исследуются на глубину в два хода (т. е. 2 хода черных и 2 хода белых); статическая оценка представляет собой сумму двух оценок: материала и подвижности фигур; для получения рабочих оценок используется минимаксная процедура; в качестве лучшего хода выбирается ход с наибольшей оценкой.

Чтобы ограничить требуемое машинное время разумными пределами, авторам программы пришлось пойти на упрощение правил игры. Вместо обычной шахматной доски (8×8 клеток) использовалась доска 6×6 клеток. Игра происходила без слонов и запрещались некоторые специальные ходы: рокировка, ход пешки на два поля и взятие пешки на проходе.

Каков же результат? Снова программа играет плохо, но теперь она способна выиграть у слабого игрока, что машина и продемонстрировала в одной из трех сыгранных ею партий. Программа допускает серьезные ошибки, которые в общем типичны для слабых игроков.

Так как это первый пример фактически играющей машины, стоит несколько детальнее рассмотреть ее особенности. На обычной (8×8) шахматной доске при выборе каждого хода имеется в среднем около 30 возможностей. Таким образом, просмотр на два хода вперед требует рассмотрения $30^4 \approx 800\,000$ продолжений. По оценке авторов программы на уменьшенной доске (6×6) число продолжений составляет примерно $20^4 = 160\,000$. Даже при таком сокращении это число все еще очень велико. Для сравнения укажем, что, по наиболее достоверным данным, человек при выборе хода рассматривает значительно менее 100 позиций [273]. Лос-Аламосская программа затрачивает на ход в среднем около 12 *мин*. При такой скорости программа должна быть весьма простой. Это видно также из ее размера — всего 600 слов. Образно выражаясь, машина успевает лишь бегло взглянуть на каждую оцениваемую позицию. Величина каждого из слагаемых оценочной функции определяется непосредственно из рассмотрения возможных продолжений: при взятии фигуры вносится соответствующее изменение в имеющийся материал, а подвижность определяется как число новых позиций, к которым можно прийти, исходя из оцениваемой позиции (и, следовательно, вычисляется почти без всяких усилий, когда исследуются все возможные продолжения).

Лос-Аламосская программа помогает оценить тот предел, до которого можно упрощать процесс в сторону минимизации количества информации, перерабатываемой при оценке каждой позиции. Программа Тьюринга помогает оценить аналогичный предел для процесса минимизации числа исследуемых продолжений. Рассмотренные выше программы, особенно Лос-Аламосская, дают реальную надежду на успех в дальнейших исследованиях. Они показывают, что машина с очень несложной программой способна по крайней мере вступить на шахматное поприще: она может обыграть начинающего игрока.

Программа Бернстейна

За последние два года шахматист и программист фирмы IBM А. Бернстейн написал шахматную программу (поле 8×8) для машины IBM-704 [121, 120]. Эта программа испытывалась на машине, и к настоящему времени на ее счету одна полностью сыгранная партия плюс значительное число разыгранных коротких этюдов. Так же, как и предыдущие программы, она укладывается в шенноновскую схему, однако в ней сделан чрезвычайно важный шаг в направлении усложнения алгоритма: программа рассматривает лишь небольшую часть возможных ходов и продолжений. Она содержит ряд подпрограмм, называемых «генераторами правдоподобных ходов». Каждый из этих генераторов связан с некоторой характеристикой позиции: безопасность короля, развитие фигур, защищенность собственных фигур, нападение на фигуры противника и т. п. Программа рассматривает не более семи возможных ходов в каждой позиции. Генераторы ходов включаются друг за другом в заданном порядке до тех пор, пока не накопится семь различных ходов.

Так же как и в Лос-Аламосской программе, варианты исследуются на два хода вперед. Однако на каждом шаге используются генераторы правдоподобных ходов, так что из каждой позиции возникает до семи вариантов. В качестве оценочной функции берется отношение двух функций — для белых и для черных. Каждая из этих функций представляет собой взвешенную сумму четырех слагаемых: «материал», «защита короля», «контролируемая площадь» и «подвижность». Программа использует минимаксную процедуру и выбирает продолжение с наибольшей рабочей оценкой.

Программа играет неровно: в некоторых случаях она обнаруживает поразительную «слепоту», но иногда выдает и серию очень хороших ходов. Насколько нам известно, эта программа ни у кого не выигрывала; единственную полностью сыгранную партию она проиграла хорошему игроку [121], а со слабыми игроками ни разу не играла.

В программе Бернстейна мы впервые встречаемся с радикальным уменьшением числа рассматриваемых продолжений (механизм отбора). При глубине анализа в два хода она проверяет лишь $7^4 \approx 2500$ возможностей из 800 000. Тот факт, что при сокращении пространства поиска в 300 раз игра программы все

еще остается сносной, означает, что механизм отбора довольно эффективен. Разумеется, правила отбора вытекают из проверенной на практике теории шахматной игры; таким образом, результаты эксперимента с этой программой показывают, что в правилах отбора удачно схвачена суть теории. С другой стороны, такое радикальное сокращение числа рассматриваемых вариантов должно привести к тому, что программа будет обнаруживать сильную тенденцию не замечать связи между ходами и их следствиями. Механизм отбора в программе Бернстейна не имеет той системы проверок и сопоставлений, которыми пользуется человек за шахматной доской. И мы обнаруживаем это, наблюдая за игрой машины. Так, в одной позиции имелась возможность последовательного нападения тремя пешками на слона. Слон был вынужден каждый раз отступать на поле, на котором его могла атаковать следующая пешка. Программа «прозевала» эту возможность из-за отсутствия генератора, который в качестве правдоподобных ходов предложил бы последовательное продвижение пешек для нападения на слона. Но это не должно нас беспокоить. Любая частная трудность устранима; например, в приведенном случае можно добавить генератор, реагирующий на соответствующую особенность позиции. Процесс корректировки правил отбора правдоподобных ходов будет идти параллельно с накоплением фактического материала относительно самой игры в шахматы и работы шахматных программ.

Любое усложнение алгоритма приводит к увеличению сложности программы. Для работы генераторов ходов и вычисления компонент оценочной функции необходима разнообразная информация о каждой позиции. Вследствие этого как размер программы, так и время, расходуемое на оценку одной позиции, оказываются большими, чем для Лос-Аламосской программы.

Из табл. 1 видно, что программа Бернстейна состоит из 7000 машинных слов, — на порядок больше, чем Лос-Аламосская (600 слов). Что же касается быстродействия, то обе программы тратят примерно одинаковое время на то, чтобы сделать ход — 8 и 12 *мин* соответственно. Увеличение размера задачи (в отношении 5 : 1) при переходе от доски 6×6 к доске 8×8 примерно компенсируется увеличением скорости машины IBM-704 по сравнению с машиной MANIAK (также порядка 5 : 1 при учете большей мощности кода первой машины). Поэтому можно считать, что при равных условиях обе программы играли бы с одинаковой скоростью. Следовательно, преимущество, состоящее в уменьшении числа рассматриваемых продолжений (1 : 300), в более сложной программе Бернстейна практически сводится на нет

увеличением объема вычислений, приходящихся в этой программе на один ход, и это несмотря на использование в программе Бернстейна фиксированных таблиц, непосредственно машинного кода и других приемов, направленных на получение максимальной скорости.

Мы провели это сравнение, чтобы обратить внимание на взаимную зависимость двух способов усовершенствования сложных программ: увеличения скорости оценок и усложнения алгоритма отбора продолжений. К сожалению, качество игры обеих программ удается сравнить лишь очень грубо. Чтобы такое сравнение было надежным, необходимо иметь вариант Лос-Аламосской программы для доски 8×8 и, кроме того, обеими программами должно быть сыграно больше партий. Поскольку вариант 8×8 Лос-Аламосской программы с точки зрения приближения к игре шахматиста должен быть лучше, чем вариант 6×6, будем считать, что Лос-Аламосская программа и программа Бернстейна играют примерно одинаково. В этом грубом приближении мы имеем две программы, достигающие одного и того же уровня игры разными путями: Лос-Аламосская — без использования отбора, за счет скорости, а программа Бернстейна — за счет сильного отбора, приводящего к большей затрате времени на рассмотрение каждой позиции.

Мы считаем нужным подчеркнуть, что равенство класса игры обеих программ — чисто случайный факт: совершенствование отбора представляет собой очень мощное средство улучшения качества работы сложных программ, а увеличение скорости — лишь очень слабое средство.

Потребуем, например, чтобы и Лос-Аламосская программа и программа Бернстейна исследовали продолжения не на два, а на три хода вперед. Тогда Лос-Аламосская программа будет затрачивать на ход примерно в 1000 (30^2) раз больше времени, чем сейчас, а программа Бернстейна — всего в 50 (7^2) раз, т. е. в 20 раз меньше. Существенная особенность шахмат — экспоненциальный рост числа продолжений при увеличении глубины анализа. Поэтому по мере возрастания глубины требуемые дополнительные усилия, затрачиваемые на исследование каждой позиции, быстро окупаются, так как замедляют рост числа подлежащих рассмотрению вариантов. Сравнение обеих программ при большей глубине анализа имеет смысл, так как естественный путь совершенствования Лос-Аламосской программы состоит в повышении ее быстродействия настолько, чтобы анализ мог проводиться на глубину в три хода. Кроме того, попытка ввести в Лос-Аламосскую программу отбор продолжений обошлась бы чрезвычайно дорого по сравнению с введением дополнительного отбора в программу Бернстейна.

Приведем еще один пример, показывающий ценность эвристик, которые позволяют исключать отдельные ветви из дерева игры. Пусть мы имеем ветвящееся дерево, которое программа исследует на глубину n ходов, и пусть это дерево имеет четыре ветви в каждой вершине. Если бы удалось удвоить скорость работы программы (т. е. заставить ее исследовать вдвое больше позиций за то же время), то программа смогла бы смотреть на полхода дальше, на глубину $n+1/2$. Если бы удалось во столько же раз усилить отбор, т. е. рассматривать две из четырех ветвей в каждой вершине, глубину анализа можно было бы удвоить.

Подведем итоги. Программа Бернстейна значительно сложнее ранее описанных программ. Хотя она в некоторых отношениях все еще пользуется стандартными упрощениями (например, фиксированной глубиной анализа), эта программа для отбора ходов использует ряд мощных эвристик. Эти эвристики, взятые из шахматной практики, резко сокращают число подлежащих рассмотрению ходов в каждой позиции.

Программа Ньюэлла, Шоу и Саймона

В нашей работе над шахматной программой, начатой еще в 1955 г., был длительный перерыв, в течение которого мы работали над программой для доказательства теорем математической логики [787—789]. В принципиальном отношении доказательство теорем и игра в шахматы представляют собой разные варианты одной и той же задачи. Эта задача заключается в нахождении эвристик, позволяющих выбрать пути, ведущие к цели, причем число возможных путей с увеличением расстояния от исходной точки до цели растет по экспоненциальному закону. В обеих задачах возникают одни и те же дилеммы — скорость или отбор, простота или сложность программы. Затраты на программирование, сопутствующие увеличению сложности, также примерно одинаковы для обеих задач. Поэтому мы недавно снова вернулись к шахматной программе, вооруженные идеями, которые возникли при работе над программой для доказательства логических теорем.

Путь, приведший нас к этим исследованиям, несколько отличается от того, которым шли упомянутые выше ученые. Мы в первую очередь стремились описать и понять процессы мышления и процесс принятия решения человеком [791, 793]. Однако как у шахматистов, так и у программистов шахматной игры избираемый подход в значительной степени диктуется структурой задачи, так что наша нынешняя программа может быть

описана в тех же терминах, что и приведенные выше программы. В ней легко различить большинство черт, свойственных ранее созданным программам: основная схема, предложенная Шенноном; понятие «мертвой» позиции, введенное Тьюрингом; генераторы ходов Бернстейна, связанные с особенностями ситуации на доске. Пожалуй, единственная особенность, общая всем остальным программам, но полностью отсутствующая в нашей (а по нашему убеждению, и у человека), — это применение аддитивной оценочной функции для сравнения возможных ходов.

Структура программы

Структура программы показана на фиг. 3. Имеется набор целей, каждая из которых соответствует некоторой особенности

	Определение цели	Генератор ходов	Статическая оценка	Генератор анализирующих ходов
Цели	Безопасность короля			
	Материальный баланс			
	Контроль над центром			
	Развитие фигур			
	Атака на короля			
	Построение пешек			

Фиг. 3. Структура программы Ньюэлла, Шоу и Саймона

шахматной позиции: «безопасность короля», «материальный баланс», «контроль над центром доски» и т. п. С каждой целью связана совокупность алгоритмов, соответствующих элементам шенноновской схемы: генераторы ходов, вычисление статической оценки и генераторы анализирующих ходов. Алгоритм объединения статических оценок в рабочие оценки для каждого из предложенных ходов и правило окончательного выбора общие для всей программы и поэтому на рисунке отсутствуют.

Цели

Программа представляет собой набор подпрограмм, каждая из которых соответствует определенной *цели*. Эти подпрограммы независимы: добавление или удаление любой из них не влияет на работу остальных. Перед каждым ходом предварительный анализ устанавливает, что данной шахматной позиции («состоянию») соответствует некоторый набор целей. Процедура определения цели, показанная для каждой цели на фиг. 3, дает информацию, которая используется при этом исходном выборе целей. Отобранные цели заносятся в список в порядке их важности. Этот список затем управляет работой остальных алгоритмов: отбора возможных ходов и исследуемых продолжений, получения статической оценки и окончательного выбора.

Очевидно, что стиль игры программы будет зависеть от того, какие цели в нее заложены и какие цели она выбирает при каждом конкретном ходе. Организация в виде набора независимых подпрограмм дает программе определенную гибкость в ходе игры в отношении выбора подлежащих рассмотрению ситуаций. (Например, цель «избежать пата» уместна лишь в некоторых окончаниях, в которых противник вынужден защищаться, а его король ограничен в движении.) Кроме того, такая структура программы предоставляет в наше распоряжение удобное средство для исследования шахматных программ, позволяя добавлять совершенно новые соображения в сложную, но еще управляемую программу.

Генерирование ходов

Генератор ходов, связанный с данной целью, предлагает ходы, направленные на достижение этой цели. Эти генераторы должны находить положительные доводы в пользу предлагаемого хода. Например, в дебюте ход d2—d4 предложит только тот генератор, который связан с «контролем над центром»; только генератор, связанный с «материальным балансом», предложит убрать фигуру, находящуюся под ударом. Эти генераторы ходов соответствуют генераторам Бернстейна с той лишь разницей, что они предназначены исключительно для генерирования очередных ходов и не используются для генерирования продолжений, которые рассматриваются при анализе данного хода. В программе Бернстейна и тем более в Лос-Аламосской программе одни и те же генераторы используются как для отыскания возможных ходов, среди которых делается окончательный выбор, так и для получения продолжений, которые необходимо исследовать для оценки последствий того или иного хода. В нашей программе последнюю функцию выполняет отдельный набор генераторов.

Оценка

Каждому ходу, предложенному одним из генераторов ходов, в процессе анализа присваивается некая оценка. Выше мы говорили, что генераторы ходов ответственны за нахождение обоснованных доводов в пользу предложенного хода. В соответствии с этим в процессе анализа выясняется лишь приемлемость хода, выданного генератором. Генератор — предлагает, алгоритм анализа — оценивает.

Оценка, присваиваемая данному ходу, получается из сочетания оценок, соответствующих каждой цели. Если угодно, ее можно рассматривать как вектор, с той лишь разницей, что этот вектор не обязательно должен сохранять неизменные компоненты на протяжении всей партии, поскольку эти компоненты определяются очередным списком целей, который составляется перед каждым ходом исходя из имеющейся позиции. Каждая компонента выражает приемлемость или неприемлемость данной позиции с точки зрения цели, соответствующей этой компоненте. Так, алгоритм статической оценки, связанный с целью «материальный баланс», проверяет только потерю или выигрыш фигуры, алгоритм «развития фигур» — только относительный выигрыш или потерю темпа, алгоритм «построения пешек» — наличие сдвоенных и изолированных пешек и т. п. Некоторые компоненты выражаются целыми числами. Например, при подсчете материального преимущества мы пользуемся условными пешечными эквивалентами: 9 пешек за ферзя, 5 за ладью и т. д. В других случаях компоненты могут принимать лишь значения 0 или 1, что означает наличие или отсутствие в позиции некоторых свойств, таких, как блокирование хода или сдвоенные пешки.

В нашей программе, как и в других программах игры в шахматы, анализ предлагаемых ходов можно разделить на три части: исследование продолжений на некоторую глубину, нахождение статических оценок и получение рабочих оценок этих ходов. В каждой достигнутой в процессе анализа позиции генераторы анализирующих ходов определяют подлежащие исследованию ветви (более подробно этот процесс будет описан ниже). В конечной для каждого продолжения позиции вычисляется векторная оценка. Далее для получения рабочей оценки предложенного хода к этим конечным статическим оценкам применяется минимаксная процедура.

Для нахождения минимаксной оценки должна иметься возможность сравнения любых двух оценок и выбора более предпочтительной из них или установления их равенства. В нашем случае это означает, что векторы оценок должны быть строго

упорядочены при любых вариациях размера и содержания списка целей. Мы применяем лексикографическое упорядочивание: множество значений каждой компоненты вполне упорядочено, а высшие (в соответствии со списком целей) компоненты полностью доминируют над низшими. При сравнении двух оценок сначала сравниваются первые компоненты. Если одна из них предпочтительнее, то и вся оценка считается более предпочтительной. Если эти компоненты равны, сравнивается вторая пара и т. д.

Окончательный выбор хода

После того как все предложенные ходы в процессе анализа получили оценки, необходимо выбрать из них какой-то один. В ранее описанных программах правило окончательного выбора вытекало из минимаксной процедуры: выбирался ход с наибольшей оценкой. Очевидность этого правила основана на предположении, что выбор производится из некоторого фиксированного множества возможных ходов. Если это не так (например, ходы предлагаются последовательно), то возможны и другие правила. Простейшее правило, которое в настоящее время применяется в программе, состоит в том, что выбирается первый же приемлемый ход. (Ход считается приемлемым, если его оценка выше установленного порога.) Исполнительная программа просматривает сверху вниз список целей. При этом включаются генераторы ходов, причем ходы, порождаемые более важными целями, рассматриваются первыми. Программа помнит ход, получивший к данному моменту наивысшую оценку, и выполняет его, если не находится ни одного «приемлемого» хода.

Другое возможное правило окончательного выбора состоит в отыскании такого приемлемого хода, который выполнял бы двоякую функцию, т. е. хода, который был бы предложен как наилучший более чем одним генератором. Для этого исполнительная программа, следуя вдоль списка целей, ищет сначала какой-либо приемлемый ход, как описано выше. После того как он найден, программа включает остальные генераторы, чтобы проверить, не будет ли этот ход предложен еще раз. Если нет, программа продолжает работать со списком ходов, проверяя, не является ли какой-нибудь дважды предложенный ход приемлемым. Если нет, делается первый приемлемый ход или лучший ход из неприемлемых. Такое правило представляется довольно разумным, поскольку понятие многоцелевого хода играет важную роль в шахматной литературе.

В третьем варианте правила окончательного выбора все цели делятся на два списка. Первый список содержит все те особенности позиций, которым обычно уделяется внимание. Во второй список входят особенности, которые редко встречаются, но могут оказать очень сильное влияние на ход партии. В этом списке окажутся цели, связанные с жертвами, замаскированными «вилками», «связками», от которых нельзя уйти за два хода, и т. п. Исполнительная программа находит приемлемый ход из первого, обычного списка, а оставшееся время тратит на поиск редко встречающихся возможностей, связанных с целями из второго списка.

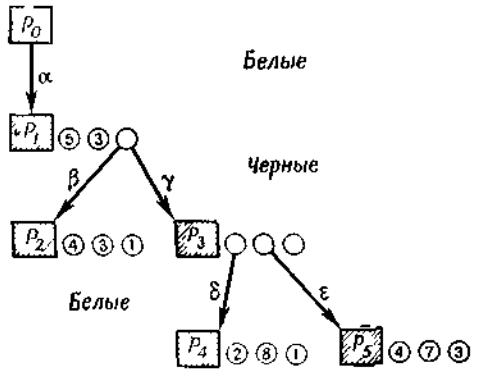
Анализ

При описании структуры программы мы не рассматривали детально механизм исследования продолжений, а просто предполагали, что некоторые продолжения исследуются, после чего для них вычисляются статические оценки и минимаксным методом находятся рабочие оценки. Между тем важность самого этого механизма очевидна. Генераторы анализирующих ходов — основное средство выбора продолжений в программе. В каждой позиции, появившейся в процессе анализа, они определяют подлежащие исследованию ветви, которые исходят из этой позиции, а следовательно, среднее число ветвей дерева и его среднюю глубину. Генераторы возможных ходов и правила окончательного выбора также влияют на количество рассматриваемых продолжений. Но они производят выбор только один раз, в то время как генераторы анализирующих ходов делают это на каждом шаге (полуходе).

Исследование продолжений основано на обобщении понятия «мертвой» позиции, введенного Тьюрингом. Напомним, что Тьюринг применял это понятие к размену фигур, когда до окончания размена нет смысла подсчитывать материал на доске. Мы используем это понятие по отношению к каждой особенности позиции. Статическая оценка для некоторой цели имеет смысл лишь в том случае, если позиция расценивается как «мертвая» в отношении этой цели, иначе говоря, если следующий ход не в состоянии сильно изменить эту компоненту статической оценки. Генераторы анализирующих ходов в каждой позиции прежде всего определяют, не является ли она «мертвой» (каждый генератор делает это по отношению к своей цели). Если нет, то генерируются ходы, которые оправданы в данной позиции и в то же время могут серьезно повлиять на статическую оценку данной цели. Таким образом, отбор подлежащих исследованию продолжений осуществляется в процессе поиска «мертвых» для всех целей позиций, в которых можно было бы дать статические

оценки. При таком способе генерирования продолжений в зависимости от ситуации меняются как число ветвей в каждой позиции, так и глубина дерева. Спокойные позиции будут порождать значительно менее разветвленные деревья, чем сложные позиции в середине партии.

Чтобы сделать алгоритм анализа более понятным, на фиг. 4 приведен схематический пример возможной ситуации. Здесь P_0 — начальная позиция, из которой машина, играющая белыми,



Фиг 4 Анализ

должна сделать ход. Стрелкой α обозначен ход, предложенный одним из генераторов. Машина делает этот ход «в уме» (иначе говоря, «рассматривает» его), получая позицию P_1. Цель анализа — получить оценку этой позиции. Полученная оценка будет принята в качестве рабочей оценки хода α. Рассматривая по очереди каждую цель из списка целей, машина пытается дать им соответствующую статическую оценку. В данном случае это можно сделать для первой и второй компонент, которым присваиваются значения 5 и 3. (В данном примере оценки для простоты даются в виде чисел. В действительности используются различные множества упорядоченных символов, причем структура каждого из этих множеств зависит от характера вычислений.)

Однако генератор, связанный с третьей компонентой, обнаруживает, что позиция не является «мертвой», и генерирует два хода, β и γ. Рассмотрение первого из них, β, приводит к позиции P_2, и делается попытка дать статическую оценку этой позиции. Далее все происходит точно так же, как и с P_1, с той лишь разницей, что на этот раз позиция оказывается «мертвой» для всех

целей, так что находится ее статическая оценка (4, 3, 1). Затем рассматривается второй ход, γ, ведущий к P_3. Попытка дать статическую оценку для P_3 наталкивается на трудность уже для первой компоненты, генератор которой выдает ход δ, ведущий к P_4. Для этой позиции получается оценка (2, 8, 1). Позиция P_3 оказывается не «мертвой» и для второй компоненты. В результате генерируется единственный ход ε, ведущий к позиции P_5, которая получает оценку (4, 7, 3). Относительно третьей компоненты позиция P_3 является «мертвой». Таким образом, исследование вариантов закончено, и все конечные позиции имеют статические оценки. Так как в позиции P_3 ход белых, белые выберут продолжение с наибольшей оценкой, т. е. ход ε с оценкой (4, 7, 3) (первая компонента доминирует). Эта оценка присваивается P_3. Теперь перед черными стоит выбор между ходом β, ведущим к P_2 (4, 3, 1), и ходом γ, ведущим к P_3 (4, 7, 3). Поскольку черные стремятся минимизировать оценку, они выберут β. Итак, в качестве рабочей оценки хода α принимается (4, 3, 1). На этом анализ заканчивается.

Минимаксная процедура проводится одновременно с генерированием ветвей. Так, например, если позиция P_5, имеющая оценку (4, 7, 3), появится раньше, чем P_4, ходы из позиции P_3 не будут больше генерироваться, так как уже очевидно, что черные предпочтут позицию P_2 позиции P_3. Оценка P_3 будет не меньше, чем оценка P_5, так как белые стремятся максимизировать оценку.

Этот алгоритм анализа нельзя назвать простым ни в теоретическом, ни в техническом отношении. Имеется много способов ограничения поиска и получения рабочих оценок, но не существует алгоритмов, гарантирующих сходимость процесса поиска. Успех в сильной степени зависит от способности находить правильные статические оценки. Чем больше позиций мы можем правильно оценить без рассмотрения возможных продолжений, тем быстрее будет заканчиваться процесс поиска. Число правдоподобных ходов, требующих оценки, также имеет значение (об этом говорилось выше в связи с программой Бернстейна), но существует предел, ниже которого это число нельзя уменьшить. Пусть, например, позиция не является «мертвой» по отношению к «материальному балансу» и одна из фигур находится под ударом. Тогда машина может попробовать: а) взять атакующую фигуру; б) усилить защиту; в) вывести атакованную фигуру из-под удара; г) положиться на имеющуюся защиту; д) прикрыться другой фигурой; е) предпринять контратаку. Ходы каждого из перечисленных типов необходимо искать и опробывать — все они правдоподобны и способны сильно повлиять на «материальный баланс».

Чтобы привести пример использования эвристики при нахождении статических оценок, представим себе, что вышеуказанная позиция появилась в процессе анализа при каком-то продолжении и что в начале этого продолжения машина уже потеряла пешку. Так как в этой позиции машина должна защищаться, шансы на возмещение пешки очень невелики. Следовательно, статическая оценка позиции не может быть выше, чем «минус одна пешка». При минимаксном подходе этого, как правило, достаточно, чтобы прекратить дальнейшее рассмотрение варианта.

Заключение

Повторим кратко описание программы. Программа организована в виде набора целей. Эти цели соответствуют основным понятиям, применяемым в шахматах: безопасность короля, проходные пешки и т. п. С каждой целью связано несколько подпрограмм:

1) алгоритм, определяющий уместность цели в данной позиции;

2) генератор ходов, который отыскивает ходы, полезные с точки зрения данной цели;

3) алгоритм для вычисления статической оценки любой позиции в отношении данной цели;

4) генератор анализирующих ходов, который находит продолжения, приводящие к «мертвым» позициям.

Возможные в данной позиции ходы предлагаются генераторами в порядке важности соответствующих целей. После того как ход предложен, он анализируется.

В процессе анализа генерируются продолжения и каждое продолжение доводится до «мертвой» позиции, а затем для этих позиций вычисляются статические оценки. Эти оценки сравниваются для получения минимаксной рабочей оценки каждого предложенного хода. Правило окончательного выбора может основываться на различных критериях, например на выборе первого предложенного хода, рабочая оценка которого больше некоторой заданной величины.

Примеры целей

В этом разделе мы в качестве примеров рассмотрим две цели и связанные с ними подпрограммы. Первый пример — «контроль над центром».

Контроль над центром

Определение. Цель заносится в список всегда, когда имеются центральные пешки, которые можно подвинуть на 4-ю горизонталь.

Генератор ходов.

1) d2—d4, e2—e4 (основные ходы);
2) помешать противнику сделать основной ход;
3) подготовить собственные основные ходы:
   а) усилить защиту поля d4 или e4,
   б) убрать фигуры, блокирующие центральные пешки.

Статическая оценка. Подсчитывается число блокирующих фигур, мешающих сделать основной ход.

Генератор анализирующих ходов. Отсутствует: статическая оценка всегда возможна.

Поясним сказанное. Цели предлагаются исходя из общей ситуации, например для стадии дебюта. Список целей для позиции составляется в результате поочередной работы программ определения цели для всех потенциально возможных целей. Каждая программа определяет, имеет ли данная цель отношение к делу в данной конкретной позиции или нет. Для цели «контроль над центром» не требуется сбора специальной информации; считается, что эта цель не имеет отношения к делу, если центральные пешки уже продвинуты на 4-ю горизонталь или дальше.

Самая важная часть программы «контроля над центром» — это генератор ходов. Генератор рассматривает два основных хода: продвижение пешки на d4 и на e4. Он предлагает эти ходы, если они разрешены правилами. Алгоритм анализа обязан (это касается всех целей) отклонить ход, если он может нанести ущерб, например если в результате такого хода противник возьмет эту пешку. Так, после 1. d2—d4, d7—d5 генератор контроля над центром предложит 2. e2 — e4, но (как мы дальше увидим) программа оценки «материального баланса» отклонит этот ход из-за потери пешки 2 ... de. Генератор «контроля над центром» не имеет никакого отношения к выяснению этих последствий.

Если ни один из основных ходов сделать нельзя, генератор ходов может либо подготовить их, либо помешать противнику сделать основные ходы. Стиль игры программы будет существенно зависеть от того, имеет ли вторая подцель приоритет над первой (как в приведенном выше описании генератора) или наоборот. Предложенный нами порядок приводит, по-видимому, к более активной и несколько лучшей игре в дебюте. Анало-

гично стиль игры зависит от того, какая пешка рассматривается первой: ферзевая или королевская.

Подцель «помешать противнику сделать основной ход» выполняется следующим образом. Прежде всего генератор ходов определяет, может ли противник сделать такой ход, пробуя этот ход и оценивая его с точки зрения противника. Если один или оба из основных ходов не отвергаются, мешающие ходы будут служить определенной цели. В этом случае генератор находит ходы, которые усиливают давление на поля d5 и e5 или связывают фигуры противника, защищающие эти поля. Среди прочих ходов для этой цели обычно предлагаются Kc3, Kf3 и продвижение пешек на c4 и f4.

При использовании генератором ходов подцели «подготовить собственные основные ходы» генератор прежде всего выясняет, почему эти ходы нельзя сделать без подготовки — потому ли, что пешка блокируется собственной фигурой, или потому, что поле на 4-й горизонтали находится под ударом. В первом случае генератор предлагает подходящие ходы для мешающей фигуры, во втором находит ходы, усиливающие защиту соответствующего поля.

Это все касалось генератора ходов, связанного с «контролем над центром». Статическая оценка для этой цели выполняет по существу «отрицательные» функции — ее задача обеспечить, чтобы не были сделаны ходы, предложенные для каких-либо других целей, но подвергающие опасности контроль над центром. Возможность того, что генератор ходов для других целей *случайно* выдаст ход, способствующий контролю над центром, попросту игнорируется. Следовательно, задача статической оценки для «контроля над центром» заключается лишь в том, чтобы не были сделаны ходы, препятствующие ходам e4 и d4. Типичным примером таких ходов могут служить ходы Cd3 и Ce3, если соответствующие центральные пешки еще стоят на 2-й горизонтали. Программа статической оценки отклонит эти ходы.

Второй пример цели — «материальный баланс». Эта цель значительно шире и сложнее, чем контроль над центром. В компетенцию этой части программы входит решение всех вопросов, связанных с материальным выигрышем или потерей при следующем ходе. Сюда не относятся угрозы типа «связок» и «вилок», которые могут повлиять на материальный баланс не раньше, чем через ход. Рассмотрение таких угроз относится к компетенции других целей. Положительный и отрицательный аспекты материального баланса должны быть включены в одну и ту же цель, поскольку они непосредственно взаимно компенсируют друг друга и часто приходится жертвовать фигуру для того, чтобы затем добиться материального перевеса.

Материальный баланс

Определение. Составляется список занятых своими фигурами полей, которые находятся под ударом, и аналогичный список полей, занятых фигурами противника. Для каждого такого поля составляется список, включающий фигуру, стоящую на этом поле, атакующие фигуры и защищающие фигуры (включая, например, обе ладьи, если они сдвоены на соответствующей вертикали или горизонтали). Статическая оценка для каждого разменного поля определяется как изменение материального баланса в результате полного размена всех атакующих и защищающих это поле фигур. Косвенные последствия размена, например потери из-за имевшихся связок, вскрытые нападения и т. п., во внимание не принимаются. Поля, получившие положительную оценку (с точки зрения защищающейся стороны), исключаются из списка. Поля с отрицательной оценкой заносятся в список в порядке убывания абсолютной величины оценки. Кроме того, составляется список всех связанных фигур.

Генератор ходов. Ходы генерируются, начиная с того поля, которое первым стоит в списке. Если это наиболее важное разменное поле занято фигурой противника, эта фигура бьется наименее ценной нападающей фигурой. Если ход отклоняется из-за того, что нападающая фигура связана, пробуется атака следующей фигурой. Если же ход отклонен по другой причине, программа отказывается от размена на этом поле и проверяет возможность размена на следующем поле из списка.

Если рассматриваемое разменное поле занято своей фигурой, генерируется целая серия возможных ходов:

а) попробовать «не ходить», чтобы выяснить, приведет ли это к потерям;

б) взять атакующую фигуру противника;

в) добавить к защите свободную от защиты других полей фигуру;

г) отойти атакованной фигурой;

д) заслонить атакованную фигуру (если заслоняющая фигура не несет других нагрузок и если противник ее при этом не возьмет);

е) связать нападающую фигуру свободной от других функций фигурой, если противник не сможет ее взять.

Статическая оценка. Для каждого разменного поля цены своих фигур складываются, а цены фигур противника вычитаются. Принята следующая система цен: ферзь — 9, ладья — 5, конь и слон — по 3, пешка — 1.

Генераторы анализирующих ходов. Позиция считается «мертвой» по отношению к этой цели, если в данной позиции нет

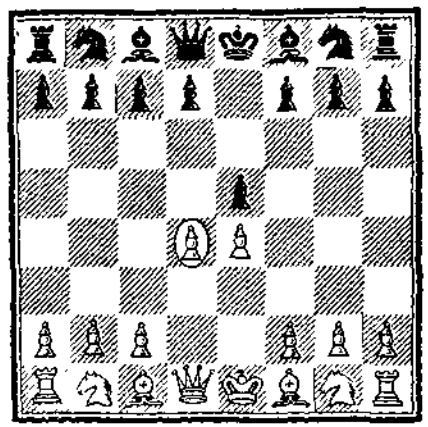
фигур, находящихся под боем, т. е. если список разменных полей пуст. В этом случае можно дать статическую оценку. В противном же случае для развязки должен быть применен один из ходов, перечисленных при описании генератора ходов. Однако для уменьшения числа рассматриваемых продолжений вводятся разного рода ограничения. Например, если в ходе размена уже понесены потери и фигура снова находится под ударом, возмещение потерянного материала маловероятно. Поэтому такая позиция считается «мертвой», и для нее вычисляется статическая оценка.

Здесь можно было, конечно, дать лишь беглый набросок эвристик, используемых при анализе материального баланса. Но уже из этого краткого описания ясно, что их должно быть очень много и они должны включать множество неявных предположений о том, что́ важно и что́ не важно на шахматной доске.

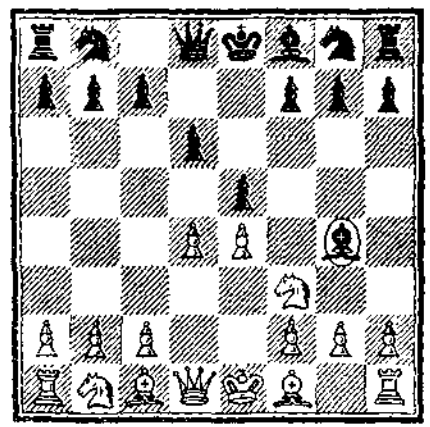
Поведение программы

Относительно поведения программы пока имеется лишь небольшой фактический материал. Программа была написана весной этого года и еще полностью не отлажена. В машинном коде запрограммированы только две цели: «материальный баланс» и «контроль над центром». Полностью разработана также цель «развитие фигур», а цель «построение пешек» — в объеме, достаточном для дебютной стадии, где ее роль в первую очередь состоит в предотвращении нежелательных структур типа «сдвоенных пешек». Этих четырех целей, расположенных в следующем порядке: «материальный баланс», «контроль над центром», «развитие фигур» и «построение пешек», по-видимому, достаточно для первой фазы дебюта. Для полного разыгрывания дебюта необходимо добавить еще несколько целей: «безопасность короля», «серьезные угрозы» и «гамбиты». В описании цели «серьезные угрозы» можно для начала ограничиться «вилками» и «связками».

Мы провели многочисленные опыты по моделированию программы вручную в типичных позициях. Взаимодействие целей показано ниже на двух примерах. В позиции, изображенной на фиг. 5, белыми играет машина. Игра началась ходом 1. e2 — e4, e7 — e5. Генератор «материального баланса» не предлагает ходов, так как в данной ситуации размен невозможен. Генератор «контроля над центром» предлагает ход 2. d2—d4 (отмеченный на схеме кружком). Этот ход совершенно резонно отклоняется, и очень поучительно рассмотреть, как это происходит. Ход предлагается для анализа. Позиция не является «мертвой» с точки зрения «материального баланса», поэтому генерируется ход черных 2 ... ed. Получившаяся позиция снова не «мертвая». Генерируется 3. Ф:d4. Теперь позиция «мертвая» в отношении «материального баланса», и программа получает для первой компоненты статической оценки значение «равенство». Этот ход не мешает движению пешек, поэтому статическая оценка со стороны «контроля над центром» оказывается приемлемой. Однако с точки зрения третьей цели — «развитие фигур» — позиция не является «мертвой», так как белые выставили незащищенную фигуру. Генерируется ход 3 ... Кс6,



Фиг 5.



Фиг. 6

и белые вынуждены отойти ферзем, потеряв темп. Поэтому ход 2. d2 — d4 отклоняется программой «развитие фигур» (ход 3 ... Сс5 приводит к потере слона и поэтому не принимается во внимание при оценке варианта).

Второй пример (фиг. 6) взят из знаменитой партии Морфи против игравших совместно герцога Брауншвейгского Карла и графа Изуара. Партия начинается так: 1. e2 — e4, e7 — e5 2. Kg1 — f3, d7 — d6. Что сделает машина в этой позиции, играя черными? Генератор «материального баланса», заметив, что размен грозит черным потерей пешки, предлагает 3 ... Cg4. Именно этот ход и был сделан герцогом и графом. Анализ приводит далее к 4. de, de. При этом появляется возможность размена ферзей, которая и проверяется: 5. Ф : d8, Кр : d8, 6. К : e5. Таким образом, это продолжение приводит к потере пешки. Поэтому рассматривается ближайший ход, при котором черные могли бы сделать лучший выбор. Так как на пятом ходе

у черных выбора нет, рассмотрим 4′ ... С : f3. Тогда 5′. gf, de 6′. Ф : d8, Кр : d8. Этот вариант отклоняется программой «развитие фигур», так как вынужденный ход королем лишает черных права на рокировку. Этого достаточно, чтобы отклонить ход 3 ... Cg4, даже не рассматривая более сильного ответа белых: 5″. Ф:f3, который выбрал Морфи. В нашей программе ход 5. gf генерируется раньше, чем 5. Ф:f3. Любой из них показывает, что ход 3 ... Cg4 ошибочен.

Эти примеры приведены, чтобы проиллюстрировать эвристику, которая нас больше всего привлекает при создании шахматных программ. Мы хотим иметь программу, которая не только делает хорошие ходы, но и приходит к ним благодаря правильным рассуждениям. Приведенные выше комментарии довольно типичны для анализа, проводимого человеком; они весьма близки также к анализу, который производит программа. Мы думаем, что это правильный путь конструирования сложных программ.

Рассмотрим другой пример: четырехцелевая программа для игры в дебюте не способна к жертвам и, напротив, всегда будет принимать гамбиты. Это происходит потому, что она не умеет сравнивать материальное преимущество с позиционным. Программа будет учитывать возможность жертв, если ввести в нее дополнительную цель, связанную с такой возможностью. Соответствующая эвристика, которой пользуется шахматист, выглядит так: не жертвуй, пока не понял, что дает эта жертва. Иначе можно выразиться и так: успех человека в игре в некоторой мере зависит от его способности вырабатывать подходящие понятия. В истории шахмат важной вехой было, например, возникновение понятия «центра» и представления о том, что означает овладение центром, контроль над центром и т. д. Не следует ожидать, что нечто эквивалентное такому понятию возникнет само собой из вычислений, основанных на учете совершенно иных особенностей позиции.

Программирование

Описываемая нами программа очень сложна. Усложнению подверглись почти все элементы основной схемы, введенной Шенноном. Для отбора ходов и получения оценок используется множество весьма специальных эвристик. Можно ожидать, что программа фактически окажется даже намного больше и запутанней и потребует значительно больше операций в расчете на одну позицию, чем программа Бернстейна.

Во введении мы отмечали тесную связь между сложностью задачи и средствами ее описания. Алгоритмы, подобные Лос-

Аламосской программе, окажутся невообразимо сложными, если их записать не в машинном коде, а, скажем, в виде электронных схем. Программа Бернстейна даже в машинном коде оказывается очень сложной; она требует значительных усилий от программиста, а в некоторых своих частях и изощренной техники программирования. Нашу же программу и вовсе невозможно записать непосредственно в машинном коде: для ее записи требуется более мощный язык.

В связи с работой над программами для доказательства теорем нами была разработана серия языков, названных «информационными» (Information Processing Language, сокращенно — IPL) [805]. Настоящая шахматная программа написана на одном из них, IPL-IV. Информационный язык — это интерпретируемый символический код. Существует записанная в машинном коде программа для машины JOHNNIAC, которая интерпретирует программу, написанную на IPL, и исполняет ее.

Во время работы в машине находятся как интерпретирующая программа, так и рабочая программа на IPL.

В этой статье невозможно дать детального описания IPL-IV или методики, применяемой при программировании игры в шахматы. IPL предназначен по существу для работы со списками. Он позволяет в процессе исполнения программы формировать чрезвычайно сложные структуры списков, не планируя заранее объем промежуточных результатов и выделение для них соответствующего объема памяти. Он позволяет также легко организовать неограниченную иерархию подпрограмм и допускает рекурсивное определение программ. Язык IPL сам по себе, т. е. до тех пор, пока мы не приступили к программированию конкретной задачи, является универсальным языком (хотя лучше всего он подходит для работы со списками, подобно тому, как алгебраические компиляторы лучше всего подходят для численной оценки алгебраических выражений).

Для программирования игры в шахматы на языке IPL составляется полный «шахматный словарь». Этот словарь состоит из набора подпрограмм, выражающих основные шахматные понятия. Сюда входят подпрограммы проверок — имеет ли одна фигура некоторое отношение к другой и находятся ли обе фигуры на одной диагонали; подпрограммы, определяющие, в каком направлении находится одна фигура по отношению к другой, или находящие первую фигуру в заданном направлении относительно другой фигуры; подпрограммы последовательного просмотра всех фигур данного типа или всех полей в данном

ряду. В этот основной словарь входит около 100 терминов. Собственно шахматная программа, как видно из этой статьи, широко использует термины шахматного словаря. Таким образом, в программе имеются четыре «языковых уровня»: машинный код JOHNNIAC, обычный IPL, основной шахматный словарь и собственно шахматная программа.

Теперь можно дать грубую оценку размера и сложности этой программы по сравнению с другими программами. В табл. 1 указано, что программа в настоящее время состоит из 6000 слов и будет, вероятно, увеличена до 16000 слов. Верхняя граница определяется емкостью магнитного барабана (JOHNNIAC не имеет магнитной ленты). Благодаря пирамидальной структуре эта программа уже сейчас намного больше программы Бернстейна, хотя трудно оценить коэффициент «расширения» при преобразовании IPL в машинный код. (Прежде всего неясно, как могла бы быть организована «эквивалентная» программа в машинном коде.) Как бы то ни было, в нашей программе лишь около 1000 слов, записанных в машинном коде, а программы на IPL занимают 3000 слов, из которых некоторые отделяет от машинного кода до 10 иерархических уровней. Кроме того, все 12000 слов на барабане будут словами IPL; никаких дополнительных данных или машинных команд не предусматривается.

По оценке, указанной в табл. 1, программа будет затрачивать от 1 до 10 *час* на ход, хотя в спокойных ситуациях, например в дебюте, ход будет занимать всего несколько минут. Даже если принять в расчет разницу в скорости IBM-704 и JOHNNIAC, наша программа, по-видимому, будет играть по крайней мере в 10 раз медленнее, чем программа Бернстейна. Этот разрыв отчасти отражает тот факт, что существующие цифровые машины не приспособлены для обработки такого рода информации, который требуется при шахматной игре [989]. Применение интерпретирующего кода типа IPL означает по существу моделирование «IPL-машины» на обычной машине. Такое моделирование требует большой вычислительной работы, помимо той, которая затрачивается на выполнение самой шахматной программы. Но указанный разрыв отражает также трудность точного описания сложных процессов: мы не умеем писать для них программы и одновременно обеспечивать высокий к. п. д. этих программ.

В свете изложенного мы считали важным исследование методики программирования и языков, подходящих для описания сложных процессов. При этом мы временно игнорировали требование, чтобы игра программы не оказалась слишком «дорогой» с точки зрения машинного времени.

Заключение

На этом мы заканчиваем обзор попыток запрограммировать игру в шахматы для вычислительной машины. В изложенных работах отчетливо видно стремление к созданию все более сложных программ и применению эвристик, обладающих большей избирательностью. Развитие идет также в направлении использования принципов игры, подобных тем, которыми пользуются шахматисты. Отчасти эта тенденция объясняется, по крайней мере в нашем случае, желанием моделировать мыслительный процесс. Однако еще в большей степени она отражает ограничения, накладываемые самой задачей на любую систему обработки информации, которая призвана ее реализовать. Мы убеждены, что любая система переработки информации (человек, вычислительная машина или любая другая), которая успешно играет в шахматы, будет пользоваться эвристиками того же типа, что и человек.

При этом мы, конечно, не забываем, что человек во всех отношениях радикально отличается от машины. Мы хотим лишь сказать, что для задач, которые невозможно решать без очень сильного отбора вариантов (а шахматы — одна из таких задач), основная цель программы — обеспечить такой отбор. Программы более высокого уровня, участвующие в этом, должны быть в основном такими же, какие бы процессы ни происходили в них на низших уровнях. Мы отнюдь не утверждаем, что программы не будут приспосабливаться к тем или иным полезным особенностям используемых для их реализации вычислительных устройств, например к высокой скорости и точности современных цифровых вычислительных машин, которые, по-видимому, позволят исследовать большее число продолжений в каждой позиции. Однако ни одно из известных нам различий (в скорости, объеме памяти и т. п.) не меняет существа задачи: поиск в пространстве возможностей, число которых растет по экспоненциальному закону. Поэтому приспосабливание к конкретным особенностям вычислительных устройств всегда будет второстепенным делом, хотя и оправдывающим затраченное на него время.

Сложность эвристических программ требует более мощного языка, чем язык элементарных машинных команд. Мы видели, что это привело к созданию новых языков для переработки информации. Но и с появлением этих мощных интерпретирующих языков общение с машиной осталось достаточно сложным делом. Поэтому необходимо сделать следующий шаг: создать программы, которые придали бы машинам способность понимать и интерпретировать человеческие команды, сравнимую с их

способностью «решать задачи» применительно к игре в шахматы и доказательству теорем.

Такой интерпретатор, который превратит машину в ученика, а человека в учителя, будет не пассивным транслятором, работающим по достаточно простому алгоритму (каковы даже самые лучшие современные интерпретаторы и компиляторы), а активной, сложной, эвристической программой. После того как наши исследования эвристических программ для игры в шахматы и других задач помогут нам понять, как создать такой интерпретатор, они в конце концов позволят нам перейти с того низкого уровня, на котором сейчас находится связь человека с машиной, на тот высокий уровень который, как мы твердо убеждены, может быть достигнут.

Приложение

Партия, сыгранная Г. Саймоном с шахматной программой

Ниже приведена партия, сыгранная шахматной программой CP-I Ньюэлла, Шоу и Саймона. Ее противником был проф. Г. А. Саймон, игравший черными. Каждый ход занимал от 2 до 50 *мин* (на машине JOHNNIAC). Программа имела три цели: материальное преимущество, контроль над центром и развитие фигур. Отсутствие целей, соответствующих безопасности короля, серьезным угрозам и прохождению пешек, существенно отразилось на игре программы и сделало ее нечувствительной к некоторым аспектам позиций. Комментарии, выделенные курсивом, принадлежат Эдварду Ласкеру. Остальные замечания сделаны Дж. Бэйлором и С. Стрэссеном.

	CP-I	Г. Саймон
1.	d4	Kf6
2.	Kc3	

Белые готовятся ходом e4 овладеть центром. Недостаток этого хода — блокирование пешки c, которая, если бы она была продвинута на c4, контролировала бы поле d5.

2.	. . .	d5
3.	Фd3?	

Этот ход подготавливает e4. Однако: 1) прежде чем выводить ферзя, следует развить более легкие фигуры, 2) ферзь становится теперь объектом нападения для черных фигур и

3) приведенный ход уменьшает подвижность белопольного слона.

Ход 3. Фd3 показывает, что в вашу программу необходимо ввести предписание развивать легкие фигуры раньше ферзя, если из других соображений не следует, что ход ферзем предпочтительнее.

3.	. . .	b6

Сомнительный ход. Вероятно, лучше было бы: а) 3 ... c5. Тогда 4. e4, de 5. K : e4, cd 6. K : f6, gf с хорошим продвижением пешек. Неплохо было бы также: б) 3 ... g6 с последующим развитием слона на f5 (используя неудачное расположение белого ферзя).

4.	e4	

Понятный ход!

4.	. . .	Cb7
5.	ed	

По-видимому, лучший ход в этой позиции. Например, 5. e5, Ke4 дает черным хороший контроль над полями d5 и e4 с последующим c5, а 5. f3 ограничивает развитие белых фигур на королевском фланге.

Ход 5. ed показывает, что ваше определение «развития фигур», по-видимому, необходимо дополнить. Следует более высоко оценивать те ходы, которые не увеличивают подвижности фигур противника.

5.	. . .	K : d5
6.	Kf3	

Белые могли усилить контроль над центром (особенно над полем d5) ходом 6. K : d5. Тогда 6 ... Ф : d5 (вариант 6 ... C : d5 не лучше); 7. c4! Фd7 8. Kf3, e5 9. Ce3, предупреждая 9 ... c5. В случае 8 ... c5 9. d5, e6 следует 10. de. В любом случае белые получают хорошую позицию.

6.	. . .	e6

Если теперь 7. K:d5, то лучший ответ со стороны черных будет 7 ... ed. После этого 8. c4 не имеет смысла, так как 8 ... dc открывает слона и изолирует центральную пешку белых.

7.	Ce2	

«Этот ход развивает фигуру и, следовательно, не может быть плохим».

7. . . . Сс7
8. Се3

Неплохо: 8... К:е3 не страшно, так как после 9. fe белые получают лучшую игру за счет появления второй центральной пешки. 8. Се3 имеет также то преимущество, что он сдерживает продвижение пешки с7. Однако более интересную позицию можно было получить, играя 8. К:d5 9. 0—0 10. Сf4 и 11. Ле1, надежно контролируя поле е5. На ход ... с5 белые могут ответить с3.

8. . . . 0—0
9. 0—0 Кd7
10. Лfe1

Этот и последующий ходы ладьями на самом деле не очень хороши. У белых еще нет (и никогда не будет!) конструктивного плана. Они просто занимают своими фигурами центральные вертикали, что вовсе не обязательно является их оптимальным положением. Обычно передвижение ладей на 1-й горизонтали служит для осуществления конкретных планов. Поэтому белым следовало попытаться продолжать: (а) 10. К:d5 и 11. с4, после чего белые ладьи, вероятно, лучше всего поставить на d1 и с1. Можно было также играть: (б) 10. Ке5, К:е5 11. de, после чего основное внимание нужно уделять вертикали d. Кроме того, 10. Ке5 увеличивает подвижность белого слона, сильно стесненного собственным ферзем.

10. . . . с5

Наконец!

11. Лad1 Фс7

Этот ход нельзя назвать хорошим, хотя он и предотвращает 12. Ке5. Например. 12. Кb5, Фb8 (чтобы быть последовательными); 13. с4!, Кb4 14. Фb1 с угрозой 15. а3 и 16. d5 для белых подходит, так что 14 ... cd 15. Кb:d4, вероятно, лучше для черных, хотя все-таки дает белым преимущество. Поэтому черным следовало бы продолжить давление на вертикаль ходом 11 ... Лс8 и не давать белым возможности играть 12. Кb5 и 13. с4. Даже после 11 ... Лс8 белые могут вполне продолжать 12. Ке5.

12. К:d5

Более сильное продолжение осталось незамеченным, но и этот ход не так уж плох; например, 12 ... cd 13. с4, cd 14. с:d4, dc 15. Ф:с4 дает белым по крайней мере равную игру.

12. . . . с:d5?

Теперь у белых есть сильное продолжение: 13. с4, после чего 13 ... cb7 14. d5, cd 15. cd, и белые получают сильную проходную пешку (угрожая 16. d6).

13. а4??

Ужасный ход. Программа решила таким путем спасти пешку а2, хотя ход с4 защитил бы эту пешку и одновременно поддержал бы атаку в центре.

Я удивлен, что ваш «контроль над центром» не предложил 13. с4 вместо 13. а4. Этот ход давал машине очень хорошую игру. Ход 13. а4 указывает на отсутствие правила (или последовательности правил), позволяющего организовать защиту пешек после ухода ладьи.

13. . . . Лас8
14. Фс3

После 14 ... cd 15. Ф:с7, Л:с7 16. К:d4 белые могут укрепить свою позицию ходом 17. с3. Тем не менее 14. Фс3 не способствует улучшению позиции. Ход 14. с4, все еще остается наилучшим.

14. . . . сf6!

Черные хорошо воспользовались «близорукостью» противника. Неплохо было бы и 14 ... Кf6 (контролируя поле е4).

15. Сb5

Разумный ход. Черные угрожали выиграть пешку путем 15 ... cd 16. Ф:с7, Л:с7 17. К:d4, с:d4 18. С:d4 и 18 ... Л:с2. Однако после приведенного выше хода белых черные должны позаботиться о защите коня. Ход 15. Сd3 не спасает пешку: 15 ... С:f3 16. gf, cd 17. С:d4 (17. Ф:с7, Л:с7 и белые не могут вернуть пешку) 17 ... Фb8! 18. Фb4, а5 19. Фb5, Сd4 20. Ф:d7, Лfd8 — сильная атака черных.

15. . . . С:f3

Хорошо: Если сначала 15 ... Лfd8, то 16. С:d7, Л:d7 17. Ке5, cd 18. С:d4, и позиция белых оказывается очень прочной.

16.	gf	Лfd8
17.	С : d7?	

Белые теряют последнюю возможность защитить пешку с2. Это можно было сделать, например, ходом 17. Фd2. Тогда 17 ... cd 18. С:d4, с:d4 19. Ф:d4, Ф:с2 20. С:d7, выигрывая партию (20 ... Лс7 21. Фf4! h6 22. Лd2!).

17.	. . .	Ф : d7
18.	b3	

Этот ход не хуже и не лучше других: в любом случае белые теряют пешку.

18.	. . .	cd
19.	Фd2!	

Очень хорошо! Белые нашли единственный способ (кроме 19. Фd3) избежать потери фигуры: связать пешку черных.

19.	. . .	Фс6!
20.	cf4	Ф : с2
21.	Ф : с2	Л : с2
22.	Лс1	

Положение белых безнадежно, но все же 22. Лd3 относительно лучше, так как мешает продвижению проходной пешки черных.

22. Лс1 указывает на отсутствие в программе правила, согласно которому следует избегать разменов в том случае, когда противник имеет материальное преимущество, если этот размен не вызван какими-либо особыми соображениями.

22.	. . .	Лс8
23.	Лсd1	

Белые продолжают игру в безнадежном положении.

23.	. . .	Лсс3
24.	b4	

«В плохих позициях нет хороших ходов!»

24.	. . .	Л : f3
25.	Cg 3	

Лучший ход в данной ситуации: белые, по крайней мере, защищаются от мата.

25.	. . .	d3
26.	Лс1	Cg5

Ход 26. Лс1 указывает на отсутствие в программе правила, позволяющего ей не попадаться на «вилку».

Лучше было 26 ... d2 с немедленным выигрышем (26 ... d2 27 Л:с2, de (Ф) + 28. Крg2, Фd1! 29. Лс8+, Cd8).

27.	Л : с2	dc
28.	Се5	с8 (Ф)
29.	Л : с8	С : с8

Белые сдались.

Это лучшее, что они могли сделать; но я уверен, что программисты просто устали!

Экспериментальные игры подобного рода дают превосходную информацию о том, какие понятия нужно ввести в программу, помимо «материального баланса», «развития фигур» и «контроля над центром», чтобы по возможности исключить ошибочные ходы.

2. НЕКОТОРЫЕ ИССЛЕДОВАНИЯ ВОЗМОЖНОСТИ ОБУЧЕНИЯ МАШИН НА ПРИМЕРЕ ИГРЫ В ШАШКИ

А. Сэмюэль

Введение

В данной работе рассматривается программирование на ЦВМ такого поведения, которое при осуществлении его людьми или животными расценивалось бы как результат обучения. Хотя здесь не место подробно останавливаться на важности разработки методов обучения машин или рассуждать о философских аспектах проблемы [1], очевидно, что значительная часть работы, ныне выполняемой человеком, будучи весьма тривиальной с точки зрения требований к интеллекту, предполагает тем не менее определенное обучение. В нашем распоряжении уже имеются вычислительные машины, обладающие надлежащей

[1] Некоторые из них имеют весьма глубокий смысл и связаны с вопросами, поднятыми Н. Гудменом в его книге «Факт, вымысел и предсказание» (Goodman N, Fact, Fiction and Forecast, Cambridge, Mass. [illegible] 1954).

способностью к переработке информации и достаточной скоростью вычислений, чтобы они могли воспользоваться разработанными методами обучения, однако наши знания об основных принципах, лежащих в основе этих методов, находятся еще в зачаточном состоянии. Ввиду отсутствия таких знаний приходится разрабатывать методы решения конкретных задач до мельчайших деталей, что связано с большими затратами времени и средств. Создание машин, обучающихся на собственном опыте, должно в конечном счете устранить необходимость во многих из таких усилий по разработке детализованных программ.

Общие принципы подхода к проблеме

С самого начала следует четко разграничить два общих подхода к проблеме обучения машин. Первый из них, который можно назвать подходом с точки зрения *нейронных сетей*, рассматривает возможность создания «обученного» поведения в случайно связанной переключательной сети (или ее модели на цифровой вычислительной машине) в результате применения некоторой системы поощрений и наказаний. Второй, значительно более эффективный подход — создание эквивалента высокоорганизованной сети, предназначенного для обучения только некоторым специальным действиям. Первый подход должен привести к развитию универсальных обучающихся машин. Сравнение размеров переключательных сетей, которые в настоящее время удается практически сконструировать или промоделировать, с размерами нейронных сетей живых организмов свидетельствует о том, что нам предстоит еще пройти большой путь до получения практически пригодных устройств[1]. Второй подход требует составления новой программы для каждого нового приложения, но его можно осуществить уже в настоящее время. Описываемые здесь эксперименты базируются на втором подходе.

Выбор задачи

В течение нескольких лет автор посвящал все свое свободное время проблеме обучения машин, уделяя основное внимание разработке методов обучения применительно к играм[1]. Игра — удобный объект для подобных исследований, ибо лишена многих сложностей, свойственных задачам, взятым из реальной жизни. Шашки, а не шахматы были выбраны из-за простоты их правил, позволяющей сосредоточить внимание на методике обучения. Независимо от сравнительных достоинств этих двух игр как средства для интеллектуального времяпрепровождения можно утверждать, что в шашках содержатся все основные свойства интеллектуальной деятельности, в которой эвристические методы и процесс обучения играют главную роль и в которой этот процесс поддается оценке.

Перечислим некоторые из этих свойств:

1. Эта деятельность не должна быть детерминированной в практическом смысле этого слова. Не существует алгоритма игры в шашки, который мог бы гарантировать выигрыш или ничью, а полный перебор всех возможных вариантов в шашечной партии включал бы около 10^{40} альтернатив при выборе ходов, так что их рассмотрение при скорости 3 альтернативы в 1 *нсек* заняло бы 10^{21} столетий.

2. Должна существовать определенная цель — выигрыш партии, и должен существовать по меньшей мере один критерий, или промежуточная цель, который имеет отношение к достижению конечной цели и выражение для которого должно быть известно. Цель в шашках — лишить противника возможности сделать ход, а доминирующий критерий — число шашек каждого цвета на доске. Важность наличия заранее известного критерия будет обсуждаться ниже.

3. Правила действий должны быть хорошо определены и заранее известны. Игры удовлетворяют этому требованию, чего, к сожалению, нельзя сказать о многих экономически важных проблемах. Хотя в принципе установление правил может быть частью процесса обучения, это — усложнение, которое можно пока не рассматривать.

4. Должен иметься некоторый запас знаний о деятельности, на которой проверяется успешность обучения.

5. Действия машины должны быть в основном похожими на действия людей, так чтобы поведение программы можно было сделать понятным человеку. Возможность заставить программу играть против человека (или другой программы) придает

[1] У Маккалок [678] сравнивает цифровую машину с нервной системой плоского червя. Распространять это сравнение на обсуждаемый случай было бы несправедливо по отношению к червю, ибо его нервная система чрезвычайно высоко организована по сравнению со случайными сетями, изучавшимися Фэрли и Кларком [313], Рочестером и др. [924] и Розенблаттом [930].

[1] Первая работающая шашечная программа для машины IBM-701 была написана в 1952 г. В 1954 г. она была переделана для IBM-704. Первая обучающаяся программа была закончена в 1955 г. и демонстрировалась по телевидению 24 февраля 1956 г.

исследованию еще больший интерес и попутно представляет убедительную демонстрацию для тех, кто не верит, что машина может обучаться.

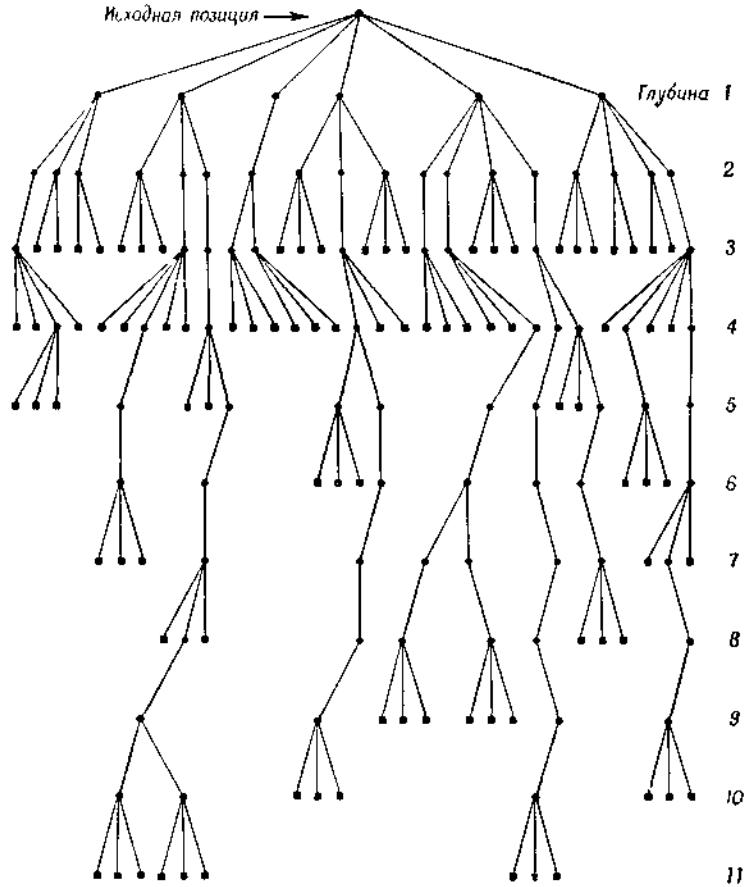
Прежде чем исследовать процесс обучения шашечной машины, мы, естественно, должны научить ее играть с соблюдением правил, т. е выразить правила игры на машинном языке и организовать ввод ходов противника и вывод ходов машины вместе со всеми данными, которые желательно иметь экспериментатору. Общие принципы такой программы были описаны Шенноном в 1950 г. применительно к шахматам. Основная программа, используемая в настоящих экспериментах, очень похожа на программу, описанную Стрейчи в 1952 г. Возможность использования машины более высокого класса (IBM-704) в сочетании с рядом частных усовершенствований методики программирования привела к тому, что игра становится довольно интересной даже без всякого обучения. Ниже будут описаны основные варианты этой программы.

Основная программа для игры в шашки

Машина играет, просматривая варианты на несколько ходов вперед и оценивая получающиеся в результате позиции почти так же, как это делает человек. Позиция на доске представляется набором машинных слов Обычно для представления любой позиции используются четыре слова 32 двоичных разряда (из 36-разрядного машинного слова IBM-704) поставлены в соответствие 32 игровым полям шашечной доски, и наличие шашек на этих полях представляется появлением единиц в разрядах соответствующего слова Просмотр вариантов начинается с определения всех возможных ходов из данной позиции Эти ходы по очереди исследуются, что приводит к образованию новых позиций. Далее процесс повторяется для каждой новой позиции (старые позиции сохраняются в памяти машины, чтобы облегчить возвращение к исходной точке), и весь процесс может быть повторен. Процедура просмотра проводится на несколько ходов вперед, как показано на фиг. 1. Получаемым в результате позициям машина присваивает числа, выражающие их относительную ценность.

Стандартный способ оценки получившейся позиции состоит в подсчете значения некоторого линейного полинома, каждый член которого характеризует какую-нибудь особенность расположения шашек на доске Было испытано большое число абстрактных схем построения такого полинома, не связанных с обычными шашечными понятиями, но ни одна из них не имела

успеха[1]. Один из возможных способов организации оценочного полинома состоит в том, что его члены с численно малыми коэффициентами рассматриваются как некоторый промежуточный



Фиг 1 Пример «дерева» ходов, исследуемого в процессе просмотра вариантов

В действительности ветвлений намного больше, чем показано на рисунке, и «дерево» может иметь до 20 уровней.

критерий по отношению к критерию, который вычисляется с помощью больших членов. Достижение промежуточных целей (рост

[1] В одной из наиболее интересных попыток позиция характеризовалась первыми и более высокими моментами черных и белых шашек относительно двух взаимно перпендикулярных осей шашечной доски Были испытаны два типа осей (параллельных краям доски и диагональных).

членов с малыми коэффициентами) показывает, что машина идет в правильном направлении, так что большие члены будут в конечном счете возрастать. Если бы программа могла смотреть вперед достаточно далеко, мы должны были бы только проверять, продолжается ли еще игра. Так как в обычной ситуации видеть настолько далеко вперед невозможно, приходится проверять что-нибудь другое, скажем число шашек. Таким образом, машина должна прослеживать каждый вариант до тех пор, пока одна из сторон не получит преимущества в числе шашек. Но даже это не всегда возможно, так что необходимо иметь тест, определяющий, не достигла ли машина позиционного преимущества и т. п. Численные выражения для различных свойств позиции затем суммируются (каждое с соответствующим весом), образуя оценочный полином.

Доминирующим оценочным параметром, согласно правилам шашечной игры, является невозможность для одной или другой стороны сделать ход[1]. Так как такая ситуация может возникнуть только один раз за всю партию, этот параметр проверяется отдельно и не включается в оценочный полином.

Следующий учитываемый параметр — материальное преимущество. Предполагается, что машине всегда выгодно уменьшать число шашек противника по сравнению со своим собственным. Изменение знака этого члена на обратный приведет к тому, что программа будет играть «в поддавки» и при наличии обучения сможет лишь учиться все лучше и лучше играть в эту игру. Если бы знак этого члена не был известен программисту, он мог бы, конечно, определить его путем испытаний, но так или иначе он должен быть фиксирован экспериментатором и оставаться постоянным во время игры; по существу это одна из инструкций, определяющих задачу машины. Выражение для подсчета материального преимущества должно быть таким, чтобы получалось хорошо известное свойство: обычно выгодно разменивать шашки, когда выигрываешь, и воздерживаться от размена, когда проигрываешь. Далее, дамки считаются более ценными, чем простые шашки. Их веса соотносятся как $3:2$[2,3]. Этот коэффициент означает, что программа будет менять 3 простые

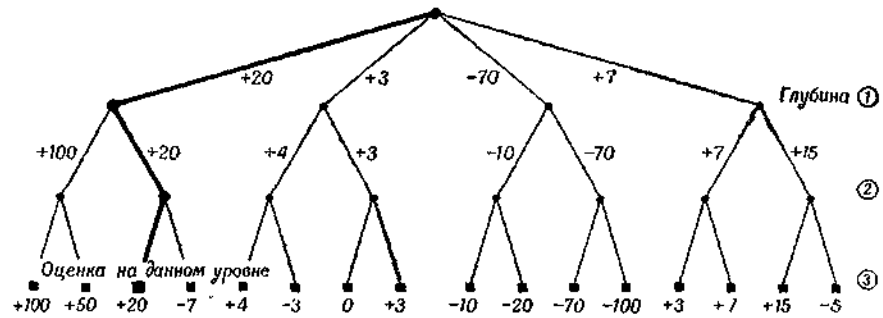
[1] Но не взятие всех шашек противника, как широко принято считать, хотя все игры кончаются обычно таким путем.

[2] Применение большего весового отношения хотя и лучше соответствует представлениям многих игроков, но, как обнаружил Стрейчи [1953], может привести к определенным логическим трудностям.

[3] В американских шашках в отличие от русских и дамки и простые шашки могут ходить только на соседнее поле. Разница между дамками и простыми шашками состоит в том, что простые шашки ходят и «бьют» только вперед, а дамки — в обоих направлениях. — *Прим. перев.*

шашки на 2 дамки или 2 дамки на 3 простые, если при этом она сможет получить некоторое позиционное преимущество.

Следующий важный вопрос — какие еще параметры должны быть выбраны в качестве членов оценочного полинома и с какими весами эти члены должны входить в полином. Есть два пути для решения этого вопроса: экспериментатор может либо сам задать эти параметры, либо предоставить машине произвести отбор. Мы обсудим в деталях первый способ в связи с исследованием процесса обучения, использующего запоминание позиций. Обсуждение различных программных методов



Фиг. 2 Упрощенная диаграмма оценки вариантов

Диаграмма показывает, что происходит обратное движение от оцененных позиций по «дереву» возможных ходов, приводящее к выбору наилучшего очередного хода. Процесс начинается с (3).
(1) машина выбирает ветвь с наибольшей оценкой
(2) предполагается, что противник выберет ветвь с наименьшей оценкой.
(3) машина выбирает ветвь с наиболее положительной оценкой.

выбора параметров и установления их весов будет проведено в следующих разделах.

Было бы неправильно в качестве очередного хода выбирать тот, который ведет к позиции с самой высокой оценкой, так как для достижения этой позиции потребовалось бы сотрудничество противника. Анализ должен проводиться с помощью *обратного* движения от позиций, которым уже присвоена некоторая оценка, по «дереву» возможных ходов. При этом каждый раз учитывается цель той стороны, чей ход рассматривается. Предполагается, что противник всегда будет пытаться минимизировать машинную оценку, в то время как машина действует так, чтобы ее максимизировать. В таком случае каждой точке ветвления, соответствующей некоторой позиции, присваивается оценка той позиции, которая получилась бы из нее в результате наилучшего на этом шаге хода. Проводя эту «минимаксную» процедуру в обратном направлении до начальной точки, определяем «наилучший ход».

Кроме того, для целей обучения начальной позиции присваивается оценка той позиции, которая находится в конце наиболее вероятной цепочки ходов. Этот процесс показан на фиг. 2. Найденный наилучший ход выполняется, сообщается на пульт управления и выводится на печать.

После этого противнику разрешается сделать свой ход, который может быть сообщен машине либо с пульта, либо с помощью перфокарт. Машина проверяет правильность хода противника, отклоняя[1] или принимая его, и процесс повторяется. Когда программа при просмотре вариантов может предсказать победу, этот факт выводится на печать. Подобным же образом программа признает свое поражение.

Ограничения, накладываемые на глубину просмотра

Рассмотрение времени, затрачиваемого на игру, приводит к необходимости ограничить длину просматриваемых вариантов некоторой довольно малой величиной. Эта длина называется «глубиной просмотра» (глубина 2 соответствует одному предполагаемому ходу машины и ожидаемому ответу противника). Глубина просмотра не фиксирована, а зависит от текущей ситуации и меняется от хода к ходу и от ветви к ветви в процессе анализа возможных ходов. В разное время было испытано множество различных способов регулирования глубины просмотра, в том числе и довольно сложные. Наиболее эффективный, хотя и очень детализированный способ в принципе прост и заключается в следующем. Программа всегда просматривает вариант на некоторое минимальное число ходов вперед. Это число в начале игры и при отсутствии обучения обычно равно 3. После просмотра на эту минимальную глубину программа будет оценивать позицию, если не имеет места ни одна из следующих ситуаций: (1) следующий ход — взятие; (2) на последнем ходе было взятие или (3) возможно предложение размена. Если имеет место одно из этих условий, программа продолжает просмотр вариантов. На глубине 4 программа остановится и оценит получившуюся позицию, если при этом не встретилась одна из ситуаций типа (1) или (3). На глубине, большей или равной 5, программа прекращает рассмотрение варианта, если на следующем ходе не должно быть взятия. На глубине 11 или большей программа прекратит просмотр, даже если на

[1] Единственное ограничение, накладываемое на процесс игры, состоит в том, что программа требует от противника делать разрешенные ходы; в частности, противник должен взять шашку, если предлагается взятие. Взятие шашки «за фук» не допускается.

следующем ходе должно быть взятие, при условии, что одна из сторон в этот момент будет иметь преимущество более чем на 2 дамки (чтобы предотвратить бесполезные исследования заведомо проигрышных или заведомо выигрышных вариантов). Программа останавливается на глубине 20 при любых условиях (так как исчерпывается объем памяти, отведенный для просмотра вариантов) и делает корректировку оценки, чтобы учесть предстоящее взятие. Наконец, специально устанавливаются глубины анализа вариантов, когда время экономится благодаря запоминанию позиций (см. ниже) и когда общее число шашек на доске становится меньше некоторого заданного числа. Все такие глубины задаются набором констант и могут быть в любой момент изменены с пульта.

Благодаря связыванию глубины просмотра с текущей ситуацией достигаются три желаемых результата. Во-первых, оценку можно производить в относительно устойчивых ситуациях — в так называемых «мертвых» позициях (по определению Тьюринга). Во-вторых, те варианты, в которых представляются большие возможности для увеличения или потери преимущества, исследуются более тщательно. Наконец, так как ветвление обычно сильно сокращается благодаря наличию ситуаций, в которых обязательно взятие, общее число рассматриваемых позиций и ходов остается в разумных пределах и более справедливо распределяется между различными возможными начальными ходами.

На практике машине обычно дается около 30 *сек* на ход. Для того чтобы увеличить, насколько это возможно, скорость работы машины, были тщательно продуманы методы просмотра таблиц, быстрой сортировки и поиска и разработано множество чисто программных приемов для полного использования всех возможностей машины IBM-704. Разумеется, любое желаемое время игры можно получить, ограничивая глубину просмотра. При слишком малой глубине получается плохая игра, а при слишком большой — чрезмерный расход машинного времени.

Варианты основной программы

Для изучения работы программы она была написана в нескольких вариантах. Один из них позволяет программе играть против самой себя, т. е. играть за обоих партнеров. Было обнаружено, что эта форма игры особенно полезна на ранней стадии обучения.

Программа может также разыгрывать партии из учебника по шашкам, записанные на перфокартах или на магнитной ленте. При работе в этом варианте программа в каждой

позиции обычным образом определяет ход, который она должна сделать, и сообщает его. Затем вместо того, чтобы сделать этот ход, программа делает ход, указанный в учебнике. Программа регистрирует свою оценку этих двух ходов и, кроме того, подсчитывает и сообщает число возможных ходов, оцененных ею выше хода, приведенного в учебнике, и число ходов, оцененных ниже. Затем машина делает ход за второго противника, и процесс повторяется. В конце партии подсчитывается коэффициент корреляции[1].

Следует отметить, что особое значение во всех этих исследованиях придается методике обучения. Необходимо постоянно сопротивляться соблазну улучшать игру машины, вводя в нее стандартные дебюты или другие «человеческие» сведения относительно техники игры. Даже при разыгрывании партий из учебника не придается никакого веса тому факту, что указанные в учебнике ходы являются, вероятно, наилучшими при данных обстоятельствах.

Для демонстрационных целей, а также для того, чтобы избежать потери машинного времени (когда противник обдумывает свой ход), иногда удобно играть несколько партий одновременно против разных противников. Для программы в ее настоящем виде наиболее подходящее для этих целей число противников 6, хотя в ряде случаев их может быть и 8.

Игра может быть начата из любого начального положения на доске, так что программу можно испытывать на эндшпилях, шашечных этюдах и т. п. В случае нестандартных начальных условий программа запоминает первоначальное расположение шашек. Время от времени и в конце каждой партии программа вычисляет различные статистические характеристики, которые помогают оценить качество игры.

Чтобы сделать программу удобной для управления, в нее было добавлено большое число других инструкций (см. приложение А), но они не имеют прямого отношения к проблеме обучения, к которой мы сейчас переходим.

Обучение с помощью запоминания позиций и его варианты

По-видимому, наиболее элементарный из достойных рассмотрения способов обучения — это вариант обучения, при котором программа просто запоминает все позиции, встретившиеся в течение игры, вместе с вычисленными для них оценками. Тогда вместо вычисления оценки можно будет найти ее в памяти, что дает экономию машинного времени. Вряд ли этот метод обучения можно считать особенно перспективным. Тем не менее, если программа использует сэкономленное время, чтобы просматривать варианты на бо́льшую глубину, она будет улучшать со временем свою игру.

[1] Этот коэффициент определяется по формуле $C=(L-H)/(L+H)$, где L — число всех возможных ходов, которые машина оценила ниже ходов, указанных в учебнике, а H — число ходов, оцененных выше ходов учебника

К счастью, способность запоминать положение на доске (информацию «глубиной 0») и видеть позиции, находящиеся на большей глубине, позволяет значительно увеличить глубину исследуемых вариантов. Чтобы пояснить это, рассмотрим самый простой случай, когда все варианты просматриваются на одну и ту же глубину, например 3. Пусть программа запоминает позиции, встретившиеся в течение игры, вместе с их минимаксными оценками. Если теперь в процессе анализа вариантов требуется оценить позицию, находящуюся на глубине 3 от оцениваемой позиции, машина ищет ее в списке позиций, встречавшихся в ранее сыгранных партиях. Если позиция найдена, ее оценка фактически уже определена просмотром на глубину 3, и если эта оценка повлияет на определение хода, который должен быть сделан, то исходная позиция будет иметь оценку глубиной 6[1], а не 3. Эта исходная позиция в свою очередь запоминается вместе со своей оценкой и может встретиться в последующих партиях. При этом глубина оценки снова увеличится и т. д. Эта процедура показана на фиг. 3. В сочетании с простым запоминанием она дает довольно сильный метод обучения.

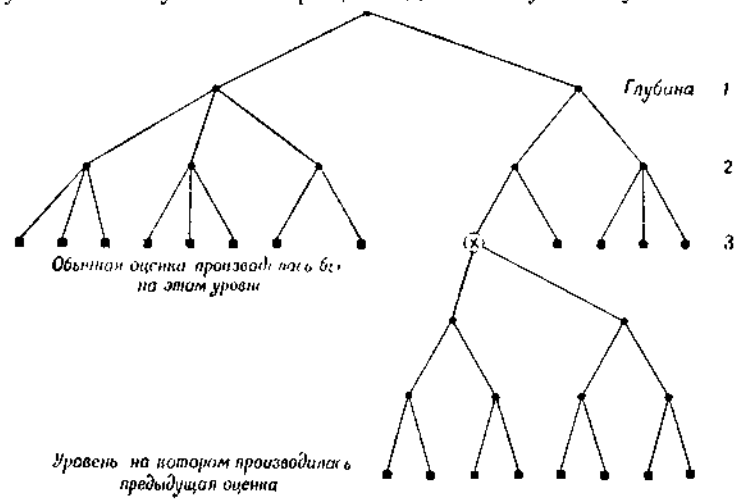
Для того чтобы обучение, происходящее по описанной выше схеме, приводило к желаемому результату, программа должна обладать некоторыми дополнительными особенностями. Во-первых, чтобы заставить программу форсировать выигрыш, в нее необходимо ввести понятие «направления». Для иллюстрации рассмотрим случай, когда имеются две дамки против одной, т. е. выигрыш обеспечен практически для всех позиций[2]. Можно предположить, что со временем программа накопит все эти позиции, причем каждая будет иметь оценку «выигрыш». Столкнувшись с такой ситуацией, программа будет просматривать все возможные варианты, и каждый из них будет приводить к выигрышу, хотя возможно, что только один из всех начальных ходов прямо ведет к цели, в то время как все остальные приводят лишь к бесполезной трате времени. Как выбрать ход в этом случае?

Хорошие результаты дает следующий способ. В памяти машины для каждой позиции хранится глубина ее оценки, и на

[1] То есть позиция, оценка которой после просмотра вариантов присваивается исходной позиции, находится на глубине 6 — *Прим. перев.*

[2] См. примечание 3 на стр. 76 — *Прим. ред.*

основании ее производится выбор между позициями с одинаковыми оценками Можно построить программу таким образом, чтобы, выигрывая, она выбирала кратчайший путь к победе, а проигрывая, применяла тактику задержки. Последний из применявшихся нами для этой цели приемов заключается в том, что абсолютная величина оценки уменьшается на некоторую малую величину всякий раз, когда ее глубина увеличивается



Фиг. 3 Упрощенное изображение процесса обучения с помощью запоминания позиций.

Информация, накопленная в результате предшествующей игры, используется для увеличения эффективной глубины минимаксной оценки.

⊗ типичная позиция, найденная в памяти машины, с оценкой, полученной от предыдущего просмотра вариантов

в процессе анализа вариантов. Если теперь программе встретятся позиции, оценки которых отличаются только глубиной, она будет автоматически делать наилучший выбор, выбирая вариант с меньшей длиной в выигрышной ситуации и более длинный в проигрышной. Понятие направления очень важно, даже при отсутствии обучения. Несколько более ранних попыток добиться обучения окончились неудачей из-за того, что это понятие не было соответствующим образом введено в машину.

Составление каталога и отбор запоминаемой информации

Так как практически число позиций, которое может храниться в памяти машины, ограничено, а время для нахождения среди них нужной позиции может оказаться чрезмерно большим,

необходимо иметь способы: 1) составления каталога позиций, находящихся в памяти, 2) стирания избыточной информации и 3) выбрасывания позиций, которые по каким-либо соображениям кажутся бесполезными. Наиболее эффективный из найденных к настоящему времени методов составления каталога заключается в следующем. В начале производится стандартизация всех позиций так, чтобы очередной ход должны были делать черные[1]. Если фактически в некоторой позиции очередной ход белых, то цвет шашек и позиция преобразуются в симметричные, а право хода передается черным. Таким образом, приблизительно в два раза уменьшается число позиций, которые надо помнить. Для тех позиций, в которых на доске находятся только «дамки», это число уменьшается в четыре раза путем отражения относительно диагоналей. При этом используется более компактное представление шашечной доски по сравнению с тем, которое применялось в предшествующей стадии игры.

После того как позиции стандартизованы, они объединяются в списки по следующим признакам: 1) число шашек на доске; 2) наличие или отсутствие материального преимущества; 3) какая из сторон владеет этим преимуществом; 4) наличие или отсутствие «дамок» на доске; 5) для какой из сторон параметр «MOVE» равен единице[2] и, наконец, 6) первые моменты относительно нормальных и диагональных осей шашечной доски. Вновь встретившиеся в ходе игры позиции запоминаются, пока их не накопится достаточное число, тогда они объединяются с основным каталогом, находящимся на магнитной ленте. Внутри каждого списка позиции записываются последовательно, рассортированными по определяющим их словам.

Списки располагаются на магнитной ленте в таком порядке, в котором они наиболее вероятно понадобятся в ходе игры, позиции, в которых каждая сторона имеет по 12 шашек, идут первыми и т. д. Описанный метод составления каталога очень важен, так как он сокращает до минимума время поиска на магнитной ленте.

При обращении к позиции, находящейся в памяти машины, соответствующий список переписывается в оперативную память, и нужная позиция ищется с помощью дихотомии.

В оперативной памяти, как правило, хранится одновременно пять или более списков в зависимости от их длины. Делая новый ход, программа обычно вызывает в оперативную память три или четыре новых списка. При этом стираются те списки, которые находятся в памяти дольше остальных.

[1] В американских шашках первый ход делают черные — *Прим. ред.*
[2] См стр 108 — *Прим ред.*

Для ограничения числа запоминаемых позиций наиболее полезными оказались два приема: один из них учитывает частоту использования позиций, второй — глубину оценки. Чтобы можно было следить за частотой использования, каждой вновь запоминаемой позиции присваивается некоторое число, называемое «возрастом». Всякий раз, когда производится обращение к находящейся в памяти позиции для исправления ее оценки или для использования ее в процедуре просмотра вариантов, «возраст» этой позиции уменьшается вдвое. Эта процедура называется *подкреплением*. В противовес этому «возраст» каждой позиции автоматически увеличивается на единицу всякий раз, когда позиции, накопленные в оперативной памяти, объединяются с позициями, записанными на магнитной ленте (это происходит приблизительно через каждые 20 ходов). Как только «возраст» какой-либо позиции достигает некоторой установленной величины, эта позиция вычеркивается из памяти. Описанная процедура представляет собой модель *забывания*. Те из вновь встретившихся позиций, которые в дальнейшем остаются неиспользованными, вскоре забываются. В то же время позиции, которые используются несколько раз подряд, получают настолько сильное подкрепление, что остаются в памяти, даже если к ним долгое время после этого не обращаются. Эта модель подкрепления и забывания была выбрана в результате размышлений о хрупкости человеческой памяти. Практика доказала эффективность такой модели.

Помимо ограничения общего числа запоминаемых позиций, желательно ограничить максимальный размер каждого списка. Как только размер списка превысит установленный максимум, из него автоматически выбрасываются позиции с меньшей глубиной оценки, так чтобы этот размер снова стал значительно меньше максимального.

Прежде чем исследовать способность описанной выше системы к обучению, необходимо задать оценочный полином. С этой целью было испытано много различных наборов параметров путем разыгрывания серии партий из учебников по шашкам и вычисления коэффициента корреляции. Значение этого коэффициента менялось от 0,2 для наихудшего из испытанных полиномов до примерно 0,6 для окончательно выбранного. Отобранный полином состоял из четырех членов (в отличие от последующих экспериментов, в которых использовалось 16 членов). Приводим эти члены в порядке убывания их важности:

1) материальное преимущество, 2) блокировка, 3) подвижность и 4) комбинированный член, учитывающий контроль над центром и продвижение шашек.

Экспериментальная проверка

После того как был выбран оценочный полином, была проведена серия игр, причем машина играла как сама с собой, так и против разных противников (некоторые из них были профессионалами). Было также разыграно много партий из учебников, в том числе и эндшпилей. Программа научилась очень хорошо разыгрывать дебют и распознавать большинство выигрышных и проигрышных конечных ситуаций за много ходов вперед, хотя ее игра в середине партии улучшилась незначительно. Программа играет сейчас значительно лучше среднего новичка, однако уровень ее игры еще нельзя назвать профессиональным.

В настоящее время память машины хранит немногим более 53 000 позиции (в среднем по 3,8 слова каждая), которые были отобраны из значительно большего числа позиций с помощью описанного выше метода отбора. Это количество еще далеко от того предела, который определяется используемыми в программе методами поиска и составления таблиц. Однако грубая оценка, основанная на частоте использования находящихся в памяти позиций в течение обычной партии (эти данные табулируются автоматически), показывает, что для того, чтобы существенно улучшить игру в середине партии, потребовалось бы увеличить имеющуюся «библиотеку» позиций по меньшей мере в 20 раз. При существующей скорости накопления новых позиций для этого потребовалось бы чрезмерно большое количество партий и, следовательно, машинного времени[1].

На основании проведенных экспериментов можно сделать следующие выводы:

1. Эффективная методика обучения с помощью запоминания позиций должна предусматривать введение в программу понятия направления и содержать отработанную систему составления каталога и запоминания информации.

2. Обучение с помощью запоминания позиций может эффективно осуществляться на машинах класса IBM-704 при условии, что информация, к которой приходится обращаться, охватывает не больше чем, грубо говоря, 10^6 машинных слов и если частота обращения к этой информации не превышает приблизительно 100 раз в минуту. Эти цифры, конечно, сильно зависят от эффективности системы составления каталога.

[1] Требуемое машинное время велико с точки зрения стоимости, но оно, вероятно, меньше того времени, которое нужно мастеру по шашкам для приобретения своей квалификации.

3. Если машина играет по простой оценочной схеме, используя лишь обучение с помощью запоминания позиций, то для достижения уровня мастера требуется помнить больше приведенного выше числа слов. Следовательно, уровень мастера не может быть достигнут на IBM-704 с помощью одних только указанных средств.

4. Игра типа шашек является подходящим объектом для разработки методик обучения и дает весьма убедительную для скептиков демонстрацию процесса обучения машины.

Метод обучения, связанный с обобщением прошлого опыта

Очевидный способ уменьшения размеров памяти, необходимой для использования прошлого опыта, заключается в обобщении этого опыта и запоминании его только в обобщенном виде. Чтобы процесс обучения был действительно эффективным, он должен идти непрерывно и представлять собой абстрагирование на нескольких уровнях. Первым шагом, сделанным в этом направлении, можно считать разработку программы, отбирающей из набора различных параметров некоторую часть для использования в качестве членов оценочного полинома, а также программы, определяющей знак и абсолютную величину весов, с которыми эти параметры входят в полином. В настоящее время оценочный полином состоит из 16 членов, отобранных из списка, содержащего 38 параметров. Член, характеризующий материальное преимущество и определяющий цель машины в игре, вычисляется отдельно и не может быть исключен программой из рассмотрения.

После целого ряда довольно неудачных попыток программа, играющая за обе стороны, была видоизменена таким образом, чтобы действовать как два различных игрока, называемых для удобства *Альфа* и *Бета*. Игрок Альфа после каждого хода обобщает свой опыт, подстраивая веса своего оценочного полинома и заменяя члены, ставшие ненужными, новыми членами, взятыми из резервного списка. Игрок Бета, напротив, использует один и тот же оценочный полином на протяжении всей партии. Программа Альфа используется для игры с человеком, а игра машины «сама с собой» представляет собой игру программы Альфа с программой Бета.

После каждой такой партии «нейтральная» часть программы сравнивает игровые способности Альфы и Беты. Если Альфа выигрывает (или имеет преимущество к моменту прекращения игры), то текущий оценочный полином, используемый Альфой, передается Бете. Если же выигрывает или имеет преимущество Бета, Альфа получает отметку «плохо». Всякий раз, когда Альфа получает определенное число «плохих» отметок (обычно порядка трех), предполагается, что она находится на неправильном пути, и производится существенное изменение ее оценочного полинома (наибольший вес приравнивается нулю). Основанием для таких действий служит то обстоятельство, что процесс обучения в целом представляет собой попытку найти абсолютный максимум некоторой функции (качества игры) в многомерном оценочном пространстве в присутствии многочисленных локальных максимумов, на которых программа может застревать. Посредством вмешательства извне можно вернуться к некоторым предыдущим условиям или произвести другие изменения, если становится очевидным, что процесс обучения не идет в нужном направлении. Однако, как правило, программа сама стремится избегать ловушек и улучшаться более или менее непрерывно.

Выяснить уровень, на котором находится программа, можно в любое время. Для этого достаточно, приостановив на время процесс обучения, заставить Альфу разыграть одну или несколько партий из учебника по шашкам, после чего сравнить ее игру с рекомендациями мастеров. Еще интереснее выставить ее против живого партнера.

Алгоритм подстраивания весов оценочного полинома

Для того чтобы Альфа могла целенаправленно изменять свой оценочный полином, необходимо дать какие-то заслуживающие доверия критерии поведения машины. Трудность состоит в том, что единственным критерием, определяющим ценность позиции, является именно этот полином. Выход из этого порочного круга находится благодаря специфическому свойству процедуры просмотра вариантов, заключающемуся в том, что от оценочного полинома не требуется, чтобы он давал особенно хорошие результаты при оценке позиций, которые по ходу игры находятся на значительном расстоянии от исходной позиции. Поэтому в качестве критерия относительного изменения позиций обоих игроков после достаточно большого числа ходов может служить более грубая оценочная функция, которой нельзя пользоваться от хода к ходу.

Пожалуй, более целесообразно использовать это свойство следующим образом. Попытаемся построить такой оценочный полином, чтобы оценка, вычисленная для текущей позиции, совпадала с оценкой, вычисленной для позиции, находящейся

в конце наиболее вероятной последовательности ходов. Очевидно, что идеальная система такого типа эквивалентна программе с неограниченной глубиной просмотра вариантов. Чем ближе мы сумеем подойти к этому идеалу, тем лучше будет играть машина[1].

Для того, чтобы можно было сравнивать значения оценочного полинома в начале и в конце достаточно длинной цепочки ходов, Альфа хранит в памяти оценки позиций, встретившихся в течение партий, вместе с весами оценочного полинома. В процессе игры Альфа сравнивает оценку, полученную на несколько ходов раньше, с минимаксной оценкой для текущей позиции. Разность между этими оценками δ используется для коррекции оценочного полинома. Если $\delta>0$, разумно предположить, что прежняя оценка была заниженной. Поэтому веса положительных членов следует увеличить, а веса отрицательных — уменьшить. Если $\delta<0$, производится противоположная операция. В этом случае можно предположить, что либо прежняя оценка была неправильной, либо плохо были выбраны последующие ходы. Следовательно, нужно увеличить веса отрицательных членов и уменьшить веса положительных. Новые веса устанавливаются в результате следующей процедуры.

В машине хранятся коэффициенты корреляции между знаками отдельных членов текущего оценочного полинома и знаком δ. После каждого хода эти коэффициенты пересчитываются с учетом числа ходов, во время которых соответствующий член входил в оценочный полином.

Вес того члена оценочного полинома (исключая член «материальное преимущество»), который имеет в настоящий момент наибольший коэффициент корреляции, полагается равным некоторому большому числу, представляющему собой целую степень числа 2. Все остальные веса также являются целыми степенями числа 2, причем отношение показателей степени любых двух весов равно отношению абсолютных величин соответствующих коэффициентов корреляции. Точнее, если отношение двух коэффициентов корреляции больше или равно n, но меньше чем $n+1$, где n — целое, то отношение соответствующих весов оценочного полинома равно 2^n. Цель этой процедуры — увеличить диапазон значений весов. Как только коэффициент корреляции меняет знак, меняется и знак соответствующего члена оценочного полинома.

[1] В этом рассуждении кроется логическая ошибка. Программа могла бы сохранять только инвариантные члены, не имеющие никакого отношения к качеству игры. Например, она могла бы подсчитывать число полей на доске. Введение члена «материальное преимущество» предотвращает такую возможность.

Неустойчивость

Следует отметить, что цепочка ходов, относительно которой вычисляется δ, состоит из двух частей. Первая часть — это уже проделанные машиной ходы, хранящиеся в памяти; вторая — ходы, предсказанные машиной. В первой части текущий оценочный полином Альфа определяет только ее собственные ходы, тогда как в предсказываемой части с помощью этого полинома определяются ходы обеих сторон. Существует соблазн увеличить чувствительность δ как индикатора изменения, увеличивая длину первой части. Однако такой путь оказывается опасным, так как веса членов, входящих в оценочный полином, могут за время от момента вычисления первоначальной оценки до момента определения δ сильно измениться. Фактически этого не удается избежать, даже если цепочка состоит из одной пары ходов. Поэтому после того, как определен ход и внесена коррекция в оценочный полином, необходимо снова вычислить значение оценочного полинома для данной начальной позиции и для дальнейшего сравнения хранить в памяти именно эту оценку, а не ту, которая использовалась для определения хода. Это соображение может показаться тривиальным, однако пренебрежение им в начальной стадии описываемых экспериментов привело к «раскачиванию», аналогичному неустойчивости электронных схем с большими задержками в цепи обратной связи.

Чтобы система была устойчивой по отношению к малым изменениям δ, устанавливается некоторый порог. Если δ по абсолютной величине не превосходит этого порога, в оценочном полиноме не производится никаких изменений. Этот порог перестраивается в процессе игры, и его величина определяется как среднее значение всех весов текущего оценочного полинома. Тот же порог используется при вычислении оценки позиции, из которой производится очередной ход. Если величина оценки меньше пороговой, она полагается равной 0.

Еще один тип неустойчивости связан с введением нового члена в оценочный полином. Очевидно, что после первого хода коэффициент корреляции этого члена примет максимальное по модулю значение, равное 1, хотя в дальнейшем он может оказаться близким к 0. Чтобы предотвратить связанные с этим сильные флуктуации, коэффициент корреляции для вновь вводимого члена вычисляется так, как если бы этот член уже несколько раз использовался и имел коэффициент корреляции, равный 0. Для этого вместо числа, указывающего, сколько раз использовался данный член, в формулу для коэффициента корреляции подставляется некоторое постоянное число (обычно 16).

После того как данный член некоторое время использовался, желательно, наоборот, сделать так, чтобы новый опыт мог изменить результаты, полученные благодаря прежнему опыту. Этого можно добиться, увеличивая знаменатель коэффициента корреляции не постепенно, а скачками и ограничивая величину этого знаменателя. Именно, на N-м ходе коэффициент корреляции C_N вычисляется по формуле

$$C_N = C_{N-1} - \frac{C_{N-1} \pm 1}{N^*},$$

где

$$N^* = 16 \quad \text{при} \quad N < 32, \quad N^* = 32 \quad \text{при} \quad 32 \leqslant N < 64,$$

$$N^* = 256 \quad \text{при} \quad N \geqslant 256.$$

После того как для δ установлена некоторая минимальная величина, представляется разумным приписывать больший вес ситуациям, приводящим к большой величине δ. В соответствии с этим вводятся два дополнительных правила: если первый член вносит свой вклад в δ, т. е. одна или обе стороны несут материальные потери, указанные выше изменения в коэффициентах корреляции удваиваются, а если величина δ настолько велика, что результатом игры почти наверняка окажется выигрыш или проигрыш, эти изменения увеличиваются в четыре раза

Замена членов

Выше упоминалось о процедуре замены членов оценочного полинома. В настоящее время программа содержит 38 различных членов (кроме члена «материальное преимущество»). Оценочный полином состоит из 16 членов, остальные 22 находятся в резерве. После каждого хода отмечается член с наименьшим коэффициентом корреляции, и для каждого члена машина накапливает число таких отметок Q. Как только Q для некоторого члена достигает определенной величины (обычно 8), этот член помещается в конец резервного списка, а на его место ставится член, находящийся в начале этого списка, но соответствующий ему коэффициент корреляции, число предыдущих использований и вес принимаются равными нулю.

Таким образом, замена членов оценочного полинома происходит в среднем через каждые 8 ходов, а заменяемый член снова входит в полином через 176 ходов.

Наблюдая за ходом этой процедуры, программа сообщает, сколько раз использовался каждый заменяемый член. Члены, которые все время выбрасываются из полинома через минимальное время после включения, можно совсем удалить из списка и заменить совершенно новыми.

Можно было бы выдвинуть возражение, что процедура отбора членов оценочного полинома из заданного списка слишком проста и что программа сама должна генерировать эти члены. К сожалению, удовлетворительного алгоритма для этой цели пока не найдено. Поскольку список задается человеком, можно было бы по крайней мере потребовать, чтобы эти члены образовывали ортогональную систему (предполагая, что это понятие имеет некоторый смысл по отношению к оценке шашечной позиции). По-видимому, наших знаний о шашках сейчас еще недостаточно, чтобы найти такую систему. Единственное практически осуществимое решение состоит в том, чтобы взять больше членов в надежде, что все существенные параметры так или иначе будут охвачены, хотя бы сложным и не самым экономным образом. Однако это не так уж плохо, ибо при попытке применить подобную методику к реальной задаче мы столкнемся, вероятно, с такой же ситуацией.

В существующем списке многие члены неявным образом связаны с понятиями, которыми пользуются хорошие игроки. Некоторые из этих понятий автору не удалось определить в точных терминах, а следовательно, не удалось и включить в программу. Некоторые члены совершенно не имеют отношения к теории шашек и были придуманы более или менее случайно, как, например, второй момент относительно диагональной оси В настоящее время список состоит из 27 простых членов и 11 комбинированных, о которых будет сказано ниже.

В заключение следует сказать несколько слов относительно точного определения членов и общих способов их оценки. Каждый член характеризует относительное положение сторон и численно равен разности оценок соответствующего параметра для каждой стороны. Изменение знака на обратный соответствует, очевидно, смене сторон. Из соображений обеспечения симметрии оценки для каждой из сторон вычисляются в соответствующие моменты времени. Рассмотрим, например, параметр, который меняется после каждого хода одной из сторон. Оценка этого параметра для черных дается после хода черных, а оценка для белых — только после хода белых. При просмотре вариантов эти индивидуальные оценки даются после каждого хода и сохраняются в памяти для последующих сравнений. Когда требуется дать оценку позиции, программа вычисляет разности между индивидуальными оценками на последнем и предпоследнем ходах. В общем мы стремились по возможности определить все параметры таким образом, чтобы их индивидуальные оценки представляли собой небольшие положительные целые числа.

Двоичные связующие члены

Кроме простых членов вышеописанного типа, в список было внесено некоторое число комбинированных, которые делают оценочный полином нелинейным. Было разработано несколько способов введения нелинейных членов, но только один из них был подвергнут детальной проверке. Из каждой пары простых членов образуются четыре комбинированных. Это делается следующим образом. Диапазон значений каждого простого члена произвольно разбивается на две области, и этим областям присваиваются числа 0 и 1. Так как большинство простых членов симметрично относительно нуля, каждому члену присваивается значение 0 или 1 в зависимости от его знака. Тогда комбинированные члены имеют вид $A \cdot B$, $A \cdot \overline{B}$, $\overline{A} \cdot B$ и $\overline{A} \cdot \overline{B}$. Эти члены вводятся в оценочный полином с регулируемыми весами и знаками и с этого момента становятся неотличимыми от остальных членов.

Так как число комбинированных членов, составленных из 27 простых, равно 1404, было решено одновременно использовать лишь небольшую часть из них и вводить новые только в случае изъятия из обращения прежних членов, оказавшихся неэффективными. Члены, используемые в настоящее время, приведены в приложении В.

Предварительные испытания алгоритма обучения с помощью выстраивания весов

Способность программы к обучению подтверждается анализом первой серии экспериментов, состоящей из 28 партий[1]. Вначале были произвольно выбраны 16 членов и всем им присвоены одинаковые веса. Первые 14 партий Альфа играла белыми, а для Беты указывался первый ход (два цикла из 7 различных первых ходов). После этого Альфа играла черными и белыми попеременно. В течение этого времени 29 разных членов подвергались замене, причем большинство из них по два раза.

Представляют интерес некоторые цифры, полученные в результате этой серии игр. Через небольшие промежутки времени программа составляла список 12 ведущих членов оценочного полинома Альфы вместе с их коэффициентами корреляции. Анализируя эти данные, можно заметить, что по крайней мере 20 различных членов имели в то или иное время наибольший

[1] Средняя продолжительность партии составляла 68 ходов (по 34 с каждой стороны). Изменение оценочного полинома производилось приблизительно через каждые 20 ходов.

вес. Некоторые из них несколько раз чередовались, а два даже появлялись во главе списка с противоположными знаками. Несмотря на то что эти колебания были более сильными в начале серии игр и со временем уменьшались, наличие их указывает на недостаточную стабильность процесса обучения. В ходе первых семи партий было произведено по крайней мере 14 изменений среди первых 10 членов списка. Альфа выиграла 3 партии и проиграла 4. Качество игры было чрезвычайно низким. В течение следующих семи партий было сделано не менее 8 изменений среди 5 первых членов. Альфа проиграла первую из этих партий и выиграла остальные 6. Качество игры постепенно повышалось, но машина все еще играла довольно плохо. В ходе следующих семи партий (с 15-й по 21-ю) было 8 изменений среди 5 первых членов. Альфа выиграла 5 партий и проиграла 2. Несколько довольно сильных любителей, игравших в этот период с машиной, пришли к заключению, что она, хотя и попадается в ловушки, но способна и сама нанести удар.

В ходе следующих семи партий (с 22-й по 28-ю) было по крайней мере 4 изменения среди первых трех членов оценочного полинома. Альфа выиграла 2 партии и проиграла 5. Программа, по-видимому, приблизилась к такому качеству игры, которое можно охарактеризовать словами «лучше среднего игрока». Детальный анализ этих результатов показал, что программа способна обучаться. Скорость обучения оказалась неожиданно высокой, но обучение было совершенно беспорядочным и не очень стабильным.

Вторая серия экспериментов

Первая серия экспериментов позволила установить некоторые из наиболее очевидных причин беспорядочного поведения. Программа была модифицирована, после чего были проведены новые эксперименты. Из всех изменений, внесенных в программу, четыре заслуживают более подробного обсуждения.

Во-первых, программа часто сбивалась с правильного пути из-за плохой игры противника. Простое решение заключается в том, что, когда $\delta > 0$, коэффициенты корреляции следует менять не столь сильно, как в случае $\delta < 0$. В модифицированной программе для случая $\delta > 0$ производится коррекция не всех членов оценочного полинома. Если оценочный полином положителен, изменяются коэффициенты только при отрицательных членах, если он отрицателен — только при положительных; коэффициенты при членах, равных нулю, не изменяются. В случае $\delta < 0$ изменение коэффициентов производится так же, как и прежде.

Второй дефект оказался связанным со слишком частым введением в оценочный полином новых членов и тенденцией этих членов занимать главные позиции, не имея для этого достаточных оснований. Этот недостаток был устранен соответствующим уменьшением частоты введения новых членов (через каждые 32 хода вместо 8).

Третий недостаток был связан с тем, что многие из встречавшихся в процессе игры позиций не участвовали в обучении из-за того, что величина δ была меньше установленного порога. В результате программа не придавала значения позициям, допускающим яркие, самобытные ходы. Несмотря на то что пока не существует точного метода, обеспечивающего правильный выбор позиций, которым следует придавать значение в процессе обучения, был найден очень простой способ уменьшения вредных последствий, проистекавших из указанного недостатка.

Этот способ состоит в том, что длина цепочки ходов, относительно которой вычисляется δ, увеличивается до тех пор, пока величина δ не станет больше пороговой. После этого производится коррекция весов оценочного полинома. В этом случае опасность возникновения неустойчивости вследствие увеличения длины цепочки ходов (которая обсуждалась в разделе «Неустойчивость») не возникает, так как при малых δ оценочный полином не меняется. Конечно, всякий раз, когда величина δ превышает пороговую, программа должна вычислить значение исправленного полинома для текущей позиции и, таким образом, снова начать пользоваться цепочками, состоящими только из одной пары ходов. Этот алгоритм устраняет недостаток, связанный с тем, что программа придает значение позициям, слишком далеко отстоящим друг от друга вдоль цепочки ходов, но вводит такую же возможность в отношении позиций, которые отстоят недостаточно далеко.

Чтобы частично компенсировать эту вновь появившуюся опасность, был изменен способ первоначальной оценки позиции. Первоначальная оценка в модифицированной программе производится не непосредственно, как раньше, а с помощью укороченного дерева вариантов (обрывающегося на первом ходе, на котором не происходит взятия шашки). Таким образом были устранены ошибки, связанные с неучетом вынужденных продолжений, и благодаря увеличению точности оценки оказалось возможным уменьшить пороговое значение δ.

Наконец, чтобы избежать передачи Бете плохого полинома в результате случайного выигрыша Альфы (или если Альфа «испортила» свой полином после того, как она добилась преимущества в начале или в середине партии), было решено, что полином должен передаваться Бете только после того, как Альфа выиграет большинство партий.

Для испытания модифицированной программы была проведена новая серия экспериментов. Чтобы уменьшить время обучения, первоначальный выбор членов оценочного полинома был сделан на основе предыдущих экспериментов, но их веса были выбраны произвольно. Исправленная программа в противоположность беспорядочному поведению в первой серии экспериментов оказалась очень стабильной, возможно, за счет несколько более низкой начальной скорости обучения.

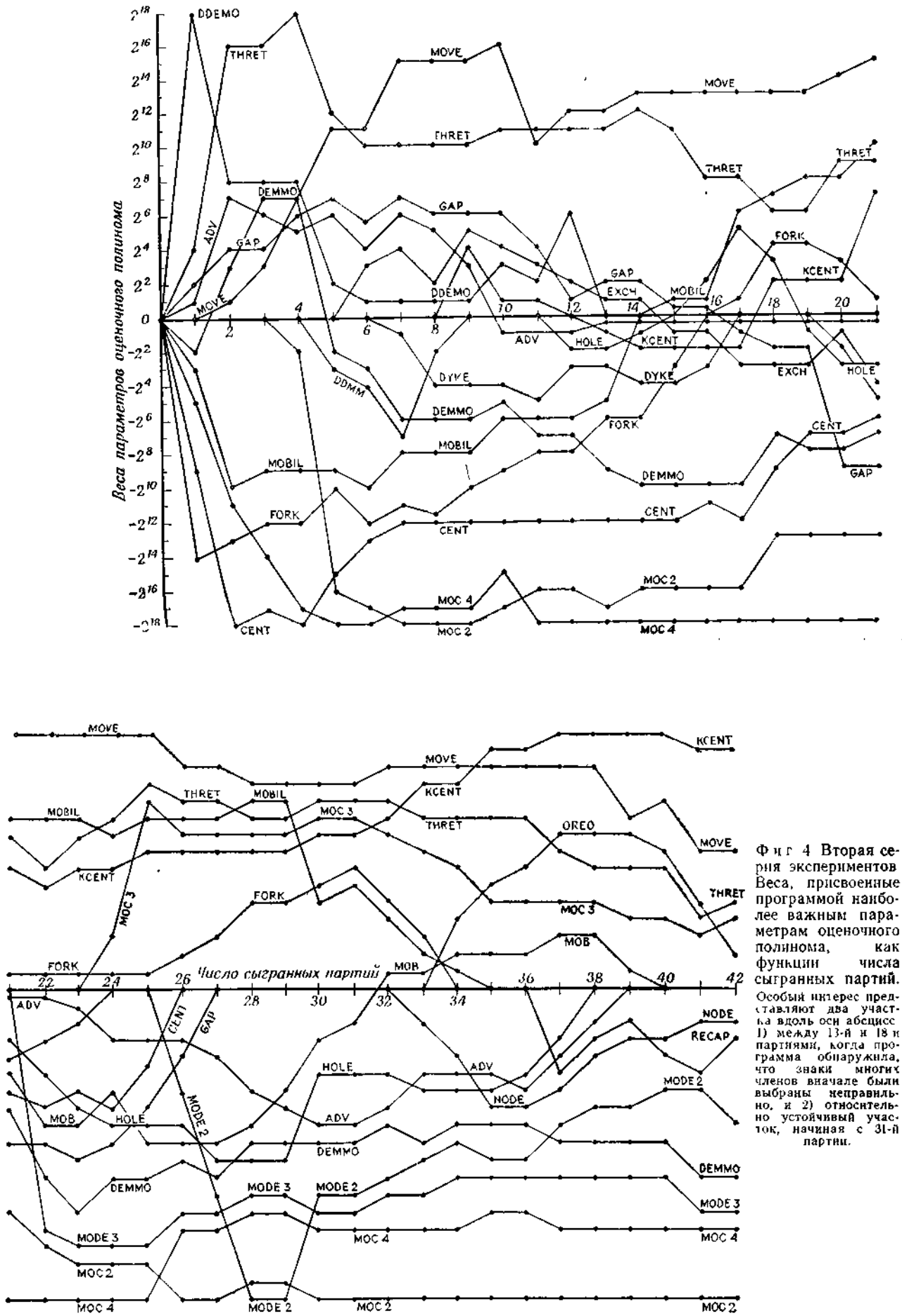
Процесс изменения весов оценочного полинома показан на фиг. 4.

Наиболее сильно изменение в поведении программы выразилось в увеличении относительного числа выигранных Альфой партий и увеличении числа ничьих. Из первых 28 партий первой серии Альфа выиграла 16 и проиграла 12. Для первых 28 партий новой серии соответствующие цифры равны 18 и 4 при 6 ничьих. Во всех случаях партии, не закончившиеся к 70-му ходу, прекращались и результат определялся, исходя из имеющейся к этому ходу позиции.

К сожалению, эти цифры непосредственно не сопоставимы, так как частота передачи полинома от Альфы к Бете во второй серии была меньше, чем в первой. Это было следствием, во-первых, изменения программы и, во-вторых, ошибки в программе, из-за которой процедура передачи полинома в течение нескольких игр не состоялась. Тем не менее значительное уменьшение числа проигрышей и возрастание числа ничьих, по-видимому, указывают на то, что процесс обучения стал намного более стабильным. Некоторые типичные партии из второй серии приведены в приложении Б.

По мере того как происходит обучение, Альфе становится все труднее улучшать свою игру, и можно ожидать, что относительное число выигрышей Альфы будет со временем уменьшаться. Если бы в оценочном пространстве встретился локальный максимум, могла бы возникнуть ситуация, в которой Альфа выигрывает менее половины партий. Если Бета достигнет этого максимума, любое незначительное изменение полинома Альфы приведет к ухудшению ее игры, и до тех пор, пока Альфа не найдет лучшую точку, будут происходить колебания около этого максимума. В поведении программы обнаруживается сходство с поведением, описанным выше, хотя для того, чтобы подтвердить это сходство, потребуется значительно больше игр, чем было проведено до сих пор.

Проведенные эксперименты позволяют сделать следующие выводы:



Фиг 4 Вторая серия экспериментов Веса, присвоенные программой наиболее важным параметрам оценочного полинома, как функции числа сыгранных партий. Особый интерес представляют два участка вдоль оси абсцисс 1) между 13-й и 18-й партиями, когда программа обнаружила, что знаки многих членов вначале были выбраны неправильно, и 2) относительно устойчивый участок, начиная с 31-й партии.

1. Простые алгоритмы обобщения прошлого опыта, типа тех, которые применялись в данной работе, могут оказаться эффективными в задачах, для которых возможно исследование дерева вариантов.

2. Размеры памяти, требуемой для этих алгоритмов, весьма невелики и не меняются в процессе работы.

3. Требуемое машинное время также находится в разумных пределах, и время одной итерации не зависит от того, в начале или в конце процесса обучения мы находимся.

4. Можно ожидать появления некоторой неустойчивости процесса, однако, по крайней мере для шашечной программы, эту неустойчивость можно устранить очень простыми способами.

5. Несмотря на несовершенство и чрезмерную избыточность используемого набора параметров, машина может в относительно короткий срок научиться играть в шашки лучше среднего игрока.

Следует, однако, заметить, что эти эксперименты не завершены достаточно большой серией игр, которая позволила бы надежно утверждать, что процесс обучения вполне стабилен или что он неизбежно приводит к наилучшему возможному выбору параметров и их весов.

Сравнение двух методов обучения

Интересно сравнить стиль игры программ, обученных методом запоминания позиций и методом выстраивания весов оценочного полинома.

Программа, использующая первый метод, способна быстро научиться копировать игру квалифицированного шашиста в дебюте. Ее игра в середине партии очень бедна, однако легко научить ее избегать явных ловушек в конце партии. Имея материальное преимущество, программа обычно выбирает прямой путь к выигрышу.

Программа, использующая выстраивание весов, так и не научилась играть в общепринятой манере, и ее игра в дебюте осталась слабой. Вместе с тем она быстро научилась хорошей игре в середине партии и, имея материальное преимущество, обычно легко одолевает противника. Интересно отметить, что после 28 партий программа все еще не научилась выигрывать эндшпиль с двумя дамками против одной в двойном углу.

Очевидно, метод запоминания более полезен в тех случаях, когда между любым действием и его результатами существует большая задержка либо когда требуется применять очень специальные методы. Напротив, подстройка коэффициентов наиболее полезна в тех случаях, когда имеется большое число возможных вариантов и можно быстро увидеть последствия того или иного действия.

Методы, включающие оба способа обучения

Следующий очевидный шаг в разработке методов обучения — комбинирование рассмотренных методов. Однако делать это надо с некоторой осторожностью, так как невозможно обновлять накопленную в памяти машины информацию после каждого изменения оценочного полинома. Одно из возможных компромиссных решений состоит в том, чтобы на ранней стадии обучения сильно ограничить количество хранимой в памяти информации, увеличивая его по мере увеличения стабильности весов в процессе обучения. Например, программу можно организовать таким образом, чтобы она вначале помнила лишь член «материальное преимущество». На некотором этапе процесса обучения можно добавить следующий член и т. д. Другое решение заключается в том, чтобы использовать выстраивание весов до тех пор, пока веса не станут относительно стабильными, а затем ввести запоминание позиций.

Дальнейшее развитие

Так как эксперименты достигли той стадии, на которой скорость обучения сильно уменьшается, можно было бы попытаться создать программу, которая сама генерирует параметры для оценочного полинома. Несмотря на отсутствие хорошего общего метода, члены полинома могут все же генерироваться на основе теорий, предлагаемых любителями игры в шашки. Такой метод идет вразрез с принципами, на которых до сих пор основывался автор, однако весьма вероятно, что мы столкнемся с необходимостью подобных компромиссов при попытках использовать методы обучения в экономических задачах.

Выводы

На основании проведенных экспериментов можно с уверенностью утверждать, что в настоящее время возможно создание обучающихся устройств, способных осуществлять сложную деятельность значительно лучше среднего человека, и что подобные устройства в конечном счете окажутся экономически выгодными в применении к реальным задачам.

Приложение А

Некоторые детали программы

Объем программы

Основная программа (без обучения)	1100 команд
Ввод очередного хода, проверка его правильности и вывод ответного хода машины	1400 „
Программы начала и прекращения игры	600 „
Программа ввода, генераторы таблиц, вывод на печать и т. п.	850 „
Вычисление различных параметров, характеризующих игру машины	700 „
Обучение с помощью запоминания позиций	1500 „
Обучение с помощью выстраивания весов	650 „
Таблицы и константы для основной программы	700 слов
Рабочий массив для основной программы	2000 „
Рабочий массив для обучения с помощью выстраивания весов	500 „
Рабочий массив для обучения с помощью запоминания позиций	Остальная часть памяти

Примерное время вычислений

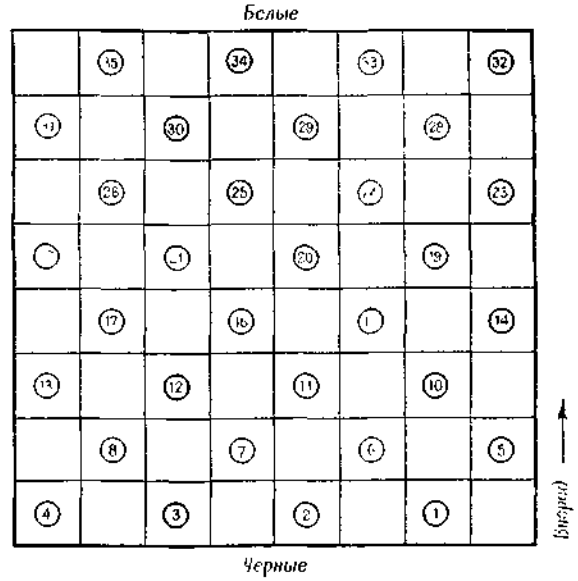
Отыскание всех возможных в данной позиции ходов	2,6 *мсек*
Выполнение хода и получение в результате этого новой позиции	1,5 „
Оценка позиции (4 члена)	2,4 „
Нахождение оценки позиции, хранящейся в памяти (при обучении с помощью запоминания позиций)	2,3 „
Оценка позиции (16 членов)	7,5 „

Представление шашечной доски в машине

Для связи с машиной используется стандартная нумерация полей шашечной доски (см. приложение Б). Внутри машины используется несколько измененная нумерация (фиг. 5). Число на игровом поле указывает номер двоичного разряда машинного слова IBM-704, соответствующего данному полю. Любая позиция представляется четырьмя такими словами. Одно из них (FA) содержит единицы в разрядах, соответствующих игровым полям, занятым теми шашками активной стороны (активной называется сторона, которая в данный момент должна делать ход), которые могут двигаться в направлении, условно обозначенном на фиг. 5 «вперед»[1]. Например, если очередь черных

[1] Здесь и ниже буквенные символы имеют следующее значение: F — вперед (Forward), B — назад (Backward), A — активная (Active), P — пассивная (Passive), L — левый (Left), R — правый (Right). — *Прим. ред.*

делать ход (черные активны), FA указывает местонахождение всех черных шашек (в том числе и дамок). Если же активны белые, FA указывает местонахождение только белых дамок, так как белые шашки могут двигаться лишь «назад». Остальные слова соответственно указывают местонахождение: BA — активных шашек, которые могут двигаться «назад»; FP — пассивных, которые могут двигаться «вперед», и BP — пассивных, которые могут двигаться «назад».



Фиг 5. Нумерация полей в машине

Для экономии памяти позиции, в которых на доске присутствуют одни дамки, представляются тремя словами, а позиции без дамок — двумя. Все эти позиции запоминаются в стандартизованном виде, как указано в основном тексте.

Возможные ходы определяются пятью словами. Одно из них своим знаком указывает, возможно ли взятие шашки противника. (Если да, то в памяти сохраняются только эти ходы.) Остальные четыре слова указывают местонахождение шашек, которые могут двигаться в четырех диагональных направлениях: RF — вправо-вперед, LF — влево-вперед, LB — влево-назад и RB — вправо-назад.

Взглянув на фиг. 5, легко заметить, что один ход вправо-вперед приводит к увеличению номера игрового поля на 4,

а один ход влево-вперед увеличивает этот номер на 5. Номера 9, 18 и 27 отсутствуют. Такая нумерация позволяет вычислять возможные ходы одновременно для всех шашек. Вычислив предварительно слово EMPTY («пустой»), которое содержит единицы во всех разрядах, соответствующих незанятым полям шашечной доски, можно вычислить, например, RF с помощью следующих четырех команд (записанных на символическом языке IBM-704):

CLA EMPTY (вызов слова EMPTY в сумматор)
ALS 4 (сдвиг влево на 4 разряда)
ANA FA (логическая операция «И» между EMPTY и FA)
STO RF (запоминание полученного слова в качестве RF)

Эта же процедура легко распространяется на ходы, связанные со взятием шашки противника. Многократное взятие рассматривается как последовательность однократных взятий, чередующихся с нулевыми ответами противника.

Приложение Б

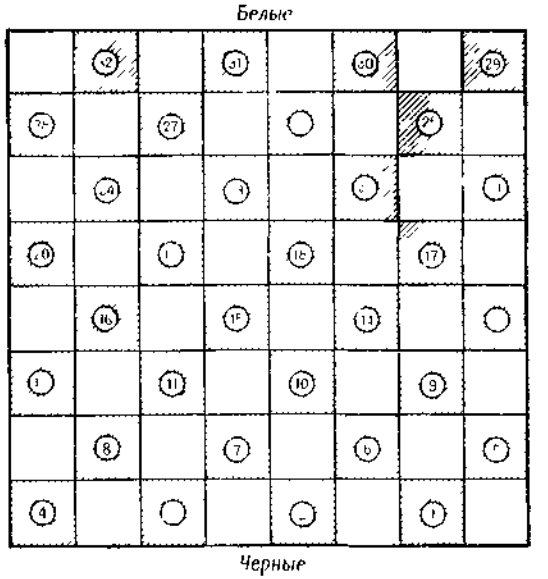
Примеры партий из второй серии экспериментов (С подстраиванием коэффициентов). См. фиг. 6 и 7

Типичные дебюты

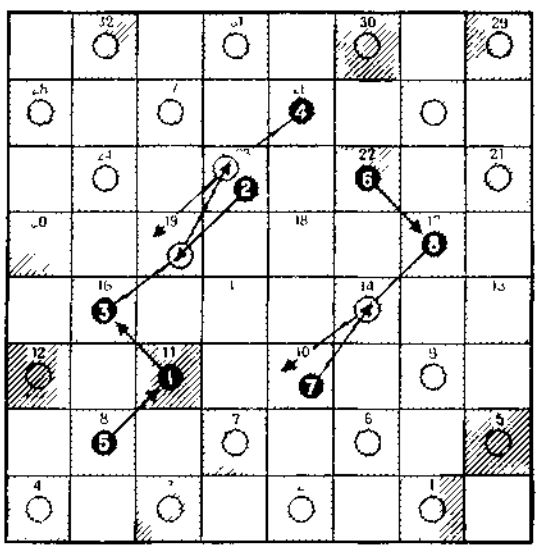
Первые 8 ходов в партиях, в которых Альфа играла черными[1] против Беты

П-4	П-6	П-12	П-17	П-19	П-21	П-31	П-37	П-39	П-41	П-43
10 14	11 16	11 16	11 16	11 16	11 16	11 16	12 16	11 16	10 14	11 16
24 19	22 18	22 17	24 20	24 20	24 20	23 18	24 20	24 20	24 20	23 19
14 18	16 20	16 20	10 14	7 11	8 11	7 11	8 12	10 15	11 15	16 23
23 14	18 14	17 13	20 11	22 17	28 24	27 23	28 24	20 11	27 24	26 19
9 18	9 18	9 14	8 15	10 14	10 14	16 20	10 14	7 16	7 10	8 11
22 15	23 14	23 18	22 17	17 10	23 18	23 19	23 18	21 17	23 18	22 17
11 18	10 17	14 23	7 11	6 15	14 23	20 27	14 23	6 10	14 23	10 14
21 17	21 14	27 18	17 10	28 24	27 18	31 24	27 18	23 19	26 19	17 10

[1] См. примечание 1 на стр. 83. — *Прим. ред.*



Фиг. 6. Обычная нумерация полей шашечной доски



Фиг. 7. Восьмиходовой дебют одной из партий программы с выстраиванием весов

См. стр. 102, партия П-43.

Типичные партии

Примеры партий, в которых Альфа играла белыми [1], первый ход Беты задавался с пульта

П-1	П-18	П-30	П-40
12 16	12 16	12 16	10 14
24 19	24 20	24 20	24 20
8 12	8 12	8 12	11 15
22 18	28 24	28 24	27 24
10 14	10 15	10 14	7 10
26 22	22 18	22 18	23 18
16 20	15 22	6 10	14 23
30 26	25 18	24 19	26 19
11 16	7 10	1 6	10 14
28 24	18 14	32 28	19 10
7 11	10 17	3 8	6 15
22 17	21 14	26 22	22 17
3 8	9 18	9 13	2 7
17 10	23 14	18 9	17 10
6 15 22	6 9	5 14	7 14
26 17	30 25	22 18	24 19
9 13	9 18	6 9	15 24
17 14	26 23	25 22	28 19
2 7	3 8	2 6	14 17
23 18	23 14	30 25	21 14
16 23	1 6	14 17	9 18
14 10	27 23	21 14 5	25 22
7 14	6 9	6 9	18 25
18 9	14 10	18 15	29 22
5 14	9 13	11 18	5 9
27 18 9	25 21	20 11 2	31 27
20 27	11 15	10 14	1 5
31 24	20 11	22 15	20 16
12 16	15 18	14 17	3 7
21 17	23 14	5 1	22 17
13 22	8 15	17 21	8 11
25 18	24 19	25 22	17 13
1 5	15 24	21 25	11 20
9 6	32 28	22 18	13 6

[1] См. примечание на стр. 83 — *Прим. ред.*

П-1	П-18	П-30	П-40
5 9	24 27	25 30	7 10
6 1	31 24	2 6	6 1
9 13	12 16	9 14	4 8
1 6	24 20	18 9	1 6
13 17	16 19	8 11	10 14
32 27	29 25	15 8	6 10
16 20	13 17	4 11	14 17
18 14	10 7	19 15	10 15
11 15	2 11	11 18	17 21
6 10	14 10	23 14	32 28
15 18	19 23	13 17	5 9
14 9	21 14	9 5	27 24
Партия прервана	23 26	12 16	20 27
	10 7	28 24	19 16
	26 30	17 22	12 19
	25 21	6 10	15 22 31
	30 26	30 25	9 14
	7 3	1 6	31 26
	11 15	25 21	14 18
	14 10	5 1	28 24
	5 9	21 17	8 11
	10 6	24 20	24 19
	15 19	16 19	21 25
	6 1	20 16	30 21
	26 22	17 13	Бета сдалась
	1 6	6 2	
	9 13	13 17	
	20 16	10 6	
	19 23	Бета сдалась	
	6 9		
	23 27		
	16 11		
	22 25		
	11 7		
	25 30		
	7 2		
	27 32		

Партия прервана на 70-м ходу

Приложение В

Вычисление значений оценочного полинома во второй серии экспериментов

Способ вычисления членов

Для нахождения численного значения каждого из 16 членов оценочного полинома вычисляется (по указанным ниже правилам) значение соответствующего параметра для рассматриваемой позиции и из него вычитается значение того же самого параметра для позиции, которая имела место перед последним ходом (с учетом перемены мест активной и пассивной сторон). Эта разность затем умножается на соответствующий вес, изменяющийся в пределах от -2^{18} до $+2^{18}$. Значение оценочного полинома принимается в качестве оценки рассматриваемой позиции для пассивной стороны.

Определения параметров

ADV (Advancement — продвижение).

За каждую пассивную шашку на 5-й и 6-й горизонталях (отсчет ведется в направлении движения пассивной стороны) дается 1, а за каждую пассивную шашку на 3-й и 4-й горизонталях дается —1.

APEX (Apex — вершина).

1 дается за каждую из следующих ситуаций: 1) на доске нет дамок; 2) одно из полей 7 или 26 занято активной шашкой и 3) ни одно из этих полей не занято пассивными шашками.

BACK (Back Row Bridge — мост первой горизонтали).

1 дается за каждую из следующих ситуаций: 1) на доске нет активных дамок и 2) два мостовых поля (1 и 3 или 30 и 32) на 1-й горизонтали заняты пассивными шашками.

CENT (Center Control I — контроль над центром I).

1 дается за каждое из полей: 10, 11, 14, 15, 18, 19, 22 и 23, занятых пассивными шашками.

CNTR (Center Control II — контроль над центром II).

1 дается за каждое из полей: 10, 11, 14, 15, 18, 19, 22 и 23, которое занято активной шашкой или может быть занято ею на следующем ходе.

CORN (Double-corner Credit — поощрение за занятие двойного угла).

1 дается за каждую из следующих ситуаций: 1) число всех активных шашек меньше или равно шести; 2) пассивных шашек больше, чем активных, и 3) активная сторона может занять одно из полей двойного угла.

CRAMP (Cramp — связка).

2 дается за позицию, в которой пассивная сторона занимает связывающие поля (13 для черных и 20 для белых) и хотя бы одно из соседних полей (9 или 14 для черных и 19 или 24 для белых), в то время как определенные поля (17, 21, 22 и 25 для черных и 8, 11, 12 и 16 для белых) заняты активной стороной.

DENY (Denial of Occupancy — блокировка).

1 дается за каждое поле, определенное в MOB (см. ниже), если на следующем ходе шашка, занимающая это поле, может быть безвозмездно взята противником.

DIA (Double-Diagonal File — неглавные диагонали).

1 дается за каждую пассивную шашку, находящуюся на диагоналях, проходящих через двойные углы.

DIAV (Diagonal Moment Value — момент относительно главной диагонали).

За каждую пассивную шашку, находящуюся на расстоянии в два хода от двойного угла, дается $^1/_2$, на расстоянии одного хода — 1, а за каждую пассивную шашку в двойном угле дается $^3/_2$.

DYKE (Dyke — дамба).

1 дается за каждую цепочку пассивных шашек, занимающую три смежных по диагонали поля.

EXCH (Exchange — размен).

1 дается за каждое поле, при занятии которого активная сторона вынуждает размен.

EXPOS (Exposure — незащищенность).

1 дается за каждую пассивную шашку, отделенную от какой-либо диагонали двумя незанятыми полями.

FORK (Threat of Fork — угроза «вилки»).

1 дается за каждую ситуацию, в которой пассивные шашки занимают два смежных поля на горизонтали и имеются три пустых поля, расположенных таким образом, что активная сторона, занявшая одно из них, делает «вилку».

GAP (Gap — разрыв).

1 дается за каждое единственное пустое поле, разделяющее по диагонали две пассивные шашки или отделяющее пассивную шашку от края доски.

GUARD (Back-row Control — контроль первой горизонтали).

1 дается, если нет активных дамок и если или мост, или треугольник Орио (см. ниже) заняты пассивными шашками.

HOLE (Hole — дырка).

1 дается за каждое пустое поле, окруженное тремя или более пассивными шашками.

KCENT (King Center Control — контроль дамок над центром).

1 дается за каждое из полей: 10, 11, 14, 15, 18, 19, 22 и 23, занятых пассивными дамками.

MOB (Total Mobility — общая подвижность).

1 дается за каждое поле, на которое активная сторона может пойти одной или более шашками независимо от того, должна ли она на этом ходе брать шашку противника.

MOBIL (Undenied Mobility — подвижность с учетом блокировки).

Параметр определяется как разность между MOB и DENY.

MOVE (Move — передвижение).

1 дается, если общий вес всех шашек (вес простой шашки — 2, вес дамки — 3) меньше 24 и если на вертикалях, начинающихся с полей 1, 2, 3 и 4, находится нечетное число шашек.

NODE (Node — узел).

1 дается за каждую пассивную шашку, окруженную по меньшей мере тремя пустыми полями.

OREO (Triangle of Oreo — треугольник Орео).

1 дается, если на доске нет пассивных дамок и если треугольник Орео (поля 2, 3 и 7 для черных и 26, 30 и 31 для белых) занят пассивными шашками.

POLE (Pole — полюс).

1 дается за каждую пассивную шашку, окруженную со всех сторон пустыми полями.

RECAP (Recapture — возврат захваченного).

Параметр идентичен параметру Exchange. (Он был введен, чтобы выяснить, какое значение имеет случайность отрезков времени, через которые параметры вводятся и выводятся из оценочного полинома.)

THRET (Threat — угроза).

1 дается за каждое поле, на которое может пойти активная шашка, создавая этим угрозу взятия пассивной шашки на следующем ходе.

Двоичные связующие члены

Ниже приводится список сокращенных обозначений этих членов.

	Блокировка (A) — общая подвижность (B)	Подвижность с учетом блокировки (A) — блокировка (B)	Подвижность с учетом блокировки (A) — контроль над центром 1 (B)
$A \cdot B$	DEMO	MODE 1	MOC 1
$A \cdot \overline{B}$	DEMMO	MODE 2	MOC 3
$\overline{A} \cdot B$	DDEMO	MODE 3	MOC 2
$\overline{A} \cdot \overline{B}$	DDMM	MODE 4	MOC 4

Оценочный полином (первые 12 членов) после 42 партий, в ходе которых было внесено 1039 изменений в коэффициенты [1]

Член	Коэффициент корреляции	Знак	Вес члена (показатель степени)	Число коррекций
MOC 2	0,45	—	18	84
KCENT	0,40	+	16	127
MOC 4	0,35	—	14	95
MODE 3	0,33	—	13	210
DEMMO	0,27	—	11	132
MOVE	0,19	+	8	91
ADV	0,19	—	8	739
MODE 2	0,19	—	8	55
BACK	0,14	—	6	6
CNTR	0,13	+	5	12
THRET	0,13	+	5	442
MOC 3	0,10	+	4	89

[1] Недавно машина сыграла еще 20 партий. Несмотря на некоторые существенные изменения в оценочном полиноме, общая картина стабилизации процесса обучения, которая наблюдается на фиг. 4, сохранилась. В ходе этих партий было сделано еще 412 коррекций весов оценочного полинома и 12 членов добавилось к списку выброшенных.

Члены, выброшенные из оценочного полинома в ходе этих 42 партий [1]

Член	Число коррекций перед выбрасыванием	Член	Число коррекций перед выбрасыванием
CORN	0	MODE 1	1
CRAMP	0	CENT	386
GUARD	0	MODE 4	0
EXPOS	162	FORK	400
DDMM	19	MOBIL	707
DYKE	115	POLE	11
MOC 1	1	HOLE	593
EXCH	445	GAP	792
DDEMO	53	MOB	608

[1] См. примечание к предыдущей таблице. — *Прим. ред.*

Приложение Г

Партия, сыгранная Р. Нили и шашечной программой Сэмюэля

Летом 1962 г. по просьбе редакторов настоящего сборника д-р Сэмюэль организовал матч между своей программой (для машины IBM-7090) и мастером по шашкам Р. Нили (R. W. Nealey), который аттестуется журналом *IBM Research News* (август 1962 г.) как «бывший чемпион штата Коннектикут и один из выдающихся игроков США».

Ниже помещена партия, в которой программа Сэмюэля выиграла у Нили, с комментариями д-ра Сэмюэля. Сам Нили следующим образом прокомментировал эту партию в журнале *IBM Research News:*

В нашей партии можно выделить несколько моментов. Все ходы вплоть до 11-го можно найти в литературе, за исключением того, что я несколько раз избегал известных продолжений в тщетной надежде сбить машину с толку. Насколько я могу судить, начиная с хода 30—26 и далее наша партия вполне оригинальна. Мне кажется очень интересным, что для достижения победы машине понадобилось сделать несколько блестящих ходов, и если бы не это, я имел неоднократно возможность свести партию вничью. Именно поэтому я и продолжал играть. Однако машина провела окончание партии без единой ошибки. В отношении эндшпиля я не встречал среди людей столь же сильных соперников с 1954 г., когда последний раз проиграл партию.

Р. Нили (белые) — Программа Сэмюэля (черные)
12 июля 1962 г., Йорктаун, шт. Нью-Йорк

Нили было предоставлено право выбора цвета, и он предпочел защищаться[1]. Затем был разыгран дебют «Old Fourteenth» («Старое 14-е начало»).

11	15		
23	19		
8	11		
22	17		
4	8		
17	13		25—22 несколько больше ограничивало бы свободу черных
15	18		
24	20		Старое 14-е начало Ли, вариант 9. 11—15 магистральный ход
9	14		
26	23		Вариант 100 Дорана, дающий равную игру
10	15		
19	10		
6	15		

28	24		Доран рекомендует ход 23—19, облегчающий игру белых
15	19		Агрессивный ход черных
24	15		
5	9		
13	6		
1	19	26	
31	22	15	
11	18		Это все еще вариант 9 Ли и вариант 100 Дорана
30	26		По-видимому, слабый ход со стороны Нили
8	11		Хороший ответ, сохраняющий контроль над центром
25	22		
18	25		
29	22		
11	15		
27	23		
15	19		
23	16		

[1] То есть выбрал белый цвет, см примечание на стр. 83. — *Прим. ред.*

РАЗДЕЛ 2

МАШИНЫ ДЛЯ ДОКАЗАТЕЛЬСТВА МАТЕМАТИЧЕСКИХ ТЕОРЕМ

Нахождение доказательств математических теорем является сложным интеллектуальным процессом. Изучение методов математического доказательства рассматривается многими как хорошая тренировка для подхода к общей проблеме «решения задач».

Однако оказывается, что те изящные доказательства, которые математики дают в научных статьях и монографиях, не позволяют достаточно глубоко проникнуть в действительные мыслительные процессы, приводящие к нахождению таких доказательств. Иногда об этих процессах удается получить беглое представление на лекции выдающегося педагога-математика.

Для большинства исследователей, работающих в этой области, притягательная сила доказательства машиной теорем лежит не столько в конечном результате (получение теорем, в том числе, возможно, новых и важных), сколько в средствах достижения этой цели (глубокое изучение структуры процесса переработки информации при математическом творчестве). Понимание этих процессов решения задач, по-видимому, явится важным шагом на пути программирования более сложных и общих процессов, связанных с решением различных интеллектуальных задач. В исследованиях доказательства теорем, так же как и в исследовании игр, простота формальной системы

позволяет бо́льшую часть усилий исследователей направить на понимание самих процессов решения, а не на формализацию задачи.

Не все работы по машинному доказательству теорем посвящены наиболее общей проблеме — способам решения задач. Некоторые исследователи, в частности Ван Хао, обращают основное внимание на результат — получение доказательств теорем. Они достигли впечатляющих результатов, используя новые и сложные математические решающие правила.

«Логик-теоретик» — программа для вычислительной машины, которая находит доказательства теорем математической логики (перечисленных в гл. 2 книги «Principia Mathematica» Уайтхеда и Рассела). Эта программа использует методы доказательства, которые ненамного лучше тех, которыми пользуется студент, только начинающий изучение «Principia».

«Логик-теоретик» был запрограммирован Ньюэллом, Шоу и Саймоном в начале 1956 г. Это была первая эвристическая программа, полностью реализованная на вычислительной машине, первая попытка проникнуть в сложные процессы мышления с помощью исследований в области искусственного разума.

Интересно отметить, что работа над программой «Логик-теоретик» привела к созданию первого машинного языка для работы со списками символов, что имело большое значение для исследований по искусственному разуму и вообще для наук, связанных с вычислительными машинами. Этот язык назван «языком для обработки информации», или сокращенно IPL[1].

Помещенная в этом разделе работа Гелернтера и его сотрудников распространяет идеи эвристического программирования на доказательство теорем эвклидовой геометрии. Его программа позволяет использовать эвристические методы там, где они наиболее эффективны, но, кроме того, в ней всюду, где это оказывается целесообразным, применяются более мощные, более формализованные методы оперирования символами. В программе для доказательства геометрических теорем специальный интерес представляет использование чертежа как эвристического приема для исследования структуры подзадач.

Группа Гелернтера для написания своей программы также разработала особый язык обработки списков. Этот язык, получивший название «язык ФОРТРАН для обработки списков» (FLPL), позволяет сочетать преимущества языка ФОРТРАН (предназначенного для численных задач) с возможностью оперирования списками символов.

[1] Подробнее о языке IPL см. статью «Краткое описание языка IPL-V», стр. 485 настоящей книги — *Прим. ред.*

3. ЭМПИРИЧЕСКИЕ ИССЛЕДОВАНИЯ МАШИНЫ «ЛОГИК-ТЕОРЕТИК»; ПРИМЕР ИЗУЧЕНИЯ ЭВРИСТИКИ

А. Ньюэлл, Дж. Шоу и Г. Саймон

Введение

Эта работа касается изучения процесса решения задач и является частью программы исследования сложных систем обработки информации. Мы разработали схему для нахождения доказательств теорем в элементарной символической логике. Составив по этой схеме программу для вычислительной машины, мы с ее помощью получили эмпирические данные, касающиеся процесса решения задач в элементарной логике. Мы назвали эту программу «машина Логик-теоретик» (ЛТ). Она была разработана для того, чтобы изучить возможность решения таких трудных задач, как доказательство математических теорем, выявление научных законов в совокупности опытных данных, игра в шахматы или понимание смысла английской прозы.

Цель настоящего исследования — разобраться в тех сложных процессах («эвристиках»), которые участвуют в решении задач. Поэтому мы не интересуемся методами, обеспечивающими решение задач, но требующими очень большого объема вычислений, а стремимся понять, каким образом, например, математик в состоянии доказать теорему, даже если он вначале не знает, как ему это сделать, и сможет ли он вообще ее доказать.

Это заставляет нас сосредоточить свое внимание на чистой теории поискового решения задач [787]. Ранее мы подробно описали программу для ЛТ, и здесь мы повторим только те детали описания, которые необходимы для того, чтобы читатель понимал полученные результаты. В другой статье [790] мы показали, как может быть составлена программа для вычислительной машины, которая должна выполнять действия, предусматриваемые ЛТ, и эта задача имеет самостоятельный интерес. Мы также откладываем до следующих статей обсуждение выводов, которые из нашей работы можно сделать для психологической теории человеческого мышления и процесса решения задач человеком. Читатель легко укажет сам на другие возможные области ее приложения, здесь же мы сосредоточим внимание лишь на природе самого процесса решения задач.

При изучении сложных систем мы будем придерживаться следующей стратегии: детальное определение системы, программирование ее для ЦВМ и эмпирическое изучение ее поведения при различных значениях параметров и в различных условиях.

Такая стратегия в настоящее время, по-видимому, является единственным удовлетворительным средством, позволяющим понять поведение таких систем. Хотя область задач, с которой имеет дело эта система (ЛТ), весьма элементарна, она служит хорошим примером трудной задачи, так как логика представляет собой изучаемый в институтских курсах предмет, достаточно трудный для большинства людей.

Наши результаты получены с несколькими программами, реализованными на машине JOHNNIAC — одной из быстродействующих ЦВМ корпорации «Рэнд». Мы опишем результаты этих экспериментов, а также проанализируем их и покажем, что они дают для понимания процесса решения задач.

Работа машины ЛТ

Сначала мы дадим конкретное представление о работе «Логика-теоретика». ЛТ — это, конечно, программа, написанная для машины JOHNNIAC, т. е. набор знаков на бумаге или отверстий на перфокартах. Однако мы можем представлять себе ЛТ также как реальную физическую машину, а работу программы как поведение такой машины. ЛТ можно отождествить с машиной JOHNNIAC, в которую введена основная программа, но не введены исходные данные.

Задачей ЛТ является доказательство теорем по элементарной символической логике, а еще точнее — по исчислению высказываний. *Исчисление высказываний* — это формализованная математическая система, содержащая выражения, построенные путем комбинации основных символов. Пять таких выражений выбирается в качестве аксиом, и существуют *правила вывода*, которые позволяют получать новые теоремы из этих аксиом и других теорем. По содержанию и по форме элементарная символическая логика очень похожа на абстрактную алгебру. Обычно переменные этой системы интерпретируются как высказывания, а аксиомы и правила вывода — как формализации логических операций, например дедукции. Однако ЛТ имеет дело с чисто формальной математической системой, поэтому нам в дальнейшем не потребуется такая интерпретация. Для понимания задачи ЛТ нам придется сообщить некоторые общие сведения из исчисления высказываний.

Постулируется множество *переменных* p, q, r, ..., A, B, C, ..., с которыми имеет дело исчисление высказываний. Эти переменные при помощи *связок* могут комбинироваться в выражения. Имея любую переменную p, можно построить выражение «не p» ($\bar{p}$). Имея любые две переменные p и q, мы можем образовать выражение «p или q» ($p \vee q$) или выражение «p влечет за собой q» (импликация, $p \rightarrow q$), где знаки $\vee$ и $\rightarrow$ являются связками. Существуют другие связки, например «и» (&), но они нам здесь не понадобятся. Если мы образовали некоторые выражения, то из них мы можем в дальнейшем строить более сложные выражения. Например, мы можем построить такое выражение[1]:

$$(p \rightarrow \bar{p}) \rightarrow \bar{p}. \qquad (2.01)$$

Кроме того, задан набор выражений, которые являются аксиомами. В качестве аксиом выбираются универсальные истинные высказывания, из которых при помощи различных правил вывода должны быть выведены теоремы. Для определенности мы в работе с ЛТ выбрали систему аксиом, определений и правил, которые используются в «Principia Mathematica», где перечислено пять аксиом:

$$(p \vee p) \rightarrow p, \qquad (1.2)$$

$$p \rightarrow (q \vee p), \qquad (1.3)$$

$$(p \vee q) \rightarrow (q \vee p), \qquad (1.4)$$

$$[p \vee (q \vee r)] \rightarrow [q \vee (p \vee r)], \qquad (1.5)$$

$$(p \rightarrow q) \rightarrow [(r \vee p) \rightarrow (r \vee q)]. \qquad (1.6)$$

Из нескольких истинных теорем можно вывести новые теоремы при помощи трех правил вывода: *подстановки* (substitution), *замены* (replacement) и *отделения* (detachment)[2].

1. Согласно правилу подстановки, любую переменную в любой теореме можно заменить на любое выражение при условии, что эту подстановку делают в этой теореме всюду, где появляется эта переменная. Например, подставляя во второй аксиоме $p \vee q$ вместо p, мы получаем новую теорему

$$(p \vee q) \rightarrow [q \vee (p \vee q)].$$

2. Согласно правилу замены, связь можно заменить ее определением и наоборот, где бы они ни встречались. По определению, «$p \rightarrow q$» означает то же, что и «$\bar{p} \vee q$». Следовательно, первое выражение можно всегда заменить вторым, и наоборот. Например, из аксиомы (1.3) заменой $\rightarrow$ на $\vee$ мы получаем новую теорему

$$\bar{p} \vee (q \vee p).$$

[1] Для простоты ссылок мы нумеровали аксиомы и теоремы в соответствии с их нумерацией в "Principia Mathematica", изд 2, т. I, Нью-Йорк, 1935

[2] В отечественной литературе правило отделения часто называют правилом заключения. — *Прим. ред.*

3. По правилу отделения, если A и $A \to B$ — теоремы, то B — также теорема. Например, из

$$(p \vee p) \to p$$

и

$$[(p \vee p) \to p] \to (p \to p)$$

получаем новую теорему

$$p \to p.$$

Для доказательства некоторого выражения в качестве исходного материала выбирают систему аксиом и уже доказанных теорем, к которым последовательно применяют различные правила до получения желаемого выражения. Доказательством является такая последовательность выражений, каждое из которых обоснованно получено из предыдущих, что вся последовательность ведет от аксиом и известных теорем к желаемому выражению.

Вот и все основные понятия символической логики, необходимые для наблюдения за ЛТ в процессе работы. ЛТ «понимает» выражения в символической логике, т. е. существует простой код, позволяющий наносить эти выражения на перфокарты для ввода их в машину. Мы сообщаем ЛТ пять аксиом и указываем ему, что он может эти аксиомы принять как истинные теоремы. ЛТ уже знает правила вывода, а также определения подстановки, замены и отделения. Затем мы даем ЛТ некоторое выражение, скажем выражение (2.01), и просим найти для него доказательство. ЛТ работает около 10 *сек*, а затем печатает следующее доказательство:

$(p \to \bar{p}) \to (\bar{p})$ (теорема (2.01), которую требуется доказать);

(1) $(A \vee A) \to A$ (аксиома (1.2));

(2) $(\bar{A} \vee \bar{A}) \to \bar{A}$ (подстановка $\bar{A}$ вместо A);

(3) $(A \to \bar{A}) \to \bar{A}$ (замена $\vee$ на $\to$);

(4) $(p \to \bar{p}) \to \bar{p}$ (подстановка p вместо A),

что и требовалось доказать.

Затем мы просим ЛТ доказать достаточно сложную теорему, обозначенную в труде Уайтхеда и Рассела как теорема (2.45), разрешив ему при этом использовать все 38 теорем, доказанных до (2.45). После приблизительно 12 *мин* работы ЛТ дает следующее доказательство:

$(\overline{p \vee q}) \to \bar{p}$ (теорема (2.45), которую требуется доказать);

(1) $A \to (A \vee B)$ (теорема (2.2));

(2) $p \to (p \vee q)$ (подстановка в (1) p вместо A и q вместо B);

(3) $(A \to B) \to (\bar{B} \to \bar{A})$ (теорема (2.16));

(4) $[p \to (p \vee q)] \to [(\overline{p \vee q}) \to \bar{p}]$ (подстановка в (3) p вместо A и $(p \vee q)$ вместо B);

(5) $(\overline{p \vee q}) \to \bar{p}$ (отделение с помощью (2) правой части (4));

что и требовалось доказать.

Наконец, ЛТ сообщают все теоремы вплоть до (2.31) (всего 28) и просят доказать, что

$$[p \vee (q \vee r)] \to [(p \vee q) \vee r].$$

ЛТ работает около 23 *мин* и затем сообщает, что он не может доказать (2.31), что он исчерпал свои возможности

Итак, что же в этом поведении ЛТ нуждается в объяснении? Примеры, рассмотренные выше, являются трудными задачами для большинства людей, причем большинство, если и находят доказательства, не знают, какими путями они этого достигли. Не существует известной простой процедуры для их нахождения. Имеются лишь различные методы для проверки того, является ли некоторое данное выражение истинным или ложным. Наилучшим из известных методов является метод *таблиц истинности*. Но эти таблицы не дают доказательства в смысле Уайтхеда и Рассела. Можно придумать «автоматические» методы для нахождения доказательств. В дальнейшем мы кратко остановимся на этом, но оказывается, что для доказательства теоремы (2.45) потребуется машинное время порядка нескольких тысяч лет.

Прежде всего необходимо пояснить, почему такие задачи являются трудными, а затем следует выяснить, с какими особенностями ЛТ связаны как его удачи, так и неудачи. Этим вопросам и будет посвящена остальная часть этой работы.

Задачи, алгоритмы и эвристики

При описании ЛТ, его среды и его поведения мы будем неоднократно использовать три понятия. Первое из них — понятие *задачи* (problem). Абстрактно говоря, человеку задана

задача, если ему даны множество возможных решений и способ проверки того, является ли данный элемент этого множества действительно решением поставленной задачи.

Причина того, почему задачи являются действительно «задачами», лежит в том, что первоначальное множество возможных решений, которое дается решающему задачу, может быть очень большим. Действительные же решения могут быть равномерно и редко распределены по этому множеству, а получение и испытание каждого нового элемента может требовать больших усилий. Таким образом, человеку, решающему задачу, фактически не «дается» множество возможных решений. Вместо этого ему сообщается некоторая процедура, позволяющая вырабатывать элементы этого множества в определенном порядке. При этом способ генерирования элементов имеет свойства, которые обычно не определяются поставленной задачей. Например, порядок в котором вырабатываются элементы, может зависеть от «стоимости» выработки элемента и т. д. Способ проверки также требует определенных затрат времени, связанных с его применением. Задачу можно решить, если эти затраты не очень велики с точки зрения времени и объема вычислений, требуемых для ее решения.

Особым и ценным свойством некоторых «генераторов решений» является даваемая ими гарантия, что если задача имеет решение, то генератор рано или поздно найдет его. Процесс, обладающий таким свойством для некоторой задачи, мы будем называть *алгоритмом* для решения этой задачи. Гарантию, которую дает подобный алгоритм, конечно, нельзя считать «пределом мечтаний», ибо ничего не сказано о стоимости или времени, которое потребуется для нахождения этого решения. Например, простой алгоритм нахождения кодовой комбинации цифр на замке сейфа состоит в полном переборе всех возможных комбинаций и проверке для каждой из них, открылся сейф или нет. Этот алгоритм является типичным процессом решения задачи: имеется генератор, который вырабатывает новые комбинации в определенном порядке, и имеется проверяющее устройство, которое устанавливает, является ли каждая новая комбинация действительно решением задачи. Такой процесс поиска является алгоритмом, так как известно, что при *некоторой* комбинации сейф откроется, и так как генератор исчерпает все комбинации за конечный промежуток времени. Однако реализация такого алгоритма связана с большими затратами, поэтому сейф с кодовым замком можно использовать для хранения ценностей, не опасаясь даже тех людей, которым этот алгоритм известен.

Процесс, который *мог бы* решить данную задачу, но не дает

никаких гарантий этого, называется *эвристическим*[1] для данной задачи. Отсутствие гарантии не означает непоправимой беды. Потери, вызванные отсутствием гарантии, зависят от стоимости этого процесса и от того, какие иные алгоритмы возможны. Для большинства встречающихся задач у нас есть только эвристики, но бывают случаи, когда для решения одной и той же задачи мы можем использовать как алгоритмы, так и эвристики. Иногда, как, например, в случае отыскания максимумов простых дифференцируемых функций, используется алгоритм приравнивания нулю первой производной; никто не станет испытывать все точки на прямой, даже если бы это было возможно. В некоторых же случаях, как, например, в шахматах, всегда играют с помощью эвристик, так как никто не может реализовать алгоритм просмотра всех продолжений игры до конца.

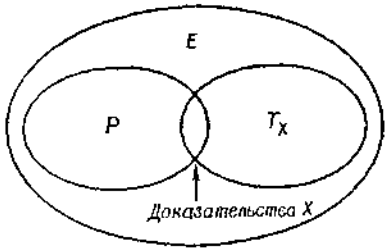
Задача доказательства теорем в логике

Нахождение доказательства теоремы в символической логике можно описать как выбор некоторого элемента из выработанного множества элементов (фиг. 1). Рассмотрим множество *всех возможных последовательностей логических выражений*, которое обозначим через E. Некоторые из этих последовательностей — очень небольшая часть — будут доказательствами. Последовательность, являющаяся доказательством, удовлетворяет следующему тесту: каждое выражение в последовательности либо 1) является одной из принятых теорем или аксиом, либо 2) получено из одного или двух предыдущих выражений в последовательности путем применения одного из трех правил вывода.

Обозначим через P *множество последовательностей, которые являются доказательствами*. Некоторые последовательности в E в качестве конечного выражения содержат *подлежащее доказательству выражение*, которое мы обозначим через X. Это множество последовательностей обозначим через T_X. Тогда нахождение доказательства данной теоремы означает выбор элемента из E, который принадлежит пересечению множеств P и T_X. Множество E вполне определяется правилами, по которым вырабатываются новые последовательности логических выражений.

[1] Как существительное слово «эвристика» встречается редко и в основном означает «искусство делать открытия». Прилагательное «эвристический» по словарю Вебстера определяется так: служащий для открытия или нахождения чего-либо. Именно в этом смысле оно употребляется в выражениях «эвристический процесс» или «эвристический метод». Для краткости мы будем использовать термины «эвристика» и «эвристический процесс» как синонимы.

Трудность доказательства теорем связана с малым числом элементов в пересечении множеств P и T_X по сравнению с числом элементов в E. Следовательно, она связана со стоимостью и скоростью «генераторов», вырабатывающих элементы E, а также со стоимостью и скоростью выполнения проверки принадлежности элемента к множеству T_X или P. Трудность доказательства зависит также от того, можно ли найти такие генераторы, которые гарантируют автоматическое выполнение некоторых условий для любого вырабатываемого элемента. Наконец, как мы увидим, трудность доказательства в сильной



Фиг. 1. Взаимное отношение множеств E, P и T_X.

E — все последовательности логических выражений; P — последовательности, являющиеся доказательствами; T_X — последовательности, оканчивающиеся выражением X.

степени зависит от эвристик, которые можно найти для руководства таким выбором.

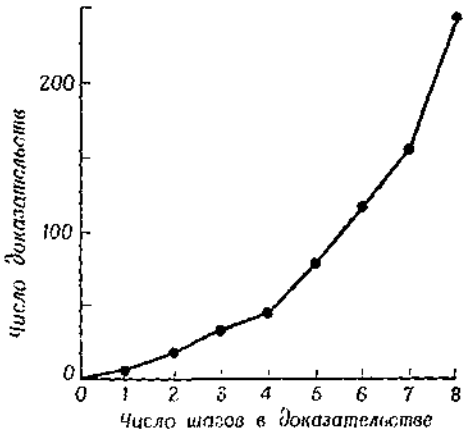
Небольшие размышления и опыт, приобретенный при попытках доказательства теорем, делают очевидным, что последовательности, являющиеся доказательствами, для определенных теорем встречаются весьма редко. Чтобы точнее выяснить, почему теоремы трудно доказывать, мы построим алгоритм для доказательства теорем. Алгоритм этот основан только на приведенных выше тестах и определениях, а не на каких-либо «глубоких» свойствах символической логики. Таким образом, он отражает существенную природу процесса доказательства теорем, т. е. природу, присущую доказательству до построения какого-либо детально разработанного метода доказательства. Мы назовем этот алгоритм «алгоритмом Британского Музея», что указывает на возможный источник подобных процедур.

Алгоритм Британского Музея

Этот алгоритм систематически строит все возможные доказательства, причем он на каждом шаге 1) устраняет дубликаты и 2) просматривает, не совпадает ли последняя теорема в доказательстве с выражением, которое требуется доказать. Для этого алгоритма множество одношаговых доказательств совпадает с множеством аксиом (так как каждая аксиома есть одно-

шаговое доказательство самой себя). Множество n-шаговых доказательств получается из множества $(n-1)$-шаговых доказательств путем выполнения всех допустимых подстановок и замен в выражениях $(n-1)$-шаговых доказательств и выполнения всех допустимых отделений пар выражений, разрешаемых рекурсивным определением доказательства[1].

На фиг. 2 показано, как растет число n-шаговых доказательств с ростом n в самом начале процесса выработки доказательства. Эта зависимость получена при применении только



Фиг. 2. Число доказательств, выработанных алгоритмом Британского Музея за первые несколько шагов.

Рассматриваются только подстановки $\bar{p}$ вместо p, $\bar{q}$ вместо q и $\bar{r}$ вместо r и замещения «или» на «влечет за собой» и «влечет за собой» на «или».

замен «или» на «влечет за собой» и наоборот, а также отрицаний переменных (например, замены $\bar{p}$ на p). Ни отделения, ни сложные подстановки (например, $q \vee r$ вместо p) не применялись. Не применялись также подстановки частных значений переменных, например p вместо q в выражении $p \vee q$. Если мы

[1] При построении алгоритма приходится принимать во внимание следующее обстоятельство, вызывающее определенные трудности. Выражение «все допустимые подстановки» требует разъяснения, поскольку число таких подстановок бесконечно. Необходимо тщательно избегать повторных выражений, которые отличаются только названиями переменных. Не входя здесь в детали, просто укажем, что эти трудности удается устранить. Существенным моментом при построении алгоритма является то, что при генерации каждого нового выражения разрешается выполнить только одну какую-либо операцию, т. е. одну замену, одну подстановку $\bar{p}$ вместо p и т. д.

включим подстановки такого типа, требующие еще трех шагов, то алгоритм будет вырабатывать около 600 дополнительных теорем, и, таким образом, множество доказательств, требующих 11 или меньшего числа шагов, будет состоять приблизительно из 1000 теорем, среди которых нет одинаковых.

Чтобы убедиться, как этот алгоритм дает доказательства конкретных теорем, рассмотрим его работу применительно к 60 с лишним теоремам, изложенным в гл. 2 «Principia Mathematica». Одна теорема (2.01) получена на 4-м шаге работы алгоритма (т. е. ее доказательство содержится среди 4-шаговых), и, следовательно, она находится среди первых 42 доказанных теорем. Три следующие теоремы [(2.02), (2.03), (2.04)] получены на 6-м шаге, т. е. находятся среди первых 115 теорем. Следующая теорема (2.05) получена на 8-м шаге, т. е. содержится в первых 246 теоремах. И только одна теорема дополнительно появляется среди первых 1000 теорем — теорема (2.07). Доказательства всех оставшихся теорем требуют сложных подстановок или отделений.

В настоящее время у нас нет способа оценить число доказательств, которое необходимо выработать, чтобы охватить доказательства всех теорем гл. 2. По нашему мнению, их должно быть порядка ста миллионов. Более того, за исключением приведенных шести теорем, нет оснований полагать, что доказательства оставшихся теорем гл. 2 могут быть получены нашим алгоритмом за небольшое число шагов.

Ввиду весьма скудной информации мы можем дать лишь весьма грубую оценку времени, которое потребуется для нахождения сложных доказательств при помощи этого алгоритма. Однако мы можем оценить время, требуемое на вырабатывание доказательств первых 250 теорем (см. фиг. 2), доказываемых не более чем за 8 шагов, предположив, что время, требуемое для рассматриваемого алгоритма, сравнимо в этом случае со временем, требуемым для ЛТ. Оно равно приблизительно 16 *мин*. Ценным свойством первой части алгоритма (до первого применения отделения) является то, что не требуется проверки на отсутствие повторов. Поэтому время подсчета для первых нескольких тысяч доказательств растет лишь линейно с числом вырабатываемых теорем. Для теорем, которые требуют отделений, необходимо производить проверку на тождественность выражений, и общее время счета растет как квадрат числа вырабатываемых выражений. При такой скорости потребовались бы сотни тысяч лет счета для выработки доказательств всех теорем гл. 2 «Principia Mathematica».

Теперь природа задачи доказательства теорем достаточно ясна. Когда последовательности выражений вырабатываются

простым и дешевым (в расчете на один вырабатываемый элемент) генератором, то вероятность того, что некоторая конкретная последовательность является желаемым доказательством, чрезвычайно мала. Это справедливо и тогда, когда генератор вырабатывает лишь последовательности, которые удовлетворяют очень тяжелым условиям, гарантирующим, что каждая последовательность является доказательством некоторой теоремы. Множество последовательностей так велико, а желаемое доказательство встречается настолько редко, что практически приемлемого объема вычислений недостаточно для нахождения доказательства при помощи такого алгоритма.

Машина „Логик-теоретик"

Если мы хотим, чтобы ЛТ мог доказывать любые теоремы, должны быть радикально изменены как порядок, в котором вырабатываются возможные доказательства, так и способ их проверки. Для этого ЛТ приходится отказаться почти от всех гарантий, которые даются алгоритмом Британского Музея. Процедуры, используемые ЛТ, не гарантируют ни того, что вырабатываемые им последовательности вообще являются доказательствами, ни того, что ЛТ когда-либо найдет доказательство, каковы бы ни были затраченные усилия. Однако *часто* желаемое доказательство этими процедурами все же вырабатывается за приемлемое машинное время.

Методы

Основной тип эвристики, которую использует ЛТ, мы назовем *методом*. Пока у нас нет точного определения метода, которое отличает его от всех других типов правил в ЛТ. Грубо говоря, метод — это *целесообразный непротиворечивый набор операций*, который, будучи примененным, и определяет в основном поиск доказательства. Методы являются самыми крупными блоками, на которые распадается программа ЛТ. Согласование и выбор методов производятся специальной управляющей программой.

Метод подстановки

Этот метод ищет доказательство определенного выражения путем поиска среди аксиом и ранее доказанных теорем такой аксиомы или теоремы, которую рядом подстановок переменных и заменой связок можно преобразовать в подлежащее доказательству выражение.

Метод отделения

Этот метод, используя правило отделения, пытается заменить доказываемое выражение новой подзадачей, при решении которой получается доказательство исходного выражения. Таким образом, если доказываемое выражение есть B, метод отделения ищет аксиому или теорему вида «$A \rightarrow B$». Если таковая найдена, A считается новой подзадачей. Теперь если можно доказать A, то, поскольку «$A \rightarrow B$» — теорема, B также будет доказано.

Методы цепеобразования

Эти методы используют транзитивность отношения импликации ($\rightarrow$) для создания новых подзадач, решение которых дает доказательство исходной задачи. Таким образом, если подлежащее доказательству выражение имеет вид «$a \rightarrow c$», метод прямого цепеобразования ищет аксиому или теорему вида «$a \rightarrow b$». Если такая теорема найдена, выражение «$b \rightarrow c$» становится новой подзадачей. Метод обратного цепеобразования работает аналогично: он ищет теорему вида «$b \rightarrow c$», и, если таковая найдена, выражение «$a \rightarrow b$» становится новой подзадачей.

Все эти методы независимы. Они носят альтернативный характер, но их можно использовать последовательно, применяя один метод для решения подзадач, выработанных другими методами. Каждый из них выполняет существенную часть доказательства. Метод подстановки фактически доказывает теоремы, а три других метода вырабатывают подзадачи, которые могут стать промежуточными выражениями в последовательности, являющейся доказательством.

Эти методы не дают гарантии, что они будут успешными. Нет никакой гарантии, что удастся найти теорему, которую можно использовать для получения доказательства методом подстановки, или теорему, которая породит подзадачу любым из трех других методов. Даже если подзадача выработана, нет гарантии, что она является частью последовательности искомого доказательства или даже вообще частью некоторой последовательности, являющейся доказательством (например, эта подзадача может оказаться ложной). Но вместе с тем эти методы гарантируют, что любая выработанная ими подзадача является частью последовательности выражений, которые оканчиваются доказываемой теоремой. (Это одно из условий того, чтобы последовательность была доказательством.) Эти методы гарантируют также, что каждое выражение последовательности получается из предыдущих по правилам вывода (второе условие

того, чтобы последовательность была доказательством). Однако нет никакой гарантии, что выбранное начало последовательности приведет к аксиомам или ранее доказанным теоремам.

Не гарантируется также, что комбинация этих четырех методов, осуществленная каким бы то ни было способом при неограниченном времени и объеме вычислений, даст необходимое множество методов, достаточных для доказательства всех теорем. Действительно, мы обнаружили теорему [(2.13) : «$p \vee \overline{\overline{p}}$»], которую ЛТ не смог доказать с помощью этих четырех методов. Все подзадачи, выработанные для (2.13), начиная с определенного места, становятся ложными и, следовательно, не могут привести к доказательству.

У нас еще нет общей теории, которая объясняла бы, почему эти методы (вместе с кратко описываемыми ниже другими приемами) превращают ЛТ в устройство, эффективно решающее задачи. В оставшейся части статьи мы подробно покажем, что это именно так. По-видимому, здесь существенно несколько обстоятельств. Во-первых, рассмотренные методы организуют последовательности, состоящие из отдельных шагов, в более крупные блоки, которые можно обрабатывать независимо. Каждый шаг может выполняться сообразно с его функцией в блоке, а сами блоки могут служить простыми переменными для более высокого уровня программы.

Следует также указать, что, помимо своей способности к «образованию блоков», методы, вырабатывающие подзадачи, работают, как правило, «назад» — от доказываемой теоремы к аксиомам или известным теоремам, а не «вперед», как это делает алгоритм Британского Музея.

Так как требуется доказать только одну теорему, а известен целый ряд истинных теорем, то эффективность отыскания доказательства в обратном направлении напоминает общеизвестный факт, в соответствии с которым иголка легко может выскочить из стога сена, в то время как найти иголку в стоге сена чрезвычайно трудно.

Управляющая (или исполнительная) программа

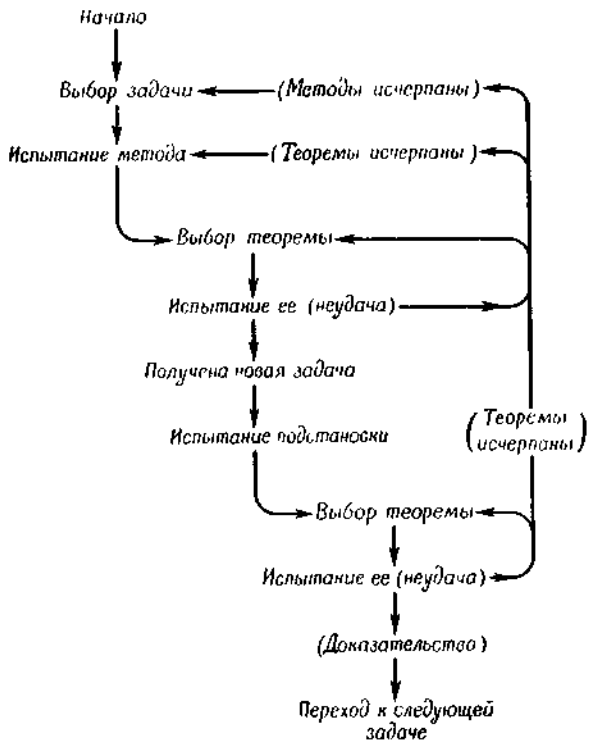
В ЛТ указанные четыре метода управляются специальной программой, блок-схема которой приведена на фиг. 3.

1. Когда ЛТ дается новая задача, сначала испытывается метод подстановки с использованием всех аксиом и известных ЛТ теорем, которые теперь хранятся в *списке теорем*.

2. Если метод подстановки не приводит к доказательству, испытывается метод отделения. Каждое успешное применение метода отделения порождает новую подзадачу. Затем делается

попытка решить ее с помощью метода подстановки. Если это опять не приводит к цели, такую подзадачу заносят в *список подзадач*.

3. Если метод отделения не дает решения при использовании всех теорем из списка теорем, то тот же самый цикл повторяется для метода прямого цепеобразования, а затем для метода



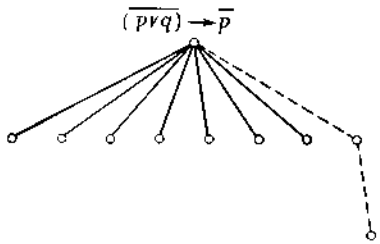
Фиг. 3 Блок-схема ЛТ.

обратного цепеобразования. А именно: пытаются выработать подзадачу, затем пытаются доказать ее с помощью метода подстановки; в случае неудачи новую подзадачу заносят в список подзадач. Из самой природы этих методов вытекает, что, если применение метода подстановки окажется успешным хотя бы для одной подзадачи, исходная теорема оказывается доказанной.

4. Если применение всех методов к исходной задаче оказалось безуспешным, управляющая программа выбирает следующую неиспользованную подзадачу из списка подзадач и выполняет для нее те же операции. Этот процесс продолжается до тех пор, пока: 1) не будет найдено доказательство; 2) не будет израсходовано время, отпущенное на эксперимент; 3) не останется свободного места в памяти машины или 4) в списке подзадач не останется ни одной неиспытанной подзадачи.

В трех примерах, рассмотренных ранее, доказательство (2.01) $[(p \to \bar{p}) \to \bar{p}]$ было получено непосредственно методом подстановки, поэтому список подзадач не использовался.

Доказательство (2.45) $[(\overline{p \vee q}) \to \bar{p}]$ было получено применением метода отделения вслед за методом подстановки. Для



Фиг. 4. Дерево подзадач для доказательства (2.45), проведенного ЛТ (все предыдущие теоремы известны).

проведения этого доказательства ЛТ пришлось выделить подзадачу и применить к ней метод подстановки. ЛТ не потребовалось выделять ни одной подзадачи из списка подзадач, поскольку подстановка была успешной.

На фиг. 4 показано дерево подзадач, соответствующее доказательству (2.45). Подзадачи на нем изображены в виде нисходящих ветвей дерева. Каждый узел на этом дереве соответствует подзадаче, а исходная задача изображается единственной вершиной дерева. Линии, идущие вниз от узла, ведут к новым подзадачам, которые вырабатываются из подзадачи, соответствующей этому узлу. Последовательность доказательства обозначена пунктирной линией. Верхняя ветвь этой линии получена методом отделения, а нижняя — методом подстановки. Другие ветви, исходящие из вершины дерева, ведут к другим подзадачам, выработанным методом отделения (но не доказуемым прямой подстановкой) до того момента, как ЛТ попытался применить теорему, ведущую к окончательному доказательству.

ЛТ, как уже указывалось, не доказал теорему (2.31) и в качестве причины указал, что он больше ничего не может «придумать» для нахождения доказательства. Это означает, что ЛТ рассмотрел все подзадачи из списка подзадач (в данном случае он состоял из шести подзадач) и не было новых подзадач для продолжения работы программ. Ни в одном из упомянутых примеров ЛТ не испытывал недостатка времени или места

в памяти, хотя это наиболее часто встречающийся исход в случаях, когда ЛТ не находит доказательства. Лишь в редких случаях ЛТ в своей работе исчерпывает весь список подзадач.

В этом разделе структура ЛТ описывалась на языке применяемых им методов. Нам остается подробнее рассмотреть вопрос, почему такая структура вместе с дополнительными приемами, которые будут описаны ниже, позволяет ЛТ доказывать теоремы за приемлемое время.

Процедура сравнения

Для доказательства даже простейших теорем алгоритму Британского Музея требуется в 5—100 (и более) раз больше времени, чем ЛТ (причем цифра 5 соответствует только одной конкретной теореме). Рассмотрим пример, взятый из начальной стадии работы алгоритма, относительно которой у нас есть подробная информация. 79-я теорема, выработанная алгоритмом (см. фиг. 2), — это теорема (2.02) «Principia» и одна из теорем, предложенных нами ЛТ для доказательства. Эта теорема $[p \rightarrow (q \rightarrow p)]$ была выработана алгоритмом примерно за 158 *сек* при помощи последовательности подстановок и замен. Эта же теорема была доказана ЛТ приблизительно за 10 *сек* методом подстановки. Причина такого различия становится очевидной, если мы обратим внимание на аксиому (1.3) $p \rightarrow (q \vee p)$, из которой теорема выводится и в той и в другой схеме.

На фиг. 5 показано дерево доказательств первых 12 теорем, полученных алгоритмом из аксиомы (1.3). Теорема (2.02) является узлом *9* на дереве и получается подстановкой $\bar{q}$ вместо q в аксиому (1.3) (при этом мы переходим в узел *5*), а затем, заменяя $(\bar{q} \vee p)$ на $(q \rightarrow p)$ в узле *5*, мы приходим в *9*. Девятая теорема, выработанная из аксиомы (1.3), является 79-й теоремой, выработанной из всех пяти аксиом, рассматриваемых совместно.

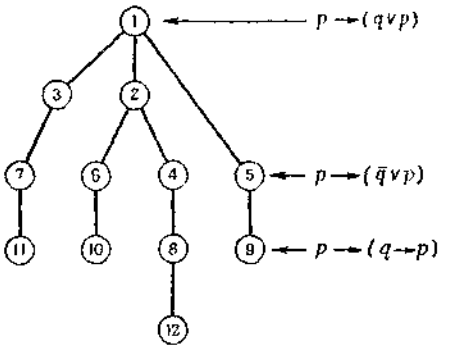
ЛТ непосредственно получает то же доказательство, используя следующую процедуру сравнения. Сравниваем аксиому с выражением, которое надо доказать:

$$p \rightarrow (q \vee p), \tag{1.3}$$

$$p \rightarrow (q \rightarrow p). \tag{9}$$

Во-первых, путем непосредственного сравнения ЛТ определяет, что основные связки совпадают. Во-вторых, ЛТ определяет, что переменные слева от основных связок совпадают.

В-третьих, ЛТ определяет, что связки справа в скобках различны. Необходимо заменить $\vee$ на $\rightarrow$, но для того, чтобы сделать это (в соответствии с определением импликации), необходимо применить отрицание к переменной, предшествующей связке $\vee$. Исходя из этого, ЛТ сначала заменяет в правой части q на $\bar{q}$, т. е. применяет отрицание, в результате чего получается выражение *5*. Теперь ЛТ может изменить $\vee$ на $\rightarrow$ и определяет, что результирующее выражение совпадает с *9*.



Фиг 5 Дерево доказательств для доказательства (2 02), проведенного алгоритмом Британского Музея (с использованием аксиомы (1 3)).

Процедура сравнения позволяет ЛТ непосредственно идти вниз по ветви из *1* через *5* в *9*, даже не исследуя других ветвей. С количественной точки зрения ЛТ просматривает только два выражения вместо восьми, т. е. скорость работы увеличивается в четыре раза. Фактически экономия времени даже больше, поскольку процедура сравнения на каждом шаге имеет дело не с целыми выражениями, а только с парами элементов.

Одна из главных причин эффективности процедуры сравнения лежит в том, что она проходит покомпонентно, так что с помощью обратной связи можно результаты подстановки или замены на каждом шаге использовать для управления следующим шагом. Эта обратная связь позволяет продолжать поиск по правильной ветви дерева возможных выражений. Для эффективного поиска не существенно, чтобы конечная цель была известна уже в самом начале работы; решающим обстоятельством является то, что в процессе поиска постоянно имеются указания

типа «тепло — холодно»[1]. Использование обратной связи приводит к тому, что когда ЛТ требуется произвести замену или подстановку, он, не прибегая к поиску, может сразу, путем прямого сравнения с доказываемым выражением, определить переменную или связку, которую надо подставить или заменить.

До сих пор мы предполагали, что ЛТ с самого начала знает, что надо использовать аксиому (1.3). Без этой информации он будет производить сравнение с каждой аксиомой по порядку, переходя к следующей, если сравнение с предыдущей не приводит к цели. Например, если попытаться сравнить теорему с аксиомой (1.2), то почти сразу (на втором шаге) выясняется, что выражением $p \vee p$ нельзя заменить p при помощи подстановки. Таким образом, процедура сравнения позволяет ЛТ отказаться от невыгодных путей поиска, а также следить за правильностью подстановок и замен.

Сравнение в методе подстановок

Процедура сравнения является существенной частью метода подстановок. Без такой процедуры этот метод является лишь той частью алгоритма Британского Музея, которая использует только подстановки и замены. Используя его, ЛТ может непосредственно или в сочетании с другими методами доказать много теорем при приемлемых затратах. Для получения материала о работе ЛТ ему последовательно были предложены для доказательства первые 52 теоремы «Principia». В каждом случае ЛТ в качестве материала для работы сообщались все аксиомы плюс все ранее доказанные в гл. 2 теоремы (вне зависимости от того, доказал ли он сам эти теоремы)[2].

[1] Здесь можно провести следующую аналогию. Последовательное изменение символов в логическом выражении до получения «правильного» выражения похоже на пробные повороты дисков на замке сейфа до получения комбинации, открывающей сейф. Предположим, что у нас есть два сейфа, причем замок каждого содержит 10 дисков, а на каждом диске имеется 10 цифр. Первый замок дает сигнал («щелчок»), когда какой-либо из его дисков стоит на правильной цифре, а второй замок щелкает только тогда, когда все 10 дисков установлены правильно. Методом случайного поиска (проб и ошибок) первый сейф будет открыт в среднем за 50 проб, а второй — за 5 миллиардов проб.

[2] Вариант ЛТ, который использовался для поиска доказательств 52 теорем, применял испытание на подобие (см. следующий раздел). Так как процедура сравнения важнее испытания на подобие, мы сначала приводим материал, касающийся процедуры сравнения, используя для этого специально отобранную выборку. Представление о размерах выборки можно получить из табл. 1. Из-за ограниченности объема памяти машины JOHNNIAC выборка ограничивалась первыми 52 из общего числа 67 теорем гл. 2 "Principia".

Из 52 теорем ЛТ доказал всего 38 (73%). Для получения этих доказательств им использовались различные комбинации методов, но метод подстановки являлся основной частью всех таких комбинаций. Почти половина доказательств (17) была получена одним лишь методом подстановки. Теоремы, доказанные одним лишь методом подстановки, по нашей субъективной оценке, имеют форму «следствий» из теорем, из которых они выведены; в тексте они расположены довольно близко друг к другу и требуют, как правило, не более трех попыток сравнения для получения одного доказательства (54 попытки на 17 теорем).

Эффективность метода подстановок по отношению к подзадачам значительно хуже, что, по нашему мнению, связано с конкретным порядком, в котором теоремы помещены в «Principia». Из 338 попыток решения подзадач методом подстановок увенчались успехом только 21 (6,2%). Таким образом, вероятность найти доказательство исходной задачи непосредственно методом подстановок составляет приблизительно $^1/_3$, в то время как соответствующая вероятность решения подзадачи, выработанной из исходной задачи, составляет всего лишь около $^1/_{16}$.

Сравнение в методах отделения и цепеобразования

До сих пор процедура сравнения рассматривалась только как часть метода подстановки, но она является также существенной частью трех других методов. Например, в методе отделения ищется теорема вида $A \rightarrow B$, где B совпадает с доказываемым выражением. Вероятность найти такую теорему ничтожно мала, если мы не разрешим изменять в дальнейшем B до совпадения с доказываемым выражением. Следовательно, как только некоторая теорема выбирается из списка теорем, ее правая часть должна сравниваться с доказываемым выражением. Аналогичная процедура используется в методах цепеобразования.

Для оценки методов отделения и цепеобразования мы можем взять те же задачи, которые использовались для оценки метода подстановок. Однако успешное сравнение в первых трех методах позволяет лишь выработать подзадачу, но не дает прямого доказательства теоремы. При каждом применении метода отделения в среднем вырабатывались три новые подзадачи. Для метода прямого цепеобразования среднее число новых подзадач на шаг равно 2,7, а для обратного цепеобразования — 2,2. Для всех методов это дает приблизительно одну подзадачу на каждые 7,5 испытанных теорем (число имевшихся в распоряжении теорем несколько менялось от случая к случаю).

При применении этих трех методов к исходной задаче вероятность успеха, как и в случае метода подстановки, была больше, чем при их применении к подзадачам. При применении их к первоначальной задаче в среднем вырабатывалось 8—9 подзадач, а при применении к выведенным из исходной задачи подзадачам число новых подзадач падало в среднем до 2—3.

При обработке первых 52 задач гл. 2 «Principia» 17 теорем были доказаны за один шаг, т. е. при однократном применении подстановки. 19 теорем были доказаны за два шага: 12 методом отделения после применения подстановки, а 7 методом прямого цепеобразования вслед за подстановкой. Еще две теоремы были доказаны за три шага. Следовательно, всего было доказано 38 теорем. Не было двухшаговых доказательств методом обратного цепеобразования, так как двухшаговые доказательства методом обратного цепеобразования являются одношаговыми для метода прямого цепеобразования. В 14 случаях ЛТ не нашел доказательства. Большинство из этих попыток было неудачным из-за ограничений по времени и памяти. Мы знаем, что одну из этих 14 теорем ЛТ вообще не может доказать, и полагаем, что он не может доказать также еще одну теорему. ЛТ может доказать большинство из оставшихся 12 теорем при наличии достаточного машинного времени и достаточного объема памяти (см. раздел о подзадачах).

Испытание на подобие

Процедура сравнения настолько сокращает необходимое число проб и ошибок при подстановках и заменах, что позволяет ЛТ успешно решать задачи. Процедура сравнения содержится во всех методах, и без нее ни один из них не давал бы удовлетворительных результатов за практически приемлемое машинное время. Однако в программе ЛТ значительные усилия затрачиваются на поиск правильных теорем, с которыми производится сравнение. Возвращаясь к работе ЛТ при доказательстве теорем гл. 2, мы находим, что из общего числа сравнений успешными были 0,3% в случае подстановок, 13,4% в случае отделений, 13,8% при прямом цепеобразовании и 9,4% при обратном цепеобразовании.

Введением процесса отбора, отбрасывающего без сравнения все те теоремы, для которых успешное сравнение маловероятно, можно сократить поиск в списке теорем. У ЛТ есть такой процесс отбора, называемый *испытанием на подобие*. По определению, два логических выражения подобны, если и в правых и в левых их частях одинаковы: 1) максимальное число уровней от главной связки к любой переменной, 2) количество различных переменных и 3) общее число вхождений переменных. Интуитивно два логических выражения «подобны», если они выглядят похожими, и выглядят похожими, если они подобны.

Рассмотрим пример:

$$(p \vee q) \rightarrow (q \vee p), \quad (1)$$

$$p \rightarrow (q \vee p), \quad (2)$$

$$r \rightarrow (m \rightarrow r). \quad (3)$$

По определению подобия, (2) и (3) подобны, а (1) не подобно ни (2), ни (3).

Во всех методах ЛТ применяет испытание на подобие ко всем выражениям, подлежащим сравнению, и использует для сравнения только те выражения, которые подобны доказываемому; в противном случае ЛТ переходит к следующей теореме в списке теорем. Как показано в табл. 1, испытание на подобие существенно уменьшает число попыток сравнения и соответственно увеличивает вероятность успешного сравнения. Особенно сильно этот эффект выражен в случае метода подстановок, для которого применение испытания на подобие на порядок уменьшает число попыток сравнения и соответственно на порядок увеличивает вероятность успешного сравнения. Для других методов число попыток сравнения уменьшается приблизительно в 4—5 раз и во столько же раз увеличивается вероятность успешного сравнения.

Таблица 1

Статистика испытаний на подобие и сравнение

Метод	Общее число рассматриваемых теорем	Число подобных теорем	Число сравнимых теорем	Доля подобных среди всех рассматриваемых теорем, %	Доля сравнимых среди всех подобных теорем, %
Подстановка	11 298	993	37	8,8	3,7
Отделение	1 591	406	210	25,5	51,7
Прямое цепеобразование	869	200	120	23,0	60,0
Обратное цепеобразование	673	146	63	21,7	43,2

Эти цифры показывают, что в среднем ЛТ получает большой выигрыш от использования испытания на подобие. Однако такой выигрыш получается не всегда. Существуют две причины, по которым весь этот большой выигрыш нельзя реализовать.

Во-первых, испытание на подобие является лишь эвристикой. Оно не дает гарантии того, что выбираются только те выражения, которые затем окажутся сравнимыми. Испытание на подобие также не дает гарантии того, что оно не отбрасывает сравнимых выражений. Как будет показано ниже, испытание на подобие редко совершает ошибки последнего типа (соответствующие статистическим ошибкам второго рода). Однако даже редкое появление подобных ошибок может дорого обойтись. Так случилось при доказательстве теоремы (2.07):

$$p \rightarrow (p \vee p). \qquad (2.07)$$

Эта теорема доказывается просто подстановкой p вместо q в аксиоме (1.3):

$$p \rightarrow (q \vee p). \qquad (1.3)$$

Однако так как испытание на подобие требует равенства числа различающихся переменных в правой части, а (2.07) и (1.3) содержат различное их число [1 для (2.07) и 2 для (1.3)], то (2.07) и (1.3) ЛТ назвал неподобными. ЛТ находит доказательство этой теоремы при помощи прямого цепеобразования, при котором он перед созданием новой подзадачи делает проверку на прямую сравнимость. Но доказательство получается примерно в пять раз длиннее, чем в случае, когда не делается испытания на подобие.

Вторая причина, почему нельзя полностью реализовать этот большой выигрыш, заключается в том, что испытание на подобие делается не «бесплатно». На деле для теорем, проходящих это испытание, затраты, связанные с испытанием на подобие, должны складываться с затратами на проведение сравнения. В следующем разделе, посвященном анализу затрат машинного времени при работе ЛТ, мы исследуем эти затраты.

Были проведены эксперименты с использованием и более слабого понятия подобия, в которых сравнивалось только число различных вхождений в обеих частях выражений. При этом испытании не совершаются упомянутые выше ошибки второго рода, и теорема (2.07) с использованием этого испытания доказывается подстановкой. Однако в остальном такая модификация весьма незначительно отразилась на результатах работы ЛТ. На примере 10 задач она дала лишь на 10% больше подобных теорем и приблизительно на 10% больше подобных подзадач. Причина, по которой эти два испытания не отличаются более основательно, состоит в том, что среди критериев, используемых для определения подобия, существует большая корреляция.

Затраты на работу ЛТ

До сих пор наше внимание было сосредоточено исключительно на характеристиках работы эвристик в ЛТ и не указывалось на огромную разницу в затратах, которые требуются для работы ЛТ и алгоритма Британского Музея. Однако ясно, что каждое дополнительное испытание, поиск, описание и т. п., обеспечивая определенное улучшение характеристик, дают и свой вклад в общие затраты. Затраты должны всегда быть сбалансированы с преимуществами, ибо всегда есть варианты эвристик, которые можно добавить в программу вместо используемых. В этом разделе мы проанализируем затраты на работу ЛТ. Требуемый объем памяти также дает свой вклад в общие затраты, но здесь мы на этом вопросе останавливаться не будем.

Измерение затрат

Программа ЛТ записана на «интерпретирующем языке» или «псевдокоде», который описывается в другой статье этого сборника[1]. ЛТ определяется на языке некоторой системы простых операций, которые в свою очередь определяются подпрограммами, написанными на языке машины JOHNNIAC. Такая простая операция дает удобную единицу измерения затрат, и все численные значения затрат у нас будут выражаться общим числом выполненных простых операций. Относительные частоты применения простых операций можно считать постоянными, и потому общее число выполненных простых операций адекватно отражает затраты. Среднее время на выполнение простой операции достаточно постоянно и приблизительно равно 30 *мсек*, хотя при очень небольшом общем числе их (менее 1000 простых операций) это число, по-видимому, составляет около 20 *мсек*.

Затраты и характеристики работы

A priori можно ожидать, что затраты, требуемые для решения логической задачи, грубо говоря, пропорциональны общему числу теорем, исследуемых в ходе решения различными методами (например, испытанных на подобие, если оно применяется, или использованных в процедурах сравнения в противном случае). Действительно, эта зависимость достаточно хорошо отражает затраты, но оценка еще улучшится, если мы присвоим теоремам, к которым применяются методы отделения и цепе-

[1] См стр 458 — *Прим ред.*

образования, больший вес, чем теоремам, к которым применяется метод подстановки.

Ниже сравниваются действительные затраты и их оценка, полученная в предположении, что число простых операций при применении метода цепеобразования в два раза больше, а при применении метода отделения в три раза больше, чем в случае применения метода подстановок. При этом включены все случаи полного испытания на подобие и не включены теоремы, доказанные подстановкой. Около 45 простых операций приходится на теорему, доказываемую методом подстановки (следовательно, 135 для метода отделения и 90 для методов цепеобразования). Как показано в табл. 2, оценки большей частью совпадают с действительными затратами с точностью до нескольких процентов, за исключением теоремы (2.06), для которой оценка является слишком низкой.

Таблица 2

Статистика затрат при методе предварительного запоминания описаний

Теорема	Полное число простых операций, тыс.	
	эксперимент	оценка
(2.06)	3,2	0,8
(2.07)	4,3	4,4
(2.08)	3,5	3,3
(2.11)	2,2	2,2
(2.13)	24,5	24,6
(2.14)	3,3	3,2
(2.15)	15,8	13,6
(2.18)	34,1	35,8
(2.25)	11,1	11,5

Существует дополнительный источник ошибок, не имеющий места в теоремах, перечисленных в табл. 2. Описания (определения), применяемые при испытании на подобие, должны быть выражены в коде логических выражений. Поскольку описания теорем используются неоднократно, ЛТ кодирует их в начале решения задачи и запоминает вместе с теоремами, так что теперь нет необходимости их заново кодировать. Однако по мере возрастания числа теорем в памяти машины не остается свободного места для хранения заранее закодированных описаний,

и ЛТ переходит на режим работы, когда они кодируются всякий раз, как потребуется. В этом случае затраты на решение задачи все еще грубо пропорциональны общему числу рассматриваемых теорем, но теперь число простых операций на теорему составляет около 70 для метода подстановок, 210 для отделения и 140 для цепеобразования.

Таким образом, наш статистический анализ затрат показывает, что в первом приближении затраты, требуемые для доказательства теоремы, пропорциональны числу теорем, которые необходимо рассмотреть для нахождения доказательства. Число рассматриваемых теорем является мерой затрат для оценки эвристики. Хорошая эвристика, обеспечивая рассмотрение «правильных» теорем в самом начале доказательства, уменьшает ожидаемое число теорем, которые должны быть рассмотрены прежде, чем будет найдено доказательство.

Оценка испытания на подобие

Как было отмечено в предыдущем разделе, для оценки улучшенной эвристики необходимо учитывать любые дополнительные вычисления, появляющиеся при использовании этой эвристики. Суммарный выигрыш может оказаться меньше, чем оценка, получаемая без такого учета; более того, дополнительные затраты могут фактически свести на нет преимущество большей избирательности. Мы собираемся ниже оценить процедуру испытания на подобие, предварительно выделяющую теоремы для применения процедуры сравнения.

С этой целью испытывался ряд теорем: сначала применялось полное испытание на подобие, затем — модифицированное (когда рассматривалось только число вхождений переменных) и наконец программа работала без испытания на подобие. Мы сравнили также результаты работы программы в случаях, когда описания кодируются заранее и когда они обрабатываются по мере необходимости.

Когда описания заготавливаются заранее, затраты при применении полного испытания на подобие меньше, чем без применения такого испытания. Экономия составляет от 10 до 60% (например, для теоремы (2.08)—3534/5206). Однако, если ЛТ приходится каждый раз заново кодировать описание, затраты с применением полного испытания на подобие становятся больше, чем при отсутствии проверки на подобие (например, для теоремы (2.45)—26 739/22 914).

Модифицированное испытание на подобие оказывается несколько лучшим. Например, при доказательстве теоремы (2.45)

потребовалось только 18 035 простых операций, в то время как в случае, когда испытание на подобие не производилось, их потребовалось 22 914 (см. выше).

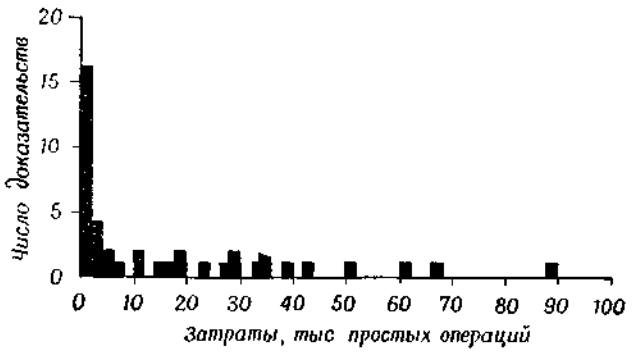
Таким образом, испытание на подобие дает минимальное преимущество и не дает улучшения, даже грубо сравнимого с тем, которое получается от процедуры сравнения, хотя мы видели (табл. 1), что характеристики эффективности сулили более существенные улучшения. Нетрудно найти причину такого расхождения. Процедура сравнения состоит в известном смысле из двух частей: первая устанавливает различия между элементами и определяет, какую корректирующую операцию надо выполнить, вторая же включает процессы подстановки и замены. В работе по сравнению большинство затрат падает на вторую часть, но если сравнение не удается, то бóльшая часть затрат экономится. Таким образом, оказывается, что процесс сравнения получается «дешевым» как раз для тех выражений, которые отбрасываются при испытании на подобие.

Возникающие проблемы

ЛТ может доказывать большое количество теорем символической логики. Однако есть много теорем, которые ЛТ не в состоянии доказать, и можно утверждать, что ЛТ достиг предела своей способности решать задачи.

На фиг. 6 показаны затраты, требуемые ЛТ для решения первых 52 задач из гл. 2 «Principia». Почти все доказательства, найденные ЛТ, содержат менее 30 000 простых операций. Среди многочисленных попыток, которые по затратам вышли за этот предел, только несколько оказались успешными, и они потребовали в сумме намного больше затрат, чем все остальные задачи. Для преобладающей части хороших доказательств потребовалось значительно меньше чем 30 000 простых операций. Для каждого доказательства, полученного методом подстановок (а таких около половины), требуется не более 1000 простых операций. Для самого длинного доказательства потребовалось 89 000 простых операций, для самого короткого — 250. По нашим оценкам, для доказательства дополнительных 12 теорем, которые, как мы полагаем, ЛТ сможет доказать, предельные затраты должны быть порядка миллиона простых операций.

Из этого следует вывод, что возможности ЛТ как устройства, решающего задачи, существенно ограничиваются задачами определенного класса. Хотя при увеличении допустимых затрат ЛТ в принципе сможет решать и другие задачи, все же для распространения возможностей ЛТ на существенно новые классы задач необходимо внести существенные изменения в его программу. Нам эта ситуация представляется типичной: хорошие эвристики дают улучшение характеристики работы на несколько порядков, но неизбежно достигается «потолок», который можно превзойти только при помощи существенно иных эвристик. Эти эвристики в свою очередь снова улучшат характеристики на несколько порядков. В следующем разделе мы проанализируем те



Фиг 6. Распределение доказательств ЛТ в зависимости от затрат.

Данные включают все успешные доказательства для первых 52 теорем гл. 2 «Principia Mathematica».

трудности, с которыми встретился ЛТ в теоремах, которые он не смог доказать. Там же будет указан общий тип эвристик, которые могут расширить диапазон эффективности ЛТ.

Дерево подзадач

Проанализируем доказательство теоремы (2.17) в предположении, что все предыдущие теоремы известны. Это именно то доказательство, для которого ЛТ потребовалось 89 000 простых операций. Воспроизводимое ниже доказательство использует в качестве правила вывода цепеобразование (каждый шаг в цепеобразовании можно рассматривать как двукратное применение отделения в строгом соответствии с системой «Principia»).

$(\bar{q} \to \bar{p}) \to (p \to q)$ (теорема (2.17), которую требуется доказать);

(1) $A \to \bar{\bar{A}}$ (теорема (2.12));

(2) $p \to \bar{\bar{p}}$ (подстановка в (1) p вместо A);

(3) $(A \to B) \to [(B \to C) \to (A \to C)]$ (теорема (2.06));

(4) $(p \to \bar{\bar{p}}) \to [(\bar{\bar{p}} \to q) \to (p \to q)]$ (подстановка в (3) p вместо A, $\bar{\bar{p}}$ вместо B и q вместо C);

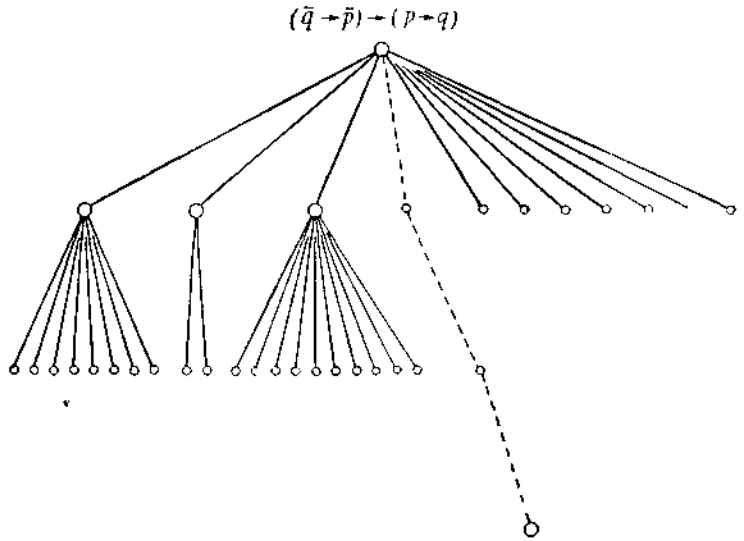
(5) $(\bar{\bar{p}} \to q) \to (p \to q)$ (отделение (4) от (3));

(6) $(\bar{A} \to B) \to (\bar{B} \to A)$ (теорема (2.15));

(7) $(\bar{q} \to \bar{p}) \to (\bar{\bar{p}} \to q)$ (подстановка q вместо A и $\bar{p}$ вместо B);

(8) $(\bar{q} \to \bar{p}) \to (p \to q)$ (цепь (7) и (5)),

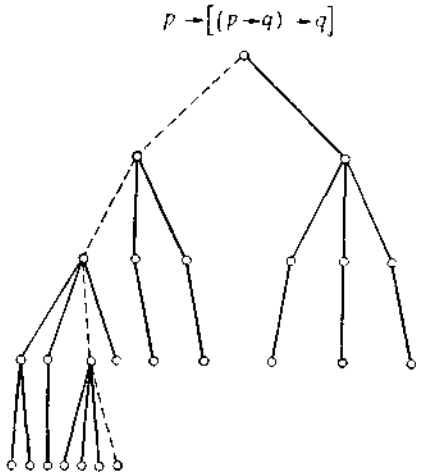
что и требовалось доказать.



Фиг. 7. Дерево подзадач для доказательства (2.17), проведенного ЛТ (все предыдущие теоремы известны).

Это доказательство длиннее, чем любое из двух приведенных ранее. На языке методов ЛТ оно ведется тремя методами вместо двух или одного: прямым цепеобразованием, отделением и подстановкой. Это приводит к естественному выводу, известному из человеческого опыта, что длина доказательства является важной характеристикой при определении общих затрат: короткое доказательство будет легким, а длинное — трудным, причем трудность будет возрастать с увеличением длины доказательства быстрее, чем линейно. В самом деле, все одношаговые доказательства требуют от 500 до 1500 простых операций, в то время как число простых операций, требуемое для двухшаговых доказательств, находится в пределах 3000—50 000. Далее, ЛТ получил только шесть доказательств длиннее двухшаговых, и они требуют от 10 000 до 90 000 простых операций.

Значение длины доказательства можно увидеть при сравнении фиг. 7, на которой показано дерево доказательства (2.17), с фиг. 4, на которой изображено дерево двухшагового доказательства для (2.45). Чтобы сделать один шаг вниз по дереву ЛТ в случае (2.17), требуется выработать и проверить намного больше подзадач. Детальное изучение этих доказательств



Фиг. 8 Дерево подзадач для доказательства (2.27), проведенного ЛТ (с использованием аксиом).

позволяет высказать следующее утверждение: в других отношениях (например, по затратам на каждую рассматриваемую теорему) эти задачи можно считать довольно сходными. Поэтому различие в общих затратах можно в основном отнести за счет разницы в числе вырабатываемых подзадач.

Рассмотрим еще несколько фактов, подкрепляющих такой вывод. На фиг. 8 показано дерево подзадач для доказательства (2.27) из аксиом. Оно является единственным четырехшаговым доказательством среди всех доказательств, сделанных ЛТ. При взгляде на это дерево сразу становится ясным, почему ЛТ смог найти указанное доказательство. Вместо того чтобы образовывать в каждом узле множество ветвей и тем самым быстро увеличивать число подлежащих просмотру подзадач, ЛТ в этом

случае вырабатывал в каждом узле только несколько подзадач. Таким путем ему удалось за приемлемое число простых операций (38 367) проникнуть на глубину четырех шагов. Если бы это дерево «ветвилось» так же, как и два других, то для получения доказательства ЛТ потребовалось бы обработать около 250 подзадач. Для этого понадобилось бы выполнить по крайней мере 250 000 простых операций. Приведенные ранее статистические данные по скорости выработки подзадач подтверждают общую гипотезу о том, что число исследуемых подзадач возрастает более или менее экспоненциально с глубиной доказательства.

Трудность состоит в том, что ЛТ использует алгоритмическую процедуру для управления процессом выработки подзадач. За исключением нескольких подзадач, которые исключаются при испытании на подобие из-за ошибок второго рода, эта процедура гарантирует, что все подзадачи, которые можно получить с помощью отделения и цепеобразования, действительно будут получены (дубликаты отбрасываются). ЛТ также использует некоторый алгоритм для определения порядка, в котором он будет пытаться решать подзадачи. Подзадачи рассматриваются в том порядке, в котором они вырабатываются, так что из неудачного рассмотрения некоторой выработанной подзадачи вовсе не следует, что доказательство не будет получено.

Ввиду такой «систематичности» управляющей программы и поскольку методы, примененные к списку теорем, содержащему в среднем 30 выражений, вырабатывают большое число подзадач, ЛТ должен находить «редкую» последовательность, ведущую к доказательству, путем поиска среди очень большого числа таких последовательностей. Для одношаговых доказательств это вообще не проблема. Для двухшаговых доказательств множество, в котором надо производить поиск, имеет еще приемлемое число членов относительно допустимого объема вычислений. Для трехшаговых доказательств требуемый для поиска объем вычислений находится на границе допустимого для ЛТ. А если добавить еще один или два шага, то объем вычислений, требуемый для поиска, превысит практически любые приемлемые границы.

Множество подзадач, вырабатываемых ЛТ, каким бы большим оно не показалось, является чрезвычайно избирательным и богатым доказательствами по сравнению с множеством, на котором ищет доказательства алгоритм Британского Музея. Следовательно, последний алгоритм за приемлемое время может находить доказательства только для простейших теорем, в то время как ЛТ находит доказательства для значительно большего числа теорем. Граница, отделяющая возможное от невозможного для любой заданной процедуры решения задач, является относительно резкой. Поэтому для дальнейшего улучшения процесса решения задач, сравнимого с тем, которое получается при переходе от алгоритма Британского Музея к ЛТ, потребуется соответствующее усовершенствование эвристик.

Модификация ЛТ

Существует много возможных способов так изменить ЛТ, чтобы он смог находить более сложные, чем двухшаговые, доказательства, причем эти способы основаны на определенных соображениях интуиции, а не на «грубой силе». Во-первых, может быть существенно снижена стоимость обработки подзадач, благодаря чему при заданных затратах можно будет обработать намного больше подзадач (не изменяя характера процесса как процесса «грубой силы»; это делает ее, однако, выполнимой с точки зрения затрат). Во-вторых, ЛТ можно так видоизменить, чтобы он выделял для обработки только те подзадачи, которые с высокой вероятностью ведут к доказательству. Один из способов сделать это — просматривать подзадачи перед занесением их в список подзадач и отбрасывать все подзадачи с малой вероятностью доказательства. Другой путь состоит в выборочном уменьшении числа вырабатываемых подзадач.

Для уменьшения числа вырабатываемых подзадач можно, например, ограничить число теорем, используемых для их выработки. На то, что такой подход может оказаться эффективным, наводит хотя бы приведенная выше статистика, которая показывает, что число подзадач, вырабатываемых некоторым методом на одну исследуемую теорему, относительно постоянно (оно составляет около одной подзадачи на 7 теорем).

Представление о том, как число теорем в списке влияет на выработку подзадач, можно получить, сравнивая деревья доказательств теорем (2.17) (фиг. 7) и (2.27) (фиг. 8). «Широкое» дерево для (2.17) получено из списка, содержащего 20 теорем, в то время как «глубокое» дерево для (2.27) получено из списка, содержащего только 5 теорем. При каждом применении какого-либо метода меньший список теорем порождал меньше подзадач.

Другим примером того же обстоятельства могут служить два доказательства теоремы (2.48), полученные с различными списками допустимых теорем. В первом случае доказательство (2.48) велось со списком, в который были включены все теоремы вплоть до теоремы (2.48). Во втором случае этот список

включал только аксиомы и теорему (2.16). Нам казалось правдоподобным, что второе доказательство должно быть более трудным, так как в этом случае приходится построить более длинную цепь доказательства. Однако это предположение оказалось ошибочным. В первом случае ЛТ доказал теорему (2.48) за два шага, затратив 51 450 простых операций. Во втором случае ЛТ доказал эту же теорему за три шага, но затратил только 18 558 простых операций, что составляет $^1/_3$ предыдущих затрат. Исследование первого доказательства показывает, что значительная доля затрат при обработке уходит на теоремы, «не имеющие отношения к делу». Это сравнение красноречиво свидетельствует о том, что устройство, решающее задачи, может быть в той же мере «захлестнуто» потоком слишком обильной информации, как и «обессилено» в результате ее недостаточности.

Мы только вскользь коснулись возможностей видоизменения ЛТ и сделали некоторые предположения относительно их потенциальной эффективности, содержащиеся в нынешнем поведении ЛТ. Все указанные ранее предложения, по-видимому, открывают достойные изучения возможности для усовершенствования программы ЛТ. Мы надеемся, что сумеем в свое время сообщить о результатах этих исследований.

Заключение

В этой статье мы привели данные о работе сложной системы обработки информации, которая в состоянии находить доказательства теорем элементарной символической логики. Мы использовали эти результаты для анализа и иллюстрации различия между систематическими, алгоритмическими процессами, с одной стороны, и эвристическими процессами, или процессами решения задач, — с другой. Показано, что эвристики позволяют программе за приемлемое машинное время решать задачи, которые могли бы быть решены алгоритмическими методами лишь ценой многолетних усилий. Наконец, были оценены границы возможностей нынешней программы ЛТ и указано несколько направлений ее усовершенствования с целью расширения этих возможностей применительно к задачам более высокого уровня сложности.

Наши исследования машины «Логик-теоретик» являются звеном в общей программе исследования сложных систем переработки информации. Цель этой программы — разработка теории подобных систем и применение ее к таким областям, как составление машинных программ, обучение людей и решение задач.

4. РЕАЛИЗАЦИЯ МАШИНЫ, ДОКАЗЫВАЮЩЕЙ ГЕОМЕТРИЧЕСКИЕ ТЕОРЕМЫ

Г. Гелернтер

Введение

Те, кто видел, как современные быстродействующие вычислительные машины обрабатывают огромный объем информации быстрее, чем мы успеваем подумать об этом, едва ли в состоянии подавить в себе мысли о том, что́ такие машины сулят нам в будущем. В исследовательском центре фирмы IBM в Нью-Йорке была разработана программа для современной вычислительной машины, предназначенная для изучения «разумного» поведения и обучения машин [363]. При этом предполагалось, что машина при решении предлагаемых ей задач работает на пределе своих возможностей. В частности, подробно исследуется возможность применения методов эвристического программирования для решения с помощью цифровых вычислительных машин трудного класса задач, которые в настоящее время считаются для них недоступными, а именно задач, для решения которых кажется необходимым вмешательство человеческого разума и интуиции. Трудно более четко охарактеризовать такие задачи, если не считать туманных замечаний, что они требуют сложных процессов принятия решений в потенциально бесконечной и неконтролируемой среде.

Однако, если ограничиться такими задачами, которые сводятся к доказательству теорем в некоторой хорошо определенной формальной системе, можно четко выявить характерные особенности этого специального круга задач.

Нам хотелось бы, чтобы наша машина могла доказывать теоремы, представленные ей в формальной системе, которая является алгоритмически *неразрешимой*. Далее, следует ожидать, что по мере того, как машина будет приобретать «опыт» в доказательстве теорем, она научится доказывать теоремы, которые раньше были ей не под силу.

Требование, чтобы машина имела дело с неразрешимыми системами, накладывает фундаментальное ограничение на ее способ действия. Обычная методика решения задач на цифровых машинах, т. е. поиск подходящего алгоритма, теперь уже неприемлема по той простой причине, что такого алгоритма, вообще говоря, не существует. Было показано [789], что даже для такого простого раздела логики, как исчисление высказываний, требуется слишком много времени для нахождения доказательства алгоритмическим методом, так как при этом приходится прово-

дить исчерпывающий поиск среди аксиом и ранее доказанных теорем в сочетании с полным развитием «доказывающей последовательности» путем систематического применения правил преобразования до достижения требуемого доказательства. О решении таким способом задач из более интересных разделов логики заведомо не может быть и речи. Поэтому остается единственный путь — создать машину, действующую на основе эвристических методов, которые обычно используются человеком в подобных ситуациях.

Эвристические методы

Эвристический метод означает применение для нахождения решения рассматриваемой конкретной задачи некоторой частной, интуитивно оправданной процедуры. То, что математик обычно использует эвристические методы, хотя бы в довольно простой форме, вполне очевидно. Для ознакомления с ролью эвристики в математическом творчестве читателю можно рекомендовать обратиться к блестящей двухтомной работе Пойа [870]. Машина, которая работала бы на основе полного набора принципов, указанных Пойа, оказалась бы превосходным устройством для решения математических задач и знаменовала бы собой большой шаг вперед на пути удовлетворения условий Тьюринга для машины, успешно справляющейся с «игрой в имитацию» [1085]. Однако создание такой машины — дело неопределенного будущего, ибо в настоящее время искусство обучения машин находится еще на слишком примитивной стадии для того, чтобы можно было говорить о переводе принципов Пойа на машинный язык.

В качестве примера задачи, более соответствующей современному состоянию вычислительной техники, мы выбрали задачу отыскания доказательств теорем элементарной эвклидовой геометрии на плоскости на уровне, скажем, старшеклассника средней школы. Эта задача в относительно чистом виде уже содержит все те трудности, с которыми нам придется встретиться на пути к поставленной выше цели. Следует отметить, что, хотя для планиметрии существует алгоритм решения, доказательства, полученные машиной, будут иной природы. Развитые ею методы окажутся столь же применимы и к решению задач в системах, в которых такого алгоритма решения не существует.

Если мы ограничили сферу наших исследований машинами, имеющими дело лишь с формальными системами, то для этого имеются достаточные основания. Во-первых, при этом четко определены как сама задача, так и то, что следует считать ее решением. Во-вторых, четко указана конечная цель — создание

эффективного устройства для доказательства теорем в некоторой неразрешимой системе. Наконец, подобно тому, как операции над числами в арифметике составляют основу работы современных вычислительных машин, так и операции с символами в формальных системах, вероятно, будут в будущем служить основой работы более сложных «машин для решения задач». Кажется очевидным, что поскольку задачи, представляющие наибольший интерес для людей, в большинстве случаев совершенно не формализованы, их необходимо выразить на языке некоторой формальной системы для того, чтобы машина могла с ними работать[1].

В таком случае «задача» предстает перед нами как некое утверждение (или цепь утверждений), выраженное в некоторой формальной логической системе. Решением задачи будет последовательность утверждений, каждое из которых содержит цепочку символов из алфавита этой системы. Последняя цепочка в решении будет собственно задачей, а первая всегда будет либо аксиомой, либо ранее доказанной теоремой в этой системе. Любая другая цепочка может быть непосредственно выведена из некоторой предшествующей системы цепочек либо сама будет аксиомой или ранее доказанной теоремой[2]. Задачей машины является выбор из запаса аксиом и теорем тех, которые могут служить основой доказательства, и вывод из них недостающих цепочек, необходимых для завершения доказательства.

В некотором смысле задача доказательства теорем имеет особенно простую природу. Как только найдена последовательность выражений, которая удовлетворяет тесту на наличие доказательства теоремы (такой тест всегда существует), можно, так сказать, «поставить точку» на этой задаче, если только не предъявляется никаких требований к «изяществу» доказательства. Но, как следует из работы Ньюэлла, Шоу и Саймона [789], любой современной вычислительной машине для доказательства более или менее простой десятишаговой геометрической теоремы путем полного перебора приводящих к доказательству последовательностей потребуется время порядка нескольких тысяч лет. Поэтому совершенно очевидно, что необходима методика вырабатывания последовательностей, которые с гораздо более высокой априорной вероятностью являются решением задачи,

[1] Критику некоторых попыток формализации нематематических научных теорий можно найти в работе [291].

[2] Машина будет использовать теорему дедукции, чтобы получить

$$\vdash \{\text{Посылки}\} \supset \{\text{Выводы}\}$$

из

$$\{\text{Посылки}\} \vdash \{\text{Выводы}\}.$$

нежели последовательности, получаемые с помощью алгоритма полного перебора.

Как и «Логик-теоретик» Ньюэлла, Шоу и Саймона, «геометрическая машина» для достижения этой цели опирается на хорошо известный аналитический метод. Метод «обратного просмотра» гарантирует машине, что каждая рассматриваемая ею последовательность действительно оканчивается на требуемом утверждении. Однако само по себе это без использования дополнительных эвристик не представляет значительного улучшения по сравнению с полным перебором, ибо преимущества метода «обратного просмотра» покупаются дорогой ценой, так как нет никакой гарантии, что каждая вырабатываемая последовательность, оканчивающаяся нужным образом, вообще служит доказательством чего-либо. Действительно, большинство цепочек, полученных таким способом, будут ложными! Но именно в этом и состоит сила аналитического метода, так как, если бы оказалось возможным найти способ обнаружения ложных последовательностей, такие последовательности можно было бы немедленно отбрасывать, что позволило бы отсекать от богато разветвленного дерева решений «сухие ветви». Набор последовательностей, вырабатываемых таким процессом, содержал бы на любой глубине поиска на несколько порядков меньше членов, и плотность возможных доказательств теоремы среди этих последовательностей была бы соответственно выше.

Именно в этом кроются дополнительные возможности геометрической машины в поиске доказательства теорем, необходимые для такой сложной формальной системы. Они не были нужны, а потому и не искались в машине «Логик-теоретик» Ньюэлла, Шоу и Саймона (см. Пойа [870]), предназначенной для исчисления высказываний.

Подобно математику, геометрическая машина использует богатые эвристические свойства чертежа для лучшего различения истинных и ложных последовательностей. Хотя чертеж используется машиной и для других целей, уже одна эвристика «отбрось как ложные любые утверждения, не удовлетворяющие чертежу» позволяет машине доказать большой класс интересных теорем, часть которых содержит некоторые тривиальные построения.

Перед тем как перейти к детальному рассмотрению структуры геометрической машины, мы сделаем замечание о двух основных (почти очевидных) принципах, которыми необходимо руководствоваться при выборе эвристик для любой машины, решающей задачи. Эвристика в значительной степени может быть уподоблена фильтру, помещенному между генератором решений и устройством, оценивающим полученные решения для заданного класса задач. Первое требование, налагаемое на такой фильтр, вытекает из того факта, что его введение в систему никогда не является бесплатным. Поэтому он должен быть достаточно «избирательным» по отношению к различным генерируемым последовательностям, чтобы приводить к общему выигрышу в эффективности при решении задач. Во-вторых, эвристика может вообще исключить из рассмотрения определенный ряд (если не все) последовательностей, которые на деле являются краткими и изящными решениями некоторых возможных задач из класса задач, для решения которых машина предназначена. Поэтому фильтр должен быть тщательно согласован с подклассом таких задач из данной области, которые считаются «интересными» и по этой причине могут быть предложены машине. Для заданного класса эвристик компромисс между такими существенно противоположными требованиями в сильной степени зависит от организации и мощности вычислительной машины и при определенных, довольно легко возникающих условиях может стать критическим. В случае экспериментов с «Логиком-теоретиком»[1], в которых изменялась «сила» конкретной эвристики (испытание на подобие), оказалось, что оптимальная «пропускная способность» такой эвристики заметно менялась в зависимости от длины задачи и числа уже накопленных в памяти теорем вследствие ограниченной емкости памяти машины

Геометрическая машина

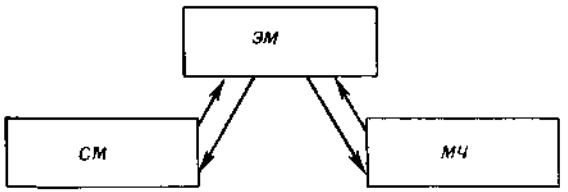
После того как был четко определен объект нашей программы исследований, оставались различные возможности для выбора тестовой задачи, помимо доказательства теорем в эвклидовой геометрии на плоскости Такой тестовой задачей, например, могло бы служить вычисление неопределенных интегралов или доказательство теорем функционального анализа. Решающим доводом в пользу выбора теорем геометрии послужила большая эвристическая ценность чертежа У творчески мыслящего ученого наиболее ценные идеи относительно путей решения задачи обычно возникают при рассмотрении некоторой модели формальной системы, в которой сформулирована задача. В случае эвклидовой геометрии семантическая интерпретация при решении задачи настолько полезна, что на практике обычно никто

[1] Создатели «Логика-теоретика» знали о таком свойстве эвристики В последующем варианте этой машины они фактически ввели некоторые синтаксические эвристики для устранения ложных подцелей Использование для этой цели семантической интерпретации исчисления высказываний (например, истинностной таблицы) превратило бы «Логика-теоретика» в тривиальное устройство.

не пытается доказать теоремы в этой системе, не набросав предварительно какой-то чертеж, если не на бумаге, то хотя бы мысленно.

Численное представление задачи с целью устранения возможных случайных совпадений на чертеже обычно гарантирует обобщение любого утверждения в данной формальной системе, правильно описывающей чертеж, за исключением утверждений, касающихся неравенств. Кроме того, когда человек решает теоретические или экспериментальные задачи, геометрия дает ему иллюстративный материал. В любом случае элементарная эвклидова геометрия понятна представителям всех отраслей науки, которым мы пожелали бы сообщить наши результаты. Наконец, нетрудно будет обобщить нашу машину так, чтобы она охватывала более интересный случай неэвклидовой геометрии. Программы с теми же возможностями в области доказательства теорем, что и наша программа для теорем из эвклидовой геометрии, было бы достаточно для доказательства большого класса нетривиальных теорем из неэвклидовой геометрии. Машина, снабженная неэвклидовым чертежом (который в подходящей аналитической форме получить не труднее, чем эвклидов), не испытывает при этом тех специфических затруднений, с которыми приходится иметь дело человеку, пытающемуся добыть определенные эвристические указания для доказательства теоремы из рассмотрения неэвклидова чертежа.

Формализация геометрии должна быть осуществлена в рамках низшего функционального анализа. Но, так как мы интересуемся машиной, отыскивающей доказательства на уровне учащегося старших классов средней школы, мы предпочли создать *более или менее подходящую для этой цели систему, следуя* программе наиболее элементарных руководств, а не на основе фундаментальной аксиоматики Тарского, Гильберта или Фордера. Мы не делали попыток обеспечить полноту или неизбыточность формализации. Если впоследствии окажется, что машина в состоянии доказать ту или иную аксиому исходя из других аксиом, соответствующая аксиома будет исключена из списка и мы сможем только приветствовать «математическое изящество» нашего автомата. Что же касается полноты, то машина пользуется теми же привилегиями, что и учащийся, который всегда предполагает (т. е. вводит в качестве дополнительных аксиом) справедливость множества «само собой разумеющихся» или «непосредственно очевидных» утверждений, касающихся, например, свойства упорядоченности точек на прямой или свойств пересечений линий на плоскости. Некоторые из этих утверждений действительно не зависят от имеющихся у него исходных аксиом и должны быть добавлены для получения полной системы. Большинство же можно вывести (правда, обычно с некоторыми трудностями) из уже известных ему аксиом. В этом процессе извлечения допущений из модели по существу нет ничего плохого, если только ученик полностью сознает, что происходит именно этот процесс (это, конечно, редкий случай, если иметь в виду среднего ученика). Вместе с тем такая ситуация позволяет значительно упростить доказательство, оставляя его в то же время осмысленным. Геометрическая машина в явном виде записывает все предположения, используемые ею в данном доказательстве. Она может, если потребуется, свести к минимуму опасность того, что доказывается лишь частный случай данной теоремы, строя добавочные чертежи для проверки общности ее предположений.



Фиг 1

Геометрическая машина фактически представляет собой частную реализацию электронной машины для обработки информации IBM-704; программа эта довольно длинная и сложная. Запрограммированная так машина состоит из трех частей: *синтаксической машины* (СМ) и *машины-чертежника* (МЧ), которые *связаны с эвристической машиной* (ЭМ), *являющейся* исполнительной (управляющей) программой Эти связи показаны на фиг. 1.

СМ оперирует с формальной системой и содержит эквиваленты большинства синтаксических эвристик, используемых «Логиком-теоретиком»[1]. МЧ содержит координатное представление доказываемой теоремы вместе с набором подпрограмм, которые дают качественное описание чертежа. Важно отметить, что, хотя для выработки описания используется аналитическая геометрия, информация, передаваемая в ЭМ (прямая связь между чертежом и формальной системой отсутствует), может иметь лишь следующую форму: «На чертеже отрезок AB, по-видимому, равен отрезку CD» или «На чертеже треугольник ABC не содержит прямого угла». Поведение системы не изменилось бы,

[1] Процесс *ценообразования*, как он определяется у Ньюэлла, Шоу и Саймона, управляется эвристической машиной

если бы МЧ была заменена устройством, способным вычерчивать фигуры на бумаге и исследовать их.

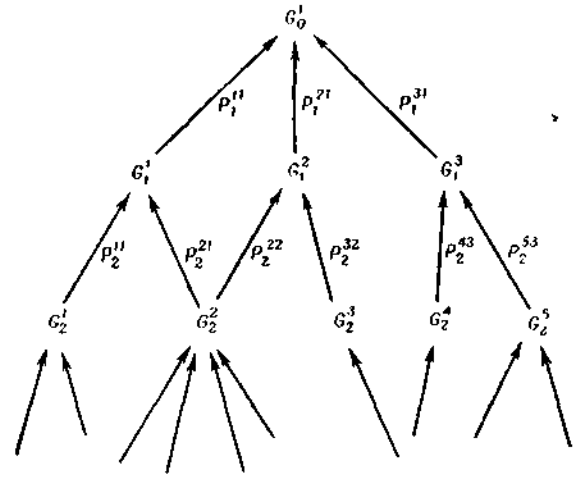
Основное назначение ЭМ в нашей первой системе (которую мы и рассматриваем в этой статье) состоит в том, чтобы сравнивать методом «обратного просмотра» последовательности, вырабатываемые СМ, с их интерпретацией на чертеже, отбрасывая те последовательности, которые не подтверждаются моделью. Кроме того, ЭМ решает и некоторые другие задачи, например организацию процесса поиска доказательства и распознавание синтаксической симметрии определенного класса последовательностей. Благодаря синтаксической симметрии поведение машины аналогично поведению математика в случае, когда ему надо доказать два синтаксически симметричных выражения A и B. В этом случае он докажет только A и скажет: «B доказывается аналогично». Использование синтаксической симметрии является важным свойством ЭМ. Оно подробно описано в работе [364].

Описанная выше процедура, очевидно, не зависит от геометрии; с соответствующей интерпретацией она применима к любой формальной системе. ЭМ применяет, однако, и некоторые дополнительные семантические эвристики, которые зависят от геометрии, но они могут «отключаться», так что поведение машины может изучаться как при наличии специальных геометрических эвристик, так и без них.

Характер машины для доказательства теорем в основном определяется ЭМ. Видоизменения и усовершенствования системы (например, введение обучения) возможны именно за счет модификации этой части программы.

Наша первая система не «вычерчивает» исходный чертеж; вместо этого ее снабжают «чертежом» в виде списка возможных координат точек, упоминаемых в теореме. Этот список точек сопровождается другим списком, в который занесены точки, соединенные отрезками. Координаты выбираются так, чтобы придать максимально возможную общность чертежам. В дальнейших разработках система будет давать свою собственную интерпретацию исходным посылкам, но, поскольку большинство задач, предназначенных для учащихся средней школы, сопровождается чертежом, мы решили, что на этом этапе можно обойтись без таких излишеств в программировании. Когда машина вычерчивает собственный чертеж, точки могут выбираться ею случайным образом с тем лишь ограничением, что они должны соответствовать исходным посылкам.

Работая методом «обратного просмотра», система вырабатывает граф решения задачи, определяемый следующим образом. Пусть G_0 — формальное утверждение, подлежащее доказательству. Назовем его целью задачи. Если G_i является формальным утверждением с таким свойством, что G_{i-1} можно непосредственно вывести из G_i, то говорят, что G_i является для задачи подцелью порядка i. Все подцели G_j, такие, что $j < i$, являются подцелями более высокого порядка, чем G_i; при этом G_0 рассматривается как подцель нулевого порядка. Узлами графа решения задачи (фиг. 2) служат G_i, причем с каждым узлом



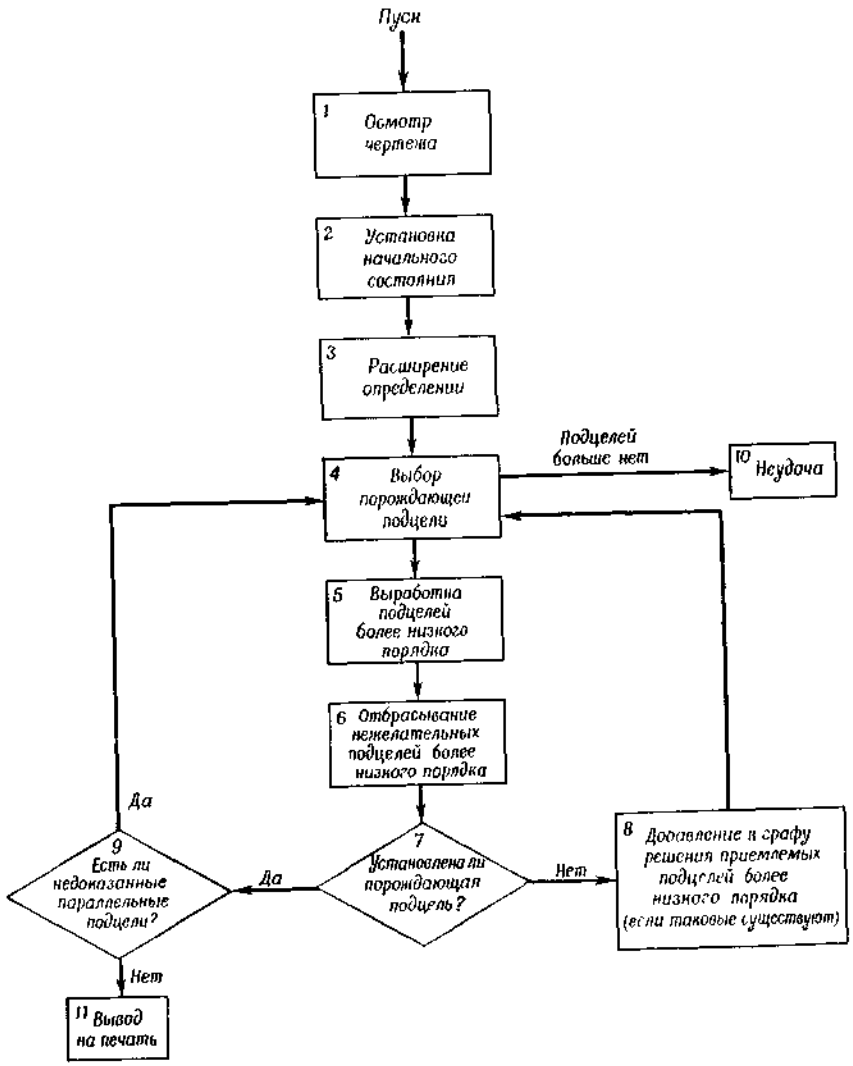
Фиг. 2. Граф решения задачи

Узлы G_i^α представляют подцели порядка i, где α — номер подцели данного порядка $P_i^{\alpha\beta}$ — преобразования G_i^α в G_{i-1}^β.

G_i прямой связью соединен по крайней мере один узел G_{i-1}. Каждая связь представляет собой преобразование G_i в G_{i-1}. Задача решена, если любая подцель G_i может быть непосредственно выведена из исходных предпосылок и аксиом. Если, как это обычно бывает в геометрии, данная подцель является совокупностью утверждений, то граф разделяется в этой точке на параллельные подцели, каждая из которых должна доказываться отдельно. В любой момент времени граф решения задачи дает полное представление о процессе поиска доказательства к этому моменту времени.

Структура ЭМ (определяющая структуру всей системы) в сильно упрощенном виде представлена на фиг. 3. МЧ и СМ являются подпрограммами в ЭМ. Работа машины заключается

в выполнении следующих операций (нумерация операций совпадает с нумерацией соответствующих блоков на фиг. 3):



Фиг 3 Упрощенная схема работы геометрической машины

1. Чертеж просматривается с целью составления трех списков, из которых один содержит все отрезки чертежа, другой — все углы, а третий — все треугольники. Каждый элемент из

списка сопровождается подсписком, дающим его подробное описание.

2. Система приводится в начальное состояние, для чего исходные предпосылки заносятся в список установленных формул, а подлежащее доказательству утверждение фиксируется на графе решений как подцель нулевого порядка.

3. В список установленных формул добавляются определения понятий и утверждений, входящих в предпосылки.

4. Из графа решения выбирается подцель, подлежащая доказательству («порождающая подцель»).

5. Из памяти выбираются соответствующие аксиомы и теоремы и, пользуясь методом «обратного просмотра», вырабатывается ряд подцелей более низкого порядка так, что, если любая из этих подцелей установлена, порождающая подцель устанавливается на основе «modus ponens» и соответствующей аксиомы (или теоремы). Если порождающая подцель признается «временно бесполезной» (шаг 8), испытываются другие построения (см. стр. 157).

6. Подцели, не удовлетворяющие чертежу, отбрасываются так же, как и те, которые на графе являются подцелями более высокого порядка (или синтаксически симметричны некоторым подцелям более высокого порядка).

7. Если любая из выработанных подцелей более низкого порядка признана действительной на основании ее наличия в списке установленных формул или исходя из чертежа, то порождающая подцель считается установленной; в противном случае:

8. В графе добавляются допустимые подцели более низкого порядка и выбирается новая порождающая подцель (шаг 4). Если в настоящий момент не построены допустимые подцели более низкого порядка для данной порождающей подцели, но такое построение возможно, эта порождающая подцель признается «временно бесполезной». Если же такое построение невозможно или если машина пытается, но не может этого сделать, то порождающая подцель обозначается как «бесполезная».

9. Если порождающая подцель установлена, она заносится в список установленных формул вместе со всеми следствиями более высоких порядков, определяемых графом. Если больше не остается параллельных подцелей, которые надлежит установить, то машина воспроизводит доказательство по графу решения и выдает его на печать (шаг 11).

10. Если на некотором этапе любая свободная подцель на графе оказывается бесполезной, то машина терпит неудачу, если только она еще раньше не исчерпала имеющегося запаса времени и памяти или терпения экспериментатора.

Именно в блоках 4, 5 и 6, в которых выбираются, развиваются и отбрасываются подцели, находятся основные эвристики. В этих подпрограммах, если угодно, и заложен «искусственный разум» нашей машины.

Некоторые первые результаты

Геометрическая машина может доказать многие теоремы в пределах заданной ей формальной системы, используя чертеж лишь для того, чтобы указать, какие подцели, вероятно, справедливы. Таким способом следующая ниже теорема была доказана менее чем за минуту[1].

Теорема. Точка, лежащая на биссектрисе угла, одинаково удалена от его сторон (см. фиг. 4 в приложении А).

Менее чем за 5 *мин* машина способна найти также доказательство следующей, сопряженной с прежней теоремы, требующее построения дополнительного отрезка.

Теорема. В четырехугольнике, в котором две противоположные стороны равны и параллельны, другие две стороны равны (см. фиг. 5 в приложении Б).

Хотя введение машиной нового элемента производит определенное впечатление, структура этого доказательства в основном тривиальна. Для получения этого нового отрезка просто соединяются две уже существующие точки. К построению нового отрезка машина пришла при помощи следующего процесса. Попытавшись развить подцели цепочки «$AB=CD$», машина не смогла найти ни одной, которая бы удовлетворяла чертежу. В таком случае обычно производится поиск другого пути на графе решения. Но когда такого пути нет (как и было в этом случае, так как порождающая цепочка является подцелью нулевого порядка), машина пересматривает те из ранее отброшенных подцелей, которые содержат понятия, не представленные на чертеже. Затем для каждой такой подцели машина рассматривает такую расширенную систему посылок, что на чертеже содержится представление каждого нового понятия. Если в расширенной системе цепочка обоснована и существуют теоремы,

[1] В описываемых ниже доказательствах использованы следующие сокращенные обозначения.

Разн. стор. $XYUV$ — точки X и Y находятся по разные стороны от прямой, проходящей через U и V.

Одн. стор. $XYUV$ — точки X и Y находятся по одну сторону от прямой, проходящей через U и V.

Предш. XYZ — точки X, Y и Z расположены на одной прямой в указанном порядке.

Коллин. XYZ — точки X, Y и Z коллинеарны.

которые позволяют вывести требуемую посылку из первоначальной системы, эта цепочка становится подцелью в расширенной системе. Добавленные посылки уточняют построение на чертеже, допускаемое теоремами, из которых эти посылки были выведены.

Однако возвратимся к нашему примеру. Подцель «$\triangle ABD$ подобен ($\cong$) $\triangle CDB$» порождается цепочкой «$AB=CD$», но требуемые треугольники не будут представлены на чертеже, пока не будет добавлена посылка «отрезок BD существует». Аксиома «две различные точки определяют отрезок» разрешает такое построение[1]. Весь процесс доказательства является вариантом вышеописанной основной эвристики и, очевидно, не зависит от конкретного вида рассматриваемой формальной системы. Заметим также, что процесс доказательства конечен, так как новые понятия не вводят новых точек, а лишь заново рассматривают старые.

Наш второй пример иллюстрирует еще одно положение. Хотя из чертежа (см. фиг. 5) ясно, что линия BD при пересечении сторон AC и AD образует внутренние накрестлежащие углы, это является следствием теоремы: «Противоположные вершины выпуклого четырехугольника находятся по разные стороны от диагонали, проходящей через две другие вершины». То, что это не верно для произвольного четырехугольника, становится ясно при рассмотрении внешней диагонали невыпуклого четырехугольника. Тогда для полной строгости решения требуется, чтобы вышеуказанная лемма, если она неизвестна, была доказана и чтобы было показано, что четырехугольник $ABCD$ может быть только выпуклым. Вместо этого машина делает обычное предположение, что диагональ образует с противолежащими сторонами четырехугольника внутренние накрестлежащие углы. Однако в отличие от обычного школьного учебника это предположение в явном виде внесено в доказательство.

Описанная система для доказательства теорем пригодна для многих задач, значительно более сложных, чем те, которые были приведены выше. Однако, когда число отдельных точек, упоминаемых в посылках, возрастает линейно, число сторон в графе решений увеличивается экспоненциально и соответствующим образом возрастает и время, требуемое для исследо-

[1] Для того чтобы треугольник был определен, наша формальная система требует, чтобы были указаны отрезки, соединяющие вершины треугольника, а также сами вершины. Это необходимо во избежание трудностей, которые в противном случае возникали бы тогда, когда в теореме упоминается большое число неколлинеарных точек Если бы наша формальная система была истинной геометрией точек, то все такие указания имелись бы непосредственно на чертеже.

вания графа. Если бы машина могла среди данной системы подцелей выбирать те, которые наиболее вероятно ведут к решению, то время на поиск можно было бы значительно сократить. Чтобы придать машине такую способность, были введены в рассмотрение две специальные геометрические эвристики.

Первая из них является программой, которая распознает определенные подцели, обычно устанавливаемые за один шаг. В эту категорию входят, например, тождества или равенства углов, которые на чертеже выступают как вертикальные углы. Такие подцели заносятся в список первоочередных целей и развиваются раньше любых других рассматриваемых подцелей.

Вторая специальная эвристика — это программа, которая определяет «расстояние» между каждой цепочкой подцели и системой цепочек посылок в некотором нечетко определенном пространстве формул. После того как будут развиты все подцели, выработанные эвристикой, следующей выбирается подцель, которая «ближе» всего к системе посылок в пространстве формул.

Поучительно проследить поведение машины при доказательстве сложных теорем как с расширенной системой семантических эвристик, так и без нее. Для теоремы: «Две вершины треугольника одинаково удалены от медианы стороны треугольника, определяемой этими вершинами» машине потребовалось около 8 *мин*, чтобы найти доказательство, опираясь только на основную эвристику (см. фиг. 6 в приложении В). Расширенная система эвристик позволяет получить доказательство за 1 *мин*. Кроме того, второе доказательство очень короткое и ведет прямо к цели, в то время как первое доказательство вслепую блуждает около прямого пути к цели, прежде чем достигает его.

Тот факт, что во втором случае потребовались только 4 цикла выработки подцелей по сравнению с 24 циклами в первом случае, отражает значительно бо́льшую эффективность расширенной системы эвристик. Во втором случае была выработана 21 промежуточная подцель, а в первом — 61. Граф решения содержал во втором случае только три уровня, а в первом 12.

Для конкретного случая задачи, предлагавшейся на выпускном экзамене по геометрии в Бруклинском техническом училище, решение с использованием расширенной системы эвристик было найдено менее чем за 5 *мин*. Когда же использовалась только основная эвристика, машина за полчаса исчерпала свои возможности, так и не решив задачи. В то же время существуют и такие задачи, для которых машина не получает выигрыша от применения дополнительных эвристик. Теорема: «Диагонали параллелограмма пересекаются» была доказана в обоих случаях приблизительно за 3 *мин*. Доказательства, полученные в обоих

случаях, были эквивалентны, хотя и не одинаковы. Задача из экзаменационных тестов Бруклинского технического училища дает пример промежуточного случая. Для этой задачи доказательства были в обоих случаях идентичны, но при работе только с основной эвристикой потребовалось почти в три раза больше времени (8 *мин* по сравнению с 3 *мин* в случае расширенных эвристик). Мы, несомненно, столкнемся и с такими случаями, когда применение расширенной системы эвристик приведет к проигрышу в эффективности, хотя до сих пор такие примеры в нашей ограниченной практике не встречались.

Заключение

Уместно еще раз подчеркнуть тот факт, что объектом наших исследований была не конструкция машины, способной доказывать теоремы из эвклидовой геометрии на плоскости или, того более, теоремы в некоторой неразрешимой системе, например в теории чисел. Скорее нас интересовала возможность использования машинами эвристических методов (или стратегий) для решения таких задач, которые на машинах нельзя решить другими методами. Машины для доказательства теорем, как таковые, являются предметом исследования математиков и логиков. Важная работа в этом направлении сделана Ван Хао и Гилмором в исследовательском центре IBM. Ван Хао написал для машины IBM-704 программу, которая может доказать все теоремы исчисления высказываний, сформулированные Расселом и Уайтхедом в их «Principia Mathematica»[1], тогда как «Логик-теоретик» мог справиться только с 38 из 52 теорем гл. 2 этого труда. Кроме того, «Логику-теоретику» потребовалось для доказательства намного больше времени, чем программе Ван Хао.

Однако Ньюэлл, Шоу и Саймон интересовались эвристическими методами, тогда как Ван Хао, а также Гилмор (программа которого работала с исчислением предикатов первого порядка) ищут алгоритмы, которые давали бы, хотя и в меньшей степени, чем решающая процедура, «интересные» доказательства за приемлемое время. Как Ван Хао, так и Гилмор пришли к выводу, что для более сложных формальных систем требуются эвристики (они предпочитают слово «стратегии»), которые необходимы им для того, чтобы сделать их алгоритмы достаточно селективными для получения интересных доказательств за приемлемое машинное время и при приемлемом объеме памяти.

---

[1] Подробнее об этой работе см. «Кибернетический сборник», № 5 (1962). — *Прим. перев.*

Исследования Ван Хао и Гилмора в большой степени относятся к новой ветви прикладной логики, впервые изложенной Ван Хао. Он назвал эту дисциплину «инференциальным анализом» [1], и, по его определению, она «оперирует доказательствами так же, как численный анализ — вычислениями» [1133]. Ожидается, что результаты инференциального анализа «приведут к машинной проверке новых математических результатов», а в конце концов «приведут к получению машинных доказательств новых и трудных теорем». Нам думается, что инференциальный анализ имеет отношение также и к проблеме «разумного» поведения машин. Однако автомат будущего, столкнувшись с реальным миром, будет, безусловно, вынужден в основном опираться на эвристики, так как для необычных формальных систем, которыми будет описываться его окружение, традиционные методы математической логики будут, вероятно, неприменимы.

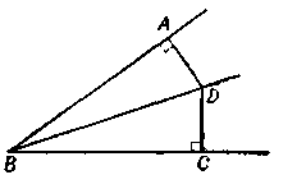
В заключение нам хотелось бы сказать несколько слов о направлении данных исследований на ближайшее будущее. Машина, описанная выше, является системой, предназначенной только для решения задач. Ее структура остается неизменной, за исключением добавления новых теорем в список аксиом. Последовательность практических задач, задаваемая машине, не будет улучшать ее работу, если только среди них не окажется теоремы, которую в дальнейшем можно использовать. Так как машина не может совершенствовать свою структуру, она всегда будет решать только ограниченный класс задач, который с самого начала предусмотрен ее создателем. По-видимому, проблема создания на этом пути машины для решения более широкого класса задач намного сложнее, чем создание машины, хотя и не столь разумной, но зато способной обучаться.

Очевиден следующий подход к проблеме внесения элемента обучения в геометрическую машину: машине дают возможность так регулировать все параметры, определяющие ее конкретные семантические эвристики, чтобы максимизировать предсказанную пригодность тех подцелей, которые оказываются полезными на практике. Таким путем машина будет улучшать согласование своих эвристических фильтров с классом задач, считающихся достаточно интересными, чтобы задаваться машине.

Еще большее значение имело бы введение программ, позволяющих машине распознавать повторяющиеся в процессе поиска доказательства образы. Будучи однажды обнаружен, такой образ позволил бы машине создавать свои собственные эвристики, основываясь на повторяемости образа в последующем доказательстве. Например, машина может заметить, что за определенными классами цепочек исходных предпосылок в доказательстве регулярно следует один и тот же первый шаг. Эвристика, выведенная из такого образа, искала бы посылки для таких цепочек и делала бы первый шаг дедукции перед тем, как начинать просматривать граф решения. Однако для этого сначала необходимо как следует разобраться в трудной проблеме распознавания абстрактных образов; кроме того, и преобразование образа в эффективную эвристику само по себе отнюдь не тривиальная задача. Но какой бы подход к обучению машины не был выбран как наиболее многообещающий, геометрическая машина будет служить превосходной системой для проведения исследований в этой области.

[1] От англ. inference — вывод, заключение. — *Прим. ред.*

Приложение А



Фиг 4.

Посылки:

Угол ABD равен углу DBC
Отрезок AD перпендикулярен отрезку AB.
Отрезок DC перпендикулярен отрезку BC.

Определения:

Угол DAB — прямой
Угол DCB — прямой

Синтаксические симметрии [1]:

CA, BB, AC, DD

Цели:

Отрезок AD равен отрезку CD.

Решение:

Угол ABD равен углу DBC.	Посылка.
Угол DAB — прямой.	Определение перпендикуляра.
Угол DCB — прямой	Определение перпендикуляра.
Угол BAD равен углу BCD.	Все прямые углы равны.
Отрезок DB.	Предположение, основанное на чертеже.
Отрезок DB равен отрезку BD.	Тождество.
Треугольник BCD.	Предположение, основанное на чертеже.

[1] Синтаксические симметрии — список таких перестановок наименований точек на чертеже, при которых формулировки предпосылок не меняются (см [364]) — *Прим. перев.*

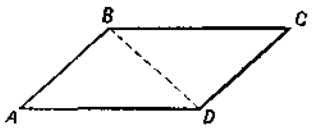
Треугольник *BAD*.	Предположение, основанное на чертеже.
Треугольник *ADB* равен треугольнику *CDB*.	Сторона — угол — угол.
Отрезок *AD* равен отрезку *CD*.	Соответствующие элементы равных треугольников равны.

Общее затраченное время: 0,32 *мин*.

Приложение Б

Посылки:

Четырехугольник *ABCD*.
Отрезок *BC* параллелен отрезку *AD*.
Отрезок *BC* равен отрезку *AD*.



Фиг. 5.

Синтаксические симметрии:

CA	*BA*	*DA*
DB	*AB*	*CB*
AC	*DC*	*BC*
BD	*CD*	*AD*

Цели:

Отрезок *AB* равен отрезку *CD*.

Я не могу решить задачу. Затраченное время: **0,88** ***мин.***

Построить отрезок *DB*.
Добавить к предпосылкам отрезок *DB*.
Решать задачу заново

Решение:

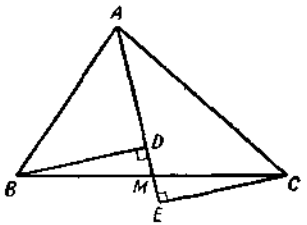
Отрезок *BC* параллелен отрезку *AD*.	Посылка.
Разн. стор. *CADB*.	Предположение, основанное на чертеже.
Отрезок *DB*.	Посылка.
Угол *ADB* равен углу *CBD*.	Внутренние накрестлежащие углы при параллельных прямых.
Отрезок *BC* равен отрезку *AD*.	Посылка.
Отрезок *BD* равен отрезку *DB*.	Тождество.
Треугольник *CDB*.	Предположение, основанное на чертеже.
Треугольник *ABD*.	Предположение, основанное на чертеже.
Треугольник *ABD* равен треугольнику *CDB*.	Сторона — угол — сторона.
Отрезок *AB* равен отрезку *CD*.	Соответствующие элементы равных треугольников

Общее затраченное время: **4,06** ***мин.***

Приложение В[1]

Посылки:

Треугольник *ABC*.
Предш. *BMC*.
Отрезок *BM* равен отрезку *MC*.
Предш. *ADM*.
Предш. *DME*.
Отрезок *BD* перпендикулярен отрезку *AM*.
Отрезок *CE* перпендикулярен отрезку *ME*.



Фиг 6.

Определения:

Угол *MEC* — прямой.
Угол *BDM* — прямой
Угол *BDA* — прямой.
Угол *BDA* равен углу *BDM*.

Синтаксической симметрии нет.

Цели:

Отрезок *BD* равен отрезку *EC*.

Решение с основными эвристиками:

Предш. *BMC*.	Посылка.
Отрезок *EC*.	Предположение, основанное на чертеже.
Угол *ECM* равен углу *BCE*.	Тот же угол.
Предш. *DME*.	Посылка.
Угол *CED* равен углу *MEC*.	Тот же угол.
Угол *BDM* — прямой.	Определение перпендикуляра.
Угол *MEC* — прямой.	Определение перпендикуляра.
Угол *BDM* равен углу *CEM*.	Все прямые углы равны.
Угол *CED* равен углу *BDM*.	Углы, равные одному и тому же углу, равны
Одн. стор *MEDB*.	Предположение, основанное на чертеже.
Одн. стор. *DACE*.	Предположение, основанное на чертеже.
Коллин. *EDM*.	Из упорядоченной коллинеарности следует коллинеарность.
Предш. *ADM*.	Посылка.
Предш. *EDA*.	Комбинаторные свойства упорядоченной коллинеарности.
Коллин *EDA*	Из упорядоченной коллинеарности следует коллинеарность.
Угол *BDE* равен углу *AEC*.	Различное обозначение одинаковых углов.

[1] См примечание на стр. 156 — *Прим. ред.*

Предш. *EMA*.	Комбинаторные свойства упорядоченной коллинеарности.
Угол *CEM* равен углу *AEC*.	Тот же угол.
Угол *BDE* равен углу *MEC*.	Углы, равные одному и тому же углу, равны.
Угол *CED* равен углу *EDB*.	Углы, равные одному и тому же углу, равны.
Разн. стор. *CBED*.	Предположение, основанное на чертеже.
Отрезок *ED*.	Предположение, основанное на чертеже.
Отрезок *EC* параллелен отрезку *BD*.	Отрезки параллельны, если внутренние накрестлежащие углы равны.
Разн. стор. *EDCB*.	Предположение, основанное на чертеже
Отрезок *CB*.	Предположение, основанное на чертеже.
Угол *BCE* равен углу *DBC*.	Внутренние накрестлежащие углы при параллельных прямых.
Угол *ECM* равен углу *DBC*.	Углы, равные одному и тому же углу, равны.
Одн. стор. *CMBD*.	Предположение, основанное на чертеже
Одн. стор. *MBEC*.	Предположение, основанное на чертеже.
Коллин. *CMB*.	Из упорядоченной коллинеарности следует коллинеарность
Угол *DBM* равен углу *BCE*.	Различное обозначение одинаковых углов
Угол *MBD* равен углу *MCE*.	Углы, равные одному и тому же углу, равны.
Угол *DMB* равен углу *EMC*.	Вертикальные углы
Отрезок *BM* равен отрезку *MC*.	Посылка.
Треугольник *BDM*.	Предположение, основанное на чертеже.
Треугольник *CEM*.	Предположение, основанное на чертеже.
Треугольник *BDM* равен треугольнику *CEM*.	Угол — сторона — угол.
Отрезок *BD* равен отрезку *EC*.	Соответствующие элементы равных треугольников.

Общее затраченное время. 8,08 *мин*.

Решение с расширенными эвристиками:

Предш. *DME*.	Посылка.
Предш *BMC*.	Посылка.
Угол *DMB* равен углу *EMC*.	Вертикальные углы.
Угол *BDM* — прямой	Определение перпендикуляра.
Угол *MEC* — прямой.	Определение перпендикуляра
Угол *BDM* равен углу *CEM*.	Все прямые углы равны.
Отрезок *BM* равен отрезку *MC*.	Посылка.
Треугольник *CEM*.	Предположение, основанное на чертеже.
Треугольник *BDM*.	Предположение, основанное на чертеже.

Треугольник *BDM* равен треугольнику *CEM*.	Сторона — угол — угол.
Отрезок *BD* равен отрезку *EC*.	Соответствующие элементы равных треугольников.

Общее затраченное время· 1,06 *мин*.

5. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ МАШИНЫ ДЛЯ ДОКАЗАТЕЛЬСТВА ГЕОМЕТРИЧЕСКИХ ТЕОРЕМ

Г. Гелернтер, Дж. Ханзен и Д. Ловленд

Введение

Ранней весной 1959 г. машина IBM-704, используя программу приблизительно в 20 000 команд, доказала свою первую теорему из элементарной эвклидовой геометрии на плоскости [365]. С того времени машина для доказательства геометрических теорем (т. е. машина IBM-704 с введенной в нее программой) нашла решения большого числа задач[1], взятых из учебников средней школы и выпускных экзаменов по планиметрии. Некоторые из этих задач следует считать достаточно трудными для среднего школьника. Действительно, едва ли какой-либо школьник, кроме разве самых способных, смог бы решить задачи из последней группы решенных машиной задач, если бы объем его предварительной «тренировки» был бы тот же, что и у машины (т. е. если бы он располагал тем же словарем геометрических понятий и тем же запасом ранее доказанных теорем).

Планируя исследования, результатом которых и явилось создание «геометрической машины», мы хотели изучить пути использования современных быстродействующих ЦВМ для решения нового и трудного класса задач, которые до сих пор считались недоступными для конечных автоматов. В особенности мы хотели, чтобы наша машина решала задачи, которые обычно считаются под силу только человеку. В одной из предыдущих работ [363] подробно излагаются причины, побудившие нас выбрать в качестве тестовой задачи доказательство геометрических теорем. Здесь мы только заметим, что процесс решения задач в геометрии удовлетворяет нашему определению «разумной деятельности», причем эта задача как раз особенно хорошо соответствовала тому подходу, который мы хотели исследовать. Тот факт, что геометрия является разрешимой системой, не имеет никакого значения для цели нашего исследования. Используемые машиной методы пригодны также для доказательства

[1] К настоящему времени доказано более 50 теорем.

теорем в системах, для которых не существует решающего алгоритма.

Мы не будем подробно останавливаться на вопросе, действительно ли наша машина ведет себя «разумно» при решении таких задач, которые люди считают «интеллектуальными». В этом вопросе психологи никак нам не помогут. Они редко рассматривают абстрактное понятие интеллекта вне зависимости от его носителя, и им еще предстоит удовлетворительно охарактеризовать разумное поведение человека. В конце концов, наблюдая за поведением людей, мы часто говорим, что они совершают разумные действия, хотя это слово имеет весьма туманное значение. Обычно эти действия связаны с решением задач, требующих принятия сложных решений в потенциально бесконечной и неконтролируемой среде. Мы были бы счастливы, если бы действия нашей машины копировали такое поведение, как бы его ни называли.

Эвристическое программирование и геометрическая машина

Геометрическая машина способна находить доказательства большого числа теорем в пределах выбранной для данного случая формальной системы (включая теоремы о параллельных линиях, о подобии и о равенстве и неравенстве отрезков и углов), не применяя какого-либо решающего алгоритма и не используя полного перебора всех возможных последовательностей, могущих привести к доказательству. Вместо этого машина опирается на эвристические методы, позволяющие предотвращать вырабатывание таких последовательностей, которые имеют низкую априорную вероятность привести к доказательству рассматриваемой теоремы.

Общая задача эвристического программирования обсуждалась Минским [732] и Ньюэллом, Шоу и Саймоном [795]. Подход авторов к этой задаче достаточно подробно описан в статьях, на которые мы уже ссылались [363, 365]. Поэтому мы здесь откажемся от подробного описания результатов, достигнутых машиной, а сосредоточим свое внимание главным образом на том, как эти результаты были достигнуты.

Здесь следует отметить, что геометрическая машина в основном работает «аналитическим» методом (идет обратно от желаемого результата). На каждом этапе поиска доказательства существует цель, которую надо «связать» с посылками задачи с помощью «мостика» из аксиом и ранее доказанных теорем. Если такой связи непосредственно установить нельзя, вырабатывается некоторое множество «подцелей» и процесс повторяется

для одной из подцелей. Для отбрасывания подцелей, вероятность которых оказаться полезными для доказательства мала, используются эвристические правила. Они выбирают из оставшихся подцелей ту, с помощью которой продолжается поиск. Кроме того, они выбирают конкретные аксиомы и теоремы, используемые для вырабатывания новых подцелей. В ряде случаев (например, при непрямом доказательстве) машина отходит от такой процедуры, но эти случаи составляют лишь малую часть в общем времени поиска.

В программе была использована предложенная Ньюэллом, Шоу и Саймоном [790] организация памяти в виде ассоциативных списков. Для того чтобы упростить задачу написания такой массивной и сложной программы, был разработан удобный язык для работы со списками, позволяющий использовать систему ФОРТРАН в качестве компилирующей программы [367]. Авторы уверены, что, если бы они широко не пользовались промежуточным языком программирования, геометрическую программу, по всей вероятности, так и не удалось бы создать.

Обзор результатов

Геометрическая машина с самого начала существовала в нескольких вариантах. Наиболее ранний и более примитивный вариант системы содержал единственную основную семантическую эвристику[1]. Однако эта первая система могла доказывать большое число интересных, хотя по общему признанию и простых теорем элементарной геометрии на плоскости[2]. Это эвристическое правило, не зависящее от конкретного вида рассматриваемой формальной системы, можно описать следующим образом. Все формулы подцелей, вырабатываемые на данном этапе поиска доказательства, интерпретируются на модели формальной системы. В нашем случае такой моделью является чертеж. Если интерпретируемая подцель обоснована на чертеже, она принимается как возможный шаг в доказательстве при условии, что она не циклическая [364]. В противном случае такая подцель отбрасывается.

В качестве эксперимента было предпринято несколько попыток доказать исключительно простые теоремы, причем вышеупомянутая эвристика «отключалась» от системы (т. е. принимались все выработанные нециклические подцели). В каждом случае

[1] Семантической называется такая эвристика, которая основана на интерпретации формальной системы, а не на структуре цепочек в этой системе.

[2] Некоторые из этих доказательств воспроизводятся в статье Гелернтера [365]

память машины быстро заполнялась первыми несколькими сотнями выработанных подцелей первого уровня, и фактически машина никогда не заканчивала выработку всех подцелей первого уровня. По нашим скромным подсчетам такая система на каждом шаге доказательства в среднем вырабатывает около 1000 подцелей. В случае же, когда используется чертеж, на каждом шаге в среднем вырабатывается 5 подцелей. При сравнении этих цифр становится очевидным, что использование чертежа является решающим фактором для нашей системы. (Заметим, что общее число подцелей на графе решения растет экспоненциально с числом принятых на каждом шаге подцелей.)

Поскольку описанная выше процедура является эвристической, иногда делаются ошибки при выборе или отбрасывании формул, служащих подцелями. Чертеж сообщается машине в координатном виде с некоторой конечной точностью. Формулы интерпретируются путем преобразования их в форму, в которой для всех входящих в формулу переменных (точек) подсчитывается численное значение их координат. Например, чтобы проверить справедливость утверждения, касающегося равенства двух отрезков, на чертеже вычисляется длина каждого отрезка, а затем они сравниваются с учетом заданного числа разрядов. Если же утверждение касается параллельных отрезков, то подсчитываются и сравниваются их наклоны. В небольшом числе случаев ошибка округления выходила за допустимые пределы, в результате чего обоснованные подцели были отброшены, а необоснованные — приняты. Однако важно заметить, что ни в одном случае такая ошибка не привела к ложному доказательству. Если обоснованные подцели отбрасываются, машина находит другие пути решения задачи. Если же принимаются необоснованные подцели, машина, конечно, не может включить их в пределы формальной системы. В наихудшем возможном случае ошибка в интерпретации может помешать машине прийти к какому-либо доказательству вообще, но такая ошибка никогда не приведет ее к необоснованному доказательству.

В этой связи должно быть ясно, что чертеж используется только для управления поиском доказательства, давая ответы «да» или «нет» на вопросы типа: «равны ли на чертеже отрезки AB и CD?» или «является ли угол ABC на чертеже прямым?» В геометрической машине нет прямой связи между чертежом и формальной системой. Поведение машины не изменилось бы, если бы координатное представление было заменено устройством, которое может чертить фигуры на бумаге и обследовать их.

После того как в описанной основной системе для доказательства теорем выработано множество подцелей, каждый член этого

множества исследуется в порядке его получения. Следующая подцель в цепочке подцелей не исследуется до тех пор, пока не будет полностью рассмотрена предыдущая. Кроме того, при получении для данной подцели подцелей следующего уровня использовались все приемлемые теоремы, имеющиеся в памяти машины.

Вскоре эта система была расширена введением эвристик, которые выбирают как подцели, так и теоремы для их вырабатывания. Эвристика выбора подцелей находит «расстояние» между цепочкой подцели и множеством посылок в нестрого определенном для этого случая пространстве формул. На каждом шаге в качестве следующей подцели выбирается такая, которая «ближе всего» к посылкам в пространстве формул. Программа выбора генератора подцели распознает определенные классы подцелей, которые обычно устанавливаются за один шаг. Для таких «насущных» подцелей немедленно применяется соответствующий генератор и делается попытка установить одношаговое доказательство (этой конкретной подцели), прежде чем вырабатывать полную систему подцелей для данной формулы.

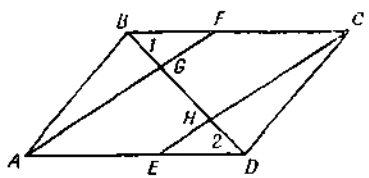
Расширенная система может доказывать ряд более сложных теорем, справиться с которыми было не под силу основной машине. Задачи же, которые доступны обеим машинам, решаются расширенной системой приблизительно в три раза быстрее, причем для этого вырабатывается около $^2/_3$ общего числа подцелей за вдвое меньшее число циклов, чем у основной машины. Средняя глубина графа решения для расширенной системы составляет около 7—9 уровней, что соответствует $^2/_3$ средней глубины для основной системы.

Добавив простую программу построения, мы расширили возможности машины, включив в рассмотрение совершенно новый класс задач, до сих пор логически недостижимый. Эта программа начинает работать только тогда, когда все другие попытки найти доказательство оказались неудачными. Она позволяет машине соединить две ранее несоединенные точки на чертеже и продолжить вновь полученный отрезок до пересечения со всеми другими отрезками на чертеже. При пересечении нового отрезка со всеми другими отрезками на чертеже образуются новые точки, которые помечаются машиной и становятся частью системы посылок данной задачи.

На этой стадии разработки геометрическая машина могла давать весьма впечатляющие доказательства (см. приложение А). Однако по мере возрастания числа точек на чертеже ее работа быстро замедлялась. Этот эффект в основном определяется тем, что в противоположность человеку, который зрительно определяет угол по его вершине и двум лучам, машина определяет

угол с помощью предиката от трех переменных — вершины и точки на каждом луче. Следовательно, равенство углов *1* и *2* на фиг. 1 можно представить 36 различными способами, так как каждый угол может быть обозначен 6 различными способами. Формальная строгость требует также, чтобы равенство углов *ADH* и *EDG*, например, было доказано, а не принято как очевидное. Должно быть ясно, что при указанных выше условиях поиск доказательства быстро вырождается в рассмотрение массы неинтересных деталей.

В современной модификации системы проблема углов решается следующим образом. Машине дают возможность использовать чертеж для отождествления данного угла с полной системой его обозначений, и она точно так же, как это делает человек, может заключать, что различные обозначения одного и того же угла имеют один и тот же смысл. В современном варианте геометрическая машина может доказывать теоремы, которые по уровню сложности соответствуют приводимой ниже.



Фиг. 1.

Теорема. Продолжение отрезка, соединяющего середины диагоналей трапеции, делит сторону трапеции пополам (см. приложение Б).

Ограничения системы

Тем, кто знаком со свойствами формальных логических систем, ясно, что, если машина не вырабатывает новых точек, все задачи решаются в пределах исчисления высказываний, правда, довольно сложной структуры. Хотя нынешняя программа построения может вырабатывать и вырабатывает новые точки, можно ожидать, что наши результаты до тех пор не будут представлять значительного интереса для логиков, пока машина не будет способной делать все возможные построения (в соответствии с полной системой аксиом с кванторами), необходимые для поиска доказательства.

Наш метод определения «хороших» (правильно составленных) цепочек в логической системе налагает также серьезное ограничение на формальную общность «машины для доказательства теорем». Чтобы достичь необходимой скорости и эффективности в обработке, правильно составленные формулы определяются схемой, а не рекурсивно. Тогда тип утверждения, которое можно сделать в такой системе, определяется имеющейся

в машине схемой. Практическое следствие этой потери общности состоит в довольно сильном ограничении свободы, с которой в геометрии можно обращаться с алгебраическими утверждениями.

В дополнение к этому имеется еще ряд менее существенных ограничений. Они являются следствием ограниченной скорости и памяти машины при решении задач сильно выраженного комбинаторного характера. Ослабление любого из указанных ограничений немедленно позволит расширить класс решаемых машиной задач как в количественном отношении, так и в отношении их сложности.

Заключение

Мы считаем, что первоначальная цель в нашей программе исследований машинного «разума» достигнута. Если бы задающий вопросы «эксперт» (судья по Тьюрингу) ограничился бы областью доказательства теорем в элементарной эвклидовой геометрии на плоскости, то от нашей машины можно было бы ожидать, что она блестяще проявила бы себя в соревновании с человеком в хорошо известной «игре в имитацию» Тьюринга [1085]. Конечно, существует много других областей (например, решение арифметических задач), в которых машина уже может успешно соревноваться с человеком. Существенным моментом здесь является то, что если «эксперт» достаточно эрудирован, он, несомненно, будет избегать задавать свои вопросы из таких областей, хотя он вполне мог бы (во всяком случае, до настоящего времени) ввести в качестве критерия, разделяющего людей и машины, задачу из планиметрии[1]. Хотя теперь мы имеем еще один аргумент в пользу того, что любая ограниченная область интеллектуальной деятельности человека может быть подчинена неумолимой логике электронов, переключающих схем и клапанов, мы предоставляем нашим коллегам — философам спорить о следствиях, вытекающих из такого утверждения, — по крайней мере до тех пор, пока не будут составлены такие программы для машин, которые дадут возможность и нам размышлять об этом.

[1] На это можно возразить (и такие возражения, несомненно, *будут* высказаны), что если задающий вопросы эксперт осведомлен о разрешимости геометрии, он определенно будет избегать этой области, предпочитая использовать для различения человека и машины явно неразрешимые разделы исчисления предикатов или теории чисел. В связи с этим мы напоминаем, что наши методы не зависят от разрешимости формальной системы, а Ван Хао и Гилмор по существу разработали доказательства теорем в неразрешимой области исчисления предикатов.

К счастью, наша работа имеет ряд более конкретных следствий, чем те, которые упоминались выше. Возможно, что наиболее важными из них являются те, которые относятся к «инференциальному анализу» — новой ветви прикладной логики, впервые введенной и рассмотренной Ван Хао [1133]. Инференциальный анализ «оперирует с доказательствами так же, как численный анализ — с вычислениями». Ожидается, что он «приведет к машинной проверке новых математических результатов» и, что более важно, «приведет к доказательству машиной новых и трудных теорем». Мы полагаем, что наши методы работы и эффективного поиска на дереве решения, а также наши результаты, касающиеся синтаксической симметрии, окажутся полезными и при решении задач инференциального анализа.

Кроме того, нами сделан некоторый вклад и в область методов, позволяющих машине осуществлять сложные информационные процессы. В работе [367] приведены результаты, касающиеся создания и использования промежуточных языков для описания процессов обработки списков. В этой работе указываются требования, которым должна удовлетворять цифровая вычислительная машина, позволяющая оптимальным образом осуществлять такие процессы. В общих чертах такая машина должна допускать:

1) обобщенную косвенную адресацию, позволяющую обращаться к ячейке, находящейся в конце упорядоченной цепочки ячеек. Эта цепочка может иметь произвольную длину, а адрес каждой следующей ячейки может находиться в левой или правой части предыдущей ячейки[1];

2) восстановление исполнительного адреса, которое дает возможность использовать имеющееся к данному моменту содержимое адресного регистра (например, последний адрес в длинной и сложной цепи косвенной адресации) в качестве адреса для следующей операции;

3) операции над частями ячейки — значительно расширенный набор операций, операндами которых служат части машинных слов. Ячейки памяти делятся на части в соответствии с заранее принятым соглашением;

4) операции поиска в списке; они играют ту же роль, которую в обычных машинах играют операции поиска в табличных массивах.

[1] Косвенное задание адреса — это задание последовательности операций, необходимых для нахождения нужной ячейки. Пример: вызов ячейки, имеющей адрес ПЛПЛ (J), осуществляется с помощью следующей последовательности операций 1) найти ячейку, адрес которой записан в левой части ячейки J; 2) найти ячейку, адрес которой записан в правой части ячейки, найденной на предыдущем шаге, и т. д. — *Прим. перев.*

Машина, предназначенная для обработки списков, должна удовлетворять также жестким требованиям, касающимся обмена между внутренней и внешней памятью. Эти требования здесь не рассматриваются.

Рассмотрим, наконец, значение нашей работы для основной проблемы — создания «разумной машины». Нам кажется, что геометрическая машина служит убедительным доводом в пользу силы и плодотворности метода эвристического программирования для решения машиной определенного класса задач. В нашей экспериментальной работе с машиной введение одной-единственной эвристики часто было эквивалентно увеличению скорости машины или емкости ее памяти в 3—5 раз.

Наша программа оказалась не пригодной для исследования простейших видов обучения, основанных на методе проб и ошибок, в основном из-за низкой скорости, с которой она приобретает опыт. Однако разумно ожидать, что геометрическая машина может еще оказаться полезной при исследовании более совершенных видов обучения, основанных на понимании смысла понятий, если только когда-нибудь станет известно, как сформулировать эту задачу.

Кроме всего прочего, наша работа дает качественную оценку порядка трудности задач, которые, по-видимому, можно решать при современном состоянии вычислительной техники. Три года назад преобладало мнение, что геометрическая машина не будет создана к настоящему времени. Сейчас же специалист вряд ли будет оспаривать, что через три года машина будет доказывать интересные теоремы из области теории чисел.

Приложение А

Посылки:

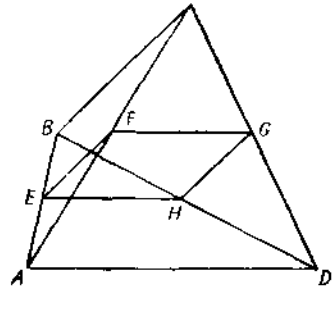
Четырехугольник *ABCD*.
Точка *E* — середина отрезка *AB*.
Точка *F* — середина отрезка *AC*.
Точка *G* — середина отрезка *CD*.
Точка *H* — середина отрезка *BD*.

Требуется доказать:

EFGH — параллелограмм.

Синтаксические симметрии:

BA, AB, DC, CD, EE, HF, GG, FH, CA, DB, AC, BD, GE, FF, EG, HH, DA, CB, BC, AD, GE, HF, EG, FH.



Фиг. 2.

Доказательство[1]:

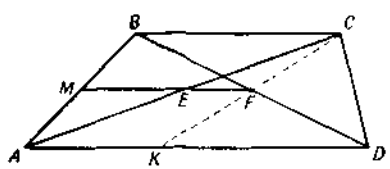
Отрезок DG равен отрезку GC.	Определение средней точки.
Отрезок CF равен отрезку FA.	Определение средней точки.
Угол DCA.	Предположение, основанное на чертеже.
Предш. DGC.	Определение средней точки.
Предш. CFA.	Определение средней точки.
Отрезок GF параллелен отрезку AD.	Отрезок, соединяющий середины сторон треугольника, параллелен его основанию.
Отрезок HE параллелен отрезку AD.	Синтаксическое сопряжение.
Отрезок GF параллелен отрезку EH.	Отрезки, параллельные одному и тому же отрезку, параллельны.
Отрезок HG параллелен отрезку FE.	Синтаксическое сопряжение
Четырехугольник $HGFE$.	Предположение, основанное на чертеже.
Параллелограмм $EFGH$.	Четырехугольник, у которого противоположные стороны параллельны,— параллелограмм.

Общее затраченное время: 1,03 *мин.*

Приложение Б

Посылки:

Четырехугольник $ABCD$
Отрезок BC параллелен отрезку AD.
Точка E — середина отрезка AC.
Точка F — середина отрезка BD.
Предш. MEF.
Предш. AMB.



Фиг 3

Требуется доказать:

Отрезок MB равен отрезку MA.

Синтаксических симметрий нет.

Я не могу решить задачу. Затраченное время: 8,12 *мин.*

Построить отрезок CF.
Продолжить отрезок CF до пересечения с отрезком AD в точке K.

Добавить к посылкам следующие утверждения:

Предш. CFK.
Коллин. AKD.

Доказательство:

Отрезок BC параллелен отрезку AD.	Посылка.
Коллин. AKD.	Посылка.
Отрезок KD параллелен отрезку BC.	Отрезки, коллинеарные параллельным отрезкам, параллельны.

[1] См примечание на стр 156 — *Прим ред.*

Разн. стор $KCDB$.	Предположение, основанное на чертеже.
Отрезок DB.	Предположение, основанное на чертеже.
Угол KDB равен углу CBD.	Внутренние накрестлежащие углы при параллельных прямых равны.
Предш. CFK.	Посылка.
Предш. DFB.	Определение средней точки.
Угол KFD равен углу CFB.	Вертикальные углы равны.
Отрезок DF равен отрезку FB.	Определение средней точки.
Треугольник FDK.	Предположение, основанное на чертеже.
Треугольник FBC.	Предположение, основанное на чертеже.
Треугольник FDK равен треугольнику FBC	Угол — сторона — угол
Отрезок KF равен отрезку CF.	Соответствующие отрезки равных треугольников равны.
Отрезок CE равен отрезку EA.	Определение средней точки.
Треугольник AKC.	Предположение, основанное на чертеже.
Предш. CEA.	Определение средней точки.
Отрезок EF параллелен отрезку AK.	Отрезок, соединяющий середины сторон треугольника, параллелен его основанию
Отрезок EF параллелен отрезку KD.	Отрезки, коллинеарные параллельным отрезкам, параллельны.
Отрезок FE параллелен отрезку BC.	Отрезки, параллельные одному и тому же отрезку, параллельны.
Предш. MEF.	Посылка.
Коллин. MEF.	Из упорядоченной коллинеарности следует коллинеарность.
Отрезок FM параллелен отрезку BC.	Отрезки, коллинеарные параллельным отрезкам, параллельны.
Отрезок FM параллелен отрезку DA.	Отрезки, параллельные одному и тому же отрезку, параллельны.
Треугольник DBA.	Предположение, основанное на чертеже.
Предш AMB.	Посылка.
Отрезок MB равен отрезку MA.	Линия, параллельная основанию треугольника и делящая одну из сторон пополам, делит другую сторону пополам.

Общее затраченное время: 30,68 *мин.*

РАЗДЕЛ 3

ДВЕ ВАЖНЫЕ ПРИКЛАДНЫЕ ЗАДАЧИ

Что могут дать исследования по искусственному разуму? Могут ли быть использованы разработанные устройства и методики для решения «реальных» практических задач? Такие вопросы нам с полным основанием могут задать руководитель предприятия, инженер или специалист по исследованию операций.

Статьи этого раздела книги описывают эвристические программы, решающие сложные проблемы из важных прикладных областей. В одной из статей приводится описание программы в области основ интегрального исчисления, в другой — задача балансирования сборочного конвейера. Общими для обеих проблем являются следующие черты:

1. Обе задачи являются задачами средней трудности для сообразительного человека с университетской подготовкой.

2. К обеим задачам легко применить эвристические методы, напоминающие методы, которые использует сообразительный человек при решении проблем.

3. Использовать программы для решения такого рода задач экономически целесообразно. В большинстве случаев было бы трудно нанять человека с соответствующей квалификацией для решения этих задач с теми же финансовыми затратами. Время, потребное машине IBM-7090 на решение этих задач, обычно намного меньше, чем понадобилось бы в среднем квалифицированному специалисту в этой области. Это верно даже несмотря на то, что современные вычислительные машины не создавались с расчетом на задачи эвристического программирования и мало приспособлены к ним. Поэтому и современные информационные языки, хорошо подходящие для эвристического программирования, используют вычислительные машины очень неэффективно.

Программа балансирования сборочного конвейера, разработанная Ф. Тонгом, представляет пример применения эвристического программирования к важной проблеме организации и управления производством. Балансирование сборочной конвейерной линии включает задачи нахождения эффективной системы расстановки рабочих и распределения заданий и рабочих мест так, чтобы максимизировать скорость конвейера или минимизировать количество рабочих, необходимых при данной скорости сборки. Разработанные для решения этой задачи более или менее прямые методы обычно оказываются практически бесполезными, так как они требуют испытания бесчисленного множества «деревьев» — комбинаций рабочих элементов задачи. Программа же Тонга в отличие от такой постановки задачи применяет набор эвристик балансирования конвейера, т. е. «хитрости ремесла», с тем, чтобы упростить ограничения, установить структуру задания и осуществить непосредственно вычислительный процесс.

Важной чертой программы Тонга является ее «уровень стремлений», эвристика постепенного ограничения усилий. В задаче балансирования конвейера могут быть легко вычислены максимум скорости сборки или минимум числа рабочих при заданной скорости, даже если действительное распределение рабочих может и не быть найдено. Программа *не пытается дать оптимального решения, а стремится дать «удовлетворительное» решение*, отличающееся от оптимального на определенный процент. Особенность этой эвристики вытекает из того, что чем ближе требуемое решение должно быть к оптимуму, тем больше требуется вычислений для нахождения этого решения, и эта связь существенно нелинейна.

Программа Слэйджла SAINT имеет дело с задачами в знакомой многим из нас области. В настоящее время язык элементарного интегрального исчисления является общепринятым для студентов университетов. Интегрирование функций (когда ответ не может быть найден стандартным приемом или интеграл не является табличным) оказывается нетривиальной, творческой задачей. Многие студенты младших курсов засиживаются до поздней ночи в поисках «ключа», который подошел бы к некоторым сложным интегралам. Обычно полагают, что тот, кто добивается успеха, не только располагает богатым запасом «приемов и преобразований», которые он может применять, но что он также обладает «чутьем», интуицией, подсказывающей ему, какие из приемов выбирать и в какой последовательности их применять. Можно, наконец, добавить также, что аналитическое интегрирование функций составляет значительную часть программы по математике на современных технических и естественных факультетах.

SAINT — программа для проведения аналитического интегрирования элементарных функций. Она использует те же «хитрости», которым обучают и которые применяют студенты при прохождении данного курса. По своим основным идеям и структуре эта программа является непосредственным отпрыском программы «Логик-теоретик» (ЛТ). Так же как и ЛТ, программа SAINT показывает, что поведение, которое, когда говорят о принятии решений человеком, определяют как «разумное» или «проницательное», на самом деле является результатом целесообразного применения некоторых эвристик для сужения пространства поиска в направлении решения.

6. ЭВРИСТИЧЕСКИЕ МЕТОДЫ БАЛАНСИРОВАНИЯ РАБОТЫ СБОРОЧНОГО КОНВЕЙЕРА

Ф. Тонг

1. Введение

В этой статье описывается эвристическая программа для балансирования работы сборочного конвейера. Применение эвристических методов вызвано тем, что данная задача, как и многие другие комбинаторные задачи, не поддается решению — в практическом смысле этого слова — методами современной прикладной математики.

В связи с тем что эвристический подход не гарантирует оптимальности решения, оценка эвристических программ основана на том, дают ли они более быстрое и (или) менее дорогостоящее решение задачи, чем другие методы. Эвристические методы, однако, еще находятся на ранней стадии развития, и многие важные вопросы, такие, как описание процесса и программирование этих методов для вычислительных машин, должны еще быть изучены. В данном исследовании как раз предпринята попытка такого изучения. Экономический эффект в работе специально не подчеркивается, хотя ее результаты достаточно интересны для того, чтобы о них сообщить.

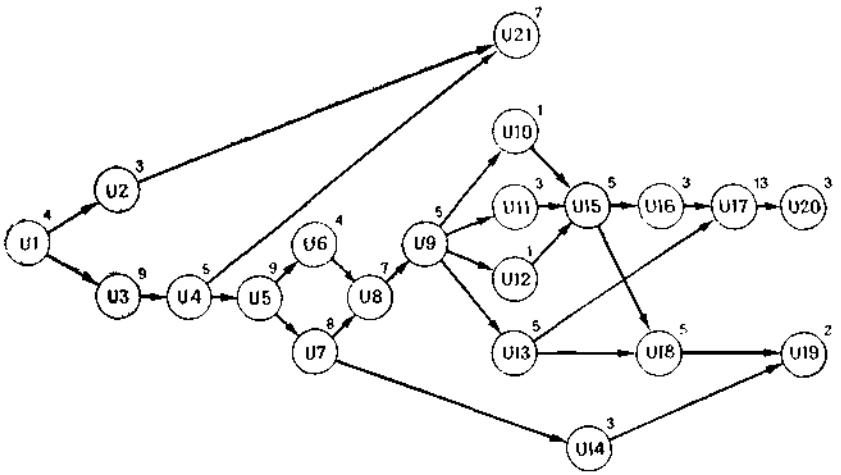
Здесь кратко излагается эвристическая программа балансирования работы сборочного конвейера и приводятся фактические результаты, полученные с помощью этой программы. Более подробное описание самой программы и дальнейшие комментарии к этому подходу и использованию цифровых вычислительных машин представлены в специальной работе [1076].

2. Балансирование сборочного конвейера

Многие виды промышленной продукции (различные бытовые приборы, автомобили и др.) собираются на конвейерных линиях. Элементарные операции, из которых состоит процесс сборки, или «задачи»[1], должны быть закреплены за рабочими местами сборочной линии. (Для наших целей «рабочее место» и «оператор» считаются синонимами.) В простейшем случае каждая элементарная задача (называемая просто «задача», или «элемент») характеризуется затратами времени на единицу продукции и локальными взаимоотношениями с другими элементарными «задачами».

[1] Под задачей (task) автор статьи имеет в виду обычную элементарную производственную операцию. — *Прим. ред.*

На фиг. 1 показана одна из таких сборок [741]. Здесь, например, выполнение элементарной задачи $U5$ требует девяти единиц времени. Она не может быть начата раньше, чем закончится выполнение задачи $U4$, и должна быть выполнена раньше, чем начнется выполнение задачи $U6$ или $U7$. Ограничение, состоящее в том, что задача $U5$ должна предшествовать задаче $U14$, не нуждается в точном описании: такая информация подразумевается записью последовательности $U5 \rightarrow U7 \rightarrow U14$.



Фиг. 1. Графическое представление 21-элементной задачи.

В производственной практике порядок операций не бывает точно закреплен. Его устанавливает инженер в зависимости от имеющегося в его распоряжении материала, требуемого срока выпуска продукции и собственных знаний технологии производства.

Установленная производственная скорость определяет максимальное время (время цикла), предназначенное для каждого рабочего места. Это означает, что количество часов за смену, деленное на число выданных за смену единиц продукции, определяет максимальное время, которое может затратить оператор на единицу продукции.

Задача балансирования работы сборочного конвейера[1] может быть сформулирована следующим образом: при заданной

[1] Существует, конечно, много других вариантов задачи балансирования работы конвейера, которые учитывают такие факторы, как неопределенность единицы времени, различные способности оператора и др. Такого рода ограничения здесь не рассматриваются.

производственной скорости (или, что эквивалентно, заданном времени цикла) определить минимальное число рабочих мест (операторов) с указанием времени и порядка следования операций.

Проблемой, тесно связанной с предыдущей, является следующая: при заданном числе рабочих (рабочих мест) определить наибольшую производственную скорость с указанием времени и порядка следования операций.

Мы будем рассматривать первый вариант задачи.

При более детальном рассмотрении задача балансирования работы сборочного конвейера приводит к следующей постановке элементарных задач:

1) каждая элементарная задача требует знания времени, затрачиваемого на единицу продукции, независимо от того, когда она совершается;

2) среди элементарных задач существует частичное упорядочение.

Оптимальное решение состоит в закреплении элементарных задач за рабочими местами так, чтобы:

1) каждая элементарная задача была закреплена за одним и только одним рабочим местом;

2) суммарное время всех элементарных задач, закрепленных за любым из рабочих мест, не превосходило некоторого максимума (времени цикла);

3) сформированные таким образом рабочие места могли быть упорядочены так, чтобы частичное упорядочение среди элементарных задач при этом не нарушалось;

4) число сформированных таким образом рабочих мест было минимальным.

Оптимальное в отношении минимума рабочих мест решение задачи показано на фиг. 1 (при заданном времени цикла, равном 20) и представлено в табл. 1.

Таблица 1

Рабочие места	Общее время	Элементарные задачи
1	18	$U1$, $U3$, $U4$
2	19	$U2$, $U5$, $U21$
3	15	$U6$, $U7$, $U14$
4	17	$U8$, $U9$, $U10$, $U11$, $U12$
5	20	$U13$, $U15$, $U16$, $U18$, $U19$
6	16	$U17$, $U20$

Реальные задачи балансирования работы сборочного конвейера значительно более сложны, чем в разобранном здесь примере. Так, типичная задача производства бытовых приборов содержала бы около 100 элементарных операций. Инженер, управляющий работой такого конвейера, действует в зависимости от имеющегося у него набора элементов с указанием их времени выполнения и собственных знаний технологического процесса. При отсутствии каких-либо формальных методов он вынужден, опираясь на свой здравый смысл, использовать метод «проб и ошибок» для отыскания приемлемого сочетания элементов.

3. Точные методы решения задачи балансирования работы сборочного конвейера

Задаче балансирования работы сборочного конвейера уделено мало внимания в литературе, посвященной вопросам организации производства; большинство работающих над этой проблемой просто ссылаются на ее обычный характер. Однако она уже была подвергнута изучению в некоторых исследованиях.

Салвсон [949] рассматривает промышленную постановку задачи и отмечает несколько трудностей, связанных с задачей балансирования, таких, например, как определение перед решением задачи минимального числа операторов с учетом различий во времени, приходящемся на различные операции. Он предлагает использовать аппарат комбинаторного анализа, но задача при этом становится слишком громоздкой, так как требует предварительного перебора всех возможных рабочих мест. Однако многие наблюдения Салвсона, касающиеся существа задачи, ценны, и его статья впервые привлекла к этой проблеме внимание научных работников, занимающихся вопросами организации и управления производством.

Джексон [510] предложил алгоритм для систематического перебора и оценки возможных вариантов решения. В одном из последующих разделов этой статьи мы сравним этот точный метод с эвристическим. Дополняя Джексона, Хелджсон и Кво [463] предложили еще один критерий для оценки возможных решений — минимизацию различий в рабочей нагрузке между рабочими местами.

Брайтон [174] разработал метод усиления «сходимости» процесса за счет перемещения элементов в пределах заданных рабочих мест так, чтобы увеличилась максимально достижимая производственная скорость. Хотя этот метод не имеет непосредственного приложения к поставленной здесь проблеме, он оказывается полезен для сглаживания рабочей нагрузки среди рабочих мест после того, как найдено решение задачи.

Митчелл [741] дополнил алгоритм Джексона включением другого обычного ограничения — зонирования. Под зонированием понимается разделение множества элементарных задач (возможно с перекрытием) на подмножества, соответствующие производственным ограничениям при сборочных операциях. В задаче балансирования работы конвейера зонирование может определяться положением продукции на конвейере или размещением производственной мощности, или и тем и другим. Например, некоторые операции могут быть произведены лишь на задней стенке предмета или только, когда он лежит на боку; кроме того, некоторые операции могут быть выполнены только на малом конвейере, присоединенном к главному. Ограничение, требующее, чтобы все элементарные задачи, закрепленные за рабочим местом, находились в одной зоне, добавляется к определению решения.

Упомянутые выше методы практически непригодны для балансирования работы большого сборочного конвейера из-за большого объема вычислений. В действительности какой-либо общей удовлетворительной схемы для решения больших комбинаторных задач, включающих частично упорядоченные связи, еще не найдено [1].

4. Эвристические программы

«Новый международный словарь английского языка» Вебстера (1959 г.) определяет прилагательное «эвристический» как «служащий для выявления или открытия». Под эвристикой мы, следуя Ньюэллу, Шоу и Саймону [794], понимаем принципы или средства, которые способствуют уменьшению поисков при решении задач. Указания в виде «сделай чертеж» при изучении геометрии, «сведи все к синусам и косинусам» при доказательстве тригонометрических тождеств или «всегда, когда можешь, делай шах — он может быть и матом» при игре в шахматы — все это хорошо известные эвристики.

Эвристические методы решения задач — это методы, экономящие затраты труда. Эвристическая программа — это механизация осуществления последовательности эвристических процедур при помощи цифровой вычислительной машины. Вычислительная машина пытается решить задачу на основе выполнения эвристической программы. В настоящее время использование цифровых вычислительных машин является средством выявления поведения сложной эвристической схемы в широком классе задач.

[1] Однако индустриальными фирмами было проведено много неопубликованных исследований по задаче балансирования работы сборочного конвейера и связанным с ней проблемам.

«Логик-теоретик» и «машина для игры в шахматы» Ньюэлла, Шоу и Саймона [789, 792] и «геометрическая машина» Гелернтера и Рочестера [363] являются примерами работающих эвристических программ в других областях.

Часто бывает неясным различие между эвристическими и неэвристическими методами решения задач. Поэтому прежде, чем сформулировать правило, по которому все методы можно разбить на эти две категории, укажем несколько общих характеристик эвристических методов:

1. Разбиение проблемы на «малые» проблемы или подпроблемы (часто посредством «анализа целей и средств») с соответствующей целевой и подцелевой организацией поведения. Например, «шахматная машина» смогла установить, что нельзя играть e7—e5 потому, что это привело бы к размену на этом поле, и в соответствии с этим она вырабатывает подцель: сперва выдвинуть пешку на защиту поля e5.

2. Использование признаков окружающей среды для определения поведения, чтобы выбрать из большого количества возможных альтернатив ценные для программы. Это означает, что между различными задачами (из более общего класса) имеется сильная взаимная зависимость, которая должна быть рассмотрена, чтобы выбрать соответствующий метод решения данной задачи. Так, методы, используемые в задаче балансирования сборочного конвейера для выбора элементов, подлежащих перемещению между группами, зависят от характеристик самих этих групп.

3. Использование рекурсивных процедур так, чтобы подпроблемы решались тем же набором приемов, что и сама проблема. Так, «Логик-теоретик» может применять тот же набор эвристик при доказательстве производного выражения, как и при доказательстве исходного положения, из которых оно было выведено.

4. Отсутствие гарантии в получении удовлетворительного решения, а часто и решения вообще. Например, «шахматная машина» вследствие пространственных и временных ограничений может оказаться не в состоянии рассмотреть некоторые продолжения, среди которых может быть и оптимальное.

Так как эвристический подход не гарантирует оптимального решения, даваемого исчерпывающим перебором, но зато компенсирует это уменьшением затрачиваемых усилий, применимость таких программ, как программа «решателя проблем», должна оправдываться числом успешно решенных задач и относительным количеством потребовавшихся для решения усилий.

В одном из последующих разделов фактические результаты работы эвристической программы балансирования конвейера сравниваются с результатами некоторых «лобовых», т. е. исчерпывающих рассмотрение всех вариантов, методов.

Эвристики, применяемые ниже при решении задачи балансирования сборочного конвейера, разработаны на основе следующих источников: обсуждение с инженерами вопроса о том, как они добиваются балансирования конвейера; изучение литературы, указанной выше; обсуждение возможных методов с коллегами, особенно с Ньюэллом, и, наконец, большая экспериментальная работа над отдельными специфическими частями проблемы.

5. Описание программы балансирования работы сборочного конвейера

Эвристический подход к балансированию работы сборочного конвейера основывается на достаточном упрощении сложной комбинаторной задачи так, чтобы она стала разрешимой (в большинстве случаев) простыми методами.

Сущность эвристической схемы определяют две рекурсивные программы[1].

На первом этапе создается иерархия значительно упрощенных задач балансирования работы конвейера путем объединения групп элементов[2] в единый сложный элемент. Каждый из этих сложных элементов представляет задачу того же класса, что и исходная задача, так как он состоит из элементов с заданным временем действия и частичным упорядочением.

На втором этапе решается простая (при малом числе элементов) задача балансирования работы конвейера, состоящая в закреплении за рабочими и выделении в подпроблеме тех сложных элементов (образующих сами по себе простые задачи), которые требуют более чем одного человека.

---

[1] В качестве базы для эвристического подхода к задаче могут быть предложены также некоторые другие приемы: 1) разделение проблемы на относительно независимые подпроблемы, решение которых находится оптимальными средствами, а перекомбинация этих частичных решений осуществляется эвристическими методами; 2) решение задачи в относительных единицах времени, принимая за единицу время самой длительной или самой короткой операции, а затем, получив результат, совершая перход к действительному времени; 3) рассмотрение мест в порядке, определяемом количеством частично упорядоченных ограничений, причем распределение начинается с тех элементов, где имеется наименьшая свобода среди элементов. Однако все эти различные подходы не рассматриваются в этой статье более детально.

[2] Под словом «элемент» в данном случае понимаются элементарные операции (задачи), совершаемые рабочими. — *Прим. ред.*

Такой подход требует эвристик как для группирования элементов в «сложные элементы» и решения упрощенной таким образом задачи, так и для восстановления деталей исходной задачи, если упрощенный вариант не приводит к решению.

На третьем этапе решения задачи с использованием фактически той же самой эвристики производится «сглаживание» рабочей нагрузки (выраженной в единицах времени) между рабочими местами. Так как наибольшее общее время, предназначенное для любого места, ограничивает скорость конвейера, то выровненный результат отвечает задаче: «Какая максимальная производственная скорость достижима при заданном числе людей?» Так как люди и время измеряются в дискретных единицах, то *невыровненное* оптимальное решение задачи: «При заданной производственной скорости минимизировать число требуемых людей» не обязательно будет оптимальным решением смежной задачи: «При заданном числе людей максимизировать производственную скорость».

Таким образом, общая схема решения проблемы предусматривает следующие этапы:

Фаза I. Повторяющееся применение процесса группирования элементов, в результате чего задача распадается на серию упрощенных задач балансирования работы конвейера, колеблющихся по сложности от первоначальной задачи до задачи, содержащей один сложный элемент.

Фаза II. Рекурсивное применение к этим упрощенным проблемам схемы расстановки людей по задачам при расчленении проблемы на такие компоненты, каждая из которых требует одного человека. Если окажется что сложные элементы проблемы требуют больше людей, чем можно обеспечить, эти элементы разделяются на составные части, которые перегруппировываются так, чтобы требовалось меньше людей.

Фаза III. Сглаживание результата перемещением задач между рабочими местами до тех пор, пока не будет достигнуто как можно более равномерное распределение времени.

В последующих разделах подробно рассматриваются эвристики, связанные с каждой из этих схем.

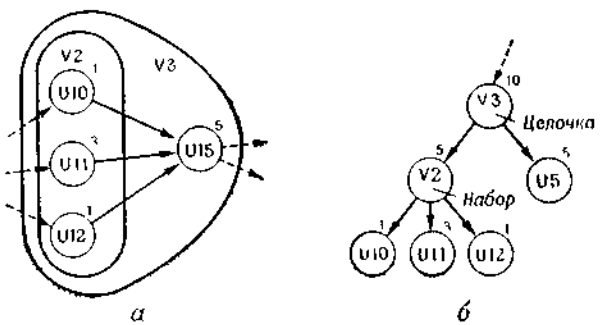
Так как такой подход не гарантирует оптимального решения полной задачи балансирования работы конвейера, мы должны иметь какое-то представление о том, какое решение следует считать удовлетворительным, и в зависимости от этого принимать или отвергать получающиеся решения. Кроме того, поскольку решение проблемы в целом не гарантирует существования решений подпроблем, то процесс решения должен продолжаться до тех пор, пока все подпроблемы не будут решены, и он должен

быть готовым к перестройке и поискам, в случае необходимости, других путей решения. Методы, удовлетворяющие этим требованиям, обсуждаются ниже.

6. Построение иерархии проблем

Упорядочивающие ограничения, присутствующие в задаче, приводят к двум естественным способам группирования элементов.

Между некоторыми элементами либо существует полное упорядочение — например, $U3$ обязательно должна предшествовать



Фиг 2 Группы элементов типа «цепочка» и «набор»

$U4$ (см. фиг 1), либо же нет никакого упорядочения, как, например, для $U10$, $U11$, $U12$. Введем понятия «цепочка» и «набор» в качестве основных характеристик типа группы в упрощенной задаче:

I. Группа соседних элементов, относительный порядок которых полностью определен, так что каждый из них, за исключением первого, имеет единственный непосредственно предшествующий элемент и каждый, за исключением последнего, имеет единственный непосредственно следующий элемент, может быть заменена одним сложным элементом, называемым *«цепочкой»*.

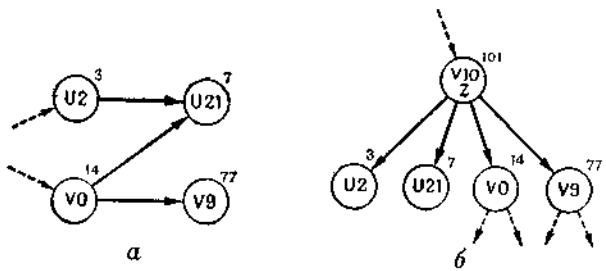
II. Группа элементов, между которыми полностью отсутствует относительное упорядочение, так что все они имеют тот же непосредственно следующий и тот же непосредственно предшествующий элемент, может быть заменена одним сложным элементом, называемым *«набором»*.

Таким образом, группа элементов $U10$, $U11$, $U12$ на фиг. 1 может быть заменена набором $V2$, а группа $V2$, $U15$ затем заменена цепочкой $V3$. На фиг. 2, *а* показано такое группирование. Однако удобнее представить группирование в виде разветвляющегося дерева, где сложный элемент имеет в качестве ветвей элементы, из которых он образован

Дерево сложного элемента $V3$ показано на фиг. 2, *б*. Отметим, что время, связанное со сложным элементом, является суммой времен составляющих его элементов.

Мы будем использовать термины «связь цепочкой», чтобы показать существование некоторого порядка (возможно, не непосредственного) между двумя элементами, и «связь набором», когда такого порядка не существует.

Хотя решение и предусматривает повторное применение группирования для создания иерархии упрощенных проблем, все



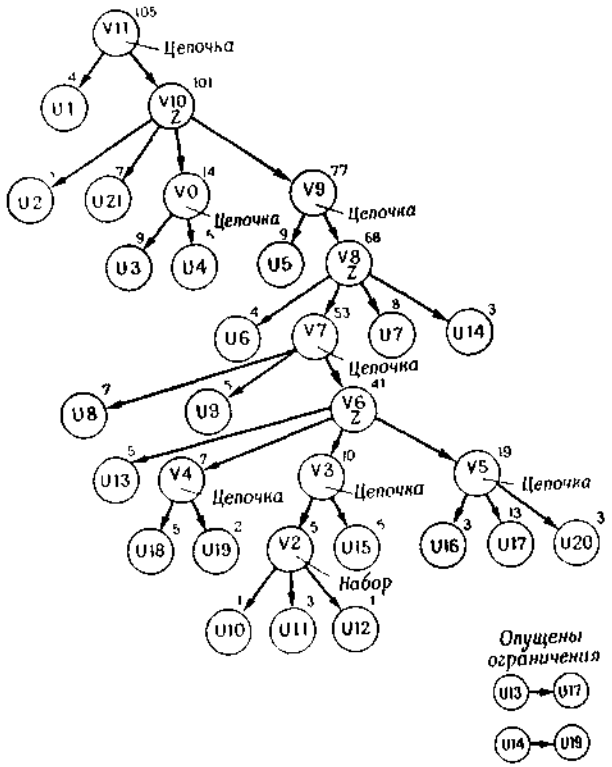
Фиг 3 Z-группирования

же этих двух типов групп оказывается недостаточно для сведения большинства реальных задач к одному сложному элементу. Мы можем продвинуться дальше, определив сперва более сложные операции группирования, а затем, в случае необходимости, устранить «мешающие» ограничения. Полученное решение затем следует проверить, чтобы выяснить, что эти отброшенные ограничения не существенны. Введенный пока единственный дополнительный сложный элемент обозначен как Z.

III. Z представляет собой группу из четырех элементов, из которых два передних имеют общий предшествующий, а два задних — общий последующий элемент. Единственным элементом, непосредственно следующим за одним из передних элементов, является один из задних; два непосредственно следующие один за другим передних элемента являются задними элементами. Задние элементы не имеют другого непосредственно предшествующего им элемента.

Пример Z и его представление в форме дерева даются на фиг. 3 (отметим, что имеется твердо установленный порядок подэлементов при представлении дерева в форме цепочки и в форме Z)

Рекурсивная процедура такого группирования может быть выполнена для любого ансамбля элементов. Под ансамблем элементов мы здесь понимаем те элементы, которые не имеют предшествующих в пределах ансамбля (называемые иначе передними), и непосредственно следующие за ними, а также следующие непосредственно за этими следующими и т д. вплоть



Фиг 4 Выход фазы I 21-элементной задачи

до тех элементов, которые не имеют последующего. Иллюстрацией к этому может служить ансамбль на фиг. 1 с передним элементом $U1$. Из-за недостатка места мы не можем в этой статье останавливаться на деталях и в дальнейшем ограничимся лишь конспективным изложением

Из группы элементов с единственным передним элементом программа пытается образовать цепочку. Когда встречается элемент, имеющий несколько непосредственно следующих за ним элементов, программа сначала ставит подпроблему группирования ансамбля с этими несколькими передними элементами, а затем уже переходит к проблеме. (Процесс этот рекурсивный, так что попытка решения этой подпроблемы может выдвинуть решение новой подпроблемы и т. д) Если подпроблема решена успешно, то продолжается выполнение проблемы высшего уровня — создание цепочки. Если попытка решения подпроблемы терпит неудачу или встречается элемент, имеющий непосредственно предшествующий ему вне ансамбля (не передний), то и проблема высшего уровня терпит неудачу.

При данном ансамбле элементов с несколькими передними элементами программа пытается создать набор или, в случае неудачи, — Z Если это можно сделать, проблема сводится к рассмотрению ансамбля с одним передним элементом, если же нельзя, то вначале ставятся независимые подпроблемы группирования отдельных ансамблей, начинающихся от каждого переднего элемента, а затем каждая подпроблема решается по очереди Когда эти подпроблемы решены, возобновляется решение следующего уровня — образование набора или Z.

При попытках решать эти подпроблемы независимо друг от друга часто обнаруживаются существующие между ними сложные упорядочивающие ограничения. Z тогда по определению включает такие ограничения. Дальнейшие ограничения, которые препятствуют завершению построения уже определенного Z, являются менее строгими, так что образование Z может быть завершено и может быть продолжено решение задачи следующего уровня

На фиг. 4 изображено дерево сложных элементов, построенное применением этой рекурсивной программы к задаче фиг 1 Текущие пояснения к этому процессу приведены в Приложении А, где рабочая последовательность программы показана в действии. Фиг 1 и 4 дают наглядную иллюстрацию этого «протокола». Первая фаза программы должна выполняться только один раз для получения набора ограничений. Построенная в результате иерархия принимается в качестве входа второй фазы работы программы.

7. Группирование операций по рабочим местам

Входами второй фазы, группирующей задачи по рабочим местам, являются:

а) иерархия задачи, полученная в результате работы фазы I; б) время цикла, определенное на основании заданной нормы выработки; в) «процент полезного времени», который дается в качестве ориентира для создания и принятия потенциальных рабочих мест Этот параметр (назначаемый самим проектировщи-

ком линии) является примерной оценкой того, насколько близко к требуемому рабочему месту могут быть сгруппированы элементы задачи. Эта мера отражает структуру и значение времени в решаемой проблеме.

Программа распределения состоит из простых рекурсивных методов по распределению рабочих (рабочих мест) по группам задач и имеет набор процедур для перегруппирования задач, если это необходимо. Эти эвристики перегруппирования обсуждаются в следующем разделе.

Рекурсивная программа распределения рабочих по группам задач заключается в следующем:

1. Вначале производится оценка потребного числа рабочих, удовлетворяющего заданной норме выработки и принятому проценту полезно используемого времени (проценту эффективности). Это число рабочих задает головной сложный элемент, представляющий отдельную группу всех элементов задачи. Затем к этому элементу применяется рекурсивная программа, описанная в п. 2 этого раздела. В случае успешного применения программы линия считается сбалансированной и фаза II окончена. В противном случае число рабочих, приписанное головному элементу, увеличивается и процедура повторяется.

2. а) Если на рассматриваемый элемент выделено более одного человека, то это число рабочих распределяется по составляющим частям элемента согласно требуемому на каждую составляющую времени. Затем этот рекурсивный метод применяется по очереди к каждой составляющей. Таким образом, в случае удачного решения каждой из подпроблем программа решает и всю проблему распределения задач по рабочим местам. Если же это число рабочих не может быть распределено по составляющим данного элемента, — если рассмотренные независимо, они требуют большего числа людей, чем имеется в распоряжении, — то необходимо применить методы перегруппирования для перемещения задач между группами так, чтобы все они были решены независимо друг от друга. Если такая перегруппировка невозможна, то управление возвращается на более высокий уровень и подается сигнал о неудаче в решении подпроблемы.

Если рекурсивной программе не удается определить одну из составляющих ниже этого уровня иерархии и сообщается о неудаче, то избыточные элементы неудавшейся подпроблемы сдвигаются в другую группу, а в случае повторной неудачи методы перегруппировки применяются для построения другого набора независимых групп. Точно так же, если разрешимые подпроблемы не могут быть найдены, управление передается на следующий более высокий уровень с указанием неудачи.

б) Если на данный элемент выделен один человек, то в случае необходимости к нему добавляются дополнительные элементы, чтобы сделать размеры группы близкими к максимальным. Эта группа обозначается как рабочее место. Управление тогда переходит к следующему более высокому уровню и дается сигнал успеха.

Подобно тому, как это проводилось в фазе I, эта рекурсивная программа имеет дело с примерами определенного класса проблем. Она решает всю проблему посредством выдвижения других, «более легких» подпроблем того же класса, решение которых ведет к разрешению более трудных проблем. В частности, фаза I программы балансирования конвейерной линии использует естественные группировки элементарных задач для построения иерархии упрощенных проблем, а в фазе II программа пытается сбалансировать линию, используя группировки предыдущего этапа как законченные единицы и перемещая задачи среди групп только тогда, когда простое распределение терпит неудачу.

8. Методы перегруппировки

Когда простая схема распределения рабочих не приводит к цели, рекурсивная программа в фазе II применяет пять методов перегруппирования: а) прямой перенос, б) обмен, в) последовательная группировка, г) полная группировка и д) исчерпывающая группировка. Какая из эвристик перегруппирования и в каком порядке может применяться — определяется характеристиками группы элементов, подлежащих перегруппировке. Эти методы используются также в фазе III работы программы, в операции сглаживания предполагаемого варианта решения задачи балансировки линии.

Все пять методов используются одной рекурсивной программой, которая рассматривает данную часть иерархии элементов («дерево проблем») и строит последовательность групп элементов, удовлетворяющих некоторым определенным условиям (максимальное общее время группы, минимальное общее время группы, положение от начала или от конца).

Программа приступает к построению первой группы, всегда начиная с наибольшего допустимого элемента, а затем строит следующие группы, исключая из рассмотрения (или разбивая на составляющие, если элемент сложный) один за другим элементы этой группы, и переходит к остающимся элементам данной части иерархии Так, если предлагалось образовать группы, насчитывающие по меньшей мере 8, но не более 16 временных единиц с конца сложного элемента V6 (см. фиг. 4), то

программа образует последовательность ($V4$, $U13$, $U20$) и ($U20$, $U17$). В программе происходит формирование каждого элемента этой последовательности в соответствии с заданным требованием, причем запоминается положение этого элемента в последовательности, что необходимо для образования следующего элемента.

Первые три метода — прямой перенос, обмен и последовательная группировка — используются для выдвижения независимо разрешимых подпроблем, которые могут быть рассмотрены в фазе II рекурсивной программы.

Прямой перенос применяется в том случае, когда рассматриваются только две составляющие. Метод заключается в попытке простого переноса элементов одной составляющей в другую, чем уменьшается количество требуемых рабочих.

Так, если на $V11$ (см. фиг. 4) выделено 6 человек (заданный цикл равен 20 единицам времени), то этим методом вначале пытаются перенести элементы, составляющие по меньшей мере 1, но не более 16 единиц времени, с $V10$ на $U1$, а фактически будет сдвигаться $V0$ (14 единиц времени).

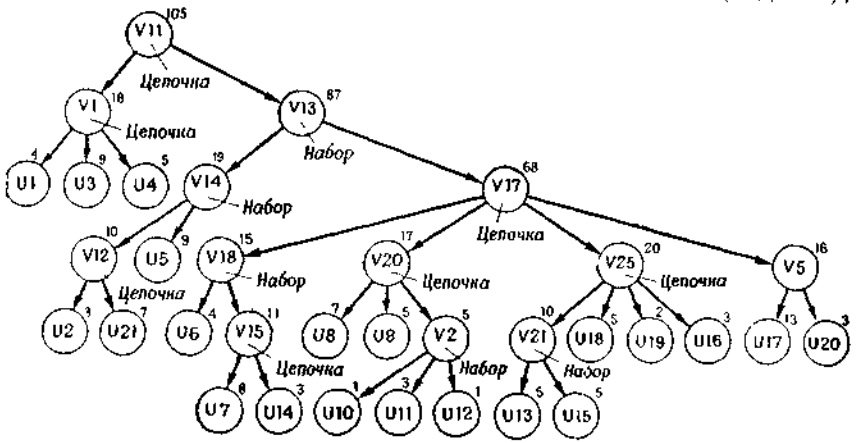
Обмен также применяется только к двум составляющим и лишь тогда, когда прямой перенос не увенчался успехом. Этим методом пытаются произвести перегруппировку путем замены элемента с временем, превышающим допустимый предел, на элемент с меньшим временем из другой составляющей. Пример этого метода приведен в следующем разделе

Последовательная группировка используется при наличии нескольких элементов. Этим методом пробуют сформировать рабочее место с приемлемым временем, начиная с головного элемента данной группы составляющих. Если оставшиеся элементы можно распределить между остальными рабочими, то метод считается успешно примененным. В случае неудачи пытаются сформировать другое рабочее место, начиная с последнего члена группы, и производится аналогичное испытание.

Допустим, например, что на $V7$ (см. фиг. 4) назначено три человека (время цикла 20 единиц). Так как $V7$ образовано из трех составляющих, требующих соответственно одного, одного и трех рабочих, то понадобятся методы перегруппировки. Методом последовательной группировки вначале объединяются $U8$, $U9$ и $V2$, и тогда, поскольку оставшаяся составляющая теперь требует 36 единиц времени, для чего необходимо только два оставшихся рабочих, метод оказывается успешным.

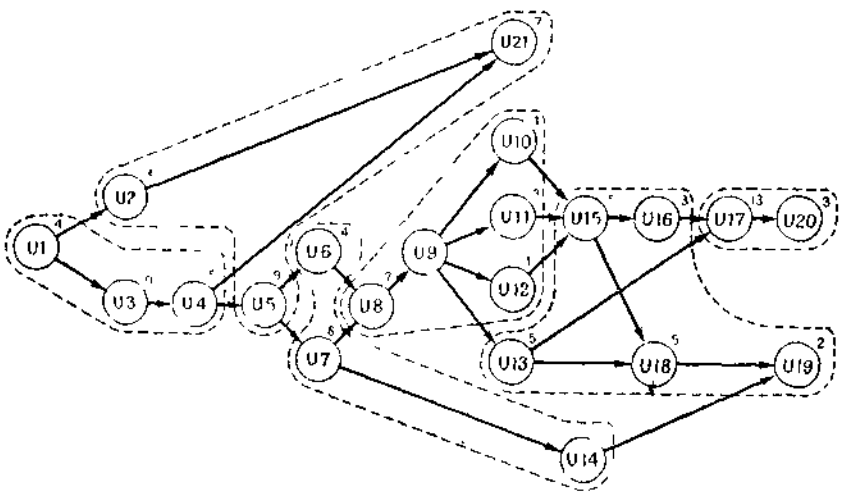
Остальные два метода — полной и исчерпывающей группировки — применяются как крайнее средство сразу ко всем оставшимся нерешенными подпроблемам.

Метод полной группировки пытается, начиная с головного (переднего) элемента группы, а затем с последнего (заднего),



Фиг 5 Выход фазы II 21 элементной задачи

формировать рабочие места, пока все элементы не будут сгруппированы. Если в какой-то период времени окажется, что метод



Фиг 6 21 элементная задача с рабочими местами Время цикла =20

не может построить рабочее место, потому что оставшихся элементов меньше, чем может быть обеспечено оставшимися рабочими, метод терпит неудачу.

Исчерпывающая группировка образует все возможные (независимые) первые рабочие места, а затем все возможные последующие для каждого из них. Этот метод по существу является исчерпывающим алгоритмом, предложенным Джексоном [510]. Ввиду большого объема вычислений, потребного для осуществления этого метода, он используется только в случаях, когда должно быть решено назначение двух рабочих. Метод позволяет все же перейти к построению на следующем, более высоком уровне фазы II (даже в случае неудачи) с наилучшими группировками, сформированными от начала или от конца, и с элементами, которые не удалось сгруппировать этим методом.

Хотя эти методы перегруппировки не совсем четки и точны, они доказали свою применимость во всех задачах, которые с их помощью пытались разрешить. Фиг. 5 показывает задачу фиг. 1 и 4 после применения к ней фазы II рекурсивной программы (время цикла равно 20 единицам, процент полезного времени — 95), а на фиг. 6 изображены рабочие места, организованные таким путем по первоначальной задаче (см. также табл. 1).

В Приложении Б приведен протокол, составленный в ходе такого распределения.

9. Сглаживание результатов балансирования

В фазе III программа балансирования конвейерной линии стремится добиться равномерного распределения работ по рабочим местам путем повторного уменьшения потребного времени на наиболее напряженных рабочих местах. Входами этой фазы работы являются иерархическое представление сбалансированного конвейера, которое может быть получено в результате работы фазы II, и время цикла.

Итеративная программа фазы III использует описанные выше (фаза II работы программы) эвристические методы перегруппировки.

Фаза III программы включает следующие шаги:

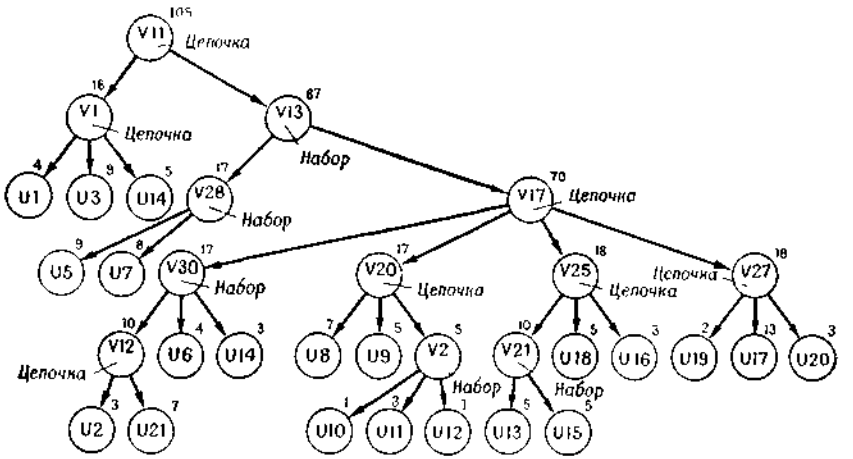
1. Вычисляется наименьшее возможное время, необходимое на полную сборку (при данном цикле). Если время для наиболее напряженного рабочего места не превышает этой границы, то дальнейшие сглаживания невозможны и программа считается оконченной.

2. Рабочие места с превышающим допустимое значение временем рассматриваются в порядке возрастания величины превышения. Пытаются выровнять распределение работ по рабочим местам, используя метод прямого эвристического переноса.

В случае удачи возвращаются к шагу 1. В противном случае переходят к шагу 3.

3. Рассматривают снова рабочие места в порядке возрастающего различия по отношению к наиболее напряженным местам. Пытаются уменьшить наибольшую разницу, применяя эвристику обмена. Если попытка оказалась удачной, возвращаются к шагу 1, в случае неудачи делается переход к шагу 4.

4. Рассматриваются еще раз и в той же последовательности «конфликтные» рабочие места. Для каждого из этих рабочих



Фиг. 7. Выход фазы III 21-элементной задачи

мест, применяя сначала первый, а затем второй эвристические методы, пытаются произвести любой перенос так, чтобы уменьшить время, даже если бывшее прежде рабочее место с меньшим временем теперь превратится в место с бóльшим временем или даже выйдет за пределы допустимого времени. Если такой перенос удается, образуется подпроблема уменьшения времени вновь сформировавшихся «конфликтных» рабочих мест. Для ее решения применяется вся приведенная программа (исключая шаг 4). В случае удачи управление передается на выполнение ступени 1; в противном случае фаза III работы программы заканчивается.

Описанная выше эвристическая программа еще не закодирована для вычислительной машины (это нетрудно сделать, поскольку она тесно связана с уже существующими для вычислительных машин программами). Но для нескольких случаев эта программа рассматривалась как программа ручного моделиро-

вания. Результаты такого моделирования для задачи фиг. 1 и 4 представлены на фиг. 7

Промоделированный выход работы фазы III программы балансирования конвейера дается в Приложении В.

10. Полученные результаты

Разработка и проверка этой эвристической программы была осуществлена на трех задачах: на 11-элементной задаче, заимствованной у Джексона [510], на 21-элементной задаче, использованной в качестве иллюстрации в данной статье и рассмотренной Митчеллом [74], и на 70-элементной реальной промышленной задаче. Хотя число примеров недостаточно, чтобы сделать обоснованные выводы об эффективности этого метода[1], можно все же сделать несколько интересных обобщений.

В табл. 2 представлены сводные данные по результатам этих исследований. Из этой таблицы видно увеличение вычислительных усилий с возрастанием размера задач на обеих фазах программы — I и II.

Таблица 2

Задача	Фаза I	Фаза II					
	число вычислительных команд IPL	время цикла	процент эффективности	количество рабочих мест	процент простоя	число вычислительных операции	число вычислительных операции на 1 рабочее место, *тыс*
11-элементная	14 478	10	95	5	8	153 075	30,6
21-элементная	51 386	20	90	6	12	141 868	23,7
		19	95	6	8	143 183	23,8
		14	95	8	6	627 809	78,5
		18	98	6	3	483 458	80,5
		21	100	5	0	760 803	152,2
70-элементная	207 194	176	93	23	9	2 495 118	108,5

[1] Необходимо проверить эту программу и на других больших задачах — не только для того, чтобы оценить экономический эффект, но и для того, чтобы убедиться, что включенные в нее эвристические методы не были выбраны случайно, лишь на основании того, что они подходят для данных частных задач

Заметно также увеличение числа вычислений на каждое рабочее место с уменьшением процента «времени простоя» для 21-элементной задачи. Эти наблюдения говорят о том, что количество вычислений, необходимых для балансирования конвейера, зависит от числа элементов в задаче, количества потребных рабочих мест и процента времени простоя. Дополнительные эксперименты с реальными задачами дадут возможность развить методы оценки потребных усилий для частных задач.

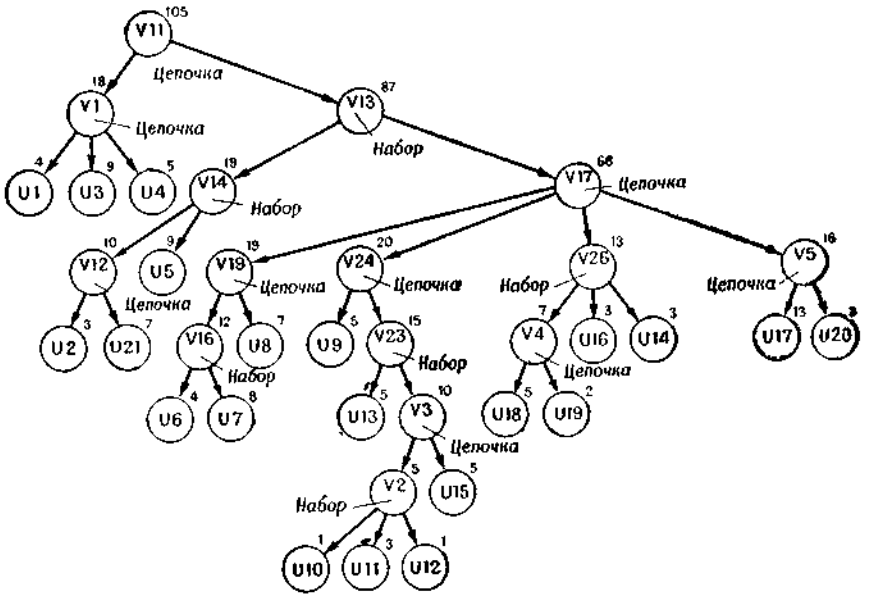
Чтобы почувствовать объем вычислительных усилий при решении задачи методом перебора, мы попытались решить 11-элементную задачу, пользуясь только методом исчерпывающей перегруппировки, т. е. по существу использовав для решения задачи алгоритм Джексона [510]. (Это только приблизительное сравнение, поскольку алгоритм в данном случае был применен к списковым процедурам и больше выиграл бы в целом при матричном представлении, чем эвристическая программа. Кроме того, поскольку метод был органически включен в процедуру, то в эту оценку входит часть затрат, не требуемых алгоритмом самим по себе.) При этих условиях программа для балансирования линии потребовала выполнения около 389 000 команд IPL, т. е. в 2,5 раза больше, чем при эвристическом подходе. Хотя учитывая все вышесказанное, эта разница и представляется не очень значительной, она будет быстро возрастать с увеличением количества элементов в задаче.

Была также проанализирована формализация задачи методом линейного программирования для 11-элементного примера (RAND Simplex Code [949]). Формализация потребовала 11 уравнений с 62 неизвестными (1 неизвестное для каждой допустимой группировки при заданном времени цикла, равном 10 единицам). Программа выполнила 15 итераций для достижения оптимального решения. Как и ожидалось, в результате получились группировки с дробным числом элементов. Чтобы действительно оценить возможности метода линейного программирования в применении к этой проблеме, придется подождать развития соответствующих методов целочисленного программирования.

В будущем мы надеемся проанализировать следующее: 1) какой объем вычислений сберегается, если исходить из точки, близкой к равновесию, а не из произвольной точки, меняющейся с изменением времени цикла, и 2) насколько полезно начальное балансирование в диапазоне интересующих нас производственных скоростей, что позволит оценить затраты труда при операции сборки.

11. Добавление зонного ограничения

Одно из преимуществ эвристического подхода к сложным проблемам с принятием решений заключается в способности переопределения задачи, добавления или исключения ограничений, накладываемых на решение проблемы. В качестве характерного примера рассмотрим зонные ограничения, исследованные в работе Митчелла [741].



Фиг. 8 Выход фазы II 21 элементной задачи с зональными ограничениями

Для каждой элементарной задачи к такому зонному ограничению относят одну или более зон, в которых только и может быть выполнена данная задача, т. е. ограничение отражает реальные ограничения производственного процесса.

Для включения нового ограничения основной процесс группирования (см. разд. 8) видоизменяется так, чтобы образовывать группы только в пределах определенной зоны или зон. Теперь программа рассматривает рабочие места так, чтобы не принимать во внимание группы, элементы которых не принадлежат определенным зонам. Изменения эти могут быть проведены достаточно просто, не затрагивая другие части программы. Зонное ограничение было промоделировано вручную на нескольких примерах. В табл. 3 представлена зональность элементов в при-

Таблица 3

Зоны	Элементарные задачи
1	*U*1, *U*2, *U*3, *U*4, *U*5, *U*7, *U*8, *U*21
3	*U*3, *U*4, *U*6, *U*7, *U*8, *U*9, *U*10, *U*11, *U*12, *U*13, *U*15, *U*16, *U*18
4	*U*13, *U*14, *U*15, *U*16, *U*17, *U*18, *U*19, *U*20

мере фиг. 1. На фиг. 8 отражены результаты видоизменения фазы II программы (при заданном времени цикла, равном 20 единицам).

12. Вопросы программирования задачи балансирования сборочного конвейера

Представленная здесь программа балансирования сборочного конвейера была описана на языке интерпретирующей системы IPL-IV для вычислительной машины JOHNNIAC. Система IPL-IV использует около 1200 из 4096 слов оперативного запоминающего устройства вычислительной машины JOHNNIAC и около 650 из 12 288 слов долговременной памяти на магнитных барабанах.

Постоянное обращение к долговременной памяти для хранения текущих данных программы позволяет вести интерпретацию программы со скоростью 9000—10 000 команд IPL в 1 *мин*. Хотя описание IPL-IV не опубликовано, родственный ему язык IPL-V, разработанный для вычислительных машин IBM-650, 704 и 709, хорошо известен [800, 805].

Преимущества применения информационных языков[1] в качестве средства исследований в различных областях техники для решения сложных задач и задач с плохой структурой подробно рассмотрены в другой работе [1076]. Основная проблема здесь состоит в том, что «параметрами», которые программисту хотелось бы свободно менять, по существу являются стратегии программы — основные принципы, практические правила и т. д. Разнообразие этих принципов не может быть оценено на ранней стадии, когда методика решения задач делает только первые шаги и ищет приемлемые методы для частного вида задач.

Для использования вычислительных машин необходимы такие системы, которые позволяли бы нам заранее не определять

[1] Такие информационные языки включают разработанную группу языков IPL (Ньюэлл и Тонг [800], Ньюэлл и др [805]); язык LISP (Маккарти [673]); COMIT (Ингве [1177]) и FORTRAN (Гелернтер и Рочестер [363]).

порядок запоминания, распределения информации и т. д. Такие системы должны позволить (возможно, за счет потери в быстродействии и эффективности использования памяти) предъявлять машине на первых порах задачи и методы решения в неполностью формализованном виде.

Списковая структура информационных языков программирования является первой ступенью в создании таких систем. Сама программа балансирования сборочного конвейера занимает примерно 6700 ячеек, из которых около 6100 размещаются во внешней памяти.

Отдав предпочтение быстрому получению результатов по сравнению с поиском эффективных процедур, мы закодировали всю структуру задачи в виде списков. После того как была разработана удовлетворительная по своим характеристикам программа, экономия машинной памяти и машинного времени могла бы быть осуществлена за счет перекодирования некоторой части информации в матричной форме. В этой связи интерес представляют статьи Брайтона [174] и Маримонта [660]. Такая запись потребовала бы изменения лишь той части программы, которая определяет значение некоторых свойств элементов. Остальная часть программы независима и должна при этом остаться без изменений. Дальнейшее сокращение операционного времени может быть получено записью в основном машинном языке некоторых операций интерпретирующей системы.

Используя язык IPL-V на машине IBM-704, можно было бы получить результаты данной программы в 5 раз быстрее, чем на машине JOHNNIAC. При такой скорости (около 1 часа для получения максимума при балансировании линии) этот подход, по-видимому, может рассматриваться как экономически рентабельный при решении некоторых сложных промышленных задач.

13. Заключение

Цели настоящего исследования шире, чем просто разработка метода для решения задачи балансирования конвейерной линии. Мы изложили в статье возможность эвристического подхода к решению сложной проблемы в конкретной области производства и рассмотрели использование новой вычислительной методики при осуществлении такого подхода. Как всегда бывает при кратком изложении, не освещенными остались многие интересные стороны этой проблемы. Но, основываясь на изложенном выше опыте, можно сделать вывод, что сочетание эвристического подхода с применением вычислительных машин, несомненно, явится полезным инструментом при решении сложных проблем.

В экономическом отношении работа программы не сопоставлялась с ценой труда инженера, который выполнил бы аналогичную работу. Однако для истинной оценки метода необходимо принять в расчет: 1) возможность сокращения числа людей, занятых на конвейере (этот вопрос еще полностью не исследован); 2) важность быстрой балансировки конвейера при большой производственной скорости и 3) значение возможности переключить квалифицированные инженерные кадры на другую, более творческую работу.

Наше исследование, кроме того, показало, что современные информационные языки, хотя и позволяют успешнее, чем обычные машинные языки, программировать такого рода задачи, все еще остаются сравнительно примитивными. Хотя трудно представить себе разработку этой программы в языке менее мощном, чем IPL-IV, все же потребуется еще очень много усилий, чтобы научиться поручать вычислительной машине процесс решения задач. Дальнейшие исследования в этой области нуждаются прежде всего в развитии методов решения задач с использованием при этом вычислительных машин как инструмента для расшифровки смысла вводимых процедур. Наличие этой техники обеспечит возможность более полного развития методов эвристического программирования.

Приложение А

Фаза I — 21-элементная задача

1. Рассмотреть цепь $U1$.
  2. Рассмотреть набор $U2$, $U3$.
    3. Рассмотреть цепь $U2$
      4. Нельзя ли свести $U2$ к $U21$?
    3 Рассмотреть цепь $U3$.
      4. Определить цепь $V0=U3$, $U4$.
      4 Рассмотреть набор $U5$, $U21$.
        5 Нельзя ли свести к другим $U21$?
    3. Рассмотреть Z $V0$, $U2$.
      4. Рассмотреть цепь $U21$.
      4 Рассмотреть цепь $U5$.
        5 Рассмотреть набор $U6$, $U7$.
          6 Рассмотреть цепь $U6$.
            7 Нельзя ли свести $U6$ к $U8$?
          6. Рассмотреть цепь $U7$.
            7. Рассмотреть набор $U8$, $U14$.
              8 Нельзя ли свести к другим $U8$?
          6. Рассмотреть Z $U7$, $U6$.
            7. Вычеркнуть ограничение от $U14$ к $U19$.
            7 Рассмотреть цепь $U8$.
              8. Определить цепь $V1=U8$, $U9$.

8. Рассмотреть набор *U*10, *U*11, *U*12, *U*13.
9. Определить набор *V*2=*U*10, *U*11, *U*12.
9. Рассмотреть цепь *V*2.
10. Определить цепь *V*3=*V*2, *U*15.
10. Рассмотреть набор *U*16, *U*18.
11. Нельзя ли свести к другим *U*18?
9. Рассмотреть цепь *U*13.
10. Рассмотреть набор *U*17, *U*18.
11. Нельзя ли свести к другим *U*17?
11 Нельзя ли свести к другим *U*18?
9. Рассмотреть *Z U*13, *V*3
10. Вычеркнуть ограничение от *U*13 к *U*17.
10. Рассмотреть цепь *U*18.
11. Определить цепь *V*4=*U*18, *U*19.
10. Рассмотреть цепь *U*16.
11. Определить цепь *V*5=*U*16, *U*17, *U*20.
10. Определить *Z V*6=*U*13, *V*4, *V*3, *V*5.
8. Рассмотреть цепь *V*6.
8. Определить цепь *V*7=*U*8, *U*9, *V*6.
7. Рассмотреть цепь *U*14.
7. Определить *Z V*8=*U*6, *V*7, *U*7, *U*14.
5. Рассмотреть цепь *V*8.
5. Определить цепь *V*9=*U*5, *V*8.
4. Определить *Z V*10=*U*2, *U*21, *V*0, *V*9.
2. Рассмотреть цепь *V*10.
2. Определить цепь *V*11=*U*1, *V*10.

Приложение Б

Фаза II — 21-элементная задача

Время цикла 20 единиц, процент полезного времени 90

1. Рассмотреть *V*11 — составляющие *U*1, *V*10; имеется в распоряжении 6 человек.
2. Применить прямой перенос.
3. Объединить *V*0, *U*1.
4. Переопределить цепь *V*11=*V*1, *V*13.
4. Определить цепь *V*1=*U*1, *U*3, *U*4.
4. Определить цепь *V*12=*U*2, *U*21.
4. Определить набор *V*13=*V*12, *V*9.
2. Назначить 1 человека к *V*1 и 5 человек к *V*13.
3. Рассмотреть *V*1 — составляющие *U*1, *U*3, *U*4; имеется 1 человек
4. Именовать *V*1 рабочим местом, время равно 18.
3. Рассмотреть *V*13 — составляющие *V*12, *V*9; имеется 5 человек.
4. Назначить 1 человека к *V*12 и 4 человека к *V*9.
5. Рассмотреть *V*12 — составляющие *U*2, *U*21; имеется 1 человек.
6. Объединить *U*5, *V*12.
7. Переопределить набор *V*13=*V*14, *V*8.
7. Определить *V*14=*U*5, *V*12.
6. Именовать *V*14 рабочим местом, время равно 19.
5. Рассмотреть *V*8, составляющие *U*6, *V*7, *U*7, *U*14
6. Включить *U*14, *U*6 в список «ожидания».
6. Назначить 3 человека к *V*7 и 1 человека к *U*7.

7. Рассмотреть *U*7; имеется 1 человек.
8. Объединить *U*7, *U*14, *U*6.
9. Определить набор *V*18=*U*6, *V*15.
9. Определить цепь *V*17=*V*18, *V*7.
9. Определить цепь *V*15=*U*7, *U*14.
9. Переопределить цепь *V*13=*V*14, *V*17.
8. Именовать *V*18 рабочим местом, время равно 15.
7. Рассмотреть *V*7 — составляющие *U*8, *U*9, *V*6; имеются 3 человека.
8. Применить метод последовательной группировки.
9. Объединить *U*8, *U*9, *V*2.
10. Переопределить цепь *V*7=*V*20, *V*6.
10. Определить *V*20=*U*8, *U*9, *V*2.
10. Переопределить *Z V*6=*U*13, *V*14, *U*15, *V*5.
8. Назначить 1 человека к *V*20 и 2 человека к *V*6.
9. Рассмотреть *V*20 — составляющие *U*8, *U*9, *V*2; имеется 1 человек.
10. Именовать *V*20 рабочим местом, время равно 17.
9. Рассмотреть *V*6 — составляющие *U*13, *V*4, *U*15, *V*5; имеются 2 человека.
10. Применить последовательную группировку.
11. Объединить *U*15, *U*13, *V*4, *U*16.
12. Определить цепь *V*23=*V*25, *V*5.
12. Определить цепь *V*25=*V*21, *U*18, *U*19, *U*16.
12. Переопределить *V*5=*U*17, *U*20
12. Определить набор *V*21=*U*13, *U*15.
12. Переопределить цепь *V*7=*V*20, *V*23
10. Назначить 1 человека к *V*25 и 1 человека к *V*5.
11. Рассмотреть *V*5 — составляющие *U*17, *U*20; имеется 1 человек.
12. Именовать *V*5 рабочим местом, время равно 16.
11. Рассмотреть *V*25 — составляющие *V*21, *U*18, *U*19, *U*16; имеется 1 человек.
12. Именовать *V*25 рабочим местом, время равно 20.
7. Определить цепь *V*17=*V*18, *V*20, *V*25, *V*5.

Приложение В

Фаза III — 21-элементная задача

Время цикла 20 единиц, процент полезного времени 90

1. Определить наименьшее возможное время для напряженного рабочего места=105/6=17+=18 единиц времени.
1. Рассмотреть *V*25, время равно 20.
2. Применить прямой перенос от *V*25 к *V*5.
3. Объединить *U*19, *V*5.
4. Переопределить цепь *V*17=*V*18, *V*20, *V*25, *V*27.
4. Переопределить цепь *V*25=*V*21, *U*18, *U*16.
4. Переопределить цепь *V*27=*U*19, *U*17, *U*20.
1. Рассмотреть *V*14, время равно 19.
2 Применить прямой перенос от *V*14 к *V*18.
2. Применить прямой перенос от *V*14 к *V*1.

2. Применить метод обмена $V14$ и $V18$.
3. Объединить $V14$, $U7$.
4. Переопределить набор $V13 = V28$, $V17$
4. Переопределить набор $V28 = V12$, $U5$, $U7$
4. Переопределить набор $V18 = U6$, $U14$.
3. Объединить $V12$, $V18$
4. Переопределить набор $V28 = U5$, $U7$.
4. Переопределить цепь $V17 = V30$, $V20$, $V25$, $V27$.
4. Определить набор $V30 = V12$, $U6$, $U14$.

7. ЭВРИСТИЧЕСКАЯ ПРОГРАММА, РЕШАЮЩАЯ ЗАДАЧИ СИМВОЛИЧЕСКОГО ИНТЕГРИРОВАНИЯ В ОБЪЕМЕ ПЕРВОГО КУРСА УНИВЕРСИТЕТА

Д. Слэйджл

Введение

Для решения элементарных задач интегрального исчисления на уровне хорошего первокурсника была составлена программа для большой быстродействующей универсальной вычислительной машины IBM-7090. Программа называется SAINT («Символический Автоматический ИНТегратор») и написана на языке LISP [674]. В данной работе, в основном соответствующей докторской диссертации автора [1024], рассматриваются программа SAINT и ее работа.

Приведем прежде всего несколько типичных примеров, отражающих внешнее поведение программы, чтобы читатель мог представлять себе ее работу более конкретно. Пусть SAINT читает на перфокартах соответствующим образом закодированную задачу: взять неопределенный интеграл $\int xe^{x^2}dx$. Менее чем через полторы минуты SAINT печатает ответ: $^1/_2\, e^{x^2}$. За исключением случаев, которые будут особо оговорены ниже, все задачи, рассмотренные в этой статье, были решены программой SAINT (заметим, что SAINT пренебрегает постоянной интегрирования, и мы здесь будем поступать точно так же). Менее 1 *мин* потребовалось программе для рассмотрения интеграла $\int e^{x^2}dx$ (который не берется в элементарных функциях), причем был выдан ответ, что она не может решить эту задачу.

Программа SAINT производит неопределенное интегрирование (операция, обратная дифференцированию, или отыскание первообразной функции). Кроме того, она берет определенные и кратные интегралы, когда они являются тривиальным обобщением неопределенного интеграла. SAINT рассматривает такие подынтегральные выражения, которые являются элементарными

функциями действительного переменного, или, как мы для краткости будем говорить, элементарными функциями. В курсе интегрального исчисления для первого года обучения обычно рассматриваются только элементарные функции. Для SAINT исключением являются гиперболические функции, которые программа не рассматривает. Элементарные функции определяются рекурсивно:

1) любая константа — элементарная функция;
2) переменная — элементарная функция;
3) сумма или произведение элементарных функций — элементарная функция;
4) f^{φ} — элементарная функция, если f и φ — элементарные функции;
5) тригонометрическая функция от элементарной функции — элементарная функция;
6) логарифмическая или обратная тригонометрическая функция от элементарной функции (ограниченная в некоторой области, если это необходимо) — элементарная функция.

В процессе работы программа SAINT использует 26 стандартных форм. Она применяет 18 типов преобразований, включая методы интегрирования по частям и замены переменных (но исключая среди прочих метод разложения на простые дроби) Так как программа SAINT использует эвристические методы, то она по определению является эвристической программой. В этой работе под эвристическим методом (или просто эвристикой) понимается метод, который помогает находить решение задачи путем правдоподобных, но не безошибочных предположений о том, что́ лучше всего делать на следующем шаге поиска решения.

Процедура неопределенного интегрирования в программе SAINT

В этом разделе мы опишем, как SAINT берет неопределенные интегралы. Структура управляющей программы SAINT похожа на структуру управляющей программы «Логика-теоретика» Ньюэлла, Шоу и Саймона [789]; это следует иметь в виду при рассмотрении п. 14 (особенно фиг. 3). Выражение «непосредственное решение», дважды упоминающееся на фиг. 3, можно грубо пояснить следующим образом. Как только выработана новая цель g, SAINT использует прямые методы, пытаясь найти ее решение. Когда это достигнуто, SAINT может занести g или некоторые подцели g во «временный список целей». Если цель g достигнута, делается попытка достичь первоначальной цели. Автор в своей работе [1024], помимо прочего, дал

полное описание системы вместе с детально разработанными примерами и соображениями о дальнейших исследованиях.

В качестве конкретного примера бегло рассмотрим вычисление интеграла

$$\int \frac{x^4}{(1-x^2)^{5/2}}\,dx,$$

выполненное машиной за 11 *мин*. Предложение, сделанное программой SAINT на первом шаге, таково: попытаться произвести замену переменных $y=\arcsin x$, которая преобразует исходное выражение к виду

$$\int \frac{\sin^4 y}{\cos^4 y}\,dy.$$

На втором шаге SAINT делает три возможных предположения:

(А) использовать тригонометрическое тождество

$$\int \frac{\sin^4 y}{\cos^4 y}\,dy = \int \operatorname{tg}^4 y\,dy;$$

(Б) использовать тригонометрическое тождество

$$\int \frac{\sin^4 y}{\cos^4 y}\,dy = \int \operatorname{ctg}^{-4} y\,dy;$$

(В) использовать замену переменных $z=\operatorname{tg}\frac{y}{2}$:

$$\int \frac{\sin^4 y}{\cos^4 y}\,dy = \int 32\,\frac{z^4}{(1+z^2)(1-z^2)^4}\,dz.$$

Здесь SAINT немедленно выносит 32 за знак интеграла.

Оценив, что (А) — самая простая из всех трех задач, SAINT предлагает замену $z=\operatorname{tg} y$, которая дает

$$\int \operatorname{tg}^4 y\,dy = \int \frac{z^4}{1+z^2}\,dz.$$

SAINT немедленно преобразует это выражение:

$$\int \frac{z^4}{1+z^2}\,dz = \int\left(-1+z^2+\frac{1}{1+z^2}\right)dz = -z+\frac{z^3}{3}+\int\frac{dz}{1+z^2}.$$

Неправильно оценив, что (Б) проще, чем

$$\int \frac{dz}{1+z^2}\,dz,$$

SAINT временно оставляет последнее выражение и переходит к (Б). Подстановка $z=\operatorname{ctg} y$ дает

$$\int \operatorname{ctg}^{-4} y\,dy = -\int \frac{dz}{z^4(1+z^2)}.$$

Теперь SAINT определяет, что интеграл

$$\int \frac{dz}{1+z^2}$$

проще и, произведя подстановку $w=\operatorname{arctg} z$, получает $\int dw$. Это выражение SAINT сразу интегрирует и, выполнив обратные подстановки, получает решение первоначальной задачи:

$$\int \frac{x^4}{(1-x^2)^{5/2}}\,dx = \arcsin x + {}^1/_3 \operatorname{tg}^3 \arcsin x - \operatorname{tg}\arcsin x.$$

Процесс неопределенного интегрирования можно описать следующим образом.

1. Цели

При каждом применении настоящей процедуры целями являются решения определенных задач, а именно выполнение интегрирования при дополнительных условиях. Позже будет описано, как программа вырабатывает цели, оперирует с ними и достигает их. Сейчас же мы ограничимся описанием того, что ниже называется «первоначальной целью», которая состоит из первоначально заданного подынтегрального выражения и переменной интегрирования.

2. Список целей

Первоначальная цель ставится первой в списке, называемом списком целей. Время от времени могут вырабатываться новые цели. Каждая вновь выработанная цель заносится в конец имеющегося списка целей.

3. Стандартные формы

Как только подынтегральное выражение вновь выработанной цели становится «стандартной формой», цель сразу же достигается с помощью подстановки. Подынтегральное выражение называется стандартной формой, если оно представляет собой частный случай одной из форм, взятой из определенного набора

форм. Например, $\int 2^x dx$ является частным случаем $\int c^v dv = c^v/\ln c$ и поэтому имеет решение $2^x/\ln 2$. Программа SAINT в современном виде использует 26 стандартных форм.

4. Преобразования алгоритмического вида

Как только оказывается, что подынтегральное выражение не является стандартной формой, делается попытка проверить, нельзя ли применить к нему преобразование алгоритмического вида. Преобразование алгоритмического вида — это преобразование, которое, когда оно применимо, всегда или почти всегда является целесообразным. Для цели преобразование называется целесообразным, если оно является шагом, приближающим достижение цели. Ниже приводятся три из используемых программой SAINT восьми преобразований алгоритмического вида.

а) постоянный множитель:

$$\int cg(v)\,dv = c\int g(v)\,dv;$$

б) разложение:

$$\int \sum_i g_i(v)\,dv = \sum_i \int g_i(v)\,dv;$$

в) линейная подстановка: если интеграл имеет вид

$$\int f(c_1 + c_2 v)\,dv,$$

то применяется подстановка $u = c_1 + c_2 v$, приводящая к интегралу вида

$$\frac{1}{c_2}\int f(u)\,du.$$

Например, в

$$\int \frac{\cos 3x}{(1-\sin 3x)^2}\,dx$$

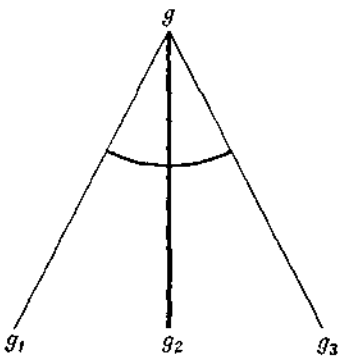
производится подстановка $y = 3x$.

5. Дерево целей „И — ИЛИ“

В результате применения к цели эвристического преобразования (см. п. 11) или преобразования алгоритмического вида вырабатываются новые цели. Эти цели опять могут в свою очередь вырабатывать цели, так что возникает определенная иерархия.

Такую иерархию удобно представлять в виде графа или дерева, ветви которого направлены вниз. При этом для облегчения

понимания используется терминология, аналогичная той, которая принята для описания обычных и генеалогических деревьев. Например, отсечение ветвей, живая и мертвая ветви, отпрыск, родитель, потомок, предок и т. д. Предположим, что нам необходимо проинтегрировать некоторое выражение или, в более общем виде, что у нас есть некоторая цель g, которую мы будем графически представлять точкой. Цель может быть преобразована в одну или несколько подцелей, которые могут быть связаны с этой целью различным образом. Рассматриваемый процесс интегрирования включает два типа связей: «И» и «ИЛИ».



Фиг. 1. Связь «И».

а) Связь «И»

Между целью и по крайней мере двумя подцелями существует связь «И», если достижение всех этих подцелей влечет за собой достижение цели. На фиг. 1 изображены связи цели с тремя подцелями. Дуга, соединяющая три ветви, обозначает связь «И».

б) Связь «ИЛИ»

Между целью и ее подцелями существует связь «ИЛИ», когда достижение любой из подцелей влечет за собой достижение цели. Пример такой связи будет дан ниже.

Из этих двух основных связей можно построить более сложные связи между целями, например такие, как изображенные на фиг. 2 (п. 12).

6. Временный список целей

Попытки выработать новые цели начинаются с процедуры IMSLN (сокращение от «Immediate Solution» — непосредственное решение (англ.)), которая описывается в п. 13. Любая цель, встретившаяся на пути IMSLN, которая не является ни стандартной формой, ни выражением, поддающимся преобразованию алгоритмического вида, заносится в конец «временного списка целей» (не путать со «списком целей»!). Затем эта цель переносится в «эвристический список целей», который описывается ниже (п. 10). Если бы эта цель заносилась не во временный

список целей, а непосредственно в эвристический список целей, то на отыскание описания цели терялось бы дополнительное время (п. 8).

7. Роль ограниченности рабочей памяти

Емкость рабочей памяти является дополнительным ограничением, налагаемым на характер доступных задач. Перед тем как приступить к применению эвристических преобразований, необходимо проверить, не исчерпана ли рабочая память. Если свободного места в памяти нет, SAINT сообщает об этом и прекращает работу. Хотя можно учитывать также и другие типы ограничений, например по времени, SAINT имеет дело только с одним ограничением — емкостью рабочей памяти. В случае когда человек берет интеграл «вручную», рабочая память измеряется числом листов бумаги или числом строк, используемых для записи окончательного и всех промежуточных результатов.

8. Описание цели

Цель извлекается из временного списка целей для получения ее «описания», т. е. упорядоченного списка «характеристик» цели. Характеристика цели — это признак, который может быть полезен либо при оценке «стоимости» ее достижения, либо для выбора соответствующих эвристических преобразований (п. 11). В программе SAINT описание состоит из 11 характеристик подынтегрального выражения [1024], включая его функциональный тип (рациональная, алгебраическая функция, рациональная функция синусов и косинусов и т. д.) и его глубину. «Глубиной» подынтегрального выражения является максимальный уровень сложности функции. Ниже приводятся значения глубины для некоторых функций:

x	имеет	глубину	0,
x^2	„	„	1,
e^{x^2}	„	„	2,
xe^{x^2}	„	„	3.

Легко видеть, что эта характеристика помогает грубо оценить сложность задачи.

9. Относительная оценка стоимости

Хотя можно попытаться использовать и другие оценки, мы в качестве относительной оценки стоимости цели просто выберем глубину ее подынтегрального выражения. Это основано на

том факте, что обычно чем глубже подынтегральное выражение, тем больше потребуется времени, памяти и т. д. для достижения такой цели.

10. Эвристический список целей

Список целей, требующих применения эвристических преобразований, или, короче, эвристический список целей, — это упорядоченный список тех целей, которые не являются ни стандартной формой, ни целями, допускающими преобразования алгоритмического вида. Цель из эвристического списка целей называется эвристической целью. Новые цели такого рода помещаются в списке целей в порядке возрастания относительных оценок стоимости.

11. Эвристические преобразования

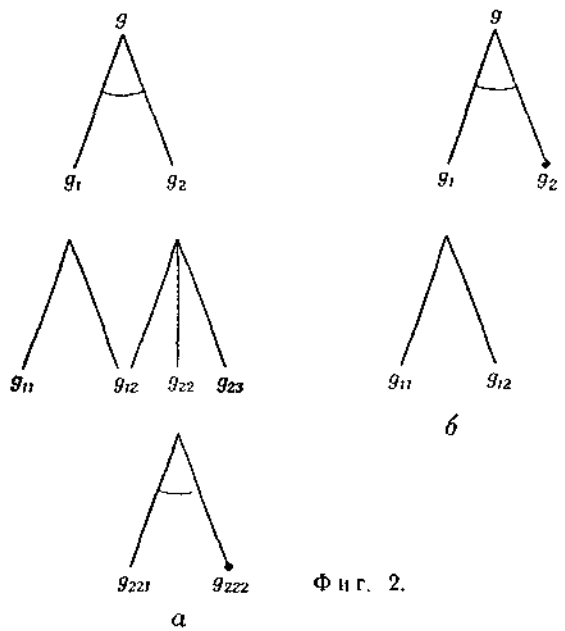
Преобразование цели называется эвристическим, когда, даже если оно пригодно и правдоподобно, все же существует некоторый риск, что оно не окажется целесообразным следующим шагом. Преобразование будет нецелесообразным, либо когда оно удаляет от решения, либо когда некоторое другое преобразование более целесообразно. Эвристические преобразования аналогичны методам отделения и прямого и обратного цепеобразования в «Логике-теоретике» Ньюэлла, Шоу и Саймона [789]. Используемые в SAINT 10 типов эвристических преобразований [1024] дают возможность применять преобразования подынтегрального выражения, подстановки и интегрирования по частям. Ниже дается только наиболее успешная эвристика — «замена подвыражения, производная которого является делителем подынтегрального выражения».

Пусть $g(v)$ — подынтегральное выражение. Для каждого не постоянного, не линейного выражения $s(v)$, не являющегося разностью двух выражений, не имеющего постоянного сомножителя и обладающего тем свойством, что число непостоянных сомножителей дроби $g(v)/s'(v)$ (после сокращения) меньше, чем число сомножителей $g(v)$, испытаем подстановку $u=s(v)$. Например, в случае $xe^{x^2}dx$ произведем подстановку $u=x^2$. (Когда SAINT пытался решить эту задачу, он использовал эту эвристику, но, к моему удивлению, воспользовался заменой $u=e^{x^2}$, которая даже несколько лучше.)

12. Отсечение ветвей на дереве целей

Как только достигнута некоторая цель g, дерево целей «усекается», т. е. некоторые тесно связанные с g цели автоматически достигаются, а другие, ставшие излишними, отсекаются.

Процедура отсечения становится понятной на следующем примере. Из фиг. 2, *а* видно, что достижение цели g_{221} позволит достигнуть цель g_{22} (так как цель g_{222} уже достигнута, что показано черной точкой). В свою очередь достижение g_{22} позволяет достигнуть цель g_2 (так как здесь имеет место связь «ИЛИ»). Так как достижение g_2 делает излишней цель g_{23}, последняя отсекается. Однако другая из порожденных g_2 целей, g_{12}, не отбрасывается, так как у нее есть и другой «родитель» — цель g_1,



Фиг. 2.

и поэтому первоначальная цель g, которая еще не достигнута, является ее прямым «предком». Первоначальной цели g нельзя достичь, достигнув одной лишь цели g_2, так как здесь имеет место связь «И», а цель g_1 еще не достигнута. Поэтому процесс отсечения ветвей будет таким, как показано на фиг. 2, *б*. Как только будет достигнута одна из целей g_{11} или g_{12}, будет достигнута и первоначальная цель g.

13. Попытки отыскать непосредственное решение

Как только выработана новая цель g, SAINT пытается достигнуть ее, используя прямые методы. Для этого SAINT может занести g или некоторые подцели, порожденные g, во временный список целей. Если g достигнута, совершается попытка достичь первоначальной цели

14. Структура управляющей программы

Ниже дается точное описание того, как все эти операции (пп. 1—13) связываются для выполнения процесса интегрирования. Первоначальная цель задается тремя выражениями, а именно: подынтегральным выражением, переменной интегрирования и емкостью рабочей памяти. Процедура имеет следующий вид (фиг. 3):

1. Если попытка отыскать непосредственное решение первоначальной цели является успешной, возвратись в исходное положение и сообщи ответ (взятый неопределенный интеграл).
2. Если исчерпана рабочая память, сообщи о неудаче.
3. Получи из временного списка целей и поставь в соответствие каждой цели ее описание и относительную оценку стоимости. Каждую цель из временного списка целей занеси в эвристический список целей согласно ее относительной оценке стоимости. Если в эвристическом списке целей не осталось целей, сообщи о неудаче.
4. Извлеки из эвристического списка целей очередную цель g_i, которая будет предметом рассмотрения следующего внутреннего цикла.
5. Если не осталось применимых к g_i эвристических преобразований, переходи опять к операции 2.
6. Примени следующее по порядку эвристическое преобразование, пригодное для g_i. Как только в результате применения такого преобразования будет выработана новая цель g, занеси ее в список целей и попытайся найти ее непосредственное решение. При этом возможны три различных случая: если в результате этой попытки достигается первоначальная цель, возвратись в исходное положение и сообщи ответ; если она не достигается, но достигается цель g_i, перейди к операции 2; в прочих случаях перейди к операции 5.

Процесс определенного интегрирования

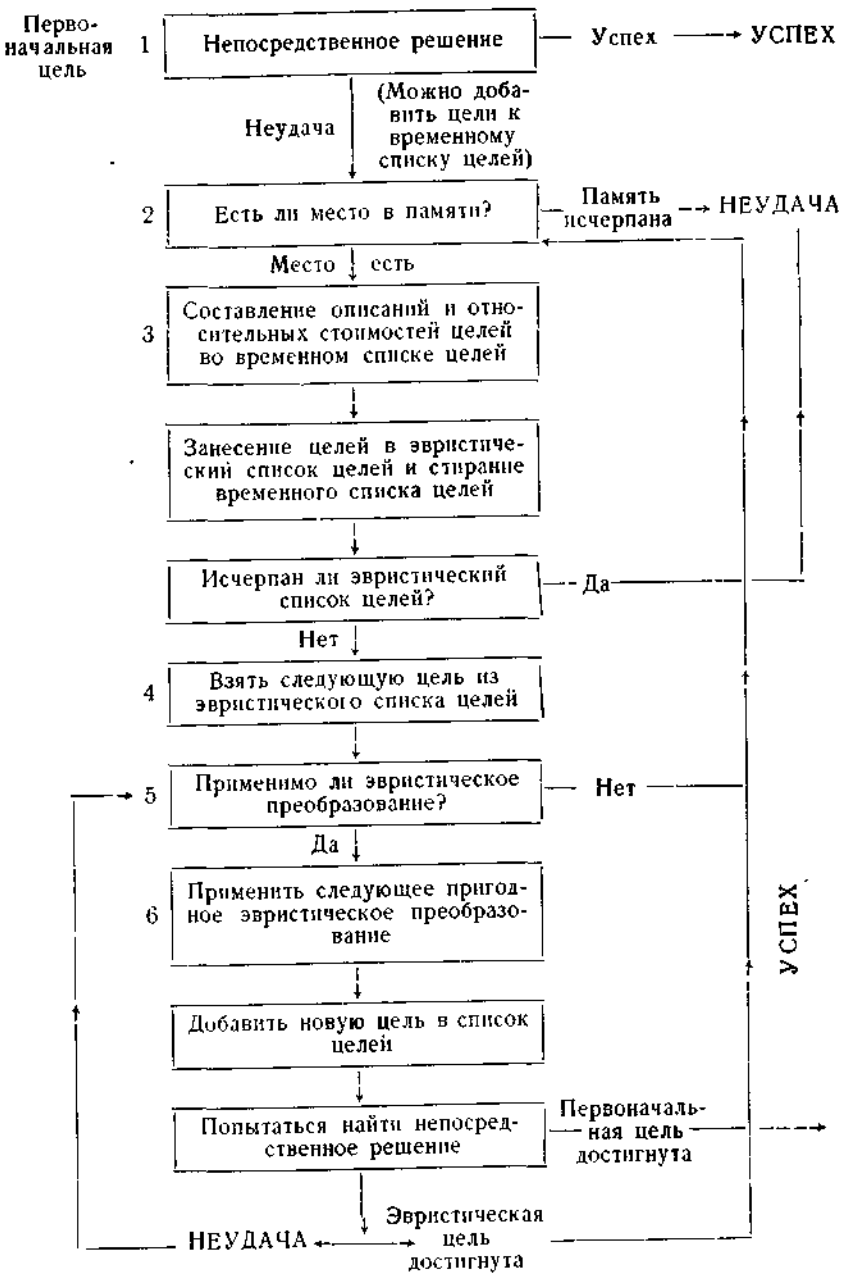
SAINT может вычислять также некоторые определенные интегралы, для чего сначала находятся соответствующие неопределенные интегралы. Так, например, для задачи

$$\int_0^3 x\sqrt{x^2+16}\,dx$$

SAINT сначала находит неопределенный интеграл

$$\int x\sqrt{x^2+16}\,dx = (1/3)(x^2+16)^{3/2}\,.$$

Затем SAINT подставляет пределы интегрирования и получает ответ: 61/3.



Фиг 3

Процесс кратного интегрирования

SAINT может осуществлять и кратное интегрирование, при условии что он может вычислить требуемые определенные интегралы. Например,

$$\int_0^1 \int_{y^2}^{2-y^2} dx\,dy.$$

Эксперименты с программой SAINT и их результаты

В этом разделе описывается несколько типичных случаев наблюдаемого поведения программы SAINT и изменение поведения программы при одной ее модификации. В работе [1024] описаны также другие эксперименты и сообщаются некоторые дополнительные детали. Для экспериментального исследования поведения программы SAINT было рассмотрено 86 задач. Главным образом с целью отладки автором были отобраны или составлены 32 задачи, которые, как с уверенностью можно было ожидать, SAINT полностью решит. Оставшиеся 54 задачи были выбраны более объективно, а именно из экзаменационных задач по интегральному исчислению, предложенных на первом курсе Массачусетского технологического института (МТИ).

Эта работа была проделана ассистентом автора Дж. Шапиро, которому было дано указание отбирать наиболее разнообразные и трудные задачи с тем условием, чтобы для их решения не требовалось применения метода разложения на простые дроби. Для оценки поведения программы мы выбрали следующие показатели.

1. Мощность

Мощность некоторого варианта SAINT указывает на размер класса задач, которые он способен решать.

2. Время

Все упоминаемые данные о времени относятся к машине IBM-7090.

3. Число подцелей и неиспользованных подцелей

Первоначальная цель не включается в число подцелей. Неиспользуемая подцель — это подцель, которая не нужна в цепочке решения.

4. Уровень

Уровнем решения является максимальный уровень подцели среди подцелей, появляющихся в процессе решения на дереве целей.

5. Эвристический уровень решения

Эта величина подобна «уровню», за исключением того, что рассматриваются только те ветви дерева целей, которые получены в результате эвристических преобразований, а не все ветви, полученные как эвристическими преобразованиями, так и преобразованиями алгоритмического вида.

А. Первоначальный (без модификаций) вариант SAINT

Программа SAINT, описанная в предыдущих разделах, пыталась решить все 86 задач, отобранные автором и его ассистентом. В этом эксперименте около половины машинного времени было затрачено на очищение уже ненужных мест в памяти для их повторного использования. Приблизительно половина оставшегося времени затрачена на «распознавание образов», т. е. на нахождение описания и на распознание того, является ли подынтегральное выражение стандартной формой, применимо ли преобразование алгоритмического вида или применимо эвристическое преобразование. Как и ожидалось, SAINT решила все 32 выбранные автором задачи. Из 54 задач МТИ SAINT решила 52 и быстро (менее чем за 1 *мин*) сообщила, что не может решить остальные две. Обе эти задачи исключены из приводимой ниже таблицы средних значений, характеризующих работу SAINT (табл. 1).

В этом разделе мы дополним таблицу средних показателей работы SAINT на задачах МТИ несколькими примерами поведения в крайних (наиболее легких и наиболее трудных) случаях. С этой целью рассматриваются только задачи МТИ, так как они выбраны более объективно. По-видимому, самой простой задачей для программы SAINT было вычислить интеграл

Таблица 1

Средние данные о работе программы SAINT

Задачи	Время, *мин*	Общее число подцелей	Число неиспользованных подцелей	Уровень	Эвристический уровень
32 задачи автора	3,3	6,4	2,0	3,5	1,0
52 задачи МТИ	2,0	4,7	0,8	2,9	0,8
Все 84 задачи	2,4	5,3	1,25	3,0	0,9

$$\int_1^2 \frac{dx}{x},$$

так как для ее решения вообще не вырабатывались подцели и потребовалось наименьшее время — 0,03 *мин*. Наибольшее время — 18 *мин* — потребовалось для вычисления интеграла

$$\int \frac{\sec^2 t}{1+\sec^2 t - 3\,\mathrm{tg}\, t}\,dt.$$

Максимальный эвристический уровень для решения этой задачи — 4. Другой задачей, решение которой имело эвристический уровень 4, было вычисление интеграла

$$\int \frac{x^4}{(1-x^2)^{1/2}}\,dx.$$

Максимальный эвристический уровень, полученный немодифицированным «Логиком-теоретиком», равен 2. Он достигался в двух из 39 решений [789]. SAINT выработала наибольшее число подцелей (18) и получила максимальный уровень (8) при решении задачи

$$\int (\sin x + \cos x)^2\,dx.$$

В 37 из 52 задач SAINT вырабатывала только те подцели, которые были необходимы в цепочке решения. Наилучшей среди 37 задач в этом отношении была задача

$$\int_{5/2}^{4} \int_{x(11-2x)}^{4(x-1)(5-x)} dy\,dx,$$

для которой SAINT выработала 16 подцелей, причем все они были необходимы в цепочке решения.

Б. BSAINT, т. е. программа SAINT, испытывающая эвристические цели в порядке их вырабатывания

Вместо того чтобы испытывать эвристические цели в порядке возрастания глубины, программа BSAINT испытывает их просто в порядке, в котором они вырабатываются. При оценке работы по времени и по числу неиспользованных подцелей SAINT работала лучше, чем BSAINT в трех из четырех задач, на которых проверялось различие в поведении.

Основные выводы

Излагаемые ниже выводы основываются на практическом опыте работы, а именно на экспериментах, описанных в предыдущих разделах, и на опыте автора, касающегося создания, структуры и характеристик работы SAINT. В пределах этого раздела для иллюстрации соответствующего вывода в скобках приводится ссылка на эксперименты, упоминаемые в предыдущих разделах.

1. Машина может обнаруживать «разумное» поведение при решении задач, т. е. поведение, которое, если бы его проявляли люди, называлось бы разумным (эксперименты из п. А).

2. В эвристическую программу легко включить программы для обработки дерева целей «И—ИЛИ» (такого, как в SAINT), которое часто оказывается полезным в сложных схемах достижения целей.

3. В SAINT распознавание образов играет очень важную роль в трех отношениях:

а) распознавание образов поглощает значительную часть программы и усилий по программированию;

б) распознавание образов используется часто, и в различных вариантах, например при отождествлении со стандартными формами, при определении преобразований алгоритмического вида и эвристических преобразований и при определении относительной оценки стоимости;

в) распознавание образов потребляет значительную часть времени при решении задач интегрирования (эксперименты из п. А).

4. В области решения задач очень полезно подразделение на три метода: стандартных форм, преобразований алгоритмического вида и эвристических преобразований. Стандартные формы в SAINT и «подстановка» в «Логике-теоретике» могут служить примерами процесса «непосредственного достижения цели», который, по-видимому, является фундаментальной частью любой схемы достижения целей. Роль исходных данных в таком процессе играет цель. Выходом будет ответ «нет» (цель еще не достигнута) или один или несколько из следующих трех ответов: 1) «да», 2) «как достигнуть цель» и 3) «достижение цели». В каждой области процесс непосредственного достижения цели должен применяться по-своему, и, так как этот процесс используется очень часто, он должен проходить очень быстро.

По-видимому, преобразования алгоритмического вида также составляют важную часть схемы достижения цели, но мы не будем их рассматривать, так как они представлены не во всех схемах, например их нет в «Логике-теоретике». Структура эвристических преобразований SAINT (соответствующих в «Логике-теоретике» методам отделения, прямого и обратного цепеобразования), по-видимому, часто удобна, но не является существенной частью схемы достижения целей.

5. Увеличение объема памяти в SAINT в 4 раза (сейчас он составляет 32 768 ячеек) могло бы привести к увеличению скорости работы в сотни раз, поскольку процедура очищения уже ненужных ячеек в памяти для их повторного использования, занимающая сейчас около половины рабочего времени, тогда лишится смысла. Кроме того, скорость работы «компилирующей» программы возросла бы приблизительно в 50 раз. Если бы особенно часто используемые функции оперирования с символом можно было бы заменить одной командой, было бы сэкономлено много машинного времени и объема памяти.

6. Скорость работы SAINT сравнима со скоростью, с которой решает задачи средний первокурсник (эксперименты из п. А). На новой большой быстродействующей машине, такой, как IBM-7030 (STRETCH), компилирующая и не улучшенная в остальном программа SAINT работала бы в 800 раз быстрее, что намного превышает скорость работы даже самого способного в этой области математика. По современным тарифам решение средней экзаменационной задачи МТИ на IBM-7090 программой SAINT обходится около 15 долл., что намного дороже, чем решение таких задач человеком. Однако решение программой SAINT задач на машине «СТРЕТЧ» обходилось бы всего лишь приблизительно в 2 долл., а если использовать компилирующую программу — только в 4 цента. Такое быстрое сокращение стоимости решения на машинах является общей тенденцией. В результате, даже не учитывая возможных улучшений программы SAINT, решение задачи машиной окажется намного дешевле, чем решение ее человеком.

РАЗДЕЛ 4

МАШИНЫ, ОТВЕЧАЮЩИЕ НА ВОПРОСЫ

Машины, отвечающие на вопросы, — это вычислительные программы, которым в пределах определенной темы можно задавать вопросы, сформулированные (с некоторыми ограничениями) на обычном человеческом языке. Проблема создания таких машин представляет как теоретический, так и практический интерес.

С теоретической точки зрения одной из наиболее интересных задач исследования искусственного разума является выяснение

принципов обработки информации, лежащих в основе процессов «понимания». Хотя многие относятся к употреблению этого термина настороженно, мы тем не менее будем его использовать, придавая ему обычный человеческий смысл. В этом заинтересованы и сами исследователи этой области, поскольку они занимаются поиском теории обработки информации в «разумном» устройстве, способном «понимать смысл» последовательностей символов, которыми являются предложения и вопросы, выраженные на нашем обычном языке.

С практической точки зрения во всей обширной области применения вычислительных машин — при поиске и восстановлении информации, в оснащенных вычислительными машинами библиотеках и учебных классах, в военных приложениях, в исследованиях по решению задач человеком совместно с машиной (чтобы назвать лишь некоторые сферы применения) — ощущается насущная необходимость в более непосредственной связи с машиной, более легкой и подвижной, чем это было возможно в прошлом.

В двух работах, помещенных в настоящем разделе, описываются важные начальные шаги в этом направлении. Программа «Бейсбол» («Baseball)» Грина и др. отвечает на вопросы, касающиеся «фактов и цифр» о бейсболе, которые хранятся в особом «Ежегоднике», заложенном в память машины[1]. Программа Линдсея «SAD SAM»[2] отвечает на вопросы о семейных родственных связях.

Основные отличия в подходе этих двух групп исследователей следующие:

1. В проекте «Бейсбол» «библиотека фактов» заранее записана в памяти машины в сравнительно легкой для обозрения форме. В проекте «SAD SAM» сведения о родственных отношениях между членами той или иной семьи не заданы заранее, а вводятся в машину в форме английских предложений. Программа, сравнивая и анализируя эти предложения, конструирует внутреннюю «карту» или «модель» родственных отношений, — информационную структуру, к которой она позднее обращается при ответах на вопросы о данной семье.

2. В проекте «Бейсбол» при определенных ограничениях, накладываемых на тип задаваемых вопросов и богатство заложенного в программу словаря, используется полноценный английский язык. В проекте «SAD SAM» используется упрощенный

[1] В США бейсбол является наиболее популярной из спортивных игр, привлекающей к себе такое же внимание, как в нашей стране футбол. — *Прим. ред.*

[2] Название программы происходит от сокращения «Sentence Appraiser and Diagrammer and Semantic Analysing Machine»

английский язык (система «Basic English», созданная для облегчения обучения языку[1]).

Обе программы, конечно, оперируют с чрезвычайно ограниченными наборами фактов. Однако на данной, исследовательской стадии выявление и тщательное изучение возможных фундаментальных механизмов «понимания» языка, пожалуй, более важно, чем примитивные попытки оперировать обширными наборами фактов.

8. «БЕЙСБОЛ» — ПРОГРАММА, АВТОМАТИЧЕСКИ ОТВЕЧАЮЩАЯ НА ВОПРОСЫ

Б. Грин мл., А. Вулф, К. Хомский и К. Лофери

Введение

Как правило, связь человека с вычислительными машинами осуществляется посредством целого ряда искусственных, стилизованных, однозначных языков, лучше приспособленных к машине, чем к человеку. Для многих будущих вычислительных систем из соображений удобства пользования и быстродействия потребуется, чтобы связь оператора с машиной осуществлялась на естественном языке. Администратору, военачальнику и ученому необходимо иметь возможность задавать машине вопросы на обычном человеческом языке и получать ответы в той же непосредственной форме. Программа «Бейсбол» является первым шагом в этом направлении.

«Бейсбол» — программа для вычислительной машины, способная отвечать на заданные по-английски вопросы, касающиеся сведений, хранящихся в памяти машины. Программа состоит из двух частей: *лингвистическая* часть считывает вопрос с перфокарты, производит его синтаксический анализ и определяет, какая информация относительно запрошенных сведений имеется; *исполнительная* часть выполняет поиск соответствующей информации в системе исходных данных и печатает ответ.

Программа написана на языке обработки информации IPL-V (Ньюэлл и Тонг [800]), в котором для ее представления используются списки и иерархии списков, называемые *списковыми структурами*. Как исходные данные, так и словарь

[1] «Basic English» включает ограниченный запас слов (300—1600) и простейшие грамматические конструкции. — *Прим. ред.*

являются такими структурами, в которых элементы информации выражены посредством пар признак — оценка[1], например: Команда (бейсбольная) = «Red Sox».

Программа работает в контексте сведений об играх в бейсбол. В данном случае такими сведениями являются месяц, день, место игры, команды, входящие в Американскую лигу бейсбола, и счет очков в каждой игре за год. При таких ограничениях контекста достаточно словаря малого объема, данные просты и их смысл ясен.

Чтобы эта первая программа была относительно простой, на характер входных вопросов наложены некоторые ограничения. Вопрос должен включать лишь одно простое предложение; запрещение сложноподчиненных предложений сильно упрощает синтаксический анализ. Исключаются логические связи, такие, как «and» (и), «or» (или) и «not» (не), а также слова, выражающие отношение, подобно «most» (большинство) и «highest» (наивысший). Наконец, исключаются вопросы, касающиеся временнóй последовательности фактов, например: «Выигрывала ли когда-нибудь команда „Red Sox" *шесть игр подряд*?». Эти ограничения носят временный характер, и в последующих вариантах программы они будут отброшены. Кроме того, они лишь незначительно уменьшают количество вопросов, на которые машина может ответить. Между простыми вопросами типа «Кому проиграла команда „Red Sox" пятого июля?» и сложными типа «Играла ли каждая команда в течение каждого месяца хотя бы один раз на каждом стадионе?» существует множество других доступных для машины вопросов.

Спецификационный список

Фундаментальным для работы программы является понятие «спецификационный список», или «спецификация». Этот список можно рассматривать как каноническое выражение смысла вопроса. Информацию, содержащуюся в вопросе, спецификация представляет в форме пар признак—оценка, например Team = = Red Sox (команда = «Red Sox»). Лингвистическая часть программы, анализируя вопрос, формирует спецификационный список Q. Этот список управляет действиями исполнительной части. Например, вопрос «Where did the Red Sox play on Yuly 7?»

[1] Оценка, или значение (Value), данного признака — одна из множества возможных конкретных реализаций данного общего понятия («признака»). — *Прим ред.*

(«Где играла команда „Red Sox" седьмого июля?») имеет следующую спецификацию:

Place	= ?	Место	= ?
Team	= Red Sox	Команда	= «Red Sox»
Month	= Yuly	Месяц	= июль
Day	= 7	День	= 7

Некоторые вопросы нельзя выразить посредством одних только главных признаков (месяц, число, место, счет и порядковый номер игры), а требуется некоторая модификация признаков. Например, в спецификационном списке вопроса «What teams won 10 games in July?» («Какие команды выиграли по десять игр в июле?») признак «Team» (команда) видоизменяется с помощью признака «Winning» (выигравшая), а признак «Game» (игра) — с помощью признака «Number of» (количество). В итоге получается следующая спецификация:

$\text{Team}_{(\text{winning})}$	= ?	$\text{Команда}_{(\text{выигравшая})}$	= ?
$\text{Game}_{(\text{number of})}$	= 10	$\text{Игра}_{(\text{количество})}$	= 10
Month	= July	Месяц	= июль

Словарь

При образовании спецификационного списка лингвистическая часть программы использует определения, выраженные парами признак—оценка, хранящиеся в ее словаре. Полное определение слова или идиомы включает: а) часть речи — для выяснения структуры вопроса; б) значение — для анализа содержания; в) указание о том, не является ли данное слово вопросительным, например «кто» и «сколько»; г) указание о том, не входит ли данное слово в какую-нибудь из уже записанных идиом. Идиомой является любой связный набор слов, функционирующий как целое и имеющий единое определение. Для слов и идиом имеются отдельные словари.

Значение слова может быть выражено в нескольких формах. Значение может характеризоваться основным или производным признаком с соответствующей оценкой. Например, значение слова *команда* — это «команда = (пустое место)», значение слов «*Red Sox*» — это «команда = „Red Sox"», а значение слова *кто* — это «команда = ?». Но значение может выражаться также подпрограммой, как в случае «модификаторов» типа *выигравшая*, *любая*, *шесть* или *сколько*. Слово *выигравшая*, например, имеет значение: подпрограмма A_1 = выигравшая. Эта подпрограмма, реализуемая при анализе содержания, присоединяет к признаку

соответствующего существительного модификатор *выигравшая*. Следует также учесть, что некоторые слова имеют больше одного значения. Значением слова *Бостон* может быть как «место=Бостон», так и «команда=„Red Sox"»[1]. Для таких слов в словаре, помимо перечня значений, имеется обозначение подпрограммы, которая в соответствии с контекстом, где встретилось данное слово, выбирает его подходящее значение. Наконец, с некоторыми словами, такими, как артикли (the), вспомогательные глаголы (did) и некоторые основные глаголы типа *играть* (play) и т. д., не связывается никакого значения.

Исходные данные

Исходные данные организованы в иерархическую структуру, подобную плану, каждый уровень которой содержит один или более элементов информации Отношения между этими элементами выражаются вхождением их в один и тот же список или в связанные друг с другом списки. Главенствующим или наивысшим уровнем структуры является признак «месяц». Для каждого месяца дальнейшие сведения разбиты по местам проведения игр. Под каждым обозначением места (для каждого месяца) расположен список всех игр, проведенных в этом месте в течение указанного месяца. Для отыскания полного набора сведений, касающихся одной игры, в иерархии прослеживается одна траектория, т. е. просматривается один список на каждом уровне. Каждая траектория содержит оценки для каждого из шести признаков, например:

Month	= July	Месяц	= июль
Place	= Boston	Место	= Бостон
Day	= 7	День	= 7
Game Serial №	= 96	Порядковый номер игры	= 96
(Team = Red Sox, Score	= 5)	(Команда = «Red Sox», счет очков	= 5)
(Team = Jankees, Score	= 3)	(Команда = «Jankees», счет очков	= 3)

Скобки указывают на то, что с каждой командой нужно сопоставить ее счет очков, что́ и производится путем занесения наименований команд и соответствующего счета очков в отдельный подсписок.

Исполнительные программы написаны так, чтобы они могли охватывать любую организацию данных. Например, для них

[1] «Red Sox» — команда города Бостона — *Прим ред.*

приемлема и такая непараллельная организация данных, при которой для всех игр вплоть до 31 июля данные располагаются, как описано выше, а для оставшейся части сезона разбиты прежде всего по местам проведения игр, а лишь затем — по месяцам. Одноуровневая структура, в которой каждая игра является списком всех ее пар признак—оценка, также приемлема для исполнительной программы. Возможность иерархической организации была предусмотрена ввиду ее общности и потенциальной эффективности. Основное правило состоит в том, что любая траектория через структуру данных, включающая один список на каждом уровне, должна содержать все факты, касающиеся одной игры. Кроме того, каждый признак, если он не входит в параллельные списки, может встретиться в любой из таких траекторий самое большее один раз.

Структура программы

Программа организована в виде ряда последовательных, в основном независимых, программ, каждая из которых оперирует с выходными данными предыдущей и выдает входные данные для последующей. Лингвистическая программа включает подпрограммы считывания вопроса, поиска значений слов в словаре, синтаксического анализа и анализа содержания. Исполнительная программа включает собственно обработку и выдачу данных на выход.

Лингвистическая программа

Считывание вопроса. Вопрос вводится в машину с перфокарт. Затем вопрос преобразуется в последовательный список слов Q.

Просмотр словаря. Каждое слово из списка Q ищется в словаре, и найденное определение слова переписывается в оперативную память. Все неидентифицированные слова выдаются на печать. (В будущем вычислительная машина с непосредственным клавишным печатающим устройством сможет попросить лицо, задавшее вопрос, определить неизвестные слова посредством слов, известных машине, и тем самым пополнить свой словарь.) Список просматривается для выявления возможных идиом. Любые связанные слова, образующие идиому, заменяются одним словом в списке Q с соответствующим определением из словаря. На этом этапе каждое слово списка связывается с его определением, указывающим часть речи, значение и, возможно, что-либо еще.

Синтаксический анализ. Основой синтаксического анализа служат части речи, являющиеся категориями синтаксиса, по-

ставленными в соответствие словам для использования их в синтаксической подпрограмме. В машину заложено 14 частей речи и несколько указателей неоднозначности. В первую очередь в вопросе ищутся неоднозначности в принадлежности слова к той или иной части речи, которые разрешаются в одних случаях просматриванием прилегающих слов, а в других случаях просмотром всего вопроса. Например, слово «score» может быть как глаголом («считать очки»), так и существительным («счет очков»); наше правило таково, что если в вопросе нет другого главного глагола, то «score» является глаголом, в противном случае — существительным.

Затем синтаксическая подпрограмма выделяет и заключает в скобки группы существительного [□] и группы предлога или наречия (□). Глагол в скобки не заключается. Эта подпрограмма скопирована с работы Харриса и др., проведенной в Пенсильванском университете [450]. Процесс выделения скобками проводится от конца вопроса к его началу. Группы существительного, например, заключаются в скобки следующим образом: определенные части речи показывают конец группы существительного; в пределах группы принадлежность слова к некоторой части речи может указывать на то, что это слово находится внутри группы, или что группа начинается с этого слова, или что это слово находится вне группы, но группа начинается с предыдущего слова. Предложные группы состоят из предлога, непосредственно предшествующего группе существительного. Последовательность, состоящая из предложной группы и группы существительного, заключается в круглые скобки. Ниже приводится пример вопроса с расставленными скобками:

[How many games] did [the Jankees] play (in [July])?

[Сколько игр] [команда «Jankees»] сыграла (в [июле])?

После окончания этой процедуры любой предлог, оказавшийся вне скобок, приписывается к группе первого существительного и вся эта группа с предлогом заключается в круглые скобки[1]. Например, вопрос «Who did the Red Sox lose to on July 5?» превращается в «[(To [who]) did [the Red Sox] lose (on [July 5])?».

После анализа групп синтаксическая подпрограмма определяет, имеет ли данный глагол (сказуемое) активную или пассивную форму, а также выделяет его субъект (подлежащее) и объект (дополнение). В частности, глагол находится в пассивной форме тогда и только тогда, когда последний в вопросе глагол главный, а предшествующий ему является одной из форм вспомогательного глагола «to be». Если в вопросах, содержащих глагол в активной форме, свободная группа существительного (группа, не заключенная в круглых скобках) находится между двумя глаголами, то она обозначается как *субъект*, а первая свободная группа существительного считается *объектом*. В противном случае субъектом считается первая свободная группа существительного, а объектом — следующая, если таковая существует. В вопросах с глаголами в пассивной форме первая свободная группа существительного обозначается как объект (так как она была бы объектом при активной форме вопроса). Все предложные группы с предлогом «by» содержат группу существительного, которая и считается субъектом. Если таких групп больше одной, то выбор среди них осуществляется позднее подпрограммой анализа содержания по смыслу.

Наконец, подпрограмма синтаксического анализа проверяет, не отмечено ли какое-нибудь слово как вопросительное. Если нет, то вырабатывается сигнал, показывающий, что на вопрос нужно ответить «да» или «нет».

Анализ содержания. При составлении спецификационного списка вопроса для исполнительной программы подпрограмма анализа содержания использует результаты синтаксического анализа и значения из словаря. Прежде всего выполняются все подпрограммы, встреченные в значении любого слова или идиомы вопроса. Есть два основных типа таких подпрограмм: подпрограммы, оперирующие со значением самого слова, и подпрограммы, которые некоторым образом изменяют значение другого слова. Подпрограмма первого типа выбирает соответствующее значение многозначного слова. Например, упомянутая выше подпрограмма, которая для названий городов решает, что́ является значением: Team$=A_t$ или Place$=A_p$ (команда$=A_к$ или место$=A_м$). Подпрограммы второго типа изменяют признак или оценку соответствующего синтаксически связанного слова. Одна из таких подпрограмм, например, на место оценки главного существительного в его группе ставит свою оценку. Так, пара Team=(blank) (команда=(пустое место)) в группе each team (каждая команда) превращается в пару Team=each (команда=каждая), а в группе what team (какая команда) превращается в пару Team=? (команда=?).

Другая подпрограмма модифицирует признак главного существительного. Так пара Team=(blank) в группе winning team (выигравшая команда) переходит в пару Team$_{(winning)}$=(blank) (команда$_{(выигравшая)}$=(пустое место)). В вопросе «Who beat the Jankees on July 4?» (Кто побил команду «Jankees» четвер-

[1] Данное и некоторые последующие замечания отражают специфические особенности английского языка, где, например, падежи передаются при помощи предлогов, а не путем изменения окончаний существительного. — *Прим. ред.*

того июля?) эта подпрограмма, найдя значение слова beat (побить, побил), модифицирует признаки объекта и субъекта так, что пары Team=? и Team=Jankees (команда=? и команда=Jankees) преобразуются в пары $Team_{(winning)}$=? и $Team_{(losing)}$=Jankees ($команда_{(выигравшая)}$=? и $команда_{(проигравшая)}$=Jankees). Другая подпрограмма комбинирует такие две операции: она одновременно изменяет и признак и оценку главного существительного. Так, пара Game=(blank) (игра=(пустое место)) в фразе «six games» (шесть игр) становится парой $Game_{(number\ of)}$=6 ($игра_{(количество)}$=6), а пара в фразе «How many games» (сколько игр) — парой $Game_{(number\ of)}$=? ($игра_{(количество)}$=?).

После выполнения этих подпрограмм в вопросе ищутся те пары признак—оценка, которые должны быть представлены в спецификации единственным вхождением. Эти пары объединяются. Например, пары Team=? и $Team_{(winning)}$=(blank) (команда=? и $команда_{(выигравшая)}$=(пустое место)) в вопросе «Who was the winning team?» («Какая команда выиграла?») должны быть слиты в одну пару: $Team_{(winning)}$=? ($команда_{(выигравшая)}$=?). Затем в результате последовательных просмотров образуются все подсписки, обусловленные синтаксической структурой вопроса. Окончательно в спецификационный список вносится вся сложная информация относительно каждой группы. В зависимости от сложности каждая группа определяет одно или больше вхождений в спецификацию. Для обеспечения промежуточного контроля машины со стороны оператора результирующий спецификационный список печатается в некотором схематическом виде.

Исполнительная программа

Обработка спецификационного списка. Спецификационный список показывает, какая часть накопленных в памяти данных относится к ответу на поставленный вопрос. Устройство обработки выделяет из исходных данных подходящую информацию и в форме списочной структуры подает ответ на выходное устройство.

Смысл исполнительной программы заключается в поиске по всевозможным траекториям, определяемым исходными данными, соответствия для всех пар признак—оценка спецификационного списка. Если на данной траектории такое соответствие найдено, то все пары, имеющие отношение к данной спецификации, заносятся в так называемый результирующий список. Соответствие для каждого конкретного спецификационного списка считается найденным, если на некоторой траектории в структуре данных есть признак из списка, а соответствующие оценки либо совпадают, либо оценкой данного признака в спецификации является «?» или «each» (каждый). В последнем случае подходящей считается любая найденная оценка этого признака.

Поиск соответствия не всегда прост. Производные и некоторые модифицированные признаки зависят от ряда признаков на некоторой траектории, и прежде чем могут быть подобраны соответствующие оценки, необходима некоторая обработка данных. Если, например, в спецификационный список входит пара Home Team=Red Sox (местная команда=«Red Sox»), то для конкретной траектории нужно прежде всего по месту проведения игр и командам найти местную команду, а лишь затем можно выяснить, подходит ли оценка «Red Sox» к оценке найденного признака. Специальной обработки требуют и подсписки, так как при различных перестановках различные выражения, входящие в подсписок, должны иногда рассматриваться раздельно, а иногда — как составляющие одно целое.

Полученный результирующий список является иерархической списковой структурой, содержащей на каждом уровне вдоль любой траектории один основной или производный признак. Каждая траектория по результирующему списку содержит информацию, полученную из одной или нескольких траекторий по исходным данным. Например, для вопроса «Where did each team play in July?» («Где играла каждая команда в июле?») для каждой команды, игравшей в июле, в результирующем списке существует единственная траектория. В результирующем списке на уровне, следующем за названием команды, находится список всех мест, в которых играла данная команда. Весь этот список является оценкой для признака Place (место). И так для каждой команды. Таким образом, каждая траектория результирующего списка может содержать информацию, собранную из многих траекторий структуры исходных данных.

Многие вопросы, подобно приведенному выше, содержат лишь одну пару вида «признак=?». В нашем примере это «Place (место)=?». Ответом на такой вопрос служит результирующий список, полученный путем однократного применения программы поиска. Дальнейшей обработки этого списка не требуется. При ответе на вопросы другого типа для каждого вхождения пары «признак=?» результирующий список нужно подвергнуть обработке. Так, в вопросе «In how many places did each team play in July?» («В скольких местах играла каждая команда в июле?») после того, как программа поиска выдаст список мест, где играла каждая команда, нужно подсчитать их количество.

Для других вопросов требуются более чем однократное применение программы поиска, а также дополнительная обработка. Если признак, содержащийся в спецификации, имеет в качестве оценки слово every (каждый), то найденные с помощью процедуры поиска списки всех оценок этого признака нужно сравнить со списком всех возможных оценок. Как ответ на вопрос результирующий список должен содержать лишь те траектории, для которых существуют все возможные оценки рассматриваемого признака. Поэтому на первом результирующем списке как на исходных данных снова выполняется процедура поиска. Теперь образуется новый результирующий список, содержащий все те траектории, для которых найдены все возможные оценки[1]. Аналогично если в вопросе имеется специальное число, например 4 teams (4 команды), то нужно выполнить подпрограмму поиска для вопроса «Which teams?» («Какие команды?»), сосчитать количество найденных на каждой траектории команд, а затем найти результирующий список для траекторий, содержащих названия четырех команд.

В общем случае вопрос может распадаться на ряд высказанных или подразумеваемых вопросов. Так как на эти вопросы нужно отвечать по очереди, необходимо несколько раз применить программу поиска с промежуточной обработкой. Входными данными для первого применения программы поиска являются исходные данные. Входом для следующего применения программы служит результирующий список. В качестве примера рассмотрим вопрос «On how many days in July did eight teams play?» («В течение скольких дней в июле играло восемь команд?»). Спецификационный список этого вопроса:

Day(number of)	= ?	День(количество)	= ?
Month	= July	Месяц	= июль
Team(number of)	= 8	Команда(количество)	= 8

На первом этапе выдается ответ на подразумеваемый вопрос Which teams? (Какие команды?). Спецификационным списком для первого применения программы поиска является:

Day	= Each	День	= каждый
Month	= July	Месяц	= июль
Team	= ?	Команда	= ?

Результатом является список дней в июле и для каждого дня — список команд, игравших в этот день. Далее устройство обработки осуществляет подсчет количества команд, игравших в каждый день, и связывает это количество с признаком Team (команда). Спецификационным списком для следующего применения программы поиска является:

Day	= ?	День	= ?
Month	= July	Месяц	= июль
Team(number of)	= 8	Команда(количество)	= 8

Результатом поиска является список дней в июле, когда играло восемь команд. После этого исполнительная программа подсчитывает количество дней, добавляет к этому результирующий список и на этом заканчивает свою работу.

Выходное устройство. Не предпринималось попытки создать устройство, печатающее ответы в виде обычных английских предложений. Вместо этого на выходе печатается лишь окончательный результирующий список. Для вопросов, требующих ответа «да» или «нет», слово «да» печатается вместе с конечным списком. Если программой поиска не найдено никаких подходящих данных, то в качестве ответа на такой вопрос печатается «нет». Во всех остальных случаях печатается «нет данных».

[1] Например, для ответа на вопрос: «Каждая ли команда играла в июле?» потребуется двухкратное применение процедуры поиска. — *Прим. ред.*

Обсуждение

Отличия системы «Бейсбол» как от систем автоматического перевода, так и от систем информационного поиска вполне очевидны. Основной задачей лингвистической части программы является осмысление вопроса, выражаемое в построении канонического спецификационного списка. Для адекватного представления смысла необходимо избавиться от неоднозначностей и рассмотреть синтаксическую структуру вопроса. Задача же программы перевода совсем другая: перевести входное сообщение с одного естественного языка на другой. При этом учитывается значение слов и исключаются двусмысленности в той степени, в какой это требуется, чтобы обеспечить правильность перевода. Однако в общем программы перевода оперируют больше над синтаксисом и меньше над смыслом, чем программа «Бейсбол».

От большинства систем информационного поиска программа «Бейсбол» отличается природой своих исходных данных. В общем случае проблема информационного поиска состоит в обнаружении нужных документов. С каждым документом сопоставлен свой набор индексов, описывающий его содержание. Для каждого входного запроса система информационного поиска должна определить соответствующие значения индексов и

отыскать все документы с такими индексами. Основной проблемой в таких системах является проблема индексации. В отличие от этого в системе «Бейсбол» признаки исходных данных четко указаны и не могут быть перепутаны. Однако производные признаки и модификаторы в системе «Бейсбол» требуют гораздо более трудоемкой обработки, чем большинство документов в программах информационного поиска. (Программа «Бейсбол» имеет близкое отношение к системе «ACSI-MATIC», описанной Миллером и др. в 1960 г. [716].)

Понятие спецификационного списка позволяет определить класс вопросов, на которые система «Бейсбол» способна дать ответ. Система может ответить на все вопросы, спецификации которых состоят из пар признак—оценка, узнаваемых программой. При этом признаки могут быть производными или модифицированными, а оценки — утвердительными или вопросительными. Спецификационный список может быть образован любой комбинацией пар признак—оценка. Многие из этих комбинаций будут бессмысленными, но на каждую из них машина может дать некоторый ответ. Так как оценки могут принимать числовые значения, множество допустимых вопросов бесконечно. Но даже если использовать только два числа, программа сможет ответить на миллионы осмысленных вопросов.

Несмотря на имеющиеся ограничения, настоящая программа — очень полезное устройство коммуникации. Любой сложный вопрос, не удовлетворяющий ограничениям, можно разбить на ряд более простых. Обычно программа отвергает вопросы, которые не может обработать. В этом случае оператор, задавший вопрос, может его перефразировать. Кроме того, оператор может всегда проверить напечатанный спецификационный список и установить, не ошиблась ли лингвистическая программа.

Дальнейшие шаги по усовершенствованию программы

В построении программы для вопросов, включающих логические связки, отрицания и слова, выражающие отношение, не должно встретиться существенных трудностей. Создание программы для вопросов, состоящих из нескольких предложений, по-видимому, также не представит затруднений, если оператор будет обозначать границы своих предложений. Используя существующие подпрограммы, система сможет по очереди сперва обработать придаточные предложения, а затем перейти к главному. Но гораздо более сложные программы понадобятся, если потребовать, чтобы синтаксическая подпрограмма наряду с анализом отдельных фраз определяла и границы между ними.

Проблема распознавания и разрешения семантических неоднозначностей остается нерешенной. Определение смысла такого вопроса, как «Did the Red Sox win most of their games in July?» («Выиграла ли «Red Sox» большинство своих встреч в июле?»), зависит от гораздо большего контекста, чем сам вопрос. Машина может либо ответить на все осмысленные версии вопроса (для данного вопроса мы знаем пять версий), либо спросить оператора, какую версию (какое толкование) он имеет в виду. Вообще способность машины спрашивать оператора явится, по-видимому, очень сильным усовершенствованием. Это позволит машине увеличивать объем своего словаря, что в свою очередь будет способствовать разрешению неоднозначностей, а возможно, даже обучению оператора пользованию программой.

Были предприняты значительные попытки, чтобы сохранить универсальность программы. В новом контексте, касающемся, например, данных о результатах голосования на выборах, бо́льшая часть подпрограмм останется неизменной. Исполнительная программа способна оперировать данными, организованными в любой тип иерархии, и не зависит от характера используемых признаков. Синтаксическая программа основана исключительно на анализе частей речи, которые легко сопоставить с набором слов в новом контексте.

В то же время некоторые подпрограммы, содержащиеся в словаре значений, специфичны только для бейсбола. Вероятно, для каждого нового контекста потребуются определенные специфические подпрограммы. Кроме того, контекст может представить ряд модификаторов и производных признаков, для определения которых потребовались бы специальные подпрограммы в исполнительной программе. К счастью, все источники таких изменений удалось изолировать в небольшом количестве специальных подпрограмм, так что основные программы остаются неизменными. Однако мы не сможем определенно утверждать, что нам удалось создать универсальную программу ответов на вопросы, до тех пор, пока нам в действительности не удастся построить программу для изменяющегося контекста.

9. ПАМЯТЬ, СПОСОБНАЯ К ВЫВОДАМ, КАК ОСНОВА МАШИН, ПОНИМАЮЩИХ ЧЕЛОВЕЧЕСКИЙ ЯЗЫК

Р. Линдсей

Общие соображения

Ученые, работающие над созданием «разумных» машин, часто спорят по основному вопросу, связанному со стратегией исследования. С одной стороны, некоторые думают, что основные

трудности в данной области можно преодолеть повышением надежности памяти машины, увеличением скорости вычисления и усовершенствования ее арифметических способностей, если все эти возможности умело использовать в усовершенствованных системах поиска и статистической обработки. С другой стороны, часть ученых считает, что для существенного прогресса в этой области необходимо прежде как-то разрешить проблемы смысла и интуиции и что решения проблем быстродействия и арифметических операций для этого еще далеко недостаточно. Этот спор может быть разрешен только с помощью убедительных доказательств, но уже сейчас очень важно решить, в каком направлении должны быть сосредоточены усилия.

Автор считает, что некоторые практически важные результаты могут быть получены на основе первого из перечисленных подходов, но основные проблемы все же будут решены лишь на базе второго. Например, использование на существующих вычислительных машинах методов информационной выборки было бы большим шагом вперед, несмотря на тривиальность этих методов в их современном виде Автоматизация поиска литературы в библиотеках и хранилищах научных документов может уже сейчас сэкономить массу времени и усилий, увеличивая вместе с тем и точность отыскания нужной литературы. Кель [540] разработал систему поиска, основанную на отыскании определенных сочетаний ключевых слов среди множества источников и получения ссылки на те документы, которые содержат искомые комбинации. Люн [627] применил прямые статистические методы для извлечения ключевых предложений из научных статей, что в результате дает полезные аннотации по соответствующему вопросу. Широкое использование этих двух методов избавило бы нас в значительной мере от угрозы потонуть в обилии информации, о чем сейчас так много приходится слышать.

И все же совершенно очевидно, что эти методы, использующие прежде всего выгоды, даваемые огромной скоростью работы вычислительных машин, сами по себе не приведут к созданию разумных машин или хотя бы машин, способных заменить обычного клерка при выполнении довольно простых операций. Дело в том, что человек даже при выполнении «неинтеллектуальной работы» делает нечто большее, чем простой подсчет частот появления или поиск ключевых слов. Человек обладает интеллектуальными характеристиками, которые в общем можно выразить словами: «он *понимает смысл* того, что слышит или читает».

Что значит «смысл» и что значит «понимание» в применении к процессам человеческого мышления, никто еще не сумел удовлетворительно объяснить. Можем ли мы тогда надеяться объяснить эти понятия достаточно точно, чтобы иметь возможность построить машину, которая понимает смысл? Прежде чем попытаться ответить на этот вопрос, сосредоточим наше внимание на более детальном рассмотрении примеров тех «вещей», которые могли бы делать машины, способные к «пониманию», но которые были бы недоступны машинам без такого понимания.

Одной из основных проблем искусственного разума является проблема механического перевода с одного естественного языка на другой. Многие рьяные защитники машинного перевода теперь понимают, что высококачественный перевод может осуществляться лишь машиной, оснащенной достаточно детализованными синтаксическими правилами, большим словарем и обладающей достаточной скоростью для определения по контексту содержания двусмысленных слов путем просмотра различных возможных вариантов. Нет сомнения, что, когда эти синтаксические правила будут открыты, вычислительная машина сможет делать достаточно хорошие переводы, и все же должно быть ясно, что такая машина никогда не осуществит действительно правильного и высококачественного перевода, не требующего предварительного и последующего редактирования со стороны опытного переводчика.

Вот хотя бы один пример такой трудности. Пусть мы хотим перевести предложения «Мальчик в доме» и «Мальчик в Париже» с английского на французский язык. По-английски это запишется так: «The boy is in the house» и «The boy is in Paris». В первом предложении предлогу «in» («в») на французском языке соответствует «dans»; во втором предложении тому же самому предлогу на французском языке соответствует предлог «à». Переводчик принимает нужный вариант исходя из того, что дом окружает человека со всех сторон, в то время как о городе этого сказать нельзя. Разумеется, данную проблему можно разрешить, исследуя все существительные с помощью индикатора, который сообщает машине, может ли указанный предмет заключать внутри себя другие предметы. Но явно неплодотворной была бы попытка обработать все идиосинкразии путем увеличения числа таких индикаторов. Бар-Хиллел [84] приводит более усложненный пример. Пусть требуется перевести на французский язык следующие два предложения: «The pen is in the box» («Перо в ящике») и «The box is in the pen» («Ящик на площадке»). Переводчику не дается другого контекста, да и никакой другой контекст не нужен, ибо он знает, что слово «pen» в первом случае может означать лишь инструмент для письма, а не отгороженный участок, в то время как для второго предложения по смыслу справедлив противоположный вариант. Таким образом, переводчик в состоянии выбрать соответствующий

эквивалент на французском языке для обоих предложений, хотя ему для этого по-французски придется применить два разных слова вместо одного английского. Здесь мы снова вынуждены увеличить объем хранящейся в машине информации, на этот раз для указания всех значений, которые может принимать каждое слово.

Проблема «понимания» может быть выражена иначе следующим образом: необходимо найти способы хранения большого количества такой детальной информации при условии сохранения объема памяти машины в приемлемых пределах.

Многие работы, затрагивающие вопросы «смысла», непосредственно не связаны с такой постановкой проблемы, а в основном рассматривают лишь проблему обозначений — каким объектам соответствует данный символ. Здесь, однако, мы сталкиваемся с проблемой: «что означает некоторое *предложение*». Осгуд [818] и Маурер [758] попытались расширить понятия «ассоциации» и «обусловливания» путем включения сюда ассоциаций между группами слов, а не только между отдельными словами.

Согласно теории Осгуда, определенное слово вызывает соответствующий внутренний ответ. Эти ответы могут быть описаны с помощью оценки их в нескольких «измерениях», например: активный — неактивный, хороший — плохой, сильный — слабый. Смысл комбинации слов есть средняя из составляющих оценок для каждого отдельно взятого слова из этой комбинации. Например, если «робкий» оценивается как умеренно неактивный, умеренно плохой и умеренно слабый и если «секретарь» оценивается как человек активный, очень хороший и умеренно слабый, то «робкий секретарь» оценивается как умеренно активный, умеренно хороший и умеренно слабый.

Согласно теории Маурера, предложения являются временными последовательностями из слов, а внутренние ответы для первых слов обусловлены в классическом смысле внутренними ответами для последующих слов. Так, например, предложение «Том — вор» обусловливает понятие «Том» понятием «вор», где мы используем неточный термин «понятие» («notion»), чтобы показать, что это не сами слова, а внутренние ответы, которые стали с ними связаны.

Как в той, так и в другой теории мера смысла совокупности слов образуется как некоторая комбинация мер для отдельных слов. Отсюда становится очевидным, что необходима более плодотворная теория, которая каким-то образом использовала бы более сложные ассоциативные связи, чем те, на которые указывают эти два автора. Схема Осгуда вообще не зависит от порядка слов, а зависит лишь от того, какие слова употребляются; схема Маурера зависит только от порядка слов и не зависит ни от каких других связей. Согласно Осгуду, предложение «Том бьет Джо» имело бы то же значение, что и предложение «Джо бьет Тома»; согласно Мауреру, предложение «Том есть вор» имело бы то же значение, что и предложение «Том бьет вора».

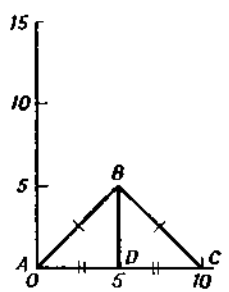
Интуитивно ясно, что понятие смысла должно включать в себя понятие «импликации»: что вытекает из данного предложения? Здесь имеется в виду не логическая импликация, а просто вывод для данного человека. Таким образом, смысл высказывания зависит от слушателей, и это, вероятно, является неустранимым требованием.

Узнавать больше, чем нам сообщили, — весьма характерное человеческое свойство, обнаруживающееся в большинстве поведений того типа, который мы называем разумным. Мы утверждаем, что эта особенность необходима и машине, предназначенной для решения действительно сложных задач по поиску информации, языковому переводу и решению проблем. Более того, нам необходимо найти эффективные способы для хранения «следствий» (импликаций), если мы хотим создать разумные машины с конечным объемом памяти, т. е. если мы хотим иметь действительно разумные машины.

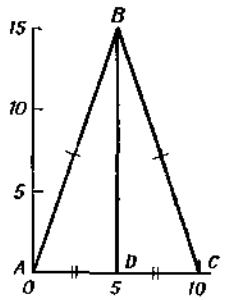
Личный опыт немедленно подсказывает нам примеры структур памяти, обладающие этими свойствами. Они часто называются «мысленными образами» (mental pictures). Гелернтер [365], например, создал «геометрическую машину», основной источник «разумности» которой заключается в способности отбрасывать большинство из формально возможных последовательностей шагов доказательства, основываясь на том критерии, что они «видимо не являются правильными». В результате машина, основываясь на посылках подлежащей доказательству теоремы, создает свой чертеж. (На самом деле, машина снабжена таким чертежом, причем проблема создания этого чертежа, хотя и оказывается сложной, не обременяет память и не снижает скорость вычисления.) Следствия из посылок в явном виде содержатся в этом чертеже наряду с некоторыми отношениями, не являющимися следствиями, однако большинство их отбрасывается. Машина затем просто отвергает из возможных подцелей (промежуточных шагов) все, что не соответствует чертежу. Например, посылки «треугольник ABC», «$AB=BC$», «A, D, C коллинеарны» и «$AD=DC$» даются вместе с координатами каждой точки, например $A(0, 0)$, $B(5, 5)$, $C(10, 0)$ и $D(5, 0)$ (фиг. 1). Машина никогда не попытается доказать, что «треугольник ABC совпадает с треугольником ABD», потому что это не справедливо для данного чертежа. Машина может определить это, вычислив относительные периметры этих треугольников. Однако

она могла бы попытаться доказать предложение «угол DAB равен углу DCB», что вытекает из посылок. Она могла бы также попытаться доказать, что «$AD=DC$», ибо это справедливо для данного чертежа, хотя и не следует из посылок. Снабдив машину более общим чертежом, например, переместив B в точку (5, 15) (фиг. 2), можно было бы избежать этой трудности.

Такие двумерные образы обладают необходимыми для нас свойствами: они хранят следствия и делают это достаточно компактно. Они имеют также обширную сферу применений, которая пока, если не считать работу Гелернтера, почти не исследована.



Фиг. 1.



Фиг. 2.

Другим примером могут служить диаграммы Венна, которые представляют собой средства, помогающие проводить логические рассуждения. Мы можем представить высказывание «все B суть A» с помощью двух соответственно обозначенных областей, одна из которых полностью лежит внутри другой. Если мы теперь в соответствии с теми же правилами добавим третью область C для представления высказывания «все C суть B», то результирующая диаграмма представляет собой импликацию «все C суть A», так как область C должна целиком лежать внутри области A. Подобные правила можно создать и для других форм высказываний, подобных тем, которые обсуждались Линдсеем [611]. В действительности для подобных ситуаций существуют более простые схемы, так как при этом не все из свойств эвклидова двумерного пространства необходимы, но, ввиду того что такая форма представления позволяет описывать и многие другие ситуации, разумные машины достигли бы некоторой экономии, используя такие универсальные представления везде, где они оказываются выгоднее создания специальных схем для каждого случая.

До сих пор мы обсуждали лишь ситуации, не вызывающие особых трудностей. Но рассуждения не всегда подчиняются правилам логики и геометрии, и поэтому даже при попытке рассматривать простые ситуации мы можем встретиться с рядом дополнительных трудностей. Наша программа была написана для работы с различными классами задач, и по мере ее изложения отчетливо выявятся возникающие при этом затруднения.

Программа делает грамматический разбор предложений, записанных на упрощенном языке «Basic English»[1], и выводит следствия, касающиеся родственных отношений людей. Для осуществления этого она создает два типа сложных структур в машинной памяти, одна из которых соответствует диаграмме высказываний, похожей на те, которые составляются студентами, а вторая соответствует обычному генеалогическому дереву. Обе они представляются в машине с помощью так называемых *списковых структур*. Списковая структура есть вид ассоциативной памяти, внутри которой каждый символ опознается индикатором, сообщающим машине адрес связанного с данным символом другого символа. Пока это все соответствует ассоциативным связям, которые являются основным понятием психологии «стимула — ответа». Однако каждому символу иногда может соответствовать целая цепочка других связанных между собой символов, в результате чего создается иерархическая организация ассоциаций в памяти. Это обеспечивает значительно большую гибкость, чем указанные выше простые ассоциации психологии «стимула — ответа» или «промежуточные ассоциации», которые недавно обсуждались и, по-видимому, являются первым шагом в направлении обобщения упрощенной схемы «стимул — реакция».

Программа грамматического разбора предложения

Грамматики языков можно описывать с помощью правил замещения, которые, когда они удовлетворяют некоторым условиям простоты, называются правилами структуры предложения (Хомский [210]). Например, простая грамматика могла бы состоять из следующих правил[2]:

1. $S \leftarrow NP + Pred$	6. $NP \leftarrow A + N$
2. $S \leftarrow NP + VP + NP$	7. $N \leftarrow$ planes
3. $S \leftarrow NP + V + NP$	8. $N \leftarrow$ man
4. $NP \leftarrow$ They	9. $A \leftarrow$ the
5. $NP \leftarrow$ planes	10. $A \leftarrow$ flying

[1] См. примечание на стр. 221. — *Прим. ред.*
[2] Обозначения указаны на фиг. 3. — *Прим. ред.*

11. $VP \leftarrow Aux + V$
12. $V \leftarrow$ are
13. $V \leftarrow$ flying
14. $Aux \leftarrow$ are
15. $Pred \leftarrow VP + NP + Ad$
16. $Ad \leftarrow$ swiftly

Эти правила интерпретируются следующим образом. Правило 12, например, означает: Когда «V» встречается в цепочке символов, оно может быть заменено на «are», или «когда «are» встречается в цепочке слов, оно может быть заменено на «V». В то время как первая интерпретация затрагивает вопросы образования предложений, вторая связана с грамматическим разбором предложения. Таким образом, мы можем образовать предложение, начав его с символа S, а затем последовательно применяя указанные правила. Например,

$$S \rightarrow NP + VP + NP \rightarrow NP + Aux + V + NP \rightarrow \text{They} + Aux + V + NP \rightarrow \text{They} + \text{are} + V + NP \rightarrow \text{They} + \text{are} + \text{flying} + NP \rightarrow \text{They are flying planes}$$

(«Они — летящие самолеты» или «Они летят самолетами»).

Различные последовательности правил могут образовывать различные предложения. С помощью приведенных выше правил можно получить также предложения, которые не соответствуют грамматике английского языка. Например, мы могли бы получить «The man are flying planes» («Человек — летящие самолеты» или «человек летят самолетами»). Надлежащая грамматика (набор правил) английского языка должна была бы исключать такие возможности. Обычно это достигается как более детальными определениями правил (обеспечивающими, например, согласование подлежащего со сказуемым), так и введением определенных «метаправил», с помощью которых определяется, какая из последовательностей применения правил является законной (см. [210, 846]).

Очевидно также, что одни и те же предложения могут получаться с помощью различных последовательностей правил. Например[1],

$$S \rightarrow NP + V + NP \rightarrow NP + V + A + N \rightarrow \text{They} + V + A + N \rightarrow \text{They} + \text{are} + A + N \rightarrow \text{They} + \text{are} + \text{flying} + N \rightarrow \text{They are flying planes.}$$

[1] Пример взят из работы Хомского [210].

Рассмотрим теперь методику непосредственного грамматического разбора, который можно было бы применить к предложению «They are flying planes swiftly» («Они летят самолетами быстро»), используя правила из приведенного нами примера. Данное предложение, разбор которого необычен, может быть исследовано следующим образом. Найти каждое слово или группу слов, которые встречаются в предложении в правой части одного из правил, и заменить их символом, стоящим в левой части правила. Применить ту же процедуру к полученной цепочке символов. Если символ или слово встречается в правой части более чем одного правила, образовать отдельные цепочки для каждого случая. Продолжать процедуру до тех пор, пока последовательность символов не будет сведена к простому символу S, отбрасывая пути, к которым эта процедура не применима. Тогда мы имеем:

They are flying planes swiftly

1. $NP + V + V + N + Ad$
   Нельзя идти дальше
2. $NP + V + V + NP + Ad$
   Нельзя идти дальше
3. $NP + V + A + N + Ad$
   $NP + V + NP + Ad$
   $S + Ad$
   Нельзя идти дальше
4. $NP + V + A + NP + Ad$
   Нельзя идти дальше
5. $NP + Aux + A + N + Ad$
   $NP + Aux + NP + Ad$
   Нельзя идти дальше
6. $NP + Aux + V + N + Ad$
   Нельзя идти дальше
7. $NP + Aux + V + N + Ad$
   $NP + VP + N + Ad$
   Нельзя идти дальше
8. $NP + Aux + V + NP + Ad$
   $NP + VP + NP + Ad$
   8.1. $S + Ad$
   Нельзя идти дальше
   8.2. $NP + Pred$
   S
   Успешный разбор

Но даже после применения большого количества правил и метаправил для отбрасывания «неграмматических» последовательностей в естественных языках останутся предложения, которые могут быть получены двумя или более различными порождающими последовательностями. И наоборот, любая последовательность, которая осуществляет грамматический разбор предложения, должна иметь возможность обнаруживать такого рода последовательности. Решение о том, какой из грамматических разборов является верным, зависит от контекста в большей степени, чем от отдельного предложения, а во многих случаях зависит также от смысла предложения и смысла различных слов.

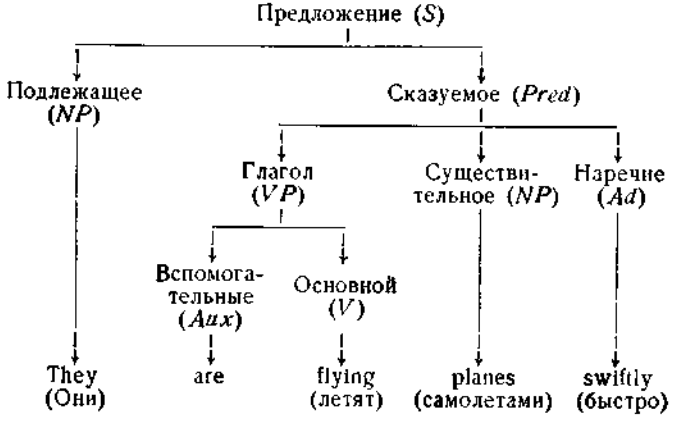
Однако, если мы даже не будем учитывать проблему выбора метода анализа данного примера, в случае когда мы имеем дело с десятками тысяч правил, необходимых для записи естественных языков, задача нахождения *любого* правильного разбора оказывается сама по себе нереальной. На деле полный набор правил оказывается настолько большим, что еще не создано ни одного такого набора хотя бы для одного естественного языка, несмотря на то что некоторые из языков изучаются уже тысячелетия. При использовании даже не очень большого числа правил описанная выше процедура грамматического разбора приводит к множеству возможных ветвей, некоторые из которых продолжают оставаться правдоподобными значительное время. Для того чтобы осуществить разбор в приемлемое время и с разумной затратой средств, полезно использовать специальную процедуру отбора.

Описанная ниже процедура, которую мы применили в нашей программе, основывается на двух предположениях, которые вполне обоснованны с точки зрения психологии. Во-первых, предполагается, что почти все последовательности, встречающиеся в исследуемом тексте, могут быть проанализированы с помощью процедуры, осуществляемой слева направо, которая вырабатывает решения относительно характера разобранных фраз без рассмотрения предложения в целом. Это уменьшает количество комбинаций правил, которые необходимо рассмотреть. Во-вторых, предполагается, что для запоминания промежуточных результатов во время разбора даже чрезвычайно длинных предложений может использоваться только очень ограниченный объем памяти. Это накладывает сильные ограничения на сложность анализируемых предложений и на типы синтаксических структур, которые не могут быть разобраны.

Окончательным результатом применения программы грамматического разбора предложения является ассоциативно организованная память, структура которой отражает взаимосвязь между словами, хотя и не дает полной информации о том, какое из правил необходимо применить первым для образования предложения. Диаграмму предложения для приведенного выше примера можно представить в виде, изображенном на фиг. 3. Каждый узел в этой диаграмме соответствует подструктуре, которая образуется во время сканирования вдоль предложения.

В качестве входного предложения программе грамматического разбора дается цепочка слов. Каждое слово может быть найдено в словаре, который указывает несколько возможных частей речи, в качестве которых может служить данное слово, причем эта совокупность слов располагается в таком порядке, чтобы наиболее часто встречающаяся функция слова была

представлена первой. Машина исследует эту последовательность слева направо, временно предполагая, что первое из слов является самой часто встречающейся функцией. Для каждой части речи машина формирует соответствующую ассоциативную структуру. Эта структура затем сохраняется на некоторое время в памяти, после чего проверяется следующее слово. Если созданная таким образом структура требует наличия дополнительного слова, то машина продолжает поиск этого слова. Если следующее слово удовлетворяет цели, то его структура объединяется со структурой первого слова так, чтобы только простая структура названия должна была быть сохранена в кратковременной памяти, а оставшуюся информацию можно было получить, указав одно лишь ее название. Если же следующее слово



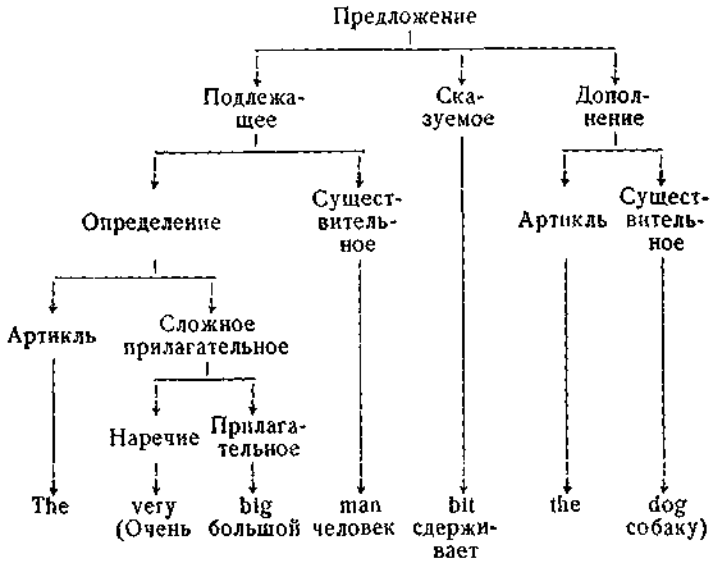
Фиг. 3.

не удовлетворяет необходимой цели, то его структура запоминается отдельно и машина продолжает поиск слов, которые полностью сформируют структуру каждого из слов, находящихся теперь в памяти.

Однако количество названий структур, которое можно записать в кратковременную память, сильно ограничено, так как машина, очевидно, должна начать комбинировать эти подструктуры, чтобы не забыть, где они находятся. Часто бывает, что машина вынуждена завершить формирование структуры несмотря на то, что она не нашла того, что искала. Это приводит к изменению обозначения части речи для одного или более слов так, что вся последовательность будет теперь совместимой с условиями. Таким образом, машина разбирает все предложение, получая промежуточные решения, накапливая подструктуры

в своей ограниченной кратковременной памяти и пересматривая свои решения лишь в случае нехватки массива памяти или в случае полной несовместимости подструктур.

Поведение машины может быть описано примерно следующим образом (фиг. 4). Первое слово — «the» (артикль). Прекрасно! Теперь необходимо найти слово, которое может служить существительным. Второе слово есть «very» («очень»), поэтому теперь мне необходимо прилагательное или наречие. Третье слово — «big» («большой») как раз является прилагательным, в котором я нуждаюсь. Поэтому следует объединить эти два



Фиг. 4.

слова в структуру «very big» («очень большой»). Теперь мне необходимо иметь существительное, которое могло бы служить подлежащим. Четвертое слово — «man» («человек») — как раз является существительным. Теперь все слова сформированы в структуру «[the (very) big] man» («очень большой человек»). Но раз мы имеем подлежащее, надо найти сказуемое. Пятое слово — «bit» («укусил», а также «сдерживает») может служить сказуемым. Итак, мы получили структуру «(the ((very) big)) man) bit» («очень большой человек укусил» (или «сдерживает»). Теперь следующее существительное или даже группа таких слов может служить дополнением. Шестое слово — «the» (артикль). Поэтому оставим его для модификации существительного. Итак, у нас остались два слова для существительного — дополнения. Последнее слово — «dog» («собака») — удовлетворяет обеим требуемым функциям. Теперь у нас есть предложение, структура которого показана на фиг. 4.

В определенном смысле эта программа представляет собой не что иное, как алгоритм, который формирует выход для любой возможной входной последовательности, состоящей из частей речи. Однако если бы программа могла только проверять входную последовательность и формировать соответствующую выходную последовательность, то в этом случае возникли бы серьезные трудности. С увеличением длины предложения размеры таблицы возрастают по экспоненте, так что даже для сравнительно небольших предложений нужна чрезвычайно большая таблица. Несмотря на то что сама программа ограничена с точки зрения сложности предложений, которые она разбирает, длина последовательности сама по себе не приводит к возрастанию сложности. Например, программа сможет проанализировать предложение, подобное следующему: «The big, black, ugly, ferocious, strong dog bit him» («Большая, черная, безобразная, свирепая, сильная собака укусила его»), где число прилагательных, которое можно включить в цепочку, ограничивается лишь объемом памяти вычислительной машины. Это становится возможным благодаря тому, что все прилагательные образуют простую подструктуру на каждом шаге и, таким образом, объем кратковременной памяти никогда не превышает допустимого. Кроме того, общий объем памяти увеличивается только линейно при увеличении длины предложения. Однако с предложениями, которые требуют формирования слишком большого количества подструктур, программа потерпит неудачу, даже если сами эти предложения относительно коротки. Описывая аналогичное устройство для формирования предложений, Ингве [1178] утверждает, что ограничения в сложности, вызванные ограничением в объеме кратковременной памяти, как раз и объясняют, почему некоторые конструкции предложений обычно не используются в естественном английском языке и называются поэтому неграмматическими.

Программы, состоящие из частей речи, обеспечивают конечный набор процессов, который может обработать в принципе бесконечное число предложений. Это программы высшего порядка по отношению к методу «просмотра таблицы» по той же причине, по которой программа, высчитывающая e^x для любого значения x, является более высокой по отношению к таблице этой функции и программе просмотра таблицы.

Очевидно, что человек должен использовать некоторое конечное множество процессов для разбора предложения, и очевидно

также, что каждое слово играет роль стимула, вызывающего соответствующие процессы. Процессы, подобные приведенным в нашей программе, очень сложны, и их результаты сильно зависят от того, какие из процессов были использованы до них. Предположение о том, что взаимодействие между этими процессами происходит именно так, как описано выше, т. е. с помощью небольшого набора ячеек кратковременной памяти, вполне соответствует нашим знаниям о природе процессов человеческого мышления. Однако несомненно, что взрослый человек располагает более богатым набором процессов, с помощью которых он эффективно классифицирует слова на более узкие категории, чем содержащие лишь несколько частей речи символы, сообщаемые нашей программе, и несомненно, что эти процессы более сложные.

Другая психологически обоснованная характеристика нашей программы заключается в том, что разбор предложения, осуществляемый нашей программой, происходит слева направо. Хотя английскую грамматику можно было бы анализировать более быстро с помощью некоторых других методов, человек обычно, за редкими исключениями, производит анализ слева направо.

Ограничения в работе нашей программы еще недостаточно хорошо выявлены. Программа не включает ни одного слова, не принадлежащего к «Basic English», представляющему собой систему грамматики и словаря, содержащего около 1700 английских слов, разработанную Огденом в 1933 г. [816]. («Basic English» является упрощенным английским языком в том смысле, что любое правильное выражение на этом языке является правильным и на «хорошем английском языке», но не наоборот.) Программа не использует некоторых знаков препинания, таких, например, как двоеточие, и не различает другие знаки, такие, как восклицательные знаки и интервалы. Кроме того, фразы и части сложного предложения могут разделяться с помощью черточек, а не запятыми. Однако программа не ограничена возможностью разбора лишь простых предложений или необходимостью подачи на вход лишь полного предложения. Это значит, что программа может оперировать со многими входами, как это имеет место в реальном письменном языке (хотя и не в учебниках грамматики). Программа всегда принимает решение, и результатом работы программы всегда является полная структура, содержащая все входы.

Окончательный результат применения программы для анализа предложения представляет собой структуру, которая связывает между собой все слова в предложении. Программа может быть логически заменена множеством описаний, перечисляющих все эти отношения, но такое множество было бы значительно более сложным и заняло бы большой объем памяти. Синтаксической сущностью предложения и является именно такая структура, в которой связь между словами определяется ее организацией.

Программа семантического анализа

После того как получена диаграмма предложения, программа делает попытку разобраться в значении некоторых слов. Вначале конструируется список всех существительных. Этот список включает в себя не только те слова, которые использовались как подлежащие или как дополнение, но также и имена, используемые в предложении в качестве притяжательных прилагательных, таких, например, как «Bill's» («Билла»).

В этом случае слова уже не рассматриваются только по их принадлежности к синтаксическим категориям. Диаграмма предложения используется как информационный накопитель, который связывает слова между собой. Программа отыскивает комбинации вида подлежащее — дополнение, в которых главное сказуемое представляет собой одну из форм глагола «to be» («быть»). Когда найдена такая комбинация, слова обозначаются как «эквивалентные» с помощью дерева соответствий (cross-referencing scheme), которое показывает, что как подлежащее, так и дополнение относятся к одному и тому же предмету или одному и тому же человеку. Затем модификация всех эквивалентных существительных группируется вместе. Далее производится поиск восьми слов, которые существуют в «Basic English» для обсуждения родственных связей: отец, мать, брат, сестра, потомок, шурин, золовка и женат (замужем). Если в предложении встретится хотя бы одно из этих слов, то их модификации подвергаются проверке с целью нахождения соответствующих имен, выступающих в роли притяжательных прилагательных или предложных дополнений, как было бы в случае, если, первоначальное предложение содержало бы выражение вида «Jane's brother» («брат Джейн») или «the father of John» («отец Джона»). Каждое такое имя собственное объединяется попарно со всеми другими, связанными с тем же вариантом родственных слов

Обработав таким образом весь набор слов, программа образует список элементарных родственных отношений. Элементы этого списка представляют собой группы по три слова («триплеты»), состоящие из двух существительных и одного слова, выражающего родственные отношения (relation word) и связывающего эти существительные.

Затем формируется «генеалогическое древо». Память вычислительной машины организуется по ассоциативному принципу, при котором одна область памяти связывается с другой. Таким образом получается структура, изоморфная диаграмме, которая будет приведена в последующем примере. Каждая семья (брачное отношение) представляется с помощью связи между мужем и женой плюс обозначения аналогичной семьи родителей мужа, родителей жены и списка потомков (ведущих происхождение от этого брака). Если имена одного из супругов, одного из родителей или некоторых из их потомков не даны, то для них резервируются места, учитывая, что они могут появиться в будущем. Окончательный вид дерева не зависит от того, была ли информация в явном виде задана в тексте или только подразумевается.

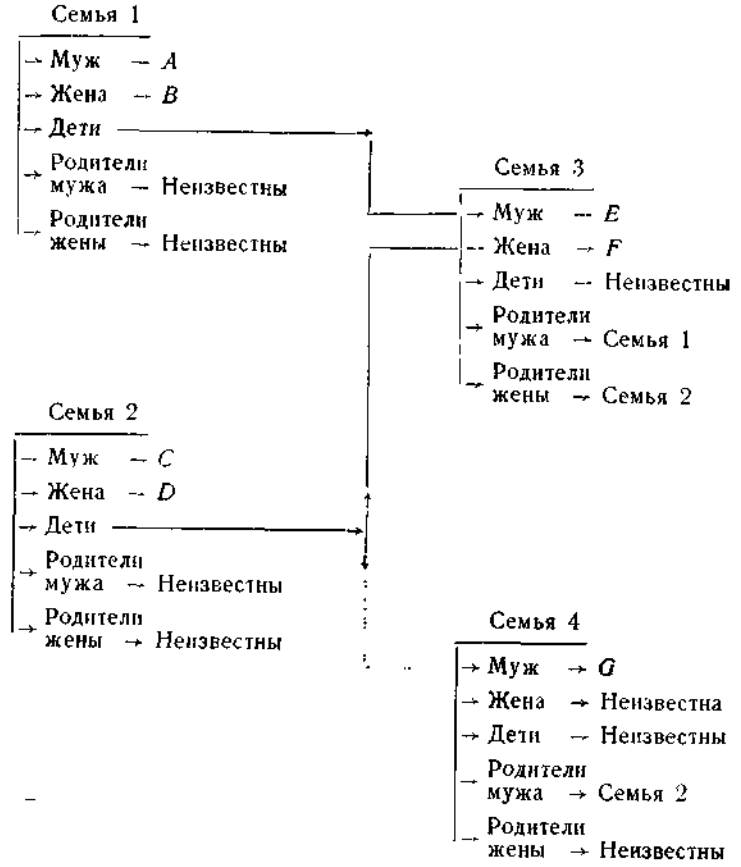
Для примера на фиг. 5 показана структура памяти, сформированная для простого генеалогического дерева. Это дерево составлено из двух основных семей, одна из которых происходит от брака между A и B, а вторая — от брака между C и D. Один из детей от первого брака E вступает в брак с одним из детей от второго брака F. Очевидно, что с помощью такого дерева могут быть описаны многие родственные связи. Однако необходимо заметить, что указанные связи представляют собой лишь односторонние связи. Этот факт приводит к необходимости в дополнительном указании фамилии семьи родителей в конце каждой семьи потомков. Таким образом, имея A и двигаясь только по направлению стрелок, мы можем определить, что E является одним из потомков. Имея E, мы снова можем проследить связи, принадлежащие A, двигаясь лишь по направлению стрелок, но это справедливо только потому, что семья, связанная с E, также содержит фамилию семьи A. Таким образом, знание факта, что F (уже обозначенный на дереве) и G (фамилия, встречающаяся впервые) являются людьми, имеющими одного общего предка, еще не достаточно для того, чтобы добавить G к списку потомков брака между A и B (точечная линия); мы должны еще переписать фамилию семьи родителей F в заново сформированную семейную ячейку G (пунктирные линии).

Генеалогическое дерево или деревья, сформированные именно таким образом, не стираются после обработки каждого предложения, а продолжают расти за счет получения дополнительной информации, приобретаемой в процессе работы.

Усложненность и большое число мелких трудностей, которые встречаются даже в этом простом типе соотношений, показывают характер тех проблем, с которыми мы встречаемся при формировании семантических структур. Более поучительной является проблема смысла, возникающая при попытке обобщить

эту программу на ситуацию с меньшими ограничениями. Рассмотрим две из наиболее важных проблем такого рода.

Даже при работе с простыми родственными связями часто случается, что порядок представления входящей информации оказывает решающее влияние на эффективность расположения



Фиг. 5.

ячеек памяти. Например, если нам первоначально сообщают, что X имеет потомков A, B, C и D, мы должны тщательно сформировать структуру для переработки этой информации. Ячейки памяти для супружеской четы X оставляют незанятыми. Если нам затем скажут, что Y имеет потомков E, F, G и H, то нам необходимо будет создать другую подобную структуру, не связанную с первой. В конце концов мы таким образом сможем

выяснить, что *B* и *H* являются братьями. Это позволяет (если не рассматривать такие усложнения, как повторные браки) разбить две структуры на простые организации, которые значительно более компактно выражают подразумеваемую информацию. Если бы нам настолько повезло, что мы уже вначале узнали бы связи между *B* и *H* (или связь семьи *X* и семьи *Y*), мы смогли бы значительно эффективнее использовать имеющуюся память. В нашей программе дополнительные структуры подлежат «стиранию»; это значит, что часть памяти, которая использовалась для таких структур, очищается для использования в других целях, когда это будет необходимо. Это чрезвычайно просто проделать с машинной памятью, легко поддающейся всяким изменениям, но память, в которой стирание происходит с большим трудом (а к такому типу, по-видимому, относится человеческая память), могла бы сильно пострадать от таких неудачных входных последовательностей.

Тем не менее разумная машина должна обладать разумностью вне зависимости от порядка поступления информации. Один из аспектов феномена «Ага!» как раз заключается в том, что многие ранее не связанные между собой элементы информации с помощью простой дополнительной информации неожиданно группируются вместе так, что многие выводы сразу же отпадают. Педагоги ограничиваются проблемой определения оптимального порядка представления изучаемого материала, но, к счастью для учащихся, человеческий мозг способен к выявлению связей даже при неоптимальных условиях.

Другая, более серьезная проблема связана с так называемым дополнительным смыслом (connotative meaning) [611]. Вероятно, в большинстве случаев множество высказываний, приводящих к некоторым определенным следствиям, содержит ряд подмножеств, которые изменяют вероятности других высказываний, не определяя ни одного из них. Таким образом, высказывание: «Джордж голосовал за Эйзенхауэра и возражал против бесплатной медицинской помощи для престарелых» более правдоподобно, чем высказывание о том, что Джордж возражал против Объединенных Наций, хотя и это в какой-то степени возможно. Совершенно ясно, что человек часто получает знания на основе этих изменяющихся вероятностей и, возможно, в большей степени, чем это было бы разумно. Но как разумно и эффективно использовать такие неясные дополнительные данные?

Рассмотрим более конкретный пример, возникающий из содержания программы родственных связей. Разберем следующее предложение: «Джо играл со своим братом Бобби в саду их тети Джейн, когда их мать позвала их домой». Это предложение содержит определенную информацию, так как очевиден

факт, что Джо и Бобби — братья. Ясно также, что Джейн является или сестрой, или золовкой матери этих детей, но не известно точно, кем именно. Если первоначальная информация связывает, скажем, Джо со многими другими и Джейн со многими другими, но не связывает родственников Джо с родственниками Джейн, то данное предложение может привести к самым различным заключениям относительно родственных связей и исключить допустимость большого числа других возможных выводов. Ряд других возможностей зависит от точного знания отношений между Джейн и матерью Джо. Задача состоит в том, чтобы выявить в этой генеалогической структуре все определенные следствия для того, чтобы устранить все связи, которые наверняка исключаются, и указать изменение вероятностей других связей, не делая, однако, еще никаких *определенных* предположений о родстве между Джейн и матерью Джо.

Описанная выше структура не способна работать даже в этом простом случае, когда ассоциации или существуют, или их нет и когда возможно только одно сочетание. Одним из решений этой задачи может быть формирование нескольких генеалогических структур — своей для каждого варианта. Это соответствует, например, ситуации, в которой студент делает чертеж остроугольного, прямоугольного или тупоугольного треугольника в соответствии с возможными случаями, для которых он хочет доказывать теоремы. Такое решение проблемы приемлемо лишь тогда, когда число альтернатив мало и когда структуры сами по себе просты. В более сложных ситуациях этот метод привел бы к необходимости слишком большого объема памяти. Поэтому желательно отразить имеющуюся неопределенность в единой простой структуре.

Для того чтобы сделать это, мы должны согласиться на увеличение количества связей. Так, в месте каждой связи в нашем первоначальном примере мы должны подставить список всех вариантов, и процесс выборки информации должен определить, что только один из них может быть верным. Когда нельзя сделать абсолютно никаких выводов, список вариантов будет содержать символ «все», показывая, что все варианты возможны. Когда позже сформировывается какой-то вывод, этот символ «все» заменяется на соответствующую связь, а когда все связи установлены, универсальный символ заменяется на список из оставшихся вариантов. Можно также составить список связей, которые наверняка невозможны. Когда нет никакого вывода, этот список будет содержать символ «ничего», который показывает, что ни один из вариантов не невозможен.

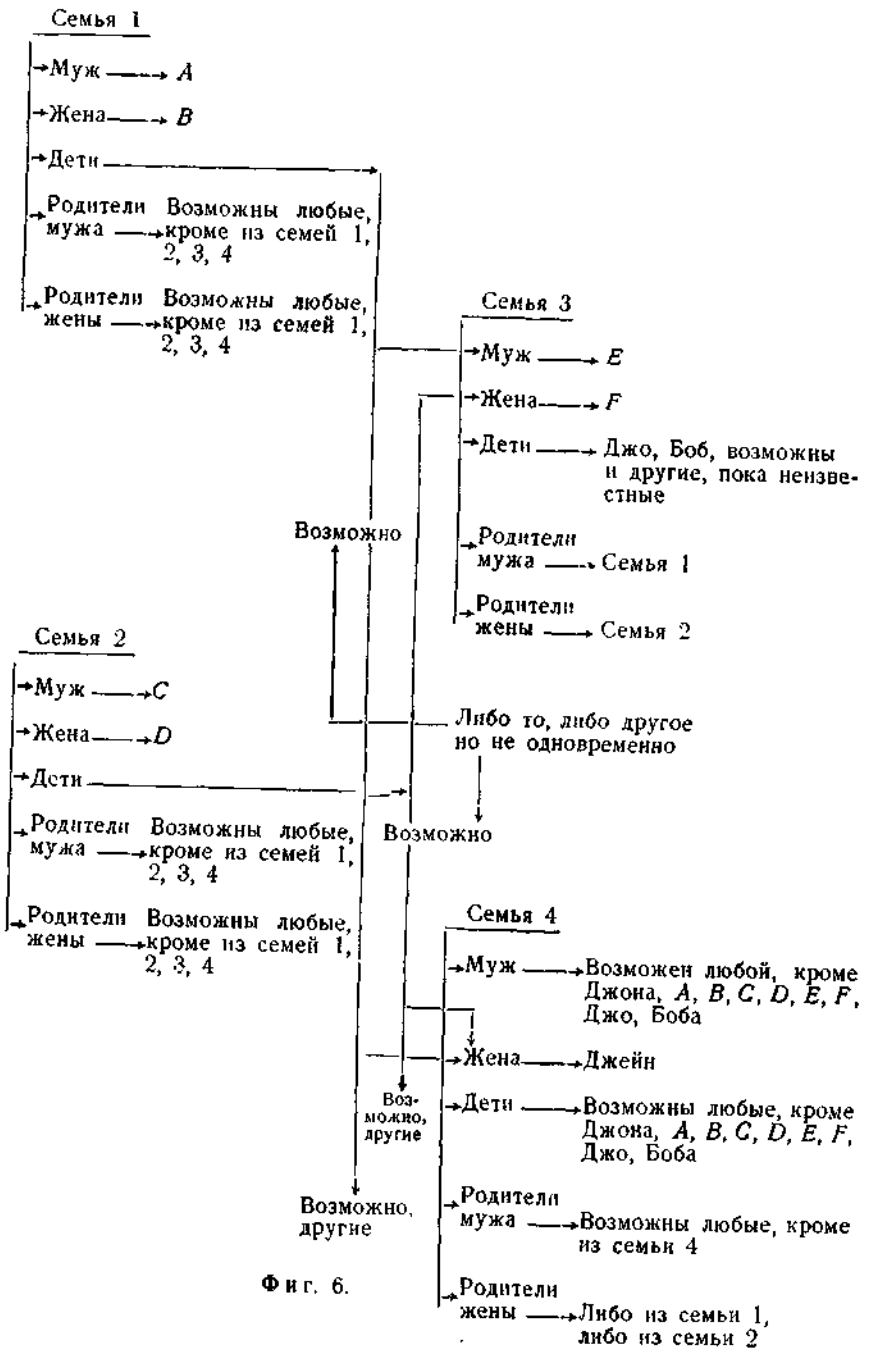
Необходимо заметить, что вероятностные связи того типа, который часто используется психологами в их гипотезах, здесь

неприменим. Иначе говоря, мы действительно не нуждаемся в связях такого типа, при которых сигнал иной раз приводит к возникновению одной ассоциации, а иногда — к другой, на чисто вероятностной основе. В приведенном примере читатель *определенно* знает, что Джейн является сестрой матери Джо *или* сестрой отца Джо, но не одновременно и той и другой. Ни один из читателей не будет, конечно, попеременно какое-то время считать Джейн сестрой матери Джо, а остальное время полагать ее сестрой отца Джо.

Однако перед нами стоят еще две проблемы. Первая заключается в невозможности или по крайней мере непрактичности оставлять слишком большое число вариантов; вторая проблема заключается в неясности того, как определить, что один из вариантов является более вероятным, чем другой. Первая проблема, по-видимому, решается человеком без запоминания всех вариантов, ибо человек не способен запомнить все возможные следствия, когда число таких вариантов слишком велико. Не вызывает сомнения тот факт, что эта проблема актуальна для машины в той же степени, что и для человека. Вторую проблему мы могли бы решить за счет упорядочения списка вариантов, помещая наиболее вероятный вариант в списке первым, или, возможно, мы могли бы решить эту задачу, приписывая вероятности каждому из вариантов. Во всяком случае установленные таким образом вероятности определят вес, который будет приписан выводам, но не определят еще те выводы, которые должны быть приняты путем исключения других выводов. В конце концов мы можем представить себе ситуацию, в которой список вариантов ограничен из-за ограниченных возможностей человека или машины, когда все остающиеся варианты постепенно устраняются за счет получения дальнейшей информации. В этом случае машина в конце концов должна указать, что где-то есть ошибки, и пересмотреть первоначальные входы, отбирая в то же время заново варианты поступающей информации на основе уже известной.

Для того чтобы закончить наш пример, приведем схему изменяющейся машинной памяти (фиг. 6), которую можно было бы использовать для решения нашей простой задачи.

Заметим наконец, что мы решили проблему «дополнительного смысла», так как неявно сохранили наше основное устройство для записи *точных* выводов, но в явном виде прибегли и к хранению *возможных* выводов. Методика, позволяющая избежать списка возможностей, окажется чрезвычайно ценной, ибо, как мы уже видели, требования к объему памяти быстро увеличиваются в случае, когда запись в памяти является полной.



Фиг. 6.

Заключение

Было показано, что проблема смысла имеет чрезвычайно большое значение для изучения природы «разумности» и что полезное определение «смысла» должно включать не только прямое значение, но и сопутствующее значение и выводы. Для того чтобы ответить на эти важные вопросы, необходимо изучить «познавательные» структуры, значительно более сложные, нежели те, на которых основывается большинство психологических теорий Основной вопрос заключается в хранении большого числа связанных между собой высказываний в виде, который эффективно использует возможности памяти машины. Иллюстрацией этого утверждения служит ссылка на программу для вычислительной машины, которая хранит синтаксические связи и извлекает и записывает семантические выводы весьма ограниченного характера.

Приведенные здесь примеры работы этой программы конкретизируют некоторые из проблем, которые необходимо решить до того, как можно будет построить машины, обладающие «разумом».

РАЗДЕЛ 5

РАСПОЗНАВАНИЕ ОБРАЗОВ

В кратком введении очень трудно дать характеристику такой сложной области работ по «искусственному разуму», какой являются исследования по «распознаванию образов».

Вначале эти исследования касались распознавания оптических изображений букв, а позднее — распознавания звуков речи. Селфридж назвал эти исследования разработкой «глаз и ушей для вычислительной машины».

В последнее время термин «распознавание образов» вполне обоснованно стали применять и к исследованиям, касающимся формирования гипотез машинами, обучения различению (классификации) в случайных сетях, процессов образования понятий человеческим мозгом и вообще схем автоматов для индуктивного вывода

Между работами по «распознаванию образов» и исследованиями «познающего» поведения имеется тесная связь, что оправдывает помещение данного раздела в этом месте книги.

Даже если мы условно отнесем статьи Фейгенбаума, Ханта—Ховленда и Фельдмана к исследованиям «распознавания», данная область исследований все равно окажется весьма слабо

представленной в этой книге. Этот пробел отчасти восполняется обзором состояния вопроса, содержащимся в статье Юра и Фосслера, помещенной в настоящем разделе, а также обсуждением соответствующих проблем в статье Минского (см. стр. 402). Другие важные работы в этой области, в частности работы Селфриджа, Кохена и других авторов, читатель найдет в библиографии.

10. ПРОГРАММА РАСПОЗНАВАНИЯ ОБРАЗОВ, КОТОРАЯ ВЫРАБАТЫВАЕТ, ОЦЕНИВАЕТ И УЛУЧШАЕТ СВОИ ОПЕРАТОРЫ

Л. Юр и Ч. Фосслер

Исторический обзор

В типичном случае программа распознавания образов или с самого начала тщательно разрабатывается для обработки конкретных наборов вводимых изображений, или же проектируется как tabula rasa[1] с некоторыми способностями улучшать свои параметры или «учиться». Программы первого типа часто не способны решать задачу распознавания большого количества изображений, которые человеческий глаз различает весьма легко [74, 412]. Самые лучшие программы этого типа, вероятно, можно приспособить для обработки новых классов изображений [424, 993]. Но в каждом таком случае, по-видимому, требуется специальное усложнение там, где на самом деле возможно упрощение, что приводит к дополнительному расходу времени и энергии как программиста, так и машины.

Программы второго типа не обладают (по крайней мере не обладали до сих пор) методами накопления опыта, которые были бы достаточно мощными для успешного применения их в интересных случаях. Машины со случайными связями[2] обнаруживают относительно слабые способности распознавания [930, 934]. (Единственным исключением является, по-видимому, модификация перцептрона Розенблатта, осуществленная Робертсом [918] Но при такой модификации перцептрон по существу можно считать машиной без случайных связей.) К настоящему времени самая хорошая машина этого типа лишь накапливает информацию (или вероятности) о дискретных ячейках входной матрицы

[1] «Чистая доска» (*лат.*). — *Прим ред.*

[2] Имеются в виду различные типы перцептронов, в которых соединения между рецептивными и ассоциативными элементами происходят случайным образом. — *Прим. перев.*

[79, 470]. Но такое обучение является необычайно слабым типом обучения (если вообще можно характеризовать такой способ накопления информации этим расплывчатым термином). Если допустить, что изображения могут изменяться, то этот тип программы работает тем хуже, чем больше варьируют изображения.

Некоторые программы сочетают в себе использование какого-то самоприспосабливающегося разделяющего оператора обработки признаков из программ второго типа и высокоэффективных операций, которые применяются в программах первого типа [285, 1098]. По-видимому, при этом достигается выигрыш в гибкости записи и возможность модификации программ; однако до сих пор нет опубликованных результатов, которые показали бы явное преимущество этих программ перед более слабыми программами (например, [79]), использующими отдельные ячейки входной матрицы таким же образом, каким комбинированные программы используют «демоны» и «операторы»[1]. Примером программы этого смешанного типа является оператор случайного выбора n элементов [129, 130]. В этой программе для составления операторов используется случайный выбор пар, пятерок и других наборов ячеек входной матрицы. В ней делается попытка дать подход к решению задач предварительного анализа и программирования. По-видимому, этот метод имеет по крайней мере такие же возможности, как и метод, основанный на анализе вероятности возбуждения одной ячейки [1091]. Но до сих пор эти возможности не продемонстрированы. И эта программа, как и большинство других обсуждавшихся (или известных авторам) программ, обычно терпит неудачу, когда требуется обработать образы, которые сильно отличаются от тех, на которых она первоначально «накапливала опыт» [1089].

Краткое описание работы программы

Неизвестные изображения подаются в машину в дискретной форме в виде матрицы 20×20 из нулей и единиц. Программа вырабатывает и составляет операторы одним из нескольких случайных методов и использует эту систему операторов для преобразования неизвестной входной матрицы в таблицу характеристик. Или же, в другом варианте, программист может задать систему заранее составленных операторов, которые его интересуют.

[1] «Демонами», или «операторами», называют различные подпрограммы, которые вырабатывают сложные обобщенные признаки изображений, так что с вырабатываемыми этими подпрограммами величинами можно обращаться так же, как с величинами, характеризующими возбужденность отдельных элементов (рецепторов) входной матрицы. — *Прим. перев.*

Эта таблица затем сравнивается с выработанными ранее и хранящимися в памяти таблицами характеристик, каждая из которых соответствует одному образу (классу изображений). В результате сравнения устанавливается название таблицы, которая наиболее близка к только что вычисленной таблице. Это название и присваивается текущему изображению. Затем программа исследует эти характеристики и в зависимости от вклада каждой из них в правильное или ложное решение о принадлежности текущего изображения увеличивает или уменьшает вес каждой из этих характеристик. В итоге эта регулировка весов ведет к отбрасыванию операторов, которые дают плохие характеристики, на что указывают малые веса, и их заменой вновь выработанными операторами.

Настоящая программа может начать работу вообще без операторов. В этом случае операторы первоначально вырабатываются программой с некоторой фиксированной скоростью до тех пор, пока не будет выработано определенное их число. Затем, используя непрерывную замену плохих операторов новыми, программа получает оптимальную систему операторов для обработки данного набора изображений.

Подробное описание работы программы

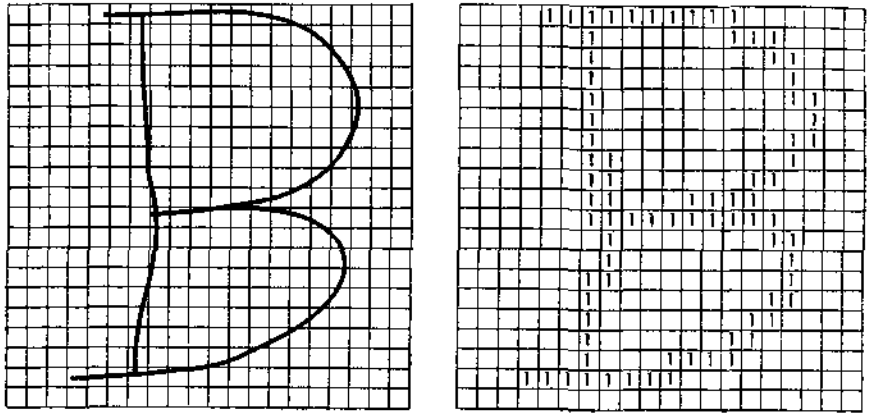
Существует несколько модификаций программы; мы приведем здесь результаты работы некоторых из них. Программа работает следующим образом:

1. Подлежащий опознаванию незнакомый образ кодируется в виде входной матрицы 20×20 элементов, состоящей из нулей и единиц (фиг. 1).

2. Проводится прямоугольная рамка вокруг входного изображения (ее стороны определяются самой левой, самой правой, самой верхней и самой нижней заполненными ячейками) (фиг. 2).

3. Входное изображение преобразуется в четыре трехразрядные характеристики при помощи системы матричных операторов по 5×5 элементов. Можно представить себе, что каждый элемент такой матрицы содержит 0, 1 или пробел. Эти малые матрицы, измеряющие локальные характеристики образа, перемещаются с шагом в одну клетку горизонтально, а затем вертикально по той части матрицы, которая лежит внутри рамки. Считается, что оператор соответствует данной входной матрице всякий раз, когда нули и единицы в матрице оператора соответствуют тем же величинам в матрице изображения, и при каждом таком соответствии положение центральной ячейки этого матричного оператора, охватывающего 5×5 элементов, временно записывается

в память (фиг. 3). Эта информация затем суммируется и приводится к единому масштабу в пределах от 0 до 7 для образования четырех трехразрядных характеристик для этого оператора. Они представляют: (1) число соответствий N; (2) среднее горизонтальное положение мест соответствий внутри прямоугольной рамки X; (3) среднее вертикальное положение мест соответствий Y и (4) среднюю величину квадрата радиального расстояния от центра рамки R.



а *б*

Фиг. 1. Неизвестное изображение поступает на вход в виде матрицы 20×20. Те ячейки, на которые попало изображение, заполняются единицами, остальные — нулями.

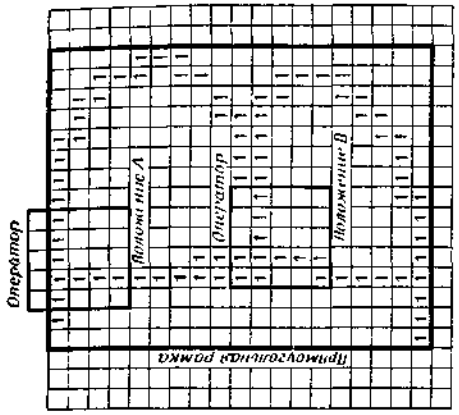
а — неизвестное изображение; *б* — закодированное изображение.

При работе машины может использоваться различное число операторов. Это означает, что число операторов либо устанавливается заранее для некоторого определенного вида работы, либо увеличивается от нуля до максимума по одному из правил, описанных ниже. Последовательность 25 чисел, которая определяет матричный оператор из 5×5 элементов, можно вырабатывать по одному из следующих правил (фиг. 4):

1) заранее составленная последовательность может быть введена в машину экспериментатором;

2) можно выработать случайную последовательность, в которой затем либо ограничивать число содержащихся в ней «единиц», либо требовать связности «единиц» в матрице 5×5 (фактически этот метод нами еще не был испытан);

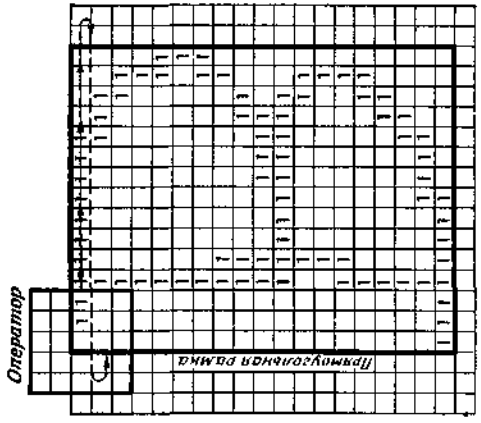
3) можно «выбрать» случайную последовательность из данной входной матрицы и модифицировать ее с помощью про-



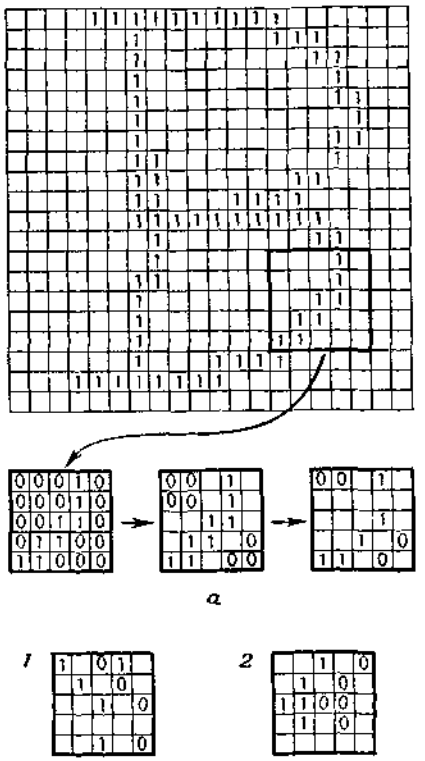
Оператор	Положение	X	Y	R^2
	A	1	0	4
	B	2	4	0
$N=2$		2	2	2

Фиг. 3 Наверху показаны два положения оператора, изображенного внизу слева, в которых он соответствует входной матрице.

Выход оператора положителен всякий раз когда его единицы и нули покрывают соответственно «единицы» и «нули» на неизвестном изображении.



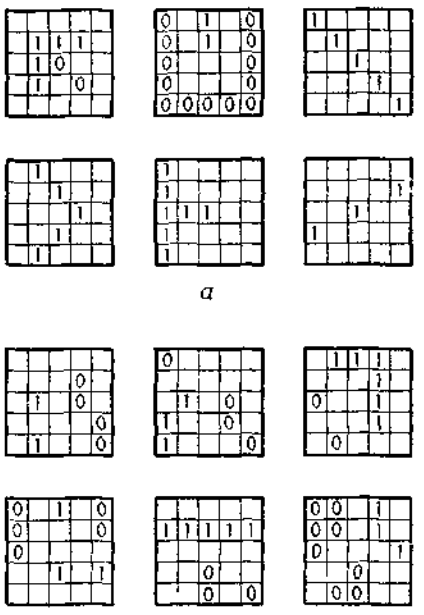
Фиг. 2 Прямоугольная рамка, которая проводится вокруг неизвестного изображения, после чего каждый из матричных «операторов» 5×5 перемещается по изображению



Фиг 4 Способы вырабатывания матричных операторов

а — выделение из входного изображения Выбирается матрица 5×5 со случайным положением центра, в которой затем выбрасываются «нули», прилегающие к «единицам», и каждая из оставшихся ячеек затем с вероятностью $1/2$ заменяется пробелом *б* — обозначение ячеек случайным образом Выбирается ячейка с «единицей», а затем через одну ячейку справа от нее помещается «нуль»

1 — от 3 до 7 «единиц» выбраны случайным образом, *2* — выбор «единиц» ограничен связностью ячеек



Фиг 5 Примеры операторов

а — типичные примеры заранее запрограммированных операторов, *б* — шесть операторов, выработанных программой в ходе эксперимента по распознаванию семи наборов из пяти образов в котором программа достигла 94% ного распознавания

цедуры, которая включает замещение некоторых клеток пробелами в выбранном операторе, чем учитываются возможные малые искажения в локальных характеристиках, сравниваемых оператором:

а) матрица 5×5 выбирается из случайного места во входной матрице;

б) все «нулевые» ячейки, прилегающие к «единичным» ячейкам, заменяются пробелами;

в) каждая из оставшихся ячеек, как «нулевых», так и «единичных», затем с вероятностью $1/2$ заменяется пробелом;

г) производится проверка, гарантирующая, что выбранный оператор не совпадает своими «единичными» ячейками ни с каким другим оператором, используемым в настоящее время или находящимся в списке операторов, недавно отброшенных программой Если выбранный оператор похож в этом смысле на один из таких операторов, то вырабатывается новый оператор, начиная с первого шага (фиг. 5).

4. Используется также второй тип оператора — комбинированный оператор, определяющий одну из 16 возможных логических или арифметических операций и две ранее подсчитанные характеристики, которые используются для получения третьей характеристики. Эти операторы вырабатываются программой случайным выбором одной из возможных операций и двух характеристик. Такой случайный процесс выработки операторов улучшается следующим образом: вырабатывается десять операторов, которые затем предварительно испытываются на последних двух примерах каждого образа, хранящихся для этой цели в памяти. Это предварительное испытание предназначено для выбора из этих операторов такого, который давал бы характеристики, более или менее инвариантные для примеров одного и того же образа, но различные для разных образов. Так как для этих операторов исходными характеристиками могут служить те, которые были получены при работе предыдущих операторов того же типа, можно построить функции большой сложности.

5. Только что описанные два типа операторов дают список характеристик, с помощью которых программа пытается распознать неизвестное изображение (фиг. 6). При этом программа накапливает в памяти подобные списки характеристик для каждого образа, который ей прежде встречался. Каждому списку характеристик в памяти соответствует набор трехразрядных чисел (от 0 до 7), каждое из которых указывает текущий вес данной характеристики.

Процесс распознавания происходит следующим образом. Вычисляются разности между характеристиками входного изображения и соответствующими характеристиками из хранящегося

в памяти списка для первого образа. Эти разности затем взвешиваются при помощи соответствующих весов каждой характеристики для данного образа. Затем при помощи общих весов, являющихся средними весов каждой характеристики по всем образам, получается взвешенная средняя разность между входным списком и данным списком в памяти. Эта средняя разность, умноженная на некоторый окончательный вес «средней разности», дает «оценку разности» для этого списка в памяти. После

Название образа	Матричные операторы									Комбинаторные операторы		
	оператор 1				оператор 2				. .	характеристика		
	N	X	Y	R^1	N	X	Y	R^2		$m-2$	$m-1$	m
?	2	2	2	2	2	3	1	6	. . .	6	7	4
A	3	3	4	1	4	0	1	1	. . .	1	2	3
B	2	2	3	2	3	3	2	5	. . .	4	7	5
C	4	5	6	5	1	0	0	4	. . .	2	1	7

Фиг. 6 Выходы операторов для неизвестного изображения заносятся в таблицу, аналогичную таблицам, хранящимся в памяти машины

подсчета оценки разности для каждого списка в памяти название списка с наименьшей оценкой печатается как название входного образа (фиг. 7).

6. После распознавания всех изображений программа модифицирует вес каждой характеристики для тех образов, которые имеют оценку разности, меньшую оценки разности для правильного образа или незначительно превышающую ее (фиг. 8). Это означает, что программа будет стремиться концентрировать внимание на случаях, для которых разделение представляется затруднительным, так как веса регулируются только для тех образов, которые близки к правильному образу в смысле оценок разности и затрудняют идентификацию входного изображения. Правильный образ сравнивается по очереди с каждым из таких близких образов. Каждая характеристика из списков в памяти отдельно исследуется для пары образов, и определяется, был ли бы выбран правильный образ, если бы выбор делался только на основании этой характеристики Если этой одной характеристики было бы достаточно для опознавания правильного образа, соответствующий вес повышается на единицу, если же на основе этой характеристики опознавание было бы неправильным,

соответствующий вес уменьшается на единицу. Наконец, если эта характеристика не дает никакой информации, например если

Образ A

Характеристики (A)	3	3	4	1	4 . . . 3
Вход (?)	2	2	2	2	2 . . . 4
Разность $\lvert A - ? \rvert$	1	1	2	1	2 . . . 1
Веса образа	2	3	1	2	0 . . . 3
Общие веса	3	3	1	1	0 . . . 3
Веса разности X	6	9	2	2	0 . . . 9

Взвешенная средняя разность	Средний вес разности	Оценка разности
$\frac{28}{27} = 1{,}04$	61	63

Образ B

Характеристики (B)	2	2	3	2	3 . . . 5
Вход (?)	2	2	2	2	2 . . . 4
Разность $\lvert B - ? \rvert$	0	0	1	0	1 . . . 1
Веса образа	4	3	2	3	2 . . . 2
Общие веса	3	3	1	1	0 . . . 3
Веса разности X	0	0	2	0	0 . . . 6

Взвешенная средняя разность	Средний вес разности	Оценка разности
$\frac{8}{32} = 0{,}25$	60	15

Фиг. 7. Схема вычислений

Сначала вычисляются разности между характеристиками входного изображения и соответствующими характеристиками в списках, хранящихся в памяти. Эти разности затем взвешиваются при помощи «общих весов» и «весов образа», давая взвешенную среднюю разность для каждого списка в памяти. Взвешенные средние разности, умноженные на соответствующие «веса средней разности», дают «оценки разности» для каждого образа в памяти В качестве названия незнакомого изображения на входе выбирается название образа с наименьшей «оценкой разности».

она одна и та же для обоих образов, то вес уменьшается с вероятностью $^1/_8$. Если образ, сравниваемый с правильным обра-

зом, имеет оценку разности выше, чем у правильного образа, то веса регулируются только в сравниваемом образе. В противном случае веса регулируются у обоих образов. Это означает, что если несколько образов получат оценку, более низкую, чем правильный образ, то веса в правильном образе будут сильно изменены, так как они будут изменяться при сравнении с каждым из этих образов.

Правильная таблица

Разность	1	4	2	3	. . .	4
Веса	4	3	2	3	. . .	1
Приращение	+1	0	+1	—1	. . .	—1
	+1	—1	—1	—1	. . .	+1
Новые веса	6	2	2	1	. . .	1

1-я неправильная таблица

Разность	2	4	5	2	. . .	2
Веса	2	3	1	4	. . .	3
Приращение	+1	0	+1	—1	. . .	—1
Новые веса	3	3	2	3	. . .	2

2-я неправильная таблица

азность	3	1	1	2	. . .	5
Веса	1	2	2	1	. . .	1
Приращение	+1	—1	—1	—1	. . .	+1
Новые веса	2	1	1	0	. . .	2

Фиг. 8 Регулировка весов для определенной таблицы.

Веса характеристик *увеличиваются*, если они давали разности, обеспечивающие правильное распознавание, и *уменьшаются* в противном случае.

Затем модифицируется список характеристик в памяти для только что обработанного образа. Если образ встретился впервые, то его список подсчитанных характеристик просто хранится в памяти вместе с названием. Если образ встретился во второй раз, то каждая из его характеристик заменяется новой с вероятностью $1/2$. При третьей и последующих встречах каждая характеристика замещается другой с вероятностью $1/4$. Так как при каждой встрече будет изменяться около $1/4$ характеристик, то после нескольких примеров обработки изображений данного образа список характеристик в памяти будет стремиться более походить на характеристики ранее обработанных изображений.

Однако, поскольку в процессе обучения будут вырабатываться операторы, дающие инвариантные характеристики для отдельного образа, то список характеристик будет представлять все обработанные примеры. Простое среднее значение величин каждой характеристики не используется потому, что это потребовало бы запоминания более трех разрядов, которые сейчас необходимы для запоминания каждой характеристики, а также запоминания числа, показывающего, сколько раз подсчитывалась каждая характеристика для каждого образа.

В другой схеме, которую мы испытывали, запоминались наибольшее и наименьшее значения, полученные для каждой характеристики, среднее которых использовалось для сравнения со входом. Эта схема очень хорошо работала во всех наших экспериментах, в которых использовалось по нескольку представителей каждого образа. Однако возможно, что при большом числе представителей все характеристики будут в конце концов иметь очень большую дисперсию, т. е. нижняя граница значений будет стремиться к 0, а верхняя — к 7.

7. Веса средней разности, которые используются на заключительном этапе процесса распознавания, обеспечивают только грубую регулировку. Эти веса первоначально имеют некоторое фиксированное значение, например 60, а затем регулируются по тем же парам образов, по которым регулируются веса отдельных характеристик. Вес для правильного образа уменьшается на N, если для N других образов оценка разности оказалась меньше, чем оценка разности для правильного образа, а веса для каждого из таких «неправильных» образов повышаются на единицу.

8. Общие веса характеристик в настоящее время подсчитываются путем усреднения весов образа по всем образам. Они указывают на общий вес каждой характеристики в процессе распознавания и служат основой для построения показателей качества характеристик, управляющих замещением операторов. Так как операторы комбинирования на основе одних характеристик вырабатывают другие характеристики, то такой показатель качества должен отражать как значение характеристики в процессе распознавания, так и роль этой характеристики в процессе создания другой характеристики, которая может оказаться важной.

9. Этот показатель качества формируется следующим образом. Сначала в таблице для показателей качества запоминается общий вес каждой характеристики. Затем, начиная с последнего оператора комбинирования и следуя от него назад по списку этих операторов, добавляют к показателям качества двух характеристик, которые оператор комбинирует, $1/2$ величины показателя

качества характеристики, соответствующей этому оператору. В заключение к каждому показателю качества добавляется удвоенное значение установившегося общего веса характеристики.

10. Когда бы ни вырабатывался новый оператор, характеристики, порождаемые этим оператором, подсчитываются для каждого из возможных образов с использованием для этой цели последнего примера для каждого образа. Эти примеры хранятся в памяти машины. Затем эти вновь подсчитанные характеристики помещаются в список характеристик для соответствующих образов. Для каждой из этих новых характеристик вес образа делается равным 1, так что такая характеристика будет иметь очень малый вес при подсчете оценки разности, пока не окажется, что она способна производить разделение. Так как общий вес характеристики есть просто среднее весов образа, то он для новой характеристики будет также равен 1. Показатель качества новой характеристики, которая не комбинируется для получения других характеристик, делается равным 3 и в дальнейшем будет увеличиваться, если оператор оказывается хорошим. Если же показатель качества становится меньше 3 (или, в случае матричного оператора, если средняя величина показателей качества его четырех характеристик становится меньше 3), оператор отбрасывается и вместо него вырабатывается новый.

Веса образа играют решающую роль как непосредственно в самом процессе распознавания, так и в получении информации, которая в конце концов определяет выработку новых операторов для замещения плохих. Так как изменение этих весов производится избирательно, основываясь на хорошем или плохом разделении для нетривиальных пар образов, то операторы, отброшенные программой, должны быть бесполезными при разделении в более трудных случаях. Кроме того, так как веса обычно изменяются более резко, когда машина делает неправильный выбор, то с достаточно большой вероятностью матричные операторы 5×5 будут выделены из неправильно опознанных образов. Хотя правила управления процессом обучения, по-видимому, во многих случаях довольно произвольны и их действие трудно описать количественно, тем не менее легко показать эффект их применения качественно. Так, например, очевидна способность программы концентрировать внимание на трудных задачах.

Интересно отметить, что программа запоминает информацию по крайней мере в трех различных местах: 1) в списке характеристик, хранящихся в памяти машины, 2) в величинах различных весов и 3) в системе операторов, используемых программой.

Результаты экспериментов с первоначальной программой

Первоначальная программа была написана для машины IBM-709, и для нее потребовалось около 2000 команд. Для обработки одного объекта требуется около 25 *сек*, если он может принадлежать к одному из пяти различных образов, и 40 *сек*, когда каждый объект может относиться к одному из десяти образов.

Хотя такие времена обработки не являются чрезмерными, они все же достаточно велики, чтобы сделать практически невыполнимым проведение очень длинных серий экспериментов.

В некоторых экспериментах, проведенных ранее, мы использовали 48 заранее запрограммированных матричных операторов. Они были разработаны для измерения таких характеристик, как прямые и кривые линии, концы вертикальных и горизонтальных отрезков и других признаков. Для испытания программы использовались семь различных наборов по пять рукописных букв: *A*, *B*, *C*, *D* и *E*. Они имели значительные искажения и вариации размеров, но не были очень сильно повернуты относительно вертикальной оси.

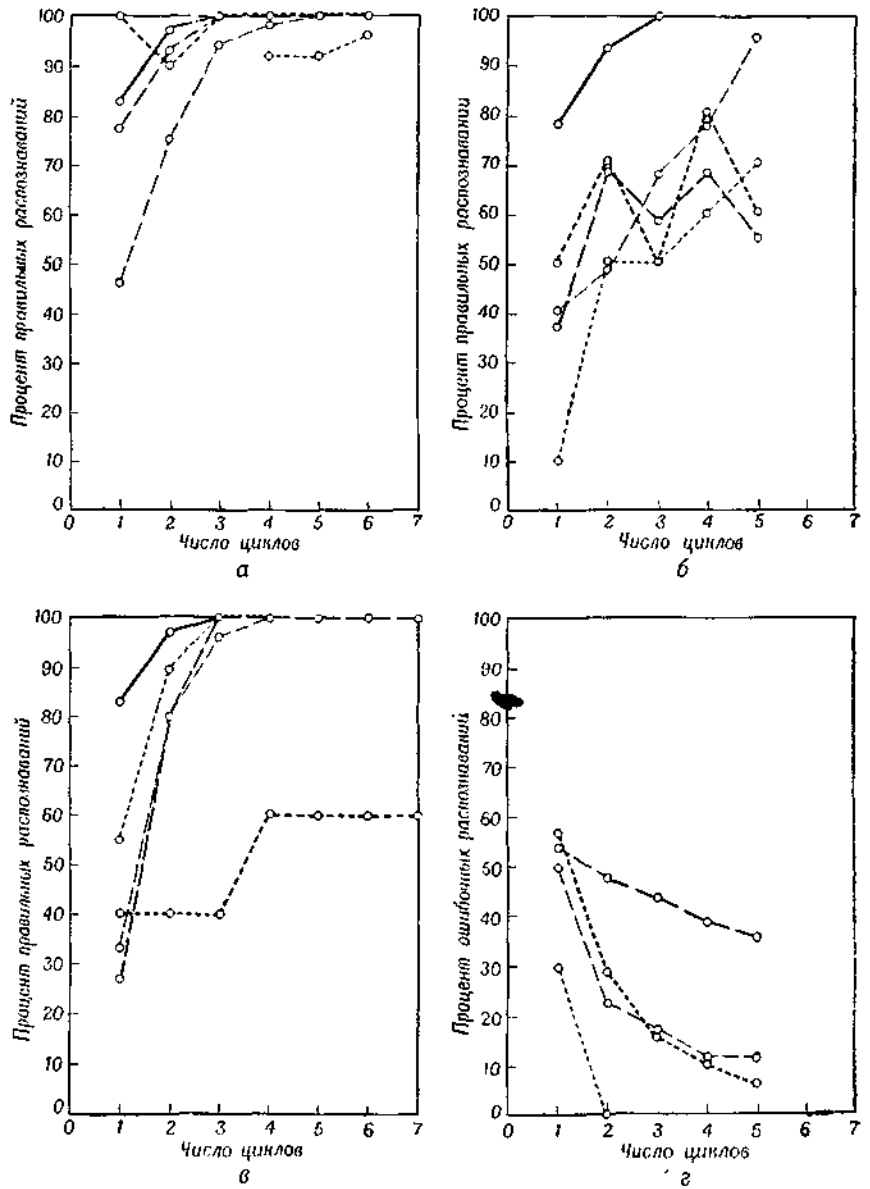
В последних трех-четырех сериях экспериментов программа правильно распознавала приблизительно от 70 до 80% изображений в зависимости от различных изменений в правилах, управляющих процессом обучения.

В одном эксперименте в качестве операторов первого уровня использовались отдельные ячейки входной матрицы. На основе этих операторов строились операторы комбинирования более высоких уровней. Процент распознавания в этом случае был немногим больше 30. Окончательно этот вариант программы испытывался без использования каких-либо заранее запрограммированных операторов. Процент распознавания для этой программы достиг 97 на знакомых и 70 на незнакомых представителях десяти букв алфавита. Программа показала также возможность распознавания простых рисунков.

Результаты экспериментов с исправленной программой

Первоначальная программа была модифицирована в основном для увеличения скорости работы, а также для упрощения некоторых логических схем. Чтобы можно было использовать логические команды IBM-709, все характеристики и их веса были представлены двоичными числами. Исправленная программа запоминала для каждого оператора девять одноразрядных чисел,

показывающих наличие или отсутствие изображения в каждой из девяти частей матрицы. Характеристики, которые вырабатываются операторами для каждого образа из памяти машины.



имеют веса либо 1 (используемые операторы), либо 0 (неиспользуемые операторы). Операторы отбрасываются, когда они дают неправильные выходы на последнем, n-м примере, для которого программа в целом дала неправильный ответ. В этой программе не подсчитывались «общие веса». Все эти изменения повлекли за собой увеличение скорости работы приблизительно в 40 раз. Таким образом, эта программа затрачивает около 1 *сек* на обработку объекта в случае десяти образов (буквы алфавита) на машине IBM-709 и менее чем 0,2 *сек* на IBM-7090. Эта программа, вероятно, слабее, чем первоначальная, так как для нее фактически нет диапазона, в пределах которого следует искать хорошую систему весов для операторов. Но из-за повышенной скорости ее работы мы проводили бо́льшую часть экспериментов именно с этим вариантом программы.

Ускоренная программа испытывалась на нескольких различных типах изображений. Результаты этих испытаний показаны на фиг. 9. В большинстве случаев результаты были примерно одинаковы как на «знакомых» изображениях (которые программа прежде обработала и на которых, следовательно, обучилась), так и на «незнакомых» изображениях (которые отличались от изображений, использованных при обучении, или рисовались различными людьми). На фиг. 9, *а* показаны результаты работы программы, когда она обучалась распознаванию рукописных прописных букв, написанных разными людьми. Эти результаты показывают относительно небольшое уменьшение процента правильных распознаваний по мере увеличения числа образов, по крайней мере вплоть до 26 букв английского алфавита. Таким образом, после того, как программа шесть раз

Фиг. 9. Результаты работы программы.

а — результаты распознавания рукописных букв. Указан процент правильно опознанных изображений для нескольких различных наборов, содержащих 26, 10 и 5 изображений. Программа испытывалась как на знакомых, так и на незнакомых изображениях. 26 букв: ——— знакомые (3 набора), — — — — незнакомые (1 набор); 10 букв: —·—·— знакомые, —·—·— незнакомые (1 набор); 5 букв: ——— знакомые (7 наборов);

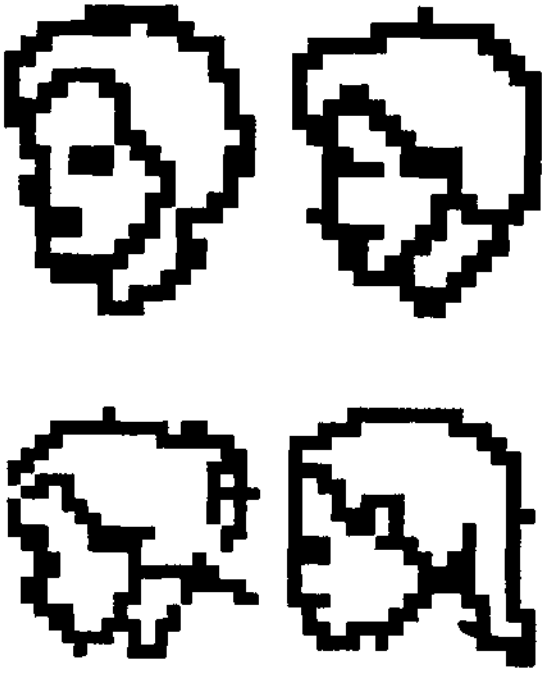
б — результаты для двух дополнительных наборов по 10 изображений: 1) контурных рисунков (различные изображения 5 человеческих лиц и 5 предметов) и 2) арабских рукописных букв (написанных одним и тем же человеком). Программа проверялась как на знакомых, так и на незнакомых образах. Стандартные образы: ——— рукописные буквы (4 набора по 10 букв); рисунки: — — — знакомые (4 набора по 10 изображений), — — — — незнакомые (1 набор); арабские рукописные буквы: —·—·— знакомые (4 набора по 10 букв), —·—·— незнакомые (1 набор).

в — результаты для двух дополнительных наборов по 5 образов: 1) устно произносимых цифр (произнесенных разными людьми); 2) рукописных отрывков (написанных разными людьми). Стандартные образы: ——— рукописные буквы (7 наборов по 5 букв); рукописные отрывки (написанные разными людьми): — — — знакомые (4 набора по 5 букв), незнакомые —·—·— (1 набор); цифры (произносимые разными людьми): — — — знакомые (4 набора по 5 цифр); — — — — незнакомые (13 незнакомых примеров).

г — результаты эксперимента по сравнению процента ошибок программы со средним процентом ошибок человека (среднее бралось по ответам 6—10 людей в каждой точке) при опознавании одного «трудного» и одного «легкого» набора «бессмысленных» образов (клякс). Оба множества содержали по 5 различных примеров каждого из 5 образов. Люди: — — — трудный набор, —·—·— легкий набор; программа: — — — трудный набор, — — — — легкий набор.

обработала 26-буквенный алфавит, процент распознавания составлял 100 на знакомых и 96 на незнакомых объектах.

На фиг. 9, *б* представлены результаты для двух наборов[1] по 10 изображений. Это были: 1) условные рисунки человеческих



Фиг. 10 Образцы схематического изображения человеческого лица, с которыми работала программа (закодированы вручную на матрице 20×20 после того, как на нее был спроектирован оригинал из газеты).

Наверху — два различных варианта мужского лица; внизу — два различных варианта женского лица (При рассмотрении рисунка читателю рекомендуется отодвинуть его на расстояние вытянутой руки и слегка прищурить глаза. — *Прим. ред.*)

лиц и простых предметов (таких, как ботинки или клещи), каждый из которых скопирован с различных изображений, которые бывают, например, на рекламных рисунках или в каталогах товаров, высылаемых по почте (на фиг. 10 показаны несколько примеров таких рисунков), и 2) рукописные арабские буквы, написанные одним и тем же человеком. Программа правильно опознавала 95% знакомых и 70% незнакомых рисунков, а после пятикратной обработки — 60% знакомых и 55% незнакомых арабских букв. На фиг. 9, *в* представлены результаты работы программы на двух различных наборах по 5 изображений в каждом: 1) преобразованные в цифровую форму и уменьшенные по уровню (огрубленные) спектрограммы звуков речи (числа «ноль», «один», «два», «три» и «четыре», произнесенные по-английски разными людьми) [1093] и 2) арабские буквы, написанные разными людьми. После четырехкратной обработки программа правильно распознала 100% как знакомых, так и незнакомых произнесенных цифр, а после трехкратной обработки она правильно опознала 100% знакомых рукописных букв. Она достигла 60%-ного распознавания незнакомых букв, но, если бы у нее были более благоприятные возможности для обучения, вероятно, произошло бы дальнейшее улучшение ее работы на этих же входных данных (раз программа правильно распознает все 100% знакомых объектов, она не получает ощутимой пользы от последующего обучения на том же материале).

[1] В наборе содержится по одному представителю каждого образа. — *Прим. ред.*

Наконец, сравнивались проценты неправильного распознавания «бессмысленных» образов программой и человеком. Образ такого типа сводит к минимуму влияние жизненного опыта человека и вытекающее отсюда влияние всевозможных ассоциаций. На фиг. 9, *г* представлены результаты двух таких экспериментов, причем в обоих экспериментах программа работала заметно лучше любого человека. Были повторены три эксперимента, ранее описанные в литературе по психологии. Во всех случаях программа работала значительно лучше, чем человек [1094].

Испытывалась также способность этой программы обрабатывать составные образы, такие, как рукописные слова и произнесенные предложения. Были дополнительно написаны простые подпрограммы, позволяющие основной программе воспринимать на входе матрицы любой длины, делать очень примитивные пробные разделения входной матрицы (границами служат каждый n-й столбец, причем n обычно порядка 7, и столбцы, у которых меньше двух заполненных ячеек) и выносить решение при большом разнообразии вариантов и различных разделениях входной матрицы. 100% правильной работы программы достигалось, когда требовалось разделить и опознать слова или фонемы в простом предложении «Did Dad say before?», произнесенном разными людьми.

На рукописных последовательностях «pattern one», «pattern two», «pattern three» (написанные разными людьми) процент

правильного распознавания букв достигал приблизительно 60. Эти эксперименты не дают сколько-нибудь адекватного представления о круге задач, которые могут встретиться с такими требованиями. Но они показывают, что программа способна по крайней мере начать обработку непрерывных объектов. И, по-видимому, сравнительно просто улучшить работу программы, добавив более эффективные методы разделения и прямой метод использования информации из контекста (например, использование частот, с которыми встречаются в данном языке определенные наборы по n букв) [1].

[1] Последний раздел статьи, содержащий обсуждение результатов, при переводе опущен, так как он в свете новейших работ не представляет большого интереса — *Прим. ред.*

ЧАСТЬ II

Моделирование процессов познания

ВВЕДЕНИЕ

Исследования, о которых говорится в этой части книги, касаются создания вычислительных программ для моделирования информационных процессов, лежащих в основе человеческого мышления. Если основная цель исследований ч. I состояла в создании программ для решения интеллектуальных задач, в настоящей части речь пойдет о создании машинных программ, решающих интеллектуальные задачи теми же путями, что и человек.

Для выяснения разницы между исследованием искусственного разума и моделированием процессов познания полезен следующий пример. Исследователь в области искусственного разума, намеренный создать машинную программу для игры в шахматы, будет вполне удовлетворен лишь в том случае, если его программа будет хорошо играть в эту игру, желательно лучше, чем самый сильный шахматист. Исследователь же, заинтересованный в моделировании познавательной деятельности некоторого конкретного шахматиста во время игры, будет неудовлетворен, если его программа будет играть в шахматы лучше (или хуже) этого шахматиста, так как такой исследователь хотел бы, чтобы его программа делала бы те же ходы, что и данный шахматист, независимо от того, являются ли эти ходы хорошими или плохими.

Зачем составлять машинные программы, решающие задачи тем же путем, что и человек?

Исследователи разрабатывают программы, решающие задачи подобно человеку, для того, чтобы лучше понять и суметь предсказывать некоторые аспекты человеческого поведения. Машинная программа является моделью, отражающей гипотезу

исследователя об информационных процессах, лежащих в основе работы мозга. Реализация программы на вычислительной машине позволяет выявить содержащиеся в этой модели предсказания. Эти предсказания сравниваются с реальным поведением человека. Обычно между полученными предсказаниями и поведением человека обнаруживаются некоторые различия; тогда модель видоизменяют для уменьшения этих различий. Затем весь процесс повторяется снова. В конечном итоге исследователь рассчитывает получить модель, которая будет хорошо предсказывать соответствующее поведение. В процессе последовательной проверки и улучшения своей модели исследователь приобретает уверенность в том, что его модель правильно отражает процессы, лежащие в основе изучаемого поведения.

Этот краткий обзор действий исследователей, использующих машинные модели для изучения процессов человеческого мышления, показывает, что эти исследователи применяют ту же методику, что и все другие ученые. Единственное отличие здесь состоит в представлении модели в виде машинной программы и использовании вычислительной машины для выявления предсказаний, к которым приводит данная модель.

В чем преимущества представления моделей человеческого мышления в виде машинных программ?

Исследователь может выразить модель человеческого поведения множеством различных способов. Возможно, самым распространенным способом является естественный язык, т. е. язык, на котором мы обычно общаемся друг с другом, будь то английский, французский, немецкий, русский или санскрит. Если бы мы хотели составить модель игры в шахматы на естественном языке, то описание модели могло бы начинаться, например, следующими словами:

> «До того как сделать ход, шахматист проверяет, находится ли его король или ферзь под угрозой . . .»

Модель игры в шахматы может быть представлена также на языке математики. Третьей же возможностью описания модели является программа для цифровой вычислительной машины.

Выбор «среды» для создания модели зависит от характеристик этой среды, характеристик модели и возможностей, которыми располагает исследователь. Естественный язык представляет чрезвычайно удобную среду, и, вероятно, всякая мыслимая модель может быть описана на естественном языке. Кроме того, модели, описанные на естественном языке, обычно легко сообщить другим лицам. Впрочем, утверждение относительно легкости описания модели и ее передачи на естественном языке

для чрезвычайно сложных моделей нуждается в уточнении. Самый большой недостаток естественного языка как среды для построения моделей, пожалуй, состоит в трудности строгого определения предсказаний некоторой модели, выраженной в этой среде. Это связано с тем, что, выражая свои мысли на естественном языке, мы используем слова с не вполне ясным значением. Так, в примере описания на естественном языке поведения шахматиста, приведенном выше, неясно, что значит слово «проверяет» и что означает «под угрозой».

Многие модели, выраженные на языке обычной математики, могут быть проанализированы точными способами. Если исследователь располагает соответствующими математическими методами или способен их разработать, он имеет возможность установить выводы, вытекающие из его модели. Если методы решения некоторой системы уравнений неизвестны или недоступны программисту, он находится в не лучшем положении, чем если бы имел модель, выраженную на естественном языке. Результатом последнего условия является ограничение исследователя, строящего модель, только классом таких моделей, для которых ему известны методы решения. К сожалению, это ограничение может оказать пагубное влияние на исследователя, заставляя его чрезмерно упрощать сложные ситуации. Как правило, математические модели поведения человека довольно изящны и просты. Иногда ограниченность математического аппарата принуждает к нежелательным компромиссам в построении моделей и уменьшает их способность к предсказанию моделируемого поведения.

Новейшей средой для построения моделей является программа для вычислительной машины. Машинная программа как среда для создания моделей обладает рядом существенных преимуществ. Одним из них является то, что с помощью такой программы может быть представлен широкий класс моделей. Ограничения, наложенные на размеры и сложность реализуемых моделей, настолько незначительны, что их в большинстве случаев можно не учитывать. Единственное существенное ограничение для составителя программы заключается в том, что все его утверждения должны быть недвусмысленными и полными. Например, прежде чем высказывание, касающееся модели шахматиста, приведенное на стр. 274, может быть запрограммировано, необходимо уточнить значение слов «проверяет» и «под угрозой». Это условие следует рассматривать скорее как преимущество, чем как стесняющее ограничение, потому что оно побуждает создателя модели к необходимой строгости при описании процессов человеческого мышления. Несмотря на свободу, которой располагает исследователь при создании машинных моделей, он сохраняет возможность точного определения вытекаю-

щих из модели выводов, так как вычислительная машина может выполнить программу и определить поведение программы в конкретной ситуации.

Построение моделей на вычислительных машинах облегчается развитием машинных языков, которые освобождают программиста от забот по микропрограммированию. Мощные алгебраические языки, вроде ФОРТРАНа и различных разновидностей АЛГОЛа составляют один класс этих машинных языков более высокого порядка. Другим классом таких языков являются языки списковой структуры, такие, как ИПЛ (IPL), ФЛПЛ (FLPL), ЛИСП (LISP) и КОМИТ (COMIT), которые были созданы для работ в области искусственного интеллекта и моделирования процессов познания. Эти языки со списочной структурой позволяют создавать динамические связи в памяти и обладают многими другими свойствами, облегчающими программирование.

Что означает анализ «на уровне переработки информации»?

При создании моделей, описываемых в этой части книги, исследователи должны принять несколько последовательных решений. Прежде всего необходимо выбрать поведение, для которого они хотят создать модель; все работы, описанные в этой части, сосредоточивают свое внимание на процессах человеческого мышления. Затем исследователям нужно выбрать способ построения моделей; все модели, описанные в этой части, представляют собой машинные программы. Наконец нужно сделать выбор относительно уровня, на котором будет анализироваться модель; во всех приведенных здесь работах исследователи выбрали в качестве уровня анализа уровень переработки информации.

Каждое явление может быть объяснено на различных уровнях. В данном случае это означает, что процессы человеческого мышления могут быть объяснены в терминах происходящих в мозгу электрических и химических процессов, в терминах нейронной организации мозга или в терминах процессов переработки информации. На первый взгляд кажется, что объяснения на каждом из этих уровней совершенно различны, поскольку мы в настоящее время еще не смогли создать общую теорию мышления, которая бы одновременно учитывала все, что происходит в мозге, в виде электрических, химических, нейродинамических и информационных процессов. Будем надеяться, что когда-нибудь все эти уровни объяснения будут объединены и между ними будут установлены взаимные отношения. В настоящее же время

для исследователей, видимо, самым разумным будет проводить работу параллельно на всех уровнях, время от времени сравнивая результаты, так как одного наилучшего уровня объяснения процессов мышления не существует. Мы можем оценивать предложенные модели на том или ином уровне лишь по их способности предсказывать поведение. На любом из этих уровней объяснения модели мышления человека могут быть реализованы в виде машинных программ, и вычислительные машины уже используются для изучения нейронов, нейронной организации и процессов переработки информации (см. библиографию).

Работы, описанные в настоящей части книги, дают анализ процессов человеческого мышления на уровне переработки информации, или на уровне оперирования символами. Исходной предпосылкой такого подхода является то, что сложные процессы мышления строятся из элементарных процессов оперирования с символами. Основной набор этих элементарных процессов может быть следующим: считывание символа, запись символа, копирование символа, стирание символа и сравнение двух символов. Другой важной предпосылкой этого подхода является возможность избирать различные пути дальнейших действий в зависимости от исхода операции сравнения. Если сравниваются два символа и при этом обнаруживается их тождество, то система переработки информации выбирает одну схему работы, т. е выполняет одну последовательность элементарных процессов. Если же символы различны, то система выполняет другую последовательность процессов.

Задача исследователя состоит в нахождении такой упорядоченной последовательности этих основных процессов, которая при наличии необходимой информации приводит к поведению, неотличимому от поведения человека, когда ему дается та же информация. «Упорядоченная последовательность этих основных процессов» и является «моделью», о которой мы прежде говорили. Таким образом, модель переработки информации для шахматиста состояла бы из последовательности этих основных процессов. Мы могли бы сообщить этой модели необходимую начальную информацию, т. е. шахматную задачу, и с помощью вычислительной машины найти предсказания, даваемые моделью.

Как создаются информационные модели человеческого мышления?

Несмотря на то что в настоящее время не существует установленной процедуры для создания модели мышления на уровне процессов переработки информации, исследователи, работы

которых представлены в настоящей части книги, вынуждены были проделать какую-то определенную последовательность шагов в процессе создания своих моделей. Перечислим основные шаги, необходимые при создании таких моделей.

Исследователь начинает с того, что определяет область человеческого поведения, которая его интересует, например «решение задач», «игра», «самообучение». Затем он обычно сосредоточивает внимание на поведении в конкретной ситуации, например при решении логических задач, игре в шахматы, распознавании бессмысленных слогов. Частная задача выбирается исследователем на основании его личных интересов или потому, что у него уже есть некоторое представление о том, как люди ведут себя при решении той или иной задачи. Для того чтобы создать предварительную модель поведения, исследователь должен иметь представление или ряд представлений о поведении, проявляющемся в данной задаче. Такое первоначальное представление может явиться плодом наблюдения за поведением людей, результатом опроса людей, что́ они делают в процессе решения задачи, или же результатом размышлений о типе устройства, которое потребовалось бы для решения данной задачи.

Следующим шагом является построение модели, т. е. составление программы для вычислительной машины, реализующей эти представления. В ходе этой работы исследователь сталкивается с недостаточностью своих первоначальных представлений. Он обнаруживает, что ему необходима дополнительная информация. Чтобы получить эту информацию, он может заново проанализировать прежние данные или провести новые эксперименты. Так исследователь сталкивается с одним из преимуществ машинного моделирования — с требованием полноты и точности в описании работы мозга. После многих видоизменений и переделок первоначальной программы получается окончательная модель.

Затем исследователь ставит перед моделью ту же задачу, что и перед живыми испытуемыми, например шахматную задачу, логическую задачу, список бессмысленных слогов. Программа испытывается с этой информацией. По существу модель можно рассматривать как «искусственного субъекта», с которым повторяется эксперимент, проведенный над живым человеком. Поведение программы, т. е. шахматные ходы, шаги при решении логических задач, ответы в задаче обучения, сравнивается затем с поведением испытуемого. Там, где это возможно, производится также сравнение тех внутренних процессов, которые приводят к наблюдаемому поведению.

Как проверяются машинные модели человеческого мышления?

Существующая методика сравнения или испытания машинной модели зависит от того, является ли данная модель моделью конкретного испытуемого или какого-то обобщенного или идеализированного испытуемого (см. ст. Фейгенбаума, стр. 305—306). Модель вербального обучения Фейгенбаума, модель обучения понятиям Ханта и Ховленда и модель социального поведения Галлахорнов — это модели обобщенного или идеализированного испытуемого. Приведенные здесь и в перечисленных работах данные касаются сравнения поведения моделей с поведением идеализированных испытуемых.

В отличие от этого модели Ньюэлла и Саймона, Кларксона, Фельдмана являются моделями конкретных испытуемых. В этом случае проводится сравнение между конкретно выбранным субъектом и конкретно выбранной моделью. При таких сравнениях возникает много проблем. Ни одна из этих проблем не является новой или необычной для машинной модели, но они по-новому освещаются благодаря способу сравнения, применяемому при этих моделях индивидуального поведения. Одной из таких проблем является выбор параметра сравнения. Другая проблема касается связи между последовательными предсказаниями модели[1]. Предсказание модели в момент t зависит от поведения ее в моменты $t-1$, $t-2$, $t-3$ и (или) от других решений данной модели. Если модель принимает решение, отличное от решения испытуемого в соответствующий момент времени, то такое расхождение может привести к дальнейшим расхождениям и (или) к ошибочным заключениям. Так, если модель шахматиста делает первый ход, отличный от первого хода шахматиста, мы вряд ли можем надеяться, что последующие ходы совпадут. Чтобы устранить эту трудность, некоторые исследователи (см. ст. Фельдмана) выбрали стратегию «возвращения модели на шаг назад» (setting the model back on the track) после принятия каждого решения. В случае шахматной игры это, например, означает, что после первого хода модели этот ход сравнивается с первым ходом шахматиста. Если ходы совпадают, делается ход за противника и модель переходит ко второму ходу. Если же первый ход модели отличен от первого хода шахматиста, это различие фиксируется и, прежде чем продолжать игру, ход машины заменяется на ход шахматиста.

[1] Предсказание модели в данный момент понимается как определенный шаг в работе модели, по которому можно вынести суждение о поведении испытуемого в аналогичной ситуации — *Прим. ред.*

Эта стратегия «возвращения на шаг назад» имеет целью устранить рассмотренную зависимость и делает каждое очередное решение модели по возможности независимым от предыдущих решений модели. Такая процедура позволяет лучше оценить качество модели, чем процедура сравнения, которая допускает нарастание ошибок (или принятие неверных решений). К сожалению, стратегию «возвращения на шаг назад» не всегда можно применить — либо ввиду отсутствия необходимой промежуточной информации (как, например, в шахматах или в эксперименте с двоичным выбором), либо же из-за трудности возвращения модели на пройденный шаг, как это имеет место в случае программы GPS.

Каковы нерешенные проблемы исследования процессов познания на моделях?

Существует два класса проблем по моделированию исследования процессов познания. Главную проблему составляют те области человеческого познания, относительно которых мы еще очень мало знаем. Перечисленные выше проблемы обучения эвристикам, индуктивного вывода и пользования естественным языком являются тремя хорошими примерами из той области, в которой наши знания о процессах человеческого мышления ничтожны. К этим областям в ближайшем будущем должен быть проявлен особый интерес.

Второй класс таких проблем образуют методические и методологические проблемы моделирования процесса познания. Три из этих проблем совершенно очевидны:

1. Испытание модели и оценка параметров. Модели редко бывают совершенно неудовлетворительными, и исследователям лишь редко приходится полностью отвергать их. Более обычной процедурой является попытка обнаружить источник ошибок и исправить имеющуюся модель [411]. С тех пор как машинные модели стали такими громоздкими и сложными, методы для обнаружения источников ошибок приобрели чрезвычайное значение. Очень важным является также эффективное использование доступной информации о процессах человеческого мышления.

2. Проведение экспериментов. Многие работы по моделированию процессов познания сводятся к изучению протокола опроса испытуемого, произносящего свои мысли вслух. Необходимо провести дальнейшую работу по этому и другим методам получения информации о процессах мышления [135]. Проведена также некоторая работа (которая, безусловно, должна быть продолжена) по определению эффективности метода «мышления вслух» [236, 354]. Необходимо разработать особые методы для явного выражения скрытых процессов мышления испытуемых. Одной из таких методик является использование искусственных языков для формулирования задач [1149]. Во всех этих областях требуется проведение большой дополнительной работы.

3. Организация и форма представления программ. Мы постоянно говорили о том, что выражение моделей в виде машинных программ допускает значительную гибкость в высказываниях. Это утверждение действительно справедливо. Однако по мере возрастания размеров и сложности моделей и программ организация этих программ вызывает ряд серьезных проблем. Языки программирования более высоких порядков и особенно языки обработки списков решают некоторые из этих проблем, но многие из них еще остаются нерешенными [808].

Как вычислительная машина повлияет на изучение процессов познания?

Использование современных цифровых машин для изучения процессов человеческого мышления может оказать двоякое влияние. С одной стороны, исследователи в области искусственного разума стремятся понять, как человек решает всякого рода интеллектуальные задачи. В этом отношении вычислительные машины внесут свой вклад в расширение наших знаний о процессах человеческого мышления. С другой стороны, вычислительная машина образует основу мощного метода для изучения процессов человеческого мышления и поэтому должна внести значительный вклад в понимание познавательных процессов. Сумма знаний о человеческом мышлении никогда не сможет «удовлетворить спрос». Однако прогноз в отношении прогресса исследования познавательных процессов человека является обнадеживающим.

РАЗДЕЛ 1

РЕШЕНИЕ ЗАДАЧ

Еще со времен древних греков изучение поведения человека при решении задач привлекало внимание ученых. Это направление получило новый стимул в связи с появлением современных вычислительных машин, что дало возможность представлять сложные поведенческие модели в виде программ для вычислительных машин и изучать эти модели при выполнении указанных программ машиной.

Авторы предлагаемой ниже статьи и их коллега Шоу являются пионерами в использовании вычислительных машин в области моделирования поведения человека при решении задач. Им удалось создать модель, поведение которой при решении логических задач удивительно похоже на поведение человека при решении тех же задач. Значение их работ для психологии заключается в составлении общей программы решения задач — GPS. В данной статье поведение одного варианта GPS сравнивается с поведением человека при решении той же задачи. В других статьях столь же успешно проводится сравнение других вариантов GPS с поведением людей.

Несмотря на то что каждый человек вел себя различно в каждом отдельном случае, практически все люди, которых Ньюэлл, Саймон и их ассистенты испытывали при решении логических задач, показали одинаковую совокупность основных принципов, используемых ими при решении задач, включающих также использование «анализа соотношения средств и цели» (means-ends analysis). Хотя и другие исследователи занимались вопросом анализа соотношения средств и цели при решении задач человеком [290], Ньюэлл, Шоу и Саймон впервые точно описали причинно-следственный анализ и испытали схему решения задач на вычислительной машине. Они провели большое количество экспериментов по данной схеме на вычислительной машине, программа которой являлась моделью поведения.

Кроме анализа соотношения средств и цели, некоторые испытуемые используют другой не менее эффективный метод, который был назван «планированием» (planning). Испытуемый, применяющий метод планирования, сначала абстрагирует или упрощает сложную задачу, а затем решает эту более простую задачу, используя информацию, полученную при решении упрощенной задачи, для решения сходной с ней сложной задачи. Этот метод был также включен в GPS и более полно описан в других статьях [795].

GPS — не просто модель поведения человека при решении логических задач. При создании GPS была сделана попытка отграничить общую часть программы, в которой в основном описан анализ соотношения средств и цели, от части, содержащей в основном описание специфических задач, связанных с заданной средой. Имея общую часть GPS и соответствующие «заданные среды», оказалось возможным запрограммировать вычислительную машину для доказательства тригонометрических тождеств [795], балансирования сборочного конвейера [1075, 1076] и создания всевозможных иных программ для вычислительной машины.

11. GPS — ПРОГРАММА, МОДЕЛИРУЮЩАЯ ПРОЦЕСС ЧЕЛОВЕЧЕСКОГО МЫШЛЕНИЯ

А. Ньюэлл и Г. Саймон

Введение

Эта статья касается вопросов психологии человеческого мышления. Предлагается теория, объясняющая, каким образом некоторые люди пытаются решать простые формальные задачи. Исследования, послужившие основой для данной теории, тесно связаны с проблемами обработки информации и построения «разумных» автоматов, а сама теория выражена в виде программы для вычислительной машины. Быстрый технический прогресс в области программирования цифровых вычислительных машин для решения творческих задач создал условия для реализации этой теории.

Часто приходится слышать, что следует проводить четкую границу между попыткой выполнить при помощи машины задачи, решаемые человеком, и попыткой *моделировать* процессы, которые человек действительно использует при решении этих же задач. Программа, которая обсуждается в данной статье, GPS (General Problem Solver — универсальный решатель задач) совмещает оба эти подхода с пользой как для того, так и для другого. Первоначально GPS описывалась как попытка создать программу для решения задач [795, 798], и в наших исследованиях она остается основным средством для изучения проблем искусственного разума. Но вместе с тем варианты GPS моделируют поведение человека [801]. Именно на этом последнем аспекте — использовании GPS в качестве теории поведения человека при решении задач — мы и хотим сосредоточить внимание в этой статье, причем особое внимание будет уделено соотношению теории и экспериментальных данных.

Чтобы дальнейшее обсуждение было более понятным, остановимся вкратце на некоторых моментах из истории психологии. В начале нашего века основным направлением психологии был ассоциационизм. Это была атомистическая доктрина, которая постулировала теорию на основе небольшого числа строго определенных элементов — ощущений или представлений, которые связываются или ассоциируются друг с другом в неизменном виде. Это была механистическая теория, объяснявшая формирование новых ассоциаций простыми и неизменными законами совпадения событий во времени и пространстве. Таковы были ее основные предположения. Согласно этой теории, поведение формируется в результате потока ассоциаций: каждая ассоциация

создавала базу для других ассоциаций и достигала новых связей с «ощущениями», поступающими из внешнего мира.

В первом десятилетии нашего века как реакция на эту теорию возникли работы Вюрцбургской школы. Отвергая идею всецело самодовлеющего потока ассоциаций, эта школа ввела понятие «задачи», или «цели» (Aufgabe), как необходимого фактора описания процесса мышления. Стоящая перед человеком «задача» дает направление его мышлению. Важным нововведением Вюрцбургской школы было систематическое использование самонаблюдения для выяснения процесса мышления и содержания сознания. В результате произошло некое слияние механицизма с феноменализмом, что в свою очередь привело к созданию двух совершенно противоположных направлений: бихевиоризма и гештальт-психологии.

Бихевиористы настаивают на том, что самонаблюдение является весьма ненадежной субъективной процедурой, бесплодность которой полностью обнаружилась во время дискуссий по поводу «внеобразного», абстрактного мышления. Бихевиористы по-новому сформулировали задачу психологии как задачу выяснения реакций организма на внешние стимулы при объективном измерении как реакции, так и стимула. Однако бихевиоризм принял также и на деле подкрепил механистическую гипотезу о том, что связи между стимулом и реакцией создаются и поддерживаются только как простые детерминированные функции внешней среды.

Гештальт-психологи, с другой стороны, встали на прямо противоположную точку зрения: они отвергли механистическую основу доктрины ассоциационистов, но признали ценность феноменологического наблюдения. Во многом они придерживались взглядов Вюрцбургской школы, согласно которым мышление есть нечто большее, чем просто ассоциация, — мышление имеет направленность, которая определяется «задачей» или «установкой» самого субъекта. Гештальт-психология разработала эту доктрину совершенно новыми путями — в терминах «холистских» принципов целостной организации.

Современная психология находится в состоянии относительно устойчивого равновесия между полюсами бихевиоризма и гештальт-психологии. Все мы восприняли главные уроки обеих школ: мы скептически относимся к субъективному в наших экспериментах и соглашаемся с тем, что все вводимые понятия в конечном счете должны быть доступны экспериментальному измерению бихевиористскими методами. Но мы признаем также, что человек представляет собой чрезвычайно сложно организованную систему и что простые схемы современных бихевиористов едва ли полностью ее отражают.

Условия эксперимента

В свете излагаемой теории рассмотрим следующую ситуацию: некий человек, студент инженерного факультета американского колледжа, сидит перед доской, на которой написаны следующие выражения:

$$\frac{(R \supset \sim P) \cdot (\sim R \supset Q)}{\sim (\sim Q \cdot P)}.$$

Это задача элементарной символической логики, но студент не знает символической логики. Он знает только, что существуют 12 правил оперирования этими выражениями, содержащие буквы, соединенные «точками» (·), «клиньями» ($\vee$), «подковами» ($\supset$) и «тильдами» ($\sim$), которые заменяют слова «и», «или», «влечет за собой» и «нет». Эти правила, приведенные в табл. I, показывают, как выражения (в «хвосте» стрелок) могут быть преобразованы в другие выражения (в «голове» стрелок). (Двусторонние стрелки показывают преобразования, которые могут быть выполнены в обоих направлениях.)

Таблица 1

Функции символической логики

Объекты (формулы) образуются построением выражений, состоящих из букв (P, Q, R, ...) и связок • (точки), $\vee$ (клинья), $\supset$ (подковы) и $\sim$ (тильды), например $P \sim Q$, $P \vee Q$, $\sim (R \supset S) \cdot \sim P$. Выражение $\sim \sim P$ полностью эквивалентно P.

Существуют 12 правил преобразования выражений (где A, B и C могут быть либо выражениями, либо подвыражениями):

R1. $A \cdot B \rightarrow B \cdot A$;
$A \vee B \rightarrow B \vee A$.

R2. $A \supset B \rightarrow \sim B \supset \sim A$.

R3. $A \cdot A \leftarrow A$;
$A \vee A \leftrightarrow A$.

R4. $A \cdot (B \cdot C) \leftrightarrow (A \cdot B) \cdot C$;
$A \vee (B \vee C) \leftrightarrow (A \vee B) \vee C$.

R5. $A \vee B \leftrightarrow \sim (\sim A \cdot \sim B)$.

R6. $A \supset B \leftrightarrow \sim A \vee B$.

R7. $A \cdot (B \vee C) \leftrightarrow (A \cdot B) \vee (A \cdot C)$;
$A \vee (B \cdot C) \leftrightarrow (A \vee B) \cdot (A \vee C)$.

R8. $A \cdot B \rightarrow A$; Относится только к основному выражению
$A \cdot B \rightarrow B$.

R9. $A \to A \vee X$. Относится только к основному выражению

R10. $\left.\begin{matrix} A \\ B \end{matrix}\right\} \to A \cdot B$. A и B — два основных выражения.

R11. $\left.\begin{matrix} A \\ A \supset B \end{matrix}\right\} \to B$. A и $A \supset B$ — два основных выражения.

R12. $\left.\begin{matrix} A \supset B \\ B \supset C \end{matrix}\right\} \to A \supset C$. $A \supset B$ и $B \supset C$ — два основных выражения.

Пример, показывающий полный ход решения испытуемым задачи:

(1)	$(R \supset \sim P) \cdot (\sim R \supset Q)$	$\sim(\sim Q \cdot P)$
(2)	$(\sim R \vee \sim P) \cdot (R \vee Q)$	Применение правила R6 к левой и правой частям (1)
(3)	$(\sim R \vee \sim P) \cdot (\sim R \supset Q)$	Применение правила R6 к левой части (1)
(4)	$R \supset \sim P$	„ „ R8 к (1)
(5)	$\sim R \vee \sim P$	„ „ R6 к (4)
(6)	$\sim R \supset Q$	„ „ R8 к (1)
(7)	$R \vee Q$	„ „ R6 к (6)
(8)	$(\sim R \vee \sim P) \cdot (R \vee Q)$	„ „ R10 к (5) и (7)
(9)	$P \supset \sim R$	„ „ R2 к (4)
10)	$\sim Q \supset R$	„ „ R2 к (6)
(11)	$P \supset Q$	„ „ R12 к (6) и (9)
(12)	$\sim P \vee Q$	„ „ R6 к (11)
(13)	$\sim(P \cdot \sim Q)$	„ „ R5 к (12)
(14)	$\sim(\sim Q \cdot P)$	„ „ R1 к (13)

Что и требовалось доказать.

Испытуемый пытается применить эти правила, хотя раньше он решал только одну подобную задачу. Экспериментатор объяснил ему, что задача состоит в том, чтобы получить выражение, написанное справа, из выражения слева, применяя приведенные 12 правил. В любой момент испытуемый может потребовать от экспериментатора применить одно из этих правил к выражению, записанному на доске. Если преобразование правильное, экспериментатор записывает новое выражение в левом столбце, а в правом пишет название правила. Ход решения задачи испытуемым показан выше (см. табл. 1). При решении задачи испытуемого просят рассуждать вслух. Его соображения записываются в «протоколе», т. е. дословной записи всего, что было сказано испытуемым и экспериментатором во время опыта. Начальная часть рассуждений испытуемого приводится в табл. 2.

Таблица 2

Протокол эксперимента для начальной части задачи

«Ну что же, взглянув на левую часть уравнения, нам сначала хочется уничтожить одну из частей, используя правило 8. Однако оно кажется слишком сложным для того, чтобы начинать с него. Тогда ... нет, нет, я не могу этого сделать, так как я тогда уничтожу либо Q, либо P в общем выражении. Я не хочу делать этого сначала. Поэтому я ищу способ избавиться от «подковы» внутри двух скобок в левой и правой частях уравнения. Но я не вижу, как бы это можно было сделать Ага, если применить правило 6 к обеим частям уравнения, то после этого станет видно, можно ли применить правило 7»

Экспериментатор пишет на доске

(2) $(\sim R \vee \sim P) \cdot (R \vee Q)$

«Я почти могу применить правило 7, но у одного R не хватает тильды. Поэтому придется поискать другое правило. Посмотрим, можно ли превратить это R в R с тильдой Собственно говоря, мне следовало бы применить правило 6 лишь к левой части уравнения. Итак, надо применить правило 6, но только к левой части».

Экспериментатор пишет на доске:

(3) $(\sim R \vee \sim P) \cdot (\sim R \supset Q)$

«Теперь я применю правило 7 в том виде, как оно представлено. Оба выражения ... простите, простите, этого нельзя сделать из-за «подковы». Тогда ... дайте посмотреть... я еще раз быстро просматриваю правила, чтобы выяснить, не смогу ли я превратить R в R с тильдой во втором уравнении, но я не вижу, как это можно было бы сделать. (Вздох) Подождите, я немножко растерялся».

Проблема объяснения

Предположим теперь, что протокол, приведенный в табл. 2, представляет собой некие данные о поведении человека, которые должны быть объяснены психологической теорией. Однако к чему это? Что, мы снова возвращаемся к самонаблюдению Вюрцбургской школы? И как нам извлечь информацию из поведения одного испытуемого, когда мы не определили те операционные критерии, которые мы хотим рассмотреть?

С точки зрения бихевиориста данная ситуация не представляет больших трудностей. Устные высказывания субъекта —

в такой же мере акты «поведения», как и движения его рук или кожно-гальваническая реакция. Субъект не занимался самонаблюдением, он просто производил непрерывный поток поведенческих актов в виде словесного отчета при решении данной задачи. Наша цель — найти адекватную модель поведения человека при решении задач, которая объясняет основные особенности этого потока поведенческих актов. Поток поведенческих актов субъекта — это не только его высказывания экспромтом, но и те команды, которые субъект дает экспериментатору и которые показывают, решает ли он данную задачу или нет.

Хотя этот способ рассмотрения поведения и отвечает на поставленные выше вопросы, он сам ставит ряд новых вопросов. Как нужно рассматривать такое изменчивое поведение? Разве речевое поведение не считается одной из самых сложных форм поведения человека? Как можно делать надежные выводы из одного эксперимента, проведенного на одном испытуемом?

Ответы на эти вопросы можно дать, опираясь на недавние выдающиеся успехи в области создания вычислительных машин, программирования и работ по искусственному разуму. Мы уже знаем, что вычислительная машина является универсальным манипулятором над символами, а не только манипулятором над числами. По существу вычислительная машина есть преобразователь систем знаков. При помощи соответствующих приемов, в особенности при помощи логических устройств адресации, этим системам знаков можно сообщить все основные характеристики лингвистических символов. Их можно воспроизводить, и можно формировать из них выражения. Мы узнали о такой абстрактной возможности после исследований Тьюринга, появившихся в середине 30-х годов, но лишь сравнительно недавно вычислительные машины стали достаточно «сильными» для того, чтобы можно было действительно исследовать возможности систем, совершающих сложные преобразования над символами.

Для наших целей наиболее важным направлением в этих исследованиях является попытка построить программы, способные решать задачи, требующие интеллектуальных усилий [365, 549, 734, 789, 792, 950, 1075]. В результате этих исследований развиты идеи и методы, которые привели к новому подходу в построении психологической теории. (Большое количество работ по искусственному разуму, в том числе и работы авторов настоящей статьи, были предприняты отчасти исходя из задач психологии; поэтому интерес психологов к этим работам не случаен.)

Итак, можно мыслить себе «разумную» программу, которая будет манипулировать символами так же, как и наш испытуемый; при подаче на ее вход выражений символической логики она будет на выходе выдавать последовательность примененных

правил, совпадающих с правилами, примененными испытуемым. Если мы будет наблюдать извне за действием такой программы, мы обнаружим, что она рассматривает различные правила и оценивает различные выражения, т. е. делает то же самое, что мы обнаруживаем в протоколе опытов с человеком. Если поведение программы будет достаточно близким к зафиксированному в протоколе поведению испытуемого, такая программа даст нам хорошую теорию процесса решения задач человеком.

В общей форме проблема кажется абсолютно ясной. Программа предписывает в абстрактных терминах (выраженных в некотором языке программирования), каким образом множество символов, хранящихся в памяти, должно подвергаться преобразованию во времени. Это вполне соответствует системе разностных уравнений, которая предписывает характер преобразования множества чисел во времени. Располагая достаточной информацией об индивидууме, можно создать программу, описывающую в системе символов поведение данного индивидуума. Каждый индивидуум при этом описывался бы своей особой программой, а те аспекты поведения при решении задач, которые не специфичны для одного лишь данного индивидуума, позволят выявить общую структуру и содержание программ для многих индивидуумов.

Но возможно ли написать программы, совершающие те же манипуляции, что и человек? Можно ли на основе такого индивидуального протокола, как приведенный в табл. 2, запрограммировать действия субъекта? Насколько хорошо такая программа будет соответствовать экспериментальным данным? Дальнейшая часть статьи посвящена ответам на некоторые из этих вопросов на основе данных приведенного выше эксперимента. Мы рассмотрим лишь поведение GPS при решении первой части задачи и подробно сравним его с поведением субъекта, отраженным в протоколе. Все это прольет свет на вопрос, в какой степени программы можно рассматривать как теории поведения человека при решении задач.

Программа GPS

Мы лишь бегло напомним существо программы GPS, подробно описанной в ранее опубликованных работах [795, 798]. GPS имеет дело с заданной средой, состоящей из *объектов*, которые могут быть преобразованы различными *операторами*; она выявляет *различие* между объектами и преобразует информацию о заданной среде в *цели* (goals). Каждая цель представляет собой совокупность данных, определяющую, в чем состоит достижение цели, позволяющую выбирать подходящую информацию,

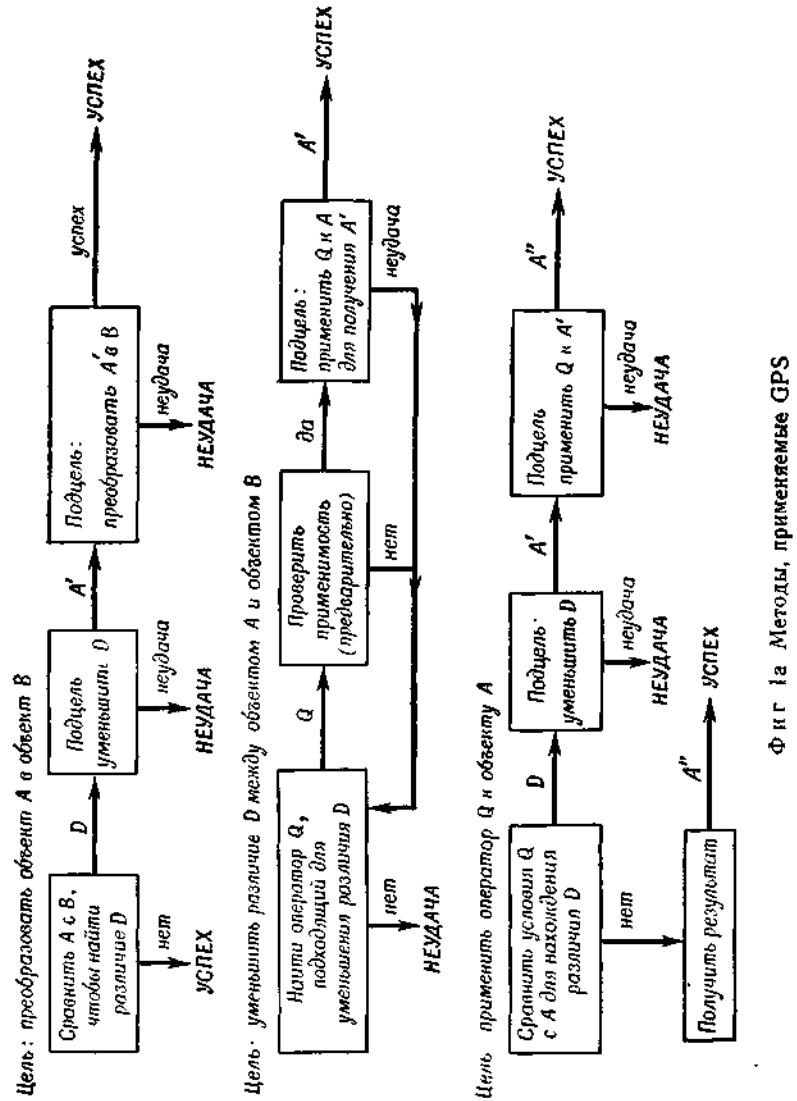
необходимую для достижения цели, и связывающую эту информацию с другими целями. Существует три типа целей:

1) преобразовать объект A в объект B;
2) уменьшить различие D между объектами A и B;
3) применить оператор Q к объекту A.

Для задачи символической логики объекты — это логические выражения, операторы — 12 правил (фактически их частные случаи), а различиями являются выражения вроде «заменить связку» или «добавить букву». Таким образом, объекты и операторы задаются самой задачей, в то время как различия являются тем, что GPS вносит в задачу. (Различия образуют пути связи операторов с тем воздействием, которое их применение оказывает на объекты.)

В целом программа GPS — это путь к достижению цели установлением ряда подцелей, достижение которых ведет к достижению главной цели. Для этого GPS располагает различными схемами, которые называются *методами*. На фиг. 1а представлены три основных метода решения, каждый из которых связан с определенной целью. Например, чтобы преобразовать объект A в объект B, объекты сначала сопоставляют, т. е. приводят в соответствие, и сравнивают элемент за элементом. Если сопоставление обнаруживает различие D между двумя объектами, то задается подцель, состоящая в уменьшении этого различия. Если эта подцель достигается, появляется новый объект A', который (как мы надеемся) уже не обладает больше различием D по сравнению с B. Тогда вырабатывается новая подцель — преобразовать A' в B. Если преобразование удается, то конечная цель достигнута за два шага — от A к A' и от A' к B.

Если цель состоит в уменьшении различия между двумя объектами, то первый шаг состоит в том, чтобы найти оператор, *применимый* к этому различию в том смысле, что оператор действует на объекты именно в отношении этого различия. В операционном смысле «применимость» оператора можно определить, используя метод сопоставления, которым мы уже пользовались, к входной и выходной формам операторов при надлежащем учете переменных. Результаты можно суммировать в таблице связей, как показано на фиг. 1б, где с каждым различием сопоставлены применимые операторы. В этой таблице приводятся также различия, распознаваемые GPS. (Этот набор несколько отличается от набора, приведенного в работе [795]; он соответствует программе, которую мы будет рассматривать в данной статье.) Если найден соответствующий оператор Q, он подвергается предварительному испытанию на применимость, один из вариантов которого дается на фиг. 1б. Если оператор выдерживает это испытание, задается подцель применения оператора



Фиг 1а Методы, применяемые GPS

Проверка применимости (предварительная):

1) Одинакова ли главная связка? (Например, $A \cdot B \rightarrow B$ не подходит для $P \vee Q$.)
2) Не слишком ли велик оператор Q? (Например, $(A \vee B) \cdot (A \vee C) \rightarrow \rightarrow A \vee (B \cdot C)$ не подходит для $(P \cdot Q)$)
3) Не слишком ли легок оператор? (Например, $A \rightarrow A \cdot A$ применимо к любому выражению.)
4) Удовлетворены ли побочные условия? (Например, правило R8 относится только к главным выражениям.)

Таблица связей [1]

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Добавить члены			×				×		×	×	×	×
Вычеркнуть члены			×				×	×			×	×
Изменить связку					×	×	×					
Изменить знак выражения					×							
Изменить знак подвыражения		×			×	×						
Перегруппировать				×			×					
Изменить положение	×	×										

[1] Крестик означает, что данное правило подходит. GPS выберет нужный вариант

Фиг 1б Применение к рассматриваемой задаче из символической логики.

к объекту. Если оператор применяется успешно, получается новый объект A', представляющий собой исходный объект A, видоизмененный в направлении уменьшения различия. (Конечно, могут произойти и другие изменения, которые могут свести к нулю полезность нового объекта.)

Если цель состоит в применении оператора, то сначала нужно установить, удовлетворяются ли условия для его применения. Предварительные испытания, описанные выше, ни в коей мере этого не гарантируют. Если условия удовлетворяются, то выход A гарантирован. Если же условия не удовлетворяются, то обнаруживается некое различие D и вырабатывается подцель уменьшить это различие — точно так же, как и в случае цели преобразования. Подобным же образом, если получается измененный объект A', создается новая подцель: применить оператор к этому новому объекту.

Эти методы образуют рекурсивную систему, которая в процессе продвижения к основной цели образует дерево подцелей. При каждой появляющейся новой трудности создается новая подцель для ее преодоления. GPS располагает рядом тестов, ко-

торые предотвращают разрастание дерева подцелей в бесполезных направлениях. Самым важным среди них является тест, применяемый к новым подцелям для уменьшения различия.

Различия в GPS определенным образом упорядочены; некоторые из них считаются менее значительными, чем другие. Этот порядок дается таблицей связей на фиг. 1б, в которой сначала указаны более трудно устранимые различия. GPS не будет испытывать подцель, если она сложнее, чем другая, более высокая подцель. Она не будет также испытывать цель, если за ней следует другая, более легкая цель. Таким образом, GPS работает по принципу: сначала устранить серьезные различия, а затем в процессе работы искать более легкие различия. Другие испытания, применяемые GPS, касаются внешних ограничений (т. е. ограничения общей «глубины» дерева цели, которую она может допустить), а также теста на тождественность новых объектов или целей с теми, которые уже появлялись ранее.

Поведение GPS при решении данной задачи

Данного выше описания вполне достаточно, чтобы можно было убедиться, хотя бы в общих чертах, в разумности схемы поведения GPS в конкретной задаче. (Мы не описали, как пользоваться двухстрочными правилами R10—R12, так как они не входят в рассматриваемый протокол эксперимента.) В табл. 3 прослежен ход начальной части решения задачи D1. Расположение строк отражает связь между подцелью и целью. Хотя методы не показаны, их можно легко вывести из появляющихся целей. Исходная задача — преобразовать L1 в L0. Сопоставление L1 и L0 показывает, что в L1 есть символы R, а в L0 их нет. Это различие приводит к выработке цели уменьшения различия, которая для краткости обозначается словом «вычеркнуть». Попытка достигнуть эту цель заключается в поиске правил, который приводит к правилу R8. Так как существуют две формы правила R8, причем обе допустимы, GPS выбирает первую из них. (Варианты правил не обозначаются, но их легко можно вывести из схемы.) Поскольку правило R8 применимо, может быть получен новый объект L2. Следуя методу для целей «преобразования», получаем на следующем шаге новую цель: преобразовать L2 в L0. Это в свою очередь приводит к новой цели уменьшения различий: найти Q для L2. Но эта задача отвергается, так как добавить символы труднее, чем убрать их. Поэтому GPS возвращается к цели 2 и ищет другое правило, которое «вычеркнет» символы. На этот раз GPS находит другую форму правила R8 и проделывает те же действия, в результате чего отвергает цель 8.

Таблица 3

Схема действий GPS в начальной части задачи

L0 $\sim(\sim Q \cdot P)$;
L1 $(R \supset \sim P) \cdot (\sim R \supset Q)$.

Цель 1: преобразовать L1 в L0,
цель 2: вычеркнуть R из L1;
цель 3: применить R8 к L1;
получаем L2 $R \supset \sim P$.
Цель 4: преобразовать L2 в L0;
цель 5: добавить Q к L2;
отвергнуть.
Цель 2;
цель 6: применить R8 к L1;
получаем L3 $\sim R \supset Q$
Цель 7: преобразовать L3 в L0;
цель 8: добавить R к L3;
отвергнуть.
Цель 2,
цель 9: применить R7 к L1;
цель 10: изменить связку на $\vee$ в левой части L1;
цель 11: применить R6 к левой части L1;
получаем L4 $(\sim R \vee \sim P) \cdot (\sim R \supset Q)$
Цель 12: применить R7 к L4;
цель 13: изменить связку на $\vee$ в правой части L4;
цель 14: применить R6 к правой части L4,
получаем L5 $(\sim R \vee \sim P) \cdot (R \vee Q)$.
Цель 15: применить P7 к L5;
цель 16: изменить знак в левой и правой частях L5;
цель 17: применить R6 к правой части L5;
получаем L6 $(\sim R \vee \sim P) \cdot (\sim R \supset Q)$.
Цель 18: применить R7 к L6;
цель 19: изменить связку на $\vee$ в правой части L6;
отвергнуть.
Цель 16;
ничего более.
Цель 13,
ничего более.
Цель 10;
ничего более.

Снова возвращаясь к цели 2 в поисках правила для устранения символов, GPS находит правило R7; GPS выбирает вариант $(A \vee B) \cdot (A \vee C) \rightarrow A \vee (B \cdot C)$, так как только этот вариант одновременно сокращает символы и имеет точку в качестве главной связки. Правило R7 нельзя применить сразу; сначала GPS обнаруживает, что в левом подвыражении есть различие в связке, а затем находит аналогичное различие в правом подвыражении. В обоих случаях GPS находит и применяет правило R6 для изменения связки из «подковы» в «клин», получая соответственно L4 и L5. Но новое выражение содержит различие в знаке, что снова приводит к правилу R6, т. е. к тому же правилу, что и раньше, но выполняющему теперь обратную функцию. Правило R6 приводит к L6, совпадающему с L4, хотя GPS не замечает их тождества. Это приводит в цели 19 к вторичному появлению различия в связке, после чего цель отвергается, так как она не дает улучшения по сравнению с целью 13. Дальнейшие попытки найти альтернативные пути изменения знаков или связок ни к чему не приводят. Этим оканчивается данный эпизод в работе программы.

Сравнение схемы поведения GPS с протоколом

Теперь мы располагаем подробной схемой действия GPS. Что может дать протокол испытания субъекта для подтверждения или опровержения нашего утверждения о том, что данная программа — детальная модель символических манипуляций, выполняемых субъектом? Какого сходства можно здесь ожидать?

Выход программы не выражен в естественном языке, который можно было бы поставить во взаимно однозначное соответствие со словами испытуемого. Мы даже не ставили перед GPS такой цели: «решать задачу и одновременно рассуждать», что было бы необходимо, если бы мы стремились к установлению столь детального соответствия в поведении человека и программы. Вместе с тем схема действий GPS, если учитывать, что мы знаем, как она была выработана, дает полный отчет о том, что являлось содержанием задачи, решавшейся GPS, и о порядке, в котором эта задача решалась. Следовательно, можно ожидать, что каждая особенность протокола, относящаяся к задаче, будет существенным образом отражена в схеме действий программы. Обратное неверно, так как многое, связанное с задачей, не было высказано субъектом в явном виде (или даже не было им осознано). Таким образом, соответствие схемы GPS и протокола является хоть и односторонним, но верным.

Начнем с первой фразы субъекта (табл. 2).

«Ну что же, взглянув на левую часть уравнения, нам сначала хочется уничтожить одну из частей, используя правило 8».

Здесь выражено желание уменьшить L1, т. е. уничтожить в нем что-то, а средством для осуществления является

применение правила 8. Это вполне соответствует целям 1, 2, 3 схемы действий GPS.

Перейдем к третьей и четвертой фразам:

«Тогда... нет, нет, я не могу этого сделать, так как я тогда уничтожу либо Q, либо P в общем выражении. Я не хочу делать этого сначала».

Здесь мы имеем прямое выражение скрытого применения правила R8, сравнение полученного выражения с L0 и отказ от этого хода действий по той причине, что мы в этом случае теряем букву, которая нужна в окончательном выражении. Трудно найти слова, которые выразили бы эту мысль еще яснее. И наоборот, если механизм программы (или что-то существенно сходное с ним) не действовал бы, было бы трудно объяснить, почему субъект произнес именно эти слова.

Одно несоответствие, однако, совершенно ясно. Субъект использовал обе формы правила 8 совместно, по крайней мере если судить по его рассуждениям, GPS же рассматривает каждую форму правила 8 отдельным пунктом. Возможно, что субъект действует так бессознательно и просто дает два результата совместно. Однако ясно, что соответствие было бы лучше, если бы GPS рассуждала следующим образом:

цель 2: вычеркнуть R из L1;
цель 3: применить R8 к L1;
получаем: L2 $R \supset \sim P$ или $\sim R \supset Q$.
цель 4 преобразовать L2 в L0;
цель 5: добавить Q к $R \supset \sim P$ или добавить P к $\sim R \supset Q$;
отвергнуть.

Дальнейшие соображения по этому поводу будут приведены ниже. Сейчас же вернемся к пропущенной нами второй фразе испытуемого:

«Однако оно кажется слишком сложным для того, чтобы начинать с него».

В программе нет ничего, что бы в точности соответствовало этому заявлению, хотя легко представить себе несколько возможных объяснений для него. Например, это могло быть просто выражением проведенного сопоставления, т. е. того факта, что L1 является настолько большим выражением, что испытуемый не может охватить его во всех деталях. Пока нет достаточных данных, чтобы определить, какая часть схемы должна соответствовать этому заявлению, поэтому это предложение должно остаться как необъясненный элемент в поведении испытуемого.

Рассмотрим теперь несколько следующих предложений из протокола:

«Поэтому я ищу способ избавиться от „подковы" внутри двух скобок в левой и правой частях уравнения. Но я не вижу, как бы это можно было сделать. Ага, если применить правило 6 к обеим частям уравнения, то после этого станет видно, можно ли применить правило 7».

Это высказывание полностью соответствует целям 9—14 схемы. Из последнего предложения совершенно ясно, что применение правила 7 является основной целью испытуемого, а для этого ему нужно изменить связки. Далее, протокол ясно показывает, что в качестве средства для достижения этого выбирается правило 6. Все три выбранных правила подтверждают, что предварительные испытания на выполнимость были проведены как испытуемым, так и программой методом цели уменьшения различия. Но, если выбор не касался основной связки, почему не было выбрано правило 5 вместо правила 6? И почему не была выбрана форма $(A \cdot B) \vee (A \cdot C) \rightarrow A \cdot (B \vee C)$ правила 7?

Однако здесь обнаруживается несоответствие схемы работы программы и протокола, так как испытуемый использует обе формы правила 6 одновременно (и, очевидно, одновременно применяет его к двум различиям), в то время как GPS рассматривает их последовательно. Это похоже на то, что происходило при использовании правила 8. Так как у нас теперь два примера параллельного сравнения, кажется вероятным, что в этом месте действительно существует разница. И снова было бы лучше, если бы GPS сработала примерно следующим образом:

цель 9: применить R7 к L1;
цель 10: изменить связку на $\vee$ в левой и правой частях L1;
цель 11: применить R6 к левой и правой частям L1;
получаем: L5 $(\sim R \vee \sim P) \cdot (R \vee Q)$.

Общей чертой этих противоречий является то, что образование сложных выражений не усложняет существенным образом методов решения. Так, в случае правила 8 два результата были получены из одной и той же заданной формы и требовали всего лишь одного сравнения. В случае с правилом 7 был проведен однократный поиск правила, примененного затем к обеим частям одновременно так, как если бы выражение рассматривалось как единое целое.

Имеются два аспекта, в которых протокол дает информацию, необъяснимую при помощи программы.

Во-первых, испытуемый применял правило 8 подсознательно (скрыто от наблюдения), но дал при этом команду экспериментатору применить правило 6 на доске. Применяемый здесь вариант GPS не проводит различия между внутренними и внешне выраженными действиями; в этом смысле данный вариант GPS — не вполне адекватная модель поведения субъекта. Эта

необходимость делать различие между скрытыми и внешне выраженными действиями обнаруживается на протяжении всего решения задачи, потому что выражения, записанные на доске, имеют характеристики памяти, значительно отличающиеся от тех, которые фактически возникают в голове.

Во-вторых, данный вариант GPS не отражает процесса поиска достаточно хорошо, чтобы он мог создать эквивалент заявления испытуемого: «Но я не вижу, как бы это можно было сделать. Ага...» Это потребовало бы точного воспроизведения в GPS листа с правилами и умения различать поиск на бумаге с поиском в памяти.

В следующих фразах говорится:

«Я почти могу применить правило 7, но у одного R не хватает тильды. Поэтому придется поискать другое правило. Посмотрим, можно ли превратить это R в R с тильдой».

Опять протокол и схема рассматривают одинаковое различие Они согласуются также в том, что раньше это различие не рассматривалось, хотя и было налицо. Тонкая структура данных опыта также хорошо согласуется со схемой. У R, стоящего в правой части, данное различие (между R и $\sim R$) выявляется скорее, чем у R в левой части, хотя оно возможно в обеих частях. Такое предпочтение, оказываемое правому R, возникает в программе (а возможно, и у испытуемого) вследствие привычки при работе с языковым текстом двигаться слева направо. Но это имеет свои последствия, так как небезразлично, начинает ли испытуемый работать с левой или правой частью выражения; это влияет на определенном этапе на весь ход последующих событий. Подобным же образом, в случае с правилом 8 субъект, по всей видимости, продвигался слева направо и сверху вниз, чтобы получить $Q \vee P$, а не $P \vee Q$. Может показаться, что речь идет о сугубо частных деталях протокола, но эти детали как раз подтверждают следующий вывод: то, что происходит внутри человеческой системы, совершенно подобно символическим преобразованиям, происходящим внутри GPS.

Следующая часть протокола:

«Собственно говоря, мне следовало бы применить правило 6 лишь к левой части уравнения. Итак, надо применить правило 6, но только к левой части».

Здесь есть сильное расхождение со схемой действий GPS, хотя, как ни странно, протокол прерывается на том же самом месте: $(\sim R \vee \sim R) \cdot (\sim R \supset Q)$. И субъект и GPS нашли, что именно правило 6 подходит для изменения знаков. Здесь GPS просто применила правило к текущему выражению, в то время как испытуемый вернулся назад и исправил предшествующее применение. В программе мы не найдем чего-либо, что соответствовало бы этому действию. Наиболее непосредственное объяснение состоит в том, что применение правила 6 в обратном направлении рассматривается субъектом как уничтожение правила 6, примененного ранее. Следуя этим рассуждениям, он выбирает более легкую (и кажущуюся менее глупой) альтернативу — исправить первоначальное действие.

Последний отрывок протокола гласит:

«Теперь я применю правило 7 в том виде, как оно представлено. Оба выражения... простите, простите, этого нельзя сделать из-за „подковы". Тогда... дайте посмотреть... я еще раз быстро просматриваю правила, чтобы выяснить, не смогу ли я превратить R в R с тильдой во втором уравнении, но я не вижу, как это можно было бы сделать. (Вздох.). Подождите, я немножко растерялся».

Здесь поведение программы снова находится в полном соответствии с протоколом. Это одна из немногих самоисправленных ошибок, с которыми мы встречались. Протокол регистрирует безуспешный поиск добавочных операторов для воздействия на различия в знаках и в связках. Окончательное высказывание, выражающее растерянность, можно истолковать как результат нескольких последовательных неудач.

Краткие выводы о соответствии между схемой и протоколом

Рассмотрим снова все соответствия и несоответствия схемы и протокола. Программа дает исчерпывающее объяснение поведения испытуемого при решении задачи, за пятью исключениями различной степени важности.

1. GPS не может отражать поведение испытуемого в двух аспектах: в различии между внутренним и внешним миром субъекта и в правильном представлении о пространстве, в котором идет поиск правила. Эти общие недостатки устранимы. Интересно было бы посмотреть, как GPS может объяснить данные об этих аспектах своего поведения.

2. Испытуемый рассматривает сложные комплексы знаков параллельно путем использования сложных выражений, в то время как GPS рассматривает их последовательно. В данном примере в конечном решении нет большой разницы, но в других условиях она может возникнуть. Теперь уже в значительной степени ясно, как нужно усовершенствовать GPS, чтобы она включала и эти указанные выше свойства.

3. В двух случаях в программе нет соответствия хорошо выраженному целенаправленному поведению человека, отраженному в протоколе. Один из них — заявление о «слишком

сложном» характере правила, — по-видимому, является просто результатом недостатка информации. Программа производит многочисленные сравнения и оценки, которые могли бы вызвать соображения того же порядка. Поэтому эта ошибка не кажется слишком серьезной.

4. Другой случай — фраза, содержащая выражение «мне следовало бы...», — представляется серьезным. Здесь речь идет о механизме (а может быть, и о целом комплексе механизмов), который отсутствует у GPS. Добавление механизма, который обеспечил бы воспроизведение одной только этой фразы, уже позволило бы значительно увеличить общие возможности программы. Например, едва ли можно указать другой путь сделать это, как придать GPS способность постоянно немного оглядываться назад на свои предыдущие действия.

5. Следует принять также общую меру предосторожности. Количество данных невелико, учитывая сложность и размеры программы. Это значит, что существует достаточно степеней свободы, чтобы сделать программу пригодной к работе с различными данными. Более важно то, что мы не знаем хорошего пути для выяснения числа соответствующих степеней свободы у данной программы, а следовательно, не знаем, насколько легко составить альтернативные программы, т. е. программы, работающие на других принципах. Единственное, что известно, — это то, что вполне можно сделать целый ряд мелких изменений, но еще никто не предложил альтернативной теории, которая дала бы доступное сопоставлению детальное объяснение процесса решения задач человеком.

Нам очень помогло бы знание того, насколько эта программа отражает индивидуальные особенности человека. Мы обсуждали здесь только данные одного опыта на одном испытуемом. Мы уже достаточно знаем о решении испытуемыми логических задач, чтобы утверждать, что в основе решения их разными людьми лежат одни и те же механизмы, но мы не можем обсуждать здесь этот вопрос более детально. Добавим лишь, что некоторые последние исследования еще более подкрепляют наши общие представления относительно процессов переработки информации, лежащих в основе человеческого мышления [172, 319, 327, 488, 715].

Заключение

В этой статье мы пытались доказать, что методы, которые использовались для создания детальной программы по решению задач, дают нам также новое сильное орудие для создания теории человеческого мышления. Они позволяют нам соединить

строгость и объективность бихевиористского подхода с богатством информации и сложным поведением, характерными для направления гештальт-психологии. Поэтому главная ценность этих методов не в том, что они дают общую схему для понимания механизмов поведения человека при решении задач (хотя они и это делают), а в том, что они в конечном счете достаточно ясно показывают, что в основе свободного поведения мыслящего человека лежит сложный, но конечный и вполне определенный комплекс правил переработки информации. Хотя это было показано в настоящее время только для небольших фрагментов поведения, полученное объяснение отличается неожиданной глубиной.

РАЗДЕЛ 2

ВЕРБАЛЬНОЕ ОБУЧЕНИЕ И ОБУЧЕНИЕ ПОНЯТИЯМ

Одной из главных проблем в исследовании мышления человека было «обучение», т. е. процесс, посредством которого поведение изменяется с течением времени. Обучение человека исследовалось в экспериментальной ситуации, в которой испытуемому предъявлялась пара объектов, называемых *стимулом* и *реакцией* (или *ответом*). После ознакомления испытуемого с набором таких пар ему предъявляется только стимул из некоторой пары и предлагается ответить соответствующей реакцией. После его ответа ему указывается правильная реакция. При исследованиях процесса обучения было испробовано большое число вариантов этого исходного эксперимента.

В широко распространенном эксперименте по вербальному обучению и стимул и реакция представляют собой бессмысленные трехбуквенные слоги. В таком эксперименте для выяснения процесса обучения исследователи варьировали число заучиваемых бессмысленных слогов, степень сходства между слогами, количество и очередность списков, заучиваемых в каждом отдельном эксперименте, скорость предъявления материала и другие аспекты эксперимента. Экспериментаторов интересовало влияние этих вариаций на время и число повторений, требующихся для заучивания списка, на количество ошибок и на способность запомнить список. В излагаемой ниже модели поведения при вербальном обучении Фейгенбаум развивает модель процессов переработки информации, лежащую в основе различительного и ассоциативного обучения у человека. Эта модель объясняет многие явления, наблюдавшиеся в указанных экспериментах. Основные процессы модели приводят к системе тестов, или

«различителей», которые разграничивают объекты друг с другом. В других процессах накапливаются ассоциативные признаки, соединяющие стимулы с реакциями.

В типичном эксперименте по обучению понятиям испытуемому также предъявляют серию стимулов и серию связанных с ними реакций. Стимулами могут быть геометрические фигуры, бессмысленные слоги, числа и т. д. Обычно имеются только две реакции, например «да» и «нет», и они связаны с определенными характеристиками стимулов. В то время как в обычном эксперименте по вербальному обучению основная задача состоит в распознавании каждого стимула так, чтобы с каждым из них можно было связать соответствующую реакцию, главной задачей эксперимента по обучению понятиям является отыскание общих черт у некоторых классов стимулов. Для проверки усвоения понятия испытуемого просят сформулировать это понятие и (или) дать правильную реакцию на некоторые добавочные стимулы. Так же как и в экспериментах по вербальному обучению, в этом случае возможно большое число вариаций этого основного эксперимента по формированию понятий.

Во второй статье этого раздела Хант и Ховленд излагают информационную модель, согласующуюся со многими явлениями, наблюдаемыми в этих экспериментах.

Проблема обучения представляет интерес не только для психологов, изучающих мышление человека, но и для исследователей в области искусственного интеллекта, стремящихся повысить эффективность программ для вычислительных машин. Читателя, интересующегося этим кругом вопросов, мы отсылаем к статье Минского (стр. 402), к работе Сэмюэля по обучению на программе для игры в шашки (стр. 71) и к сообщению Ньюэлла, Шоу и Саймона о программе «Логик-теоретик» (стр. 113). Фейгенбаум и Саймон [320] обсуждают возможные применения описанной в данном разделе модели «вербального обучения» к проблеме искусственного разума. Хант [504] построил некоторые другие модели формирования понятий, имеющие отношение к искусственному интеллекту.

12. МОДЕЛИРОВАНИЕ ВЕРБАЛЬНОГО ОБУЧЕНИЯ

Э. Фейгенбаум

Введение

Цель настоящей статьи состоит в детальном описании информационной модели элементарных процессов обучения человека символам. По этой модели составлена программа для цифровой вычислительной машины, названная EPAM (Elementary Perceiver and Memorizer — элементарная программа восприятия и запоминания).

Программа EPAM представляет собой точную формулировку информационной теории вербального обучения и противопоставляется другим ранее предложенным теориям вербального обучения[1]. Она представляет собой попытку вполне точного описания некого экономного и правдоподобного механизма, объясняющего заучивание бессмысленных слогов. Критическая оценка EPAM должна в конечном счете базироваться не на том, насколько она интересна как обучающаяся машина, а на ее возможности объяснять и предсказывать явления вербального обучения.

Обсуждение проблемы моделирования вербального обучения мне хотелось бы предварить несколькими краткими замечаниями о том классе информационных моделей, к которому принадлежит EPAM.

а) Это модели умственных (mental) процессов, а не анатомических структур мозга. Это *психологические* модели работы мозга, не включающие ни физиологических, ни нейрологических предположений и не содержащие также попыток объяснить процессы переработки информации в терминах более элементарных нервных процессов.

б) В этих моделях мозг рассматривается как *устройство по переработке информации*, снабженное органами чувств в качестве входных каналов и исполнительными органами в качестве выходных устройств и обладающее внутренними программами проверки, сравнения, анализа, перегруппировки и хранения информации.

в) Центральный механизм обработки информации предполагается последовательным, т. е. способным выполнять одновременно только одну операцию (или очень небольшое их число).

г) Основной единицей в этих моделях служит *информационный символ*, т. е. некий набор двоичных элементов, при помощи которых, согласно предположению, данные о внешнем мире отображаются в мозге.

д) Эти модели существенно *детерминированные*, а не вероятностные. Случайные переменные не играют в них значительной роли.

---

[1] Примерами количественных (или квазиколичественных) теорий вербального обучения служат теории Халла и др. [498], Джибсона [379] и Аткинсона [66]

Исходный эксперимент

Еще на ранних стадиях развития науки психологи изобрели экспериментальную методику для упрощения исследований вербального обучения человека. Так, «простейшим» экспериментом является заучивание наизусть бессмысленных слогов, объединенных в пары или в последовательные списки.

Запоминаемый материал обычно состоит из трехбуквенных слов с гласной посередине и согласными по краям. Бессмысленные слоги выбираются так, чтобы в английском языке они не имели никакого смыслового значения. Например, CAT (кошка) не является бессмысленные слогом, а XUM — является [1].

Один из основных вариантов такого эксперимента по заучиванию состоит в следующем:

а) составляется набор бессмысленных слогов, сгруппированных в пары, скажем всего 12 пар;

б) испытуемый садится перед демонстрационным устройством, с помощью которого ему поочередно предъявляются пары слогов;

в) сначала испытуемый видит левый член пары (*единица стимула*) и пытается назвать второй член (*единицу реакции*);

г) через короткий промежуток времени появляется единица реакции, так что испытуемый видит одновременно обе единицы;

д) через несколько секунд весь цикл повторяется с новой парой слогов, и так продолжается до тех пор, пока не будут предъявлены все пары (составляющие одну *пробу*);

е) пробы повторяются обычно до тех пор, пока испытуемый не научится давать правильные реакции на все стимулы. После каждой пробы устраивается сравнительно короткий перерыв;

ж) перед каждой последующей пробой пары переставляются случайным образом.

Такой способ проведения этого эксперимента называют *парно-ассоциированным предъявлением*.

Другой вариант этого эксперимента называется *предъявлением с предсказанием последовательности*. В этом случае бессмысленные слоги (скажем, в количестве 10 или 12) располагаются в некоторую последовательность, порядок которой не меняется в последовательных пробах. Когда испытуемый видит n-й слог, он должен назвать $(n+1)$-й. Через несколько секунд ему предъявляется $(n+1)$-й слог, и он должен назвать

[1] Никогда нельзя гарантировать того, что с бессмысленным слогом не связано никаких ассоциаций. Подготовлены специальные списки бессмысленных слогов, в которых они размещены с учетом их «ассоциативного значения», что позволяет экспериментатору всегда обеспечивать их «бессмысленность».

$(n+2)$-й и т. д. Эксперимент заканчивается, когда испытуемый научился правильно предугадывать все слоги.

Были созданы многочисленные варианты этого эксперимента [1]. Явления заучивания хорошо исследованы, они устойчивы и воспроизводимы. К типичным особенностям поведения испытуемых, например, относятся следующие явления:

а) чаще наблюдаются отказы от ответа, чем ошибочные ответы;

б) ошибочные ответы, как правило, вызваны тем, что испытуемый путает сходные стимулы или похожие реакции;

в) испытуемый может забыть ассоциацию, которую он с успехом воспроизводил в нескольких пробах подряд. После этого она может снова появиться и снова исчезнуть. Это явление названо *осцилляцией* [2];

г) пусть испытуемый сначала заучил список слогов или пар слогов x, а затем другой аналогичный список y. Тогда при последующей проверке списка x испытуемый показывает более низкие результаты, чем при проверке через такое же время без вклинивания списка y. Такое забывание называют «ретроактивным торможением». Ошибочные ответы при повторном испытании обычно оказываются слогами из списка y. Это явление быстро исчезает. Обычно после одного повторения списков список x снова полностью восстанавливается в памяти;

д) обучение требует тем большего числа проб, чем более сходны слоги-стимулы.

Информационная модель

В этом разделе мы опишем структуру EPAM и присущие ей процессы.

EPAM не является моделью отдельного испытуемого. В этом отношении она не похожа на модели двоичного выбора, описанные Фельдманом [327]. Дело в том, что индивидуальные различия лишь незначительно сказываются на результатах описанного выше исходного эксперимента.

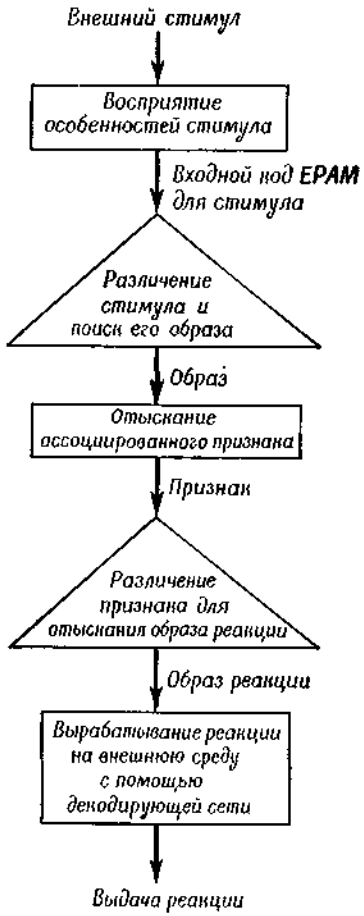
Мы утверждаем, что существуют некоторые элементарные информационные процессы, без которых индивидуум не может обойтись при различении, запоминании и ассоциировании вер-

[1] За подробным обзором этого вопроса отсылаем к статье Ховленда в книге Стивенса [1052].

[2] См. Халл [497]. Фактически Халл употреблял выражение «осцилляция у порога припоминания», что отражает его теоретические взгляды.

бальных единиц, и что эти информационные процессы участвуют во всякой познавательной деятельности любого индивидуума[1].

ЕРАМ не охватывает, разумеется, всего множества таких процессов. Ясно также, что процессы, включенные в последнюю модификацию ЕРАМ, являются существенными и фундаментальными.



Фиг. 1. Исполнительный процесс ЕРАМ получения ассоциации, связанной со стимулом

Обзор: исполнение и обучение

Теоретически ЕРАМ можно подразделить на две подсистемы: систему исполнения и систему обучения. В режиме исполнения ЕРАМ выдает реакции на стимулы. В режиме обучения ЕРАМ обучается различать и ассоциировать стимулы.

Система исполнения более проста. Ее схема дана на фиг. 1. Когда стимул обнаружен, он кодируется в ходе *перцептивного процесса*, вырабатывающего внутреннее его представление (*входной код*). *Дискриминатор* сортирует входные коды в *различительной* сети, представляющей собой дерево оценок и ответвлений, и отыскивает в памяти *образ* стимула. После этого отыскивается и передается в дискриминатор ассоциированный с этим образом *признак* реакции. Дискриминатор сортирует признак в сети и находит образ реакции, т. е. реакцию в той форме, в какой она хранится. Образ реакции буква за буквой декодируется *генератором реакций* в другой различающей сети к форме, удобной для последующего его вывода вовне. Этот выход и является реакцией.

В системе обучения процессы более сложны. Процесс обучения

[1] Некоторые информационные модели по праву считаются моделями психической деятельности индивидуального субъекта, как, например, модель Фельдмана [326]. Другие же, как модель ЕРАМ, относятся к человеку вообще. Возможны и смешанные модели, часть процессов в которых является общей

различению формирует различение; в ходе этого процесса наращивается сеть испытаний и ветвей. Ассоциативный процесс формирует ассоциации между образами посредством запоминания указателя ответа с образом стимула. Эти процессы будут подробно описаны несколько ниже.

Последующие разделы, посвященные информационной модели, содержат подробное описание процессов в обеих системах и их структур.

Ввод в ЕРАМ: внутреннее представление внешних данных

Мы делаем следующие предположения о процессах символического ввода при предъявлении обучаемому бессмысленных слогов. *Воспринимающая система* принимает сырую информацию извне и кодирует ее *внутренними символами*. Последние содержат описательную информацию об особенностях внешних стимулов. Предполагается, что незнакомый трехбуквенный слог кодируется по отдельным буквам, поскольку буквы являются хорошо знакомыми единицами для взрослого испытуемого[1]. Конечным результатом процесса восприятия является внутреннее представление бессмысленных слогов — список внутренних символов (т. е. список списков двоичных элементов), содержащий описательную информацию о буквах бессмысленного слога. Это, по терминологии Минского [734], «характеристика» бессмысленного слога.

В действительности я не программировал этот процесс восприятия. Исходя из целей данной модели, я приписал различным буквам алфавита кодовые обозначения на основе 15 геометрических признаков букв. Испытание и изучение модели на настоящем этапе требуют от водного кода только следующего:

а) чтобы разряды кода достаточно простым образом соответствовали признакам реальных букв;

для всех людей, а часть может быть различной у разных индивидуумов (например, программа GPS Ньюэлла, Шоу и Саймона [795]). С другой стороны, информационные модели можно подразделять по соотношению в них «жестких» (т. е. необходимых и неизменных) и «стратегических» (т. е. являющихся результатом выбора тех или иных стратегий в процессе управления) процессов. Мне кажется очевидным, что более гибкие программы служат скорее моделями индивидуумов, так как содержат возможность внутреннего выбора «стратегии». Примером «жесткой» модели служит читающая машина Линдсея [610], а гибкой — модель формирования понятий с помощью консервативной стратегии сосредоточения (Уикелгрин [1148], Брюнер, Гуднау и Остин [172]); первая отображает человека вообще, а вторая предсказывает поведение лишь отдельной личности.

[1] Я исхожу из того же основного механизма восприятия, что и Селфридж [973] и Диннин [284], программа которых сканирует очертания букв и воспринимает их простые топологические особенности.

б) чтобы коды букв имели большую избыточность, т. е. включали значительно больше разрядов, чем это необходимо для различения букв алфавита.

Коротко говоря, внутреннее представление бессмысленного слога — это список списков двоичных единиц, причем каждый подсписок кодирует с большой избыточностью одну из букв слога.

Когда дана последовательность таких входных данных, сущность проблемы обучения состоит в следующем: 1) *различение* каждого кода с другими, ранее заученными, позволяющее давать на них разные ответы, и 2) *ассоциирование* информации об «ответном» слоге каждой пары с информацией о «стимулирующем» слоге, с тем чтобы за подачей стимула следовала правильная реакция.

Различение и запоминание: растущее дерево образов

Займемся сначала структурой, отложив на время обсуждение происходящих в ней процессов.

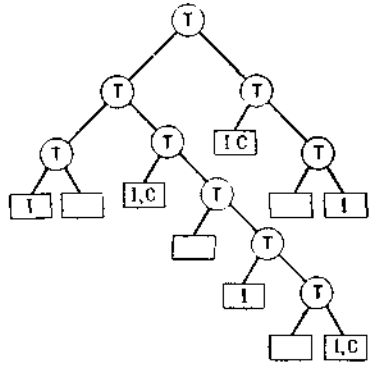
Различительная сеть. Это первая информационная структура в ЕРАМ. В любой момент обучения в ней отражена вся ранее полученная различительная информация. В ней нет ничего нового — это обычное сортирующее дерево, или декодирующая сеть. Небольшая сеть такого типа изображена на фиг. 2. На концах ветвей дерева расположены так называемые *списки образа* (image lists), в которых может запасаться символическая информация. В узлах хранятся программы, называемые *тестами*. Они проверяют характеристики введенного кода и передают сигнал в левую или правую ветвь. В каждом списке образа имеется список символов, называемый *образом*. Каждый образ является копией некоторого кода на входе или его части. Этими терминами я буду пользоваться в дальнейшем при описании процессов в сети.

Интерпретатор сети. Различительная сеть проверяется и преобразуется в ряде процессов, из которых наиболее важным является интерпретация сети, помещающая введенный код в соответствующее место в сети и составляющая ассоциированный с ним список образа. *Такой процесс обработки представляет существо чисто ассоциативной памяти; содержащаяся в стимуле информация сама ведет к отысканию ассоциированной с ним информации.* Интерпретация сети — очень простой процесс. В нем отыскивается тест в самом верхнем узле дерева, и выполняется его программа. Появляющийся сигнал сообщает, переходить ли к следующему тесту по левой или по правой ветви. Так повторяется до тех пор, пока не будет достигнут конец одной

из ветвей дерева. Тогда выдается название списка образа и процесс заканчивается. Такую сортировку элементарных кодов в статичной сети проводит различитель (дискриминатор) исполнительной системы.

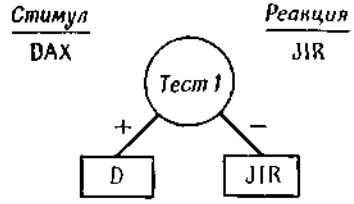
Обучение различению. Это процесс, ведущий к наращиванию сети. Вначале обучающаяся система получает только набор простых процессов для наращивания сети и запасания новых образов на концах ее ветвей, а не готовую сеть.

Чтобы разобраться в процессах различения и запоминания, рассмотрим конкретный пример запоминания бессмысленных слогов. Допустим, что первая пара



Фиг 2 Типичная различительная сеть ЕРАМ

T — различительный тест в узле, *I* — образ в конце ветви, *I, C* — образ и признак в конце ветви, прямоугольник без символа — пустой конец ветви.



Фиг. 3. Различительная сеть после заучивания двух первых слогов

Признаки не показаны Соблюдается условие не добавлять избыточных тестов Тест 1 работает с первой буквой

слогов (стимул — реакция) из списка уже заучена (не будем пока уточнять, как формировались звенья ассоциации). Пусть первой парой было DAX—JIR. В этой точке различительная сеть состоит из двух ветвей, как показано на фиг. 3. Поскольку слоги различаются уже по первой букве, первый тест будет, вероятно, проверять какой-то признак, по которому различаются D и J. Никаких других тестов пока не требуется.

Заметим, что образ JIR запоминается полностью. Ответный образ должен также запоминаться полностью, чтобы дать необходимую информацию для *выработки* реакции; в то же время для того, чтобы дать информацию, требуемую для *распознавания* стимула, достаточно запоминания лишь части образа стимула. Обучающаяся система сама по мере наращивания информационной сети определяет, сколько требуется информации об образе стимула, на основе ошибок, которые указывают на неправильное различение.

Возвращаясь к нашему простому примеру, предположим, что, следующей парой слогов будет PIB—JUK. В данном состоянии для двух новых слогов в сети нет конечных накопителей, так как различительная способность сети может соответствовать только двум объектам. Входной код для PIB подвергается сетевой интерпретации. Допустим, что первый тест направляет его в отмеченную знаком плюс ветвь на фиг. 3. Поскольку имеется различие между уже хранящимся образом (с первой буквой D) и новым кодом (с первой буквой P), попытка запасти образ PIB в этой конечной ветви нарушила бы запасенную там ранее информацию.

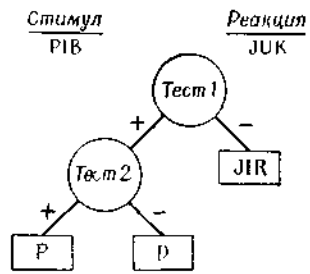
Ясно, что требуется развить еще далее различительную способность. Для этого выявляется различие запасенного ранее образа со вновь поступающим кодом. После отыскания различия для него создается новый тест, который помещается в ту точку сети, где различение не удалось. Далее создается образ нового объекта и оба образа — старый и новый — запасаются на концах ветвей нового теста, что и разрешает конфликт[1]. Тогда сеть приобретает вид, изображенный на фиг. 4, где показан второй тест, различающий по каким-то признакам буквы P и D.

Теперь проводится сетевая интерпретация вводного кода для JUK. Поскольку первый тест не может (при описанных выше допущениях) обнаружить различий между входными кодами JUK и JIR, код JUK направляется в то же окончание, что и JIK. Затем производится поиск различий, которые, конечно, можно обнаружить только во вторых или в третьих буквах.

Очередность пометки. Которая буква должна быть избрана следующей для поиска различий? В последовательной машине типа EPAM этот поиск должен выполняться в некоторой очередности, которая не обязательно должна быть фиксированной и произвольно заданной. Она может быть переменной и адаптивной.

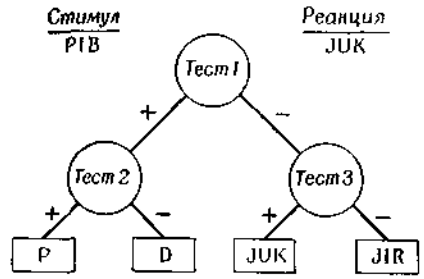
[1] При описанном процессе сеть наращивается при добавлении к памяти каждой новой единицы. С точки зрения переработки информации больше всего времени в системе тратится на процессы сличения и наращивания сети. В общем случае ценой небольших дополнительных усилий можно устанавливать больше одного различия и добавлять к сети больше одного различительного теста. Помещение в сеть каждого избыточного теста порождает «пустой» список образа. Если впоследствии некоторый слог попадет в этот пустой список образа, его образ можно будет запасти без наращивания сети. Между малыми сетями, которые приходится наращивать каждый раз, и большими, которые обеспечены избыточными тестами и обширным резервом пустых списков образа, существует некая золотая середина. С этим «структурным параметром» проводились эксперименты, в которых обнаружилось, что в данном случае золотой серединой является добавление при наращивании сети одного или двух избыточных тестов. Я не берусь, однако, настаивать на общности этого частного результата.

Для этого в EPAM имеется *очередность пометки букв слога*, предписывающая в любой момент последовательность сличения букв. Поскольку наблюдалось, что испытуемый обычно смотрит на крайние буквы раньше, чем на средние, очередность пометки первоначально задавалась так: первая буква, третья буква, вторая буква. Если при сличении в какой-нибудь букве обнаруживается различие, номер этой буквы продвигается вперед в списке очередности. Следовательно, раньше будут просматриваться буквы, более богатые различиями. В нашем примере, поскольку



Фиг. 4. Различительная сеть фиг. 3 после дополнительного заучивания слога PIB.

Тест 2 работает с первой буквой.



Фиг. 5. Различительная сеть фиг. 4 после дополнительного заучивания слога JUK.

Тест 3 работает с третьей буквой.

между образом JIR и кодом JUK в первой букве не будет обнаружено различий, просматриваются третьи буквы, в которых различие обнаруживается (R и K). Для этого различия строится тест, и сеть снова наращивается. Итог показан на фиг. 5. Порядок пометки при этом изменяется: третья буква продвигается на первое место.

Мы не будем развертывать дальше наш пример, поскольку все существенные особенности наращивания сети уже показаны.

Ассоциирование образов: обработка с помощью признаков

Различительная сеть и ее интерпретация связывают коды внешних объектов с внутренними образами и списками образов. Но основной эксперимент по заучиванию требует, чтобы информация, содержащаяся в стимуле, каким-то путем приводила к информации реакции и вызывала ответ. Для ассоциирования внутренних образов друг с другом (т. е. реакции со стимулом) при весьма незначительном усложнении исходного механизма можно применить то же понятие различительной сети.

Ассоциирование образа стимула с образом реакции производится с помощью запоминания вместе с образом стимула части закодированной информации о реакции. Эту информацию называют *признаком* (cue). Признак имеет ту же форму, что и входной код, хотя и содержит обычно значительно меньше информации. Признак ассоциированного образа может помещаться интерпретатором сети в различительную сеть для восстановления этого образа. Если мы, например, в сети на фиг. 3 записываем вместе с образом стимула букву J как признак реакции JIR, то сортировка этого признака позволит правильно восстановить образ реакции. *Внутренние ассоциации ЕРАМ строятся посредством засылки вместе с образом стимула информации, достаточной для отыскания в сети образа реакции в момент ассоциирования.*

Количество необходимой информации определяется в процессе ассоциирования методом проб и ошибок. На основании зафиксированной очередности букв первая по очереди буква добавляется к признаку. Затем признак сортируется интерпретатором сети и образуется образ реакции. При этом может оказаться, что воспроизведен не тот образ, так как, если в признаке отсутствует необходимая для теста информация, интерпретатор от данного теста случайным образом (с равными вероятностями) выбирает правую или левую ветвь[1]. В процессе ассоциации неправильно выбранная реакция сразу обнаруживается непосредственным сличением с входным кодом реакции. После этого к признаку добавляется следующая по очереди буква, и процесс повторяется до тех пор, пока не будет получен правильный ответ; тогда ассоциирование считается завершенным.

Отметим два важных возможных случая. Во-первых, в описанном выше процессе признак, не гарантирующий нахождение адекватной реакции, может по случайному стечению обстоятельств привести к выбору правильного образа реакции. Такая «счастливая удача» впоследствии обычно приводит к ошибкам в реакциях.

Во-вторых, предположим, что построенный признак оказался достаточным для выбора нужного образа реакции при *данном* ассоциировании, т. е. *в данный момент времени*, когда ассоциируются два признака. Но при дальнейшем наращивании сети для накопления в памяти новых образов признак, ранее обеспечивавший правильную выборку, может стать неудовлетворительным и не обеспечит воспроизведения нужного образа. В ЕРАМ звенья ассоциаций «датированы» и всегда могут быть нарушены при дальнейшем обучении. Реакции могут оказаться «невыученными» или временно «забытыми» не из-за уничтожения информации в памяти, а вследствие временной потери доступа к информации за счет наращивания сети. Если такого рода неудачу в ассоциации можно обнаружить с помощью обратной связи от естественной или экспериментальной ситуации, то помеха легко устраняется добавлением информации в признак. В противном случае реакция может расстроиться на более или менее длительное время. Значение этого явления, пожалуй, станет яснее после обсуждения результатов моделирования ЕРАМ.

[1] Это единственный случай использования случайных переменных в ЕРАМ. Мы хотели бы вовсе обойтись без них, но не нашли удовлетворительного и физиологически правдоподобного адаптивного механизма для принятия решения. Все же случайный механизм дает лучшие результаты, чем также использовавшийся нами механизм фиксированного выбора ветви.

Внешний и внутренний уровни реакции

Следует различать понятия, связанные с выбором внутреннего образа реакции в ЕРАМ и преобразованием этого образа в реакцию, выдаваемую во внешнюю среду.

Поиск реакции. Предъявляется слог-стимул. Его входной код сортируется в различительной сети для отыскания того списка образа, в котором находится нужный признак. Признак сортируется в сети для отыскания другого списка образа, который содержит предлагаемый образ реакции. Реакция не возникает, если нет признака или при сортировке отбирается пустой список образа.

Образование реакции. Этой цели служит фиксированная различительная (декодирующая) сеть, по предположению уже обученная, которая преобразует коды букв внутренних образов к форме, принятой для выдачи вовне. Образ реакции декодируется буква за буквой интерпретатором сети в декодирующей сети для букв.

Организация задачи обучения

Заучивание бессмысленных слогов путем описанного выше процесса требует времени. ЕРАМ — последовательная система, поэтому отдельные слоги могут обрабатываться в ней в определенной последовательности, которая, однако, не задается произвольно, а является результатом исполнительного процесса более высокого ранга, служащего для того, чтобы должным образом «направлять внимание» ЕРАМ. Эти так называемые *макропроцессы* мы здесь не будем описывать и обсуждать; полное их изложение можно найти в работе Фейгенбаума и Саймона [322].

Конкретизация модели: программа ЕРАМ для вычислительной машины

Модель ЕРАМ была реализована в виде программы на информационном языке IPL-V (Ньюэлл и др. [805]) [1] и испытана на машине IBM-7090 в Беркли и в лаборатории корпорации «Рэнд». В более ранней работе автора [318] приведены описание машинной реализации программы, а также полная программа в указанном языке и структура данных для ЕРАМ (по состоянию на октябрь 1959 г.).

Информационный язык IPL-V, представляющий собой списковый язык обработки информации, оказался весьма удобным для модели ЕРАМ по следующим основным причинам:

а) Основные процессы в IPL-V производятся непосредственно и явным образом со *структурами типа списков*. Различные информационные структуры в ЕРАМ (например, различительная сеть, список образа) особенно легко обрабатывать как списки. Действительно, различение по самому своему определению есть списковая структура простого типа.

б) В ряде случаев полезно, а в некоторых и необходимо вместе с символами запасать описательную информацию об этих символах. В IPL-V предусмотрены специальные структуры и процессы для подобных задач.

в) В IPL-V легко записать иерархию подпрограмм, и это позволяет без затруднений программировать те сложные управляющие последовательности, которые используются в ЕРАМ.

Экспериментальные исследования с помощью ЕРАМ

Процедура исследования поведения ЕРАМ достаточно проста. Мы составили «экспериментальную» программу и задали исходную информацию, соответствующую конкретному эксперименту. Затем эта «экспериментальная» подпрограмма провела ЕРАМ через все шаги данного эксперимента. Полная запись предъявленных стимулов и полученных реакций, а также сеть в ее окончательном виде, выдавались на печать. При желании могла выдаваться и любая другая информация о процессах в ЕРАМ и о состоянии ее памяти.

Было проведено большое число экспериментов по парному ассоциированию и предсказанию последовательности слогов. Проводилось также моделирование других классических экспериментов по заучиванию бессмысленных слогов. Полные результаты этих экспериментов по моделированию и сравнение поведения ЕРАМ с поведением человека послужат предметом следующей статьи. Здесь же мы приведем лишь несколько кратких примеров, показывающих, каких результатов мы ожидали и какие получили.

[1] См также стр 458 настоящей книги — *Прим ред.*

А. Обобщение стимула и реакции

Данный пример будет описан в психологических терминах. Если X и X' — сходные стимулы, а Y — правильная реакция на стимул X, то появление реакции Y в ответ на предъявление X' называется *обобщением (генерализацией)* стимула. Аналогично, если Y и Y' — сходные реакции, появление Y' после предъявления X называется обобщением реакции. Обобщение — явление, свойственное поведению всех испытуемых, и оно было обнаружено также у ЕРАМ как следствие процесса реагирования и структуры различительной сети. Дело в том, что для памяти ЕРАМ сходны те «стимулы», входные коды которых передаются на одну и ту же конечную ветвь, и сходны те «реакции», которые запасаются в близких частях сети (по этой причине и появляются ошибочные реакции, когда информация признака недостаточна).

Б. Осцилляции и ретроактивное торможение

Об этих явлениях мы уже говорили выше. Осцилляции и ретроактивное торможение проявляются в поведении ЕРАМ как следствие простых механизмов различения, различающего обучения и ассоциации. Они ни в коем случае не создавались специально. Появление довольно сложных явлений, наподобие этих, укрепляет уверенность в том, что исходные идеи модели были правильны.

Эти явления описываются здесь совместно, поскольку их происхождение одинаково. По мере заучивания слогов различительная сеть наращивает новые разветвления. Такое наращивание означает присоединение новых тестов, а это в свою очередь означает, что во всех сортируемых объектах будет просматриваться больше информации. Важным классом сортируемых объектов является набор признаков. Содержащаяся в признаке информация, достаточная в некоторый момент времени для уверенной ассоциации, может в некоторый последующий момент оказаться недостаточной. Как указано выше, это может повести к ошибочной реакции. Причиной ошибок является исключительно процесс выучивания новых слогов. В случае осцилляции новыми слогами служат те, которые вместе попадают в один выученный список. В случае ретроактивного торможения новыми слогами

служат слоги другого списка той же различительной сети. В обоих случаях причины ошибочной реакции одинаковы. Поэтому перед нами родственные явления (гипотеза, которая не рассматривалась сколько-нибудь обстоятельно психологами).

В модели ЕРАМ термин *интерференция* уже не просто описательный — у него есть точное, операциональное значение. Процесс интерференции последующего обучения с предыдущим приобретает вполне конкретные черты.

В. Забывание

Обычное объяснение забывания тем или иным путем сводится к простой и привлекательной идее о том, что накопленная в мозге информация физически разрушается со временем (например, исчезновение «следов памяти» или стирание в машине старой информации при записи новой). Однако такое объяснение не вяжется с тем повседневным наблюдением, что каждый из нас может при определенных условиях вспомнить подробную и, по всей видимости, маловажную информацию по истечении длительного периода времени. Другое, не столь наглядное объяснение связывает забывание не с разрушением информации, а с тем, что заученный материал теряется и становится трудно доступным в большой разрастающейся ассоциативной сети.

ЕРАМ, как легко видеть, забывает и хорошо заученные реакции. Это забывание является прямым следствием последующего обучения. Забывание это временное: потерянную информацию можно реконструировать путем запоминания дополнительной информации признаков. ЕРАМ оказывается механизмом, который объясняет явление забывания при отсутствии утраты информации. Насколько нам известно, это первая конкретная демонстрация такого типа забывания в вычислительной машине.

Заключение. Некоторые перспективы

Проверка выдвинутой теории информационных процессов была проведена путем моделирования экспериментов различного типа и подробного сопоставления количественных и качественных аспектов реального поведения с поведением модели. Саймон и автор данной статьи уже провели ряд модельных экспериментов. По мере дальнейшего исследования вербального заучивания потребуется проведение новых экспериментов.

Мы экспериментировали с различными «органами чувств» на ЕРАМ, соответствующими «зрительному» входу и «письменному» выходу, «слуховому» входу и «устному» выходу, «мускульным» входу и выходу. Каждому из этих случаев соответствует своя схема кодирования на входе и своя различительная сеть. Теперь возможны ассоциации как внутри сетей, так и между сетями и преобразования одних способов представления в другие. Так, у ЕРАМ, как и у любого из нас, есть нечто вроде «внутреннего слуха» и «внутреннего зрения». Мы проводили ассоциативное обучение ЕРАМ чтению наподобие того, как начинают обучение чтению маленьких детей, однако находимся еще лишь в самом начале такого рода исследований.

Однако в одном вопросе модель ЕРАМ, как и все существующие теории заучивания наизусть, оказалась неэффективной. Речь идет о выяснении того, между чем образуется ассоциация: между символами или между знаками этих символов. ЕРАМ не может, например, заучить последовательность, в которой один и тот же слог встречается дважды, не может различить первое и второе вхождение этого слога. Для решения этой задачи мы сконструировали (и сейчас проверяем) процессы построения, запоминания и реагирования для цепочек ассоциаций знаков [322].

13. МАШИННАЯ МОДЕЛЬ ФОРМИРОВАНИЯ ЧЕЛОВЕЧЕСКИХ ПОНЯТИЙ

Э. Хант и К. Ховленд

Введение

Что такое понятие? Обычное использование данного слова не всегда четко. Выражения вида «понятие силы», «понятие возмездия» и «понятие собаки» являются достаточно расплывчатыми. Чёрч [222] предложил определение, которое было фактически принято психологами, работающими в области экспериментов по «обучению понятиям». Идея Чёрча заключается в том, что элементам некоторого множества объектов может быть присвоен любой символ (или *наименование*). Для любого произвольного объекта существует правило, касающееся описания этого объекта, с помощью которого можно решить, принадлежит ли данный объект к тому множеству объектов, для которого используется данное наименование. Правило решения в этом случае и есть «понятие» наименования, а множество объектов образует содержание этого наименования.

В обычном эксперименте обучения понятиям испытуемому предъявляют серию стимулов (объектов или примеров), которой дается определенное название, и другую серию стимулов,

имеющую другое наименование, или серию стимулов, состоящую из других наименований. Так, например, первое множество можно назвать «собаки», второе множество — «не собаки» или «кошки, волки, овцы и т. д.». Таким образом, необходимы определенные программы для классификации примеров в соответствии с названиями, задаваемыми экспериментатором. Это наши упорядочивающие программы. Иногда различные стимулы, которым дано одно и тоже наименование, имеют некоторые общие характеристики, например во всех «положительных» (позитивных) примерах могут содержаться три треугольника. В иных случаях не существует ни одного общего связующего элемента, но зато существуют общие взаимоотношения, например во всех «положительных» примерах верхняя и нижняя фигуры могут быть одного размера, хотя сами фигуры могут быть большими, средними или малыми в каждом ряду. Составление машинной программы требует описания связи между основными элементами стимулов, поэтому при моделировании нам необходимо иметь программу описания.

Наконец, различные типы наборов стимулов могут быть организованы различным образом с точки зрения различных типов логической связи. Иногда понятие характеризуется совместным присутствием двух или большего числа характеристик. Такие понятия относятся к конъюнктивным понятиям (например, «большая красная фигура»). Другие понятия включают различные подмножества характеристик, это так называемые дизъюнктивные понятия (например, «красные *или* большие фигуры»). Набор программ решения дает различные варианты определения формы ответа.

Программа должна уметь моделировать различные условия, при которых проводятся эксперименты. В качестве примеров некоторых из таких различий в условиях, которые должны быть промоделированы, можно упомянуть следующие.

В процессе изучения могут меняться количество и сложность стимулов, а также скорость предъявления новых стимулов. Примеры могут оставаться в поле зрения испытуемого, или же он должен опираться на свою память. Различные задачи по обучению понятиям можно сравнивать между собой по ряду параметров, например по таким: логическая сложность правильного ответа, соотношение числа существенных и несущественных параметров, порядок предъявления задач различных типов и предъявление испытуемому информации с помощью «положительных» или «отрицательных» примеров.

В течение эксперимента испытуемый может давать значительное количество различных по своему характеру ответов. Испытуемые могут устно описывать свои промежуточные и окончательные гипотезы относительно характеристик понятия. Эти ответы могут явиться для нас ключом к пониманию принципов переработки информации данным человеком. С этой точки зрения они служат дополнительными оценками в нашей работе по созданию моделей. Время, которое испытуемый затрачивает на выдачу ответа при различных меняющихся условиях эксперимента, также является полезной оценкой ответа. Можно также анализировать ошибки, которые испытуемый делает при последовательном определении названий показываемых ему стимулов. При проведении эксперимента следует отдавать предпочтение более объективной записи результатов, и нашей основной целью является предсказание этих результатов с помощью моделирования на вычислительной машине.

Чтобы дать теоретическое объяснение обучению понятиям, мы воспользуемся аналогией «черного ящика». Мы попытались составить программу для вычислительной машины, которая при наличии закодированных сигналов в качестве исходных данных будет выдавать закодированные ответы, которые можно использовать для предсказания ответов испытуемого. Нашей целью являлось точное предсказание ответов, а не создание устройства для формирования гипотез или же воспроизведения предварительно полученных протоколов. Мы здесь не затрагиваем вопросов, связанных с самой задачей распознавания геометрических образов. Геометрические образы используются нами в качестве стимулов лишь из-за их удобства, а также вследствие того, что они представляют собой стимулы, которые могут быть описаны с помощью ранее распознанных стимулов. Мы надеемся, что сможем сделать некоторые выводы относительно процессов образования понятий вне зависимости от частной формы стимулов.

Сообщения о результатах наших экспериментов частично уже опубликованы. В настоящей статье мы детальнее коснемся вопросов программирования нашей машинной модели образования понятий. Модель была полностью завершена и отлажена и применялась в ряде экспериментов. После описания модели мы приведем некоторые результаты, полученные при моделировании, и обсудим возможные модификации нашей модели.

Программа образования понятий написана на списковом языке обработки информации для вычислительных машин IBM-709—7090. Первые программы были написаны на информационном языке IPL-V — интерпретирующем языке обработки списков, созданном Ньюэллом, Шоу и Саймоном [805]. По некоторым техническим соображениям мы в настоящее время переписываем программу на языке LISP — языке обработки списков, созданном Маккарти [674]. У нас не было достаточного опыта

работы с системой LISP, чтобы можно было сравнить эти два языка в применении к нашему типу задач. Поскольку основные принципы LISP и IPL-V тождественны, мы в дальнейшем не будем делать различия между этими двумя языками.

Описание программы

Программа состоит из двух блоков исходных данных, определяемых программистом перед началом работы программы, и пяти подсистем для обработки данных. Перед началом моделирования программист определяет последовательность задач, набор параметров и набор списков. Последние два представляют возможности модели испытуемого при решении задачи. Исходные данные задачи остаются неизменными на протяжении всего опыта, характеристики же модели испытуемого могут изменяться самой программой. Задачи фиксируются путем описания примеров (стимулов), указания названий (классов) и условий предъявления стимулов. Это определяет требования к памяти, числу предъявляемых стимулов за определенный интервал времени и т. д. Все условия, используемые для описания задачи, записываются в списке свойств, состоящем из символов, характеризующих данную задачу.

Каждый пример (т. е. объект, который должен быть классифицирован) представляется с помощью символа, чей список свойств определяет принадлежность данного символа к определенному классу, и списка пар признак — оценка, дающих формальное описание объекта. Так, в нашем предыдущем примере «большой красный треугольник» будет содержать в своем описательном списке следующие пары: название класса — «альфа», размер — большой, цвет — красный, форма — треугольный. Список формального описания дает самую подробную информацию об объекте, которая только доступна программе. Описание более высокого уровня, основанное на отношениях между элементами формального списка, могут быть сконструированы самой программой.

Признаки (или «размерности») представляют собой как бы «степени свободы» изменения свойств объектов. Мы использовали «анализ размерностей» объектов, который определяет конечное множество со встроенной структурой для описания объектов [487]. Признаки организованы также в «наборы», или группы. Эти группы являются подмножествами множества всех признаков, рассматриваемых совместно во время их распознавания и формирования ответов.

Спецификация модели «испытуемого» содержит две обширные категории величин: численные параметры и начальные

условия, используемые для управления программой. О них мы будем говорить, когда речь пойдет об их работе в модели.

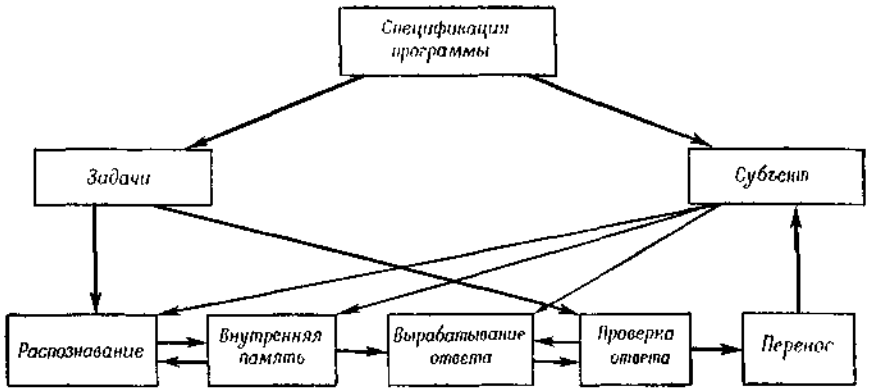
На фиг. 1 показаны каналы связи между различными подсистемами модели. Существуют две основные группы подсистем. Первая, как показано на фиг. 1, есть система распознавания и памяти. Ее задача заключается в выборе информации из формального описания представленных примеров и сохранения этой информации для дальнейшего формирования ответа и работы группы контроля.

После проверки списка признаков данной задачи программа определяет условия предъявления стимулов. Если эти условия *не требовали бы запоминания показанных стимулов человеком* (т. е. если примеры показываются испытуемому и затем остаются в его поле зрения), то при предъявлении стимула к внутренней памяти добавляется название каждого показанного примера вместе с его полным формальным описанием. Мы не утверждаем, что испытуемые в состоянии видеть все, что есть в сложном примере в течение времени его предъявления. Однако, если условия предъявления таковы, что испытуемый может заново просмотреть пример, мы не видим оснований для применения специальной программы распознавания.

Если же пример показывается лишь однажды, а потом устраняется, то испытуемому остается лишь запомнить информацию, которую он получил во время предъявления примера. В этом случае необходима специальная программа «распознавания». В нашей модели имеется довольно примитивный метод считывания примеров в памяти. В начальную спецификацию модели испытуемого включен список, состоящий из наборов признаков, которые считываются в порядке, в котором они расположены в списке. При каждом предъявлении примеров из определенной задачи наша программа читает все наборы признаков, которые когда-либо считывались. Если при этом программа получает достаточное количество информации для отличения рассматриваемых примеров от предыдущих, то процесс чтения оканчивается. Если же информации не хватает, то читается новый набор признаков и заново повторяется тест различения. Новые наборы признаков дополняются до тех пор, пока не будут использованы все признаки или пока не появится возможность отличить данные примеры от предыдущих. Когда программа чтения оканчивается, во внутренней памяти начинают работать соответствующие описания (некоторая часть формального описания).

Для задач, в которых существуют требования к памяти, используется модель человеческой памяти с ограниченным объемом [502]. Параметры испытуемого определяют некоторое

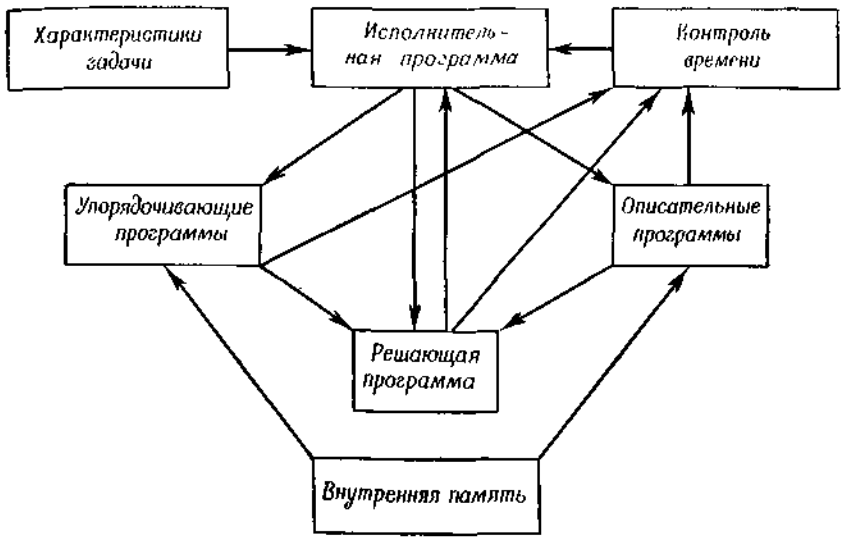
количество ячеек в памяти. Они располагаются отдельно для создания памяти предъявления примеров. Каждый новый пример записывается случайным образом в одну из ячеек. Предыдущая запись, если она имелась, стирается. Таким образом, вероятность того, что модель испытуемого в состоянии обработать пример, уменьшается с увеличением числа показанных примеров. Мы считаем, что такая модель является лишь грубым приближением к человеческой памяти, хотя она, как было показано, в некоторых случаях позволяет предсказывать вероятность использования определенной информации.



Фиг. 1 Схема организации программы.

На фиг. 1 видно, что связь между элементами распознающей памяти и элементами, контролирующими формирование ответа, в данной системе носит весьма косвенный характер. Возможно, что такая система не является самой эффективной. Схему объединения подсистем мы рассмотрим на основе другой модели.

«Сердцем» нашей модели является подсистема вырабатывания ответа. Схема ее внутренней работы показана на фиг. 2. Блок выработки ответа находит правила двоичных решений для различения между обозначением названия и его дополнением. При этом в каждый момент он рассматривает лишь один набор признаков. Наборы признаков выбираются в порядке, определяемом текущим описанием модели испытуемого. Если уже существует ответ, который включает определенный набор признаков, то такой набор будет исключен из рассмотрения при формировании ответа. «Исполнительная программа» системы вырабатывания ответа начинает работать тогда, когда находится набор признаков, для которого не пригоден ни один из текущих ответов. Затем программа подготавливает *исполнительный список*, содержащий названия трех подпрограмм, которые будут выполняться в порядке, определяемом исполнительным списком. Содержание внутренней памяти используется в качестве выхода для первой и второй из этих трех программ. Они в свою очередь дают выход для третьей (последней) подпрограммы исполнительного списка. Успешное завершение работы третьей программы приводит к «пробному» определению понятия.



Фиг. 2 Процесс выработки ответа

Исполнительная программа выбирает подпрограммы для оперативного списка из трех «справочных» списков. Эти списки первоначально определяются программистом как часть списка описания испытуемого. Во время моделирования они могут быть изменены.

Первый справочный список содержит названия *упорядочивающих программ*. Каждая из этих программ делит примеры во внутренней памяти на два множества: примеры, работающие позитивно («положительные»), и примеры, работающие негативно («отрицательные»). Эти две категории являются взаимоисключающими и охватывают все примеры в памяти. При моделировании мы испытали таким образом три полученные упорядочивающие программы. Одна из них помещает в множество всех «положительных» примеров все примеры, принадлежащие

к классу, который был определен программистом как класс, для которого находится понятие. Вторая текущая возможная подпрограмма использует обратную процедуру, помещая те же самые примеры в «отрицательные» множества. (Если программист указал, что есть несколько классов одинаковой важности, то эти две программы используют название класса самого последнего из предъявленных примеров.) Третья, упорядочивающая, программа определяет как «положительные» все те примеры, которые имели название класса наименьшего множества, представленного во внутренней памяти; при этом предполагается, что существует по меньшей мере два примера в одном множестве.

Другой справочный список содержит название программ, которые формируют рабочее описание всех примеров в памяти. Эти программы связывают с каждым примером во внутренней памяти определенное описание, полученное за счет изменения той части формального описания, которая включена в текущий набор признаков. В наших экспериментах мы использовали две программы описания. Одна из них просто переписывает необходимые признаки и оценки из формального описания в рабочее описание. Другая программа определяет новые признаки, основанные на связи между оценками признаков формального описания. Для создания рабочего описания использовались следующие правила:

1. Новый признак определяется для каждой пары признаков (источника), которые количественно оцениваются по одной и той же шкале. Для каждого примера оценка нового признака принимает значение «равно», «больше» или «меньше» в зависимости от результатов сравнения оценок первоначальной пары признаков из этого примера.

2. В том случае, если во всех хранящихся в памяти примерах два признака источника имеют общую оценку, для каждой пары признаков источников определяется новый признак в списке формального описания. (Общая оценка не обязательно должна появляться у обоих признаков в одном и том же примере.) В зависимости от результатов сравнения оценок первоначальных величин в рассматриваемом примере оценка нового признака в каждом из примеров принимает значение: «тот же» или «другой».

Фактически при моделировании упорядочивающие и описательные программы применяются последовательно. Но они функционально параллельны: выход одной программы не оказывает влияния на выход другой. Обе они дают выход для решающей программы. Этот выход состоит из всех примеров во внутренней памяти, переклассифицированных и переписанных. Решающая

программа пытается найти метод для различения «положительных» и «отрицательных» примеров. Различение всегда начинается как определение «положительных» примеров, даже в том случае, если они являются дополнением класса, для которого программа пытается определить понятие.

Настоящая модель содержит три программы решения. Первые две из них пригодны для решения задач по обучению конъюнктивным понятиям (задачи, в которых для формулировки ответа достаточно одной только логической связки «И»). Третья программа представляет собой более длительную, более сложную и обладающую большей общностью условную процедуру.

Обе «конъюнктивные» программы в качестве первой операции вносят в список те признаки, которые имеют лишь одну оценку во всем множестве «положительных» примеров. Если не существует такого списка, то и не существует ни одного конъюнктивного определения «положительного» примера. Если такой список существует, он обрабатывается двумя программами другим способом. Первая конъюнктивная программа исследует все признаки для того, чтобы определить, имел ли хотя бы один из них ту же оценку в «отрицательном» примере, какую он имел во всех «положительных» примерах. Вторая программа проверяет все «отрицательные» примеры, чтобы узнать, имел ли хотя бы один из них полную конъюнкцию пар признак — оценка, которые являются общими для всех «положительных» примеров. Программа повторяет ответ в том случае, если не найдено ни одного такого примера. Таким образом, обе программы в случае их успешной работы определяют конъюнктивное понятие, которое можно использовать для примеров во внутренней памяти.

Третья, решающая программа (условная программа) представляет собой рекурсивную функцию, которая при ее небольшом видоизменении сможет придать модели испытуемого способность ответить на любую задачу по обучению понятиям. В ее настоящем виде она обладает способностью решать задачи ограниченной сложности по созданию дизъюнктивных понятий.

Условная программа вначале опознает ту из пар признак — оценка, которая наиболее часто встречается в «положительных» примерах. Затем она выдает два подсписка — работающий позитивно и работающий негативно, каждый из которых содержит эту пару. Для двух подсписков используется первая конъюнктивная программа. Если это приводит к успеху, то она заново дает ответ, который может быть использован для любого последующего примера, имеющего соответствующую пару признак — оценка. Если происходит так, что условная программа выдает лишь подсписок «положительных» примеров, то ответ

представляет собой оценку простого признака, находящегося на рассмотрении. Если в следующем примере не появляются пары признак — оценка, то вопрос о принадлежности данного примера к определенному классу остается открытым.

Если ответ получается не по такому принципу или если остаются неклассифицированные примеры, повторяется условная программа, опускающая первоначально рассматриваемые пары признак — оценка и все классифицированные ранее примеры. Результат применения условной и конъюнктивной программ составляет второй «условный» ответ. Данная процедура повторяется до тех пор, пока не будут рассмотрены все примеры во внутренней памяти или все признаки.

В результате получается правило классификации, состоящее из цепочки предложений (высказываний) о простых конъюнктивных ответах и правилах, при которых они применяются (например, «красный треугольник», «зеленый круг»). Цепочка высказываний может иметь любую длину, но каждое высказывание должно содержать лишь две пары признак — оценка. Это ограничение можно было бы устранить, применив второе конъюнктивное правило взамен первого. Можно было бы также ввести условное правило n-го уровня путем рекурсивного применения условной программы к подспискам до тех пор, пока не будут классифицированы все примеры. Результирующая процедура даст правило для решения всех задач по образованию понятий. Эта процедура не обязательно должна представлять собой самое компактное высказывание точного правила.

Когда исполнительная программа выбирает исполнительный список, она в сущности использует шаблон для ответа на каждую задачу. Если задача имеет ответ, включающий в себя необходимые характеристики, не учтенные упорядочивающей программой и программой описания, оперирующими в определенных наборах признаков, и если такой ответ не принадлежит к определенному логическому типу, то существует исполнительный список, который отыщет этот ответ.

Способ, которым наша первая модель изменяет свою структуру, показан на фиг. 2. Вначале выбирается множество признаков. Затем из справочных списков, содержащихся в описании испытуемого, выбирается первый исполнительный список. Этот список использует всегда те программы, которые находятся в начале каждого справочного списка. Если невозможно получить ответ при помощи исполнительного списка, то программа описания или программа решения (выборочно) изменяется до тех пор, пока не будет заново сформирован первоначальный исполнительный список. Когда это достигается, выбирается новая упорядочивающая программа. Изменение решающей программы и упорядочивающей программы происходит до тех пор, пока снова не повторится исполнительный список, в результате чего выбирается новая упорядочивающая программа.

Когда не остается больше ни одной упорядочивающей программы, набор признаков заменяется следующим из списка испытуемого в порядке записи этих наборов. Процесс заканчивается в том случае, если сформирован ответ, если рассмотрены все наборы признаков или превышено допустимое время работы. Как это осуществляется, будет сейчас показано.

Во время решения определенной задачи порядок наборов признаков остается неизменным. Однако за время формирования ответа могут временно меняться справочные списки для программ описания и решения. Это осуществляется за счет передвижения символа в списке связи с первого места на последнее всякий раз, когда он устраняется из исполнительного списка. Один из способов моделирования индивидуальных различий заключается в изменении первоначального порядка программ в справочных списках.

Как мы уже указывали, существует механизм контроля времени, который может прерывать процесс формирования ответа. Каждой программе справочного списка приписан индексный номер. Эти номера определяются программистом как часть исходных данных вместе с числом, соответствующим времени, за которое модель испытуемого должна найти ответ. В зависимости от условий предъявления примеров это число может представлять собой время, отпущенное модели испытуемого на решение всей задачи, или время между предъявлениями двух стимулов. Каждый раз, когда работает программа, содержащаяся в исполнительном списке, ее индексный номер вычитается из числа, соответствующего сигналу времени, которое первоначально было установлено равным числу, соответствующему отпущенному времени. Когда число, соответствующее сигналу времени, превращается в нуль, формирование ответа прекращается (возможно, что к этому времени справочные списки для описания и решения уже перестроены) и управление передается от исполнительной программы решения более высокому уровню.

Соответствующий каждой программе индексный номер можно рассматривать как количество «затрат», т. е. как «стоимость» определенной программы обработки информации для испытуемого. Успех в решении каждой задачи зависит от сложного взаимоотношения между правилами по изменению порядка программ в справочном списке, оценкой индексного числа и оценкой отпущенного времени. Изучение этих зависимостей является одной из наших наиболее увлекательных исследовательских задач.

Эта модель, как она сейчас запрограммирована, имеет независимый контроль времени. Всякий раз, когда программе предъявляется новый пример, он исследуется для определения того, принадлежит ли он к классу, который предсказывается для него текущими ответами. Если такого согласования нет или (в случае условных ответов) если не предсказывается принадлежность ни к одному из классов для этого примера, начинает работать программа формирования ответа. При правильном предсказании включение блока формирования ответа происходит только в том случае, если в описании задачи предписана «низкая» скорость предъявления стимулов.

Всякий раз, когда формируется ответ, используются описательный и исполнительный списки. В момент решения задачи (т. е. после того, как были показаны все примеры) те наборы признаков, которые были связаны с ответом, и те программы, которые появились в списках, приведших к успеху, передвигаются в начало соответствующих справочных списков. Таким образом, характеристики испытуемого, которые были вначале определены программистом, изменились самой программой.

Процедура «переноса» обучения[1] имеет очень интересное психологическое приложение. Наш искусственный испытуемый обладает способностью позитивного и негативного переноса только тогда, когда решена предыдущая задача. Перенос почти всецело зависит также от формы задачи, которая непосредственно предшествовала данной. Однако мы не знаем, справедливо ли это для решения задач человеком.

Моделирование и оценка результатов

Предлагаемая модель создавалась не на «пустом месте». Некоторое время мы затратили на рассмотрение других моделей, которые, однако, не были запрограммированы [488]. Кроме того, мы изучали протоколы аспирантов Йельского университета, пытавшихся решать задачи по «формированию понятий». Эти задачи имели три логически правильных ответа: дизъюнкцию, конъюнкцию и отношение. (Данная задача была уже описана ранее [503], и мы имели некоторые сведения относительно трудностей, встречающихся при таком исследовании.) Каждый испытуемый встречался в опыте с тремя условиями предъявления стимулов. Модель, о которой мы только что рассказали, дала наилучшие ответы по сравнению со всеми другими моделями, которые мы могли сконструировать. Создавая ее, мы изменяли порядок и обозначения символов в справочных списках, но в остальном модель не менялась.

[1] См. ниже, стр. 332. — *Прим. ред.*

Ввиду того что каждый испытуемый решал три задачи, мы имели возможность проверить наши процедуры «переноса» и не теряли слишком много времени из-за заранее определенного порядка. Результаты нашего сравнения были в основном обнадеживающими. Однако они не могут рассматриваться как строго обоснованные, так как для составления программы мы использовали протоколы испытаний.

Некоторые более обнадеживающие факты были получены в другом эксперименте, где испытуемым предлагалось решать ряд задач [992]. Это было совершенно обособленное исследование. Испытуемым предлагалось найти правила классификации для каждого из шести возможных типов расщепления восьми примеров, каждый из которых находился в одном из двух наборов по четыре примера и описывался одним из двух значений трех признаков. Испытуемые могли решать (и притом очень быстро) задачи, в которых необходимая информация выводилась на основе одного признака. Это могла делать и наша модель испытуемого. Как испытуемый, так и его модель довольно легко определяли, что задача состоит из «цепочки» двух условных высказываний (например, «большой и красный» или «малый и белый»). В третьем случае испытуемые и их модели не были в состоянии создать рабочее правило для классификации «типа VI», в которой создание ответа требовало описания каждого примера для довольно трудного правила об изменении оценок. В одной из ситуаций испытуемые проделывали это лучше, чем модели их поведения. Если верный ответ может быть получен в виде простого правила с одним исключением, то наша программа считает такую задачу сложной. Испытуемые также считают эту задачу сложной, однако не настолько сложной, как задачи «типа VI». Результаты данного моделирования и, в частности, расхождение между моделью и поведением испытуемых, о котором мы упоминали, заставили нас рассматривать другие рекурсии в программе условного решения.

Аналогичный эксперимент, результаты которого еще не опубликованы, проводился Хантом и Уэллсом. В их эксперименте ответ получался на основе пяти обычно используемых логических связок между двумя элементами. «Таблица истинности» формировалась и предъявлялась испытуемому в геометрической форме. Например, связь «p и q» могла быть представлена как «красная и звезда». Пять задач показывались испытуемому пять раз в различной последовательности, причем каждый испытуемый решал все пять задач в одной из этих последовательностей. Когда писалась эта работа, моделирование и анализ эксперимента не были еще окончательно завершены, однако все же можно указать некоторые предварительные результаты.

Наблюдается хорошее совпадение между нашими моделями и некоторыми протоколами. Как машинная модель, так и испытуемые очень чувствительны к порядку, в котором предъявляются задачи, но их реакции не настолько сходны между собой, как нам хотелось бы. Необходимо поэтому создать новую процедуру «переноса». В эксперименте, который не был непосредственно связан с моделированием, Уэллс изучал приемы, которыми испытуемый обучался методам решения дизъюнктивных задач. Мы надеемся, что результаты этих экспериментов дадут ключ к выяснению природы процедур «переноса», которые нам следовало бы включить в нашу модель.

Мы не претендуем на то, что дали полное объяснение процесса обучения понятиям! Думается, что многие согласятся с нами. При создании нашей модели многие решения были приняты без достаточного теоретического или экспериментального обоснования. Некоторые из них наверняка не верны. Но какие именно?

Вероятно, нам придется изменить программы памяти и распознавания. Некоторые из известных явлений памяти не могут быть воспроизведены с помощью простых моделей памяти с временным размещением информации. Например, не может быть воспроизведен эффект подобия нескольких стимулов. При наложении признаков наша модель работает по принципу «все или ничего», т. е. мешающий стимул или полностью разрушает запись предыдущих стимулов в памяти, или абсолютно не влияет на нее. Это, очевидно, не является окончательным ответом на вопрос о работе памяти при обучении понятиям.

Были рассмотрены две различные системы памяти. Одна из этих систем представляет собой расширенный вариант системы с частичным размещением. Вместо того чтобы записывать одно «кодовое слово» (в действительности списковую структуру), представляющее собой всю известную информацию о стимуле, в одном списке размещения, можно было бы записать несколько «кодовых слов» в нескольких списках размещения. Каждое из этих «слов» будет представлять собой определенный тип информации о некоторой части данного стимула. Запись каждого «слова» производилась бы независимо в каждом из списков размещения. Кодовые слова, относящиеся к одному и тому же примеру (стимулу), снабжаются данными о взаимном расположении в памяти. Когда необходимо получить информацию из памяти, схема каждого примера будет воссоздаваться вновь из первоначальной системы перекрестных связей. Однако, ввиду того что поступающие примеры могут записываться независимо в каждый из списков размещения, некоторые кодовые слова могут быть восстановлены на прежних местах. Степень этого

восстановления будет зависеть от подобия между примером, который должен появиться, и стимулом, который следует за предъявлением этого примера. Такая система будет очень чувствительна к эффекту подобия стимулов.

С другой стороны, мы могли бы использовать и систему ассоциативной памяти. Вместо того чтобы пытаться запомнить элементы информации непосредственно, мы могли бы создать «ассоциации» между названиями и характеристиками стимулов. Это является основным принципом методики, используемой многими психологами, изучающими проблемы обучения. Механизмы, которые реализуют такой вид памяти, подробно исследовались Розенблаттом [930—933]. Существует также определенная аналогия между этим подходом и механизмами классификации, составляющими основу «Пандемониума» Селфриджа [977]. Использование такой системы памяти потребовало бы изменения всех принципов работы нашей модели.

Ассоциативные схемы обычно содержат в себе механизм для обучения понятиям. Они, видимо, также потребуют некоторого «градиента обобщения». Проведенные недавно эксперименты [991, 992] показали, что тенденция к символическому кодированию стимулов при выработке понятий играет большую роль, чем обобщение, основанное на подобии стимулов. По этим причинам мы на первых порах отказались от рассмотрения механизмов ассоциативной памяти.

В данной модели мы подвергаем формальное описание примера двум преобразованиям. При предъявлении примера происходит проверка параметров формального описания, чтобы определить, какая информация должна быть записана в памяти. Через некоторое время та часть формального описания, которая записана в памяти, преобразуется с целью создания рабочего описания. Эти две процедуры можно было бы объединить, если бы описательная программа, находящаяся в данный момент в начале справочного списка описательных программ, могла быть непосредственно применена к стимулу до того, как он будет записан в память.

Такая процедура обладает преимуществом в смысле экономии объема памяти. Вместо того чтобы иметь две отдельные области во внутренней памяти для рабочего и постоянного описания, в этом случае необходимо было бы хранить лишь одно описание. Но за выигрыш объема памяти мы расплачиваемся потерей информации. По определению любое рабочее описание можно получить из формального описания. Все рабочие описания не могут быть получены одно из другого. Например, если нам известно, что пример содержит две фигуры одного и того же цвета, мы все же не знаем, что это за цвет. В результате

способность нашей модели испытуемого использовать определенную программу описания в данный момент будет очень сильно зависеть от той программы описания, которая использовалась ранее.

Более тщательно должно быть исследовано влияние «установки» в момент предъявления стимула на последующие характеристики памяти. Результаты некоторых экспериментов [592, 593] указывают, что «установка» представляет собой скорее функцию способа поиска в памяти, нежели способа записи элементов информации в память. В литературе имеются также довольно противоречивые данные о «скрытом обучении»; этот термин используется для описания экспериментов, в которых животные, научившиеся реагировать на раздражитель *A* в присутствии раздражителя *B*, не связанного с текущими потребностями животного, в результате быстрее обучаются впоследствии реакции на раздражитель *B*. На основе существующих экспериментальных данных еще не ясно, как при моделировании обучения понятиям процедура распознавания символов должна быть соединена с процедурой формирования ответов.

В настоящей модели, видимо, не учтены надлежащим образом процессы «переноса». «Перенос» определяется как влияние предыдущего опыта в решении задач на решение той задачи, с которой испытуемый сталкивается во время опыта. Вначале мы решили использовать простой способ представления «переноса», когда испытуемый пробует применить все приемы, которые в прошлом оказывались эффективными. Основной результат моделирования логических связей, выполненного Хантом и Уэллсом, состоял в доказательстве того, что требуется введение другой процедуры «переноса».

В соответствии с традициями классической теории обучения мы присвоили каждой программе в справочном списке некий переменный числовой индекс. Этот индекс позволяет определить вероятность выбора данной программы. Такой метод представления процесса обучения является, вероятно, наиболее обычным и простым. Основное возражение против него состоит в том, что такой метод предполагает наличие «головных счетчиков» и по существу превращает нашу программу в цифровую модель непрерывного процесса.

В отличие от метода ассоциативных индексов новый метод является методом порядковой перестройки программ в справочном списке. Проблема, связанная с порядковыми перестройками, состоит в том, что они не позволяют указать переменное расстояние между программами списка. Предположим, что мы рассматриваем каждую задачу по обучению понятиям как соревнование между программами из одного и того же справочного списка. Победу одерживает та из них, которая находит место в исполнительном списке, приведшем к решению. Каков должен быть «выигрыш» времени у программы в позиции *n*, чтобы она могла завоевать следующую позицию? Должна ли она перескакивать через позиции? Как мы уже указывали, эти вопросы в настоящее время исследуются.

Вполне возможно, что нам придется полностью изменить применяемый метод «переноса». В настоящее время наша модель записывает ответы вместе с относящейся к ним информацией о применимых программах. Мы могли бы присоединить к программам информацию о тех задачах, в которых эти программы оказались применимыми. Тогда необходимо было бы разработать способы для быстрого извлечения моделью испытуемого ключевых характеристик задачи во время формирования ответа. Затем надо было бы исследовать программы, чтобы выяснить, в чем с точки зрения прошлого опыта они оказались полезными при решении данного типа задач.

С задачей «переноса» непосредственно связана задача выбора ответа во время обучения. Наша нынешняя модель перестраивает свой порядок выбора ответов после того, как задача решена. В момент решения задачи управление выбором ответа происходит на основе отсчета времени, который не зависит от управляющей программы. В настоящее время не используются какие-либо промежуточные вычисления для выбора следующего элемента информации, который необходимо поместить в рассматриваемый список. В другой модели это могло бы стать решающим фактором. Анализ соотношения средств и цели, примененный в программе «Логик-теоретик» [790], широко использует промежуточные вычисления. Амарел [16] предложил модель для решения сходной задачи, где промежуточные вычисления осуществляли управление формированием ответа.

Наши работы по моделированию и анализ экспериментальных данных убедили нас в том, что весьма желательно иметь некоторый способ, который обеспечивал бы, чтобы выбор некоторого элемента в исполнительном списке зависел от результатов исполнения ранее выбранных программ. Остается неясной оптимальная степень такой зависимости. Бартлетт [91] в совершенно другом контексте приводит пример, который может помочь в прояснении этого вопроса. Он сравнивает решение задач с обдумыванием двигательных реакций, например при игре в теннис. Существуют моменты, в которые можно изменить последовательность ответов, а между двумя такими моментами последовательность действий осуществляется без перерывов. Экспериментальная задача состоит в идентификации ответов и выборе соответствующих моментов.

Мы отдаем себе отчет в том, что до сих пор наша модель использовалась в большей степени для выявления недостающих в настоящее время экспериментальных данных, чем для объяснения процесса обучения понятиям. При создании модели мы должны были сознательно ограничить диапазон исследований. В результате мы пришли к некоторым выводам относительно экспериментов, которые надлежит провести. Вполне возможно, что в типичном эксперименте по обучению понятиям смешиваются три процесса: память, распознавание и процесс символического решения задач. Еще неясно, можно ли эти процессы рассматривать как части единого акта «формирования понятий». Каждый из них можно программировать отдельно. Кроме того, мы столкнулись с проблемой «переноса», т. е. влияния, которое текущая гипотеза испытуемого оказывает на усвоение им последующей информации, и влияния недостатка времени на процесс обработки информации. Решение этих задач и явилось одним из главных результатов нашей работы по составлению программ обучения понятиям.

Сравнение с другими работами в данной области

С формальной точки зрения наша задача тесно связана с моделями распознавания образов. Как программирование устройства для «распознавания образов», так и программирование «формирующего понятия» устройства предполагают создание механизма, который, оперируя в заданной среде символов, формирует определенные ответы из стимулов, принадлежащих к заранее определенным множествам. На основе этой математической тождественности по крайней мере один из критиков — Келлер [541] — выразил мнение, что задачи такого типа необходимо рассматривать совместно, не разделяя «психологический» и «нейронный» подходы. Несмотря на то что такое рассмотрение может оказаться полезным при создании теорем относительно канонической формы классификации, оно не может быть приемлемо при создании моделей. Что касается нашего подхода, то он коренным образом отличается от подхода, применяемого в знакомых нам исследованиях по распознаванию образов.

Наиболее существенное различие заключается в способе, с помощью которого мы предварительно кодируем стимулы. В опытах по распознаванию испытуемые обычно воспринимают стимулы, закодированные в виде проекции на экран (сетчатку). Результатом распознавания образа является последовательность битов, каждый из которых соответствует наличию или отсутствию освещенности определенной части экрана. То же представление можно было бы использовать и для временных образов. Каждый бит соответствует наличию или отсутствию у стимула некоторой характеристики.

Мы предполагаем, что существует признаковое и оценочное кодирование [487], и рассматриваем такие данные восприятия, которые можно выразить словами. В опыте же по распознаванию испытуемый создает свой собственный код. Каждая схема кодирования, созданная этим испытуемым, будет специфична для каждого типа используемых стимулов (зрительный, слуховой и т. д.). Ввиду того что нас интересовало оперирование с кодированными элементами, мы обошли эту проблему соответствующей организацией нашей программы и за счет открытых указаний испытуемому в процессе эксперимента.

Наша модель отличается от большинства моделей распознавания образов также характером используемых процессов. Имеющиеся модели распознавания, по крайней мере те, которые были созданы Селфриджем с сотр. [976, 977] и Розенблаттом [934], являются в основном параллельно работающими устройствами, использующими большое число избыточных тестов с коррекцией ошибок. Наша же программа представляет собой последовательно работающее устройство, которое пытается сформировать один, пусть сложный, но зато свободный от ошибок тест классификации. Мы не видим причин, по которым эти два подхода должны были бы взаимно исключать друг друга. Наша программа связана с моделированием символического процесса. То, что эти две задачи формально подобны между собой, еще не означает, что они представляются одинаковыми в процессе решения задач.

В принципе нельзя выдвинуть возражения против использования модели распознавания как входа для модели обучения понятиям. Комбинированная система могла бы выработать свои собственные признаки и оценки, а затем оперировать с ними. Практически говорить о создании такой схемы еще, конечно, преждевременно. Но она представляет собой цель, к которой в перспективе следует стремиться.

Непосредственно проблемой обучения понятиям занимались Кохен [566] и Амарел [16]. Кохен ограничил свою программу формированием «понятий», основанных на конъюнктивном правиле, включающем стимулы, определяемые последовательностью битов. Его программа состоит в применении алгоритмов к информации о множестве объектов, содержащейся *в данный произвольный момент* в памяти. Программа позволяет строить случайные догадки относительно правильного выбора понятия. Эти догадки могут взвешиваться при помощи «доверительных коэффициентов», удовлетворяющих постулатам Пойа [870]

относительно правдоподобных рассуждений. Одно из открытий Кохена [565] в работе с использованием метода Монте-Карло, заключалось в том, что изменения в значении «доверительных коэффициентов» можно использовать для оценки вероятности того, что данный ответ правилен, еще до фактического получения ответа.

Амарел [16] предложил устройство, которое генерирует программу, позволяющую сопоставить аргумент и его значение в символической логике. Основная идея его предложения, которую мы всецело одобряем, состоит в использовании промежуточных результатов для «контроля» за дальнейшим формированием ответа.

Но ни Кохен, ни Амарел не занимались непосредственно моделированием поведения человека. Эти различия в целях исследования и характеристиках программы образуют основные особенности нашей работы по сравнению с другими работами в данной области.

Грубо говоря, наша программа сходна с программами обработки списковых структур, составленными в Технологическом институте им. Карнеги группой «Рэнд корпорейшн», возглавляемой Ньюэллом, Шоу и Саймоном, а также Маккарти [674] и его сотрудниками в Массачусетском технологическом институте. В особенности связана с нашими исследованиями работа Фейгенбаума [318], который разработал программу, моделирующую обучение парным ассоциациям. В качестве части своей программы он включил подпрограмму для селективного выбора характеристик стимулов. По мере накопления опыта в среде стимулов система записывает в свою память все большее число признаков, что позволяет ей производить все более тонкое распознавание. Логические принципы работы распознающей системы Файгенбаума, и в частности ее способность к отбрасыванию признаков, не содействующих распознаванию, также были использованы в нашей программе.

Наша программа, хотя она уже работает, ни в коей мере не является завершенной. Почти каждый новый эксперимент по моделированию указывает на возможности ее улучшения. Мы собираемся продолжить исследования в области обучения понятиям путем использования некоторой модели обработки информации.

Нам хотелось бы лишь предупредить о следующем. Ни наша модель, ни какая-либо другая не привели к постановке значительного числа новых экспериментов. А это ведь является традиционным критерием использования научной модели, и, видимо, в дальнейшем с этим встретимся как мы, так и другие исследователи в этой области. Мы не думаем, чтобы польза от применения машинных моделей в психологии в настоящее время была доказана или опровергнута. Суд продолжается. Мы, естественно, надеемся на его благосклонное решение.

РАЗДЕЛ 3

ПРИНЯТИЕ РЕШЕНИЯ В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ

В большинстве случаев решения принимаются в условиях неопределенности, т. е. когда принимающий решение не знает последствий стоящих перед ним альтернатив. С такой проблемой принятия решения встречается, например, военачальник, посылающий свои войска в бой. В аналогичном положении находится издатель, принимающий решение о тираже выпускаемых им книг. Экономисты, математики и статистики исследовали вопрос о том, как «следует» вести себя в таких ситуациях моделям, в то время как психологи изучали, как люди ведут себя в подобных ситуациях. В двух статьях, включенных в этот раздел, рассматриваются вопросы последней категории. В обеих статьях вычислительная машина используется для создания перерабатывающих информацию моделей поведения отдельных субъектов, принимающих решение в условиях неопределенности.

Фельдман описывает изучение поведения в эксперименте с двоичным выбором. В этом эксперименте субъекта просят предсказать, какое из двух событий произойдет в каждой из серии испытаний («проб»). С помощью данных, полученных от испытуемых, «думающих вслух» в эксперименте с двоичным выбором, Фельдману удалось составить модели познавательных процессов, лежащих в основе поведения при двоичном выборе. Эти модели отражают процессы формирования гипотез, которые происходят у испытуемых в этих экспериментах. Одна из этих моделей с прилагаемым протоколом приводится в следующей статье.

Кларксон в своей работе занимается изучением поведения человека при выборе портфеля ценных бумаг. Он исследовал, как банковский служащий, ведающий инвестициями, выбирает портфель акций для вложения капитала, если ему известны соответствующие ограничения, налагаемые законами штата, цели вложения капитала и рыночная конъюнктура. Проведенный Кларксоном анализ поведения человека, принимающего решение, позволил ему составить программу для вычислительной машины, которая в контрольных случаях сумела с большой

точностью предсказать поведение банковского служащего. В четырех таких случаях модель отобрала 29 акций — то же их число, что и банковский доверенный, и правильно предсказала количество акций в каждом портфеле. Модель также очень хорошо определила число держателей акций каждого вида. Только в пяти из 29 операций выбора модель выбрала акции, отличные от выбранных банковским служащим. Эти результаты указывают на то, что модель Кларксона является отличным предсказателем поведения банковского доверенного.

Значение обеих этих моделей выходит за пределы тех частных ситуаций принятия решения, для которых они были созданы. Работа Фельдмана указывает на то, что кажущаяся простой задача представляется испытуемому очень сложной ситуацией. Метод формирования и проверки гипотез, которым пользуется испытуемый, вполне сходен с поведением исследователя, изучающего того же испытуемого. Испытуемый ищет некоторые «образы» (pattern) в последовательности событий, экспериментатор же ищет «образы» поведения испытуемого. Оба стараются обнаружить регулярность, в существовании которой и тот и другой глубоко убеждены.

Модель Кларксона дает важное подтверждение правильности модели принятия решения человеком, описывающей принятие решения в условиях ограниченной памяти и при наличии довольно маленького набора простейших правил. Ни испытуемый в эксперименте с двоичным выбором, ни банковский доверенный не следуют раз и навсегда принятым стратегиям для «рационального» поведения в окружающей их среде. Детальный анализ, подобный, например, проведенному Фельдманом и Кларксоном, дает полезную информацию о том, как ведут себя люди в конкретных ситуациях, требующих принятия решений, и предлагает объяснение поведения в других ситуациях.

14. МОДЕЛИРОВАНИЕ ПОВЕДЕНИЯ В ЭКСПЕРИМЕНТЕ С ДВОИЧНЫМ ВЫБОРОМ

Дж. Фельдман

Введение

Современные быстродействующие вычислительные машины применяются для моделирования больших, сложных систем с целью облегчения их изучения. Одной из таких систем, изучавшихся с помощью моделирования на вычислительной машине, являлся человек. Настоящая статья является еще одним вкладом в непрерывно растущие усилия по изучению процессов мышления человека при помощи моделирования их на вычислительной машине. Исследования, представленные в этой статье, касаются вопросов моделирования поведения отдельных испытуемых в эксперименте с двоичным выбором [326]. В первом разделе статьи дается описание эксперимента, второй содержит обзор модели, а в третьем подробно описывается модель конкретного испытуемого.

Эксперимент с двоичным выбором

В эксперименте с двоичным выбором испытуемому предлагается предсказать, какое из двух событий E_1 или E_2 произойдет в каждой из некоторой серии проб. После того как испытуемый делает предсказание, ему сообщается, какое из событий произошло в действительности. Последовательность событий обычно задается некоторым случайным механизмом, например таблицей случайных чисел. При осуществлении пробы может произойти одно и только одно событие. Событиями могут быть вспышки света или появление определенных символов на карточках. Обычно от испытуемого требуют попытаться сделать как можно больше правильных предсказаний.

В отличие от этого в описываемом ниже эксперименте испытуемого просили «думать вслух» так, чтобы, помимо самого предсказания, излагать также его обоснование. Высказывания испытуемого заносились в протокол. Требование «думать вслух» ставилось для того, чтобы увеличить информацию о процессах, происходящих в мозге испытуемого при решении им поставленной задачи.

Такая методика эксперимента применялась уже в некоторых классических исследованиях поведения при решении задач [290, 461], а также позднее при изучении процесса мышления с помощью моделирования его на вычислительной машине [226, 798]. При сравнении поведения «думающих вслух» и «не думающих вслух» испытуемых в эксперименте с двоичным выбором не было обнаружено какого-либо значительного различия [326]. Событиями в данном эксперименте являлись символы «крестик» и «галочка». «Галочка» появлялась в 142 случаях из 200, а «крестик» — в остальных 58 случаях. Символы записывались на пронумерованных карточках размером 3×5 *дюймов*.

После того как испытуемый делал свое предсказание для пробы t, ему показывалась данная карта t, содержащая «крестик» или «галочку». В то время когда испытуемый предсказывает событие пробы t, он может видеть лишь предыдущее событие пробы $t-1$.

Протокол с записью высказываний испытуемого D.H. и экспериментатора, которым являлся автор данной статьи,

составленный в течение часового эксперимента с двоичным выбором, представлен в приложении. Символы «крестик» и «галочка» при описании всюду обозначаются соответственно как P и C.

Основная модель

Для того чтобы промоделировать поведение одного испытуемого в эксперименте с двоичным выбором, модель поведения должна быть представлена в виде машинной программы. Если затем программе предлагается предсказывать те же самые последовательности событий, что и испытуемому, то поведение программы, т. е. предсказания и их обоснования, можно сравнить с поведением испытуемого. Если поведение программы является приемлемой копией поведения испытуемого, то программа является по меньшей мере удовлетворительным объяснением поведения испытуемого.

Уровень, на котором производится анализ поведения, фактически определяется отчетом самого испытуемого. Не было сделано попытки пойти дальше, перейдя к исследованию более фундаментальных процессов, лежащих в основе человеческого поведения, например перейти на уровень нейрологии или исследования химических процессов. Таким образом, эта модель является попыткой установить связь между соображениями или гипотезами, которые испытуемый выдвигает в своих предсказаниях, и предшествующими гипотезами, предсказаниями и событиями. Предполагается, что испытуемый активно выдвигает гипотезы о структуре серии событий. Эти гипотезы проверяются путем использования их для предсказания событий, и если предсказание данного события правильно, гипотеза обычно сохраняется, если же предсказание неправильно, то выдвигаются новые гипотезы.

Модель испытуемого D. H.

Модель каждого испытуемого основывается на детальном изучении протоколов и на некоторых общих предположениях о поведении человека. По-видимому, целесообразно прежде всего подробно описать модель для испытуемого D. H., протокол поведения которого приведен в приложении.

Гипотезы

Модель предлагает два типа гипотез относительно данной серии событий. Гипотезы первого типа касаются схемы чередования событий. Модель имеет набор из следующих девяти схем:

1) последовательность одних C;
2) последовательность одних P;
3) простое чередование;
4) $2C$ и $1P$;
5) $1C$ и $2P$;
6) $2P$ и $2C$;
7) $3P$ и $3C$;
8) $4P$ и $4C$;
9) $4P$ и $3C$.

Модель может предположить, что серия событий происходит согласно одной из этих схем, и использовать гипотезы этого типа, чтобы предсказать события данной пробы t.

Предсказания по первым двум схемам (последовательностей только из одних C или только из одних P) для опыта t независимы от событий, предшествующих пробе t. Предсказания по другим схемам (схемы чередования) зависят от этих предшествующих событий. Так, если испытуемый предлагает схему «простое чередование» для пробы t, а событием пробы $t-1$ было C, то предсказанием для опыта является P.

Чтобы облегчить выработку предсказания по схеме чередования для опыта t, схемы записываются как классифицирующие сети. Например, схема «$2C$ и $1P$» представляется в следующем виде:

Является ли событие пробы $t-1$ событием C?
  Нет — предсказываю для пробы t событие C.
  Да — является ли событие пробы $t-2$ событием C?
    Нет — предсказываю для пробы t событие C.
    Да — предсказываю для пробы t событие P.

Второй тип гипотез, который может предложить модель,— это «антисхемная» гипотеза или гипотеза «противоположного вида» («anti-pattern or guess-opposite hypothesis»). Например, модель может предположить, что событие в пробе t будет противоположно тому событию, которое предсказывается данной схемой. Этот тип гипотез является модельным представлением понятия «ошибка игрока», так как «здравомыслящие люди» предсказывают «решку», если до этого семь раз выпадал «орел»

В самом общем виде гипотеза включает две составляющие: схемную и антисхемную компоненты. Выбор гипотезы получается отысканием схемной составляющей. Если гипотеза имеет антисхемную компоненту, то выбирается гипотеза, противоположная предсказанию по схеме. Если же гипотеза не имеет антисхемной компоненты, то гипотеза выбирается предсказанием схемной компоненты. Таким образом, в то время как предсказание по схемной гипотезе: «последовательность из одних C» всегда есть C, предсказание по гипотезе «антисхема последовательности из одних C» всегда есть P.

Основной цикл

Основной цикл модели состоит в следующем: модель использует гипотезу для предсказания события в пробе t. Затем это событие происходит. В «фазе I» модель «объясняет» событие опыта t с помощью объяснительной гипотезы. В «фазе II» формируется «предсказательная гипотеза» для пробы $t+1$. Модель использует эту «предсказательную гипотезу», чтобы определить событие в опыте $t+1$. После этого происходит событие опыта $t+1$ и цикл повторяется.

Фаза I

Необходимость этой фазы определяется в основном тем, что модель должна «объяснить» каждое событие. Приемлемым объяснением является гипотеза, дающая возможность предсказать событие в данной пробе. Процесс, происходящий в фазе I, представлен логической программой в табл. 1.

Таблица 1

Фаза I модели испытуемого D.H. в эксперименте с двоичным выбором

A. Смогла ли схемная компонента предсказательной гипотезы в опыте t правильно предсказать событие пробы?

B. Да — объяснительная гипотеза для пробы t является схемной компонентой предсказательной гипотезы для пробы t.

C. Нет — включаются схемы, которые смогли правильно предсказать событие в пробах t и $t-1$. Схема предсказательной гипотезы для пробы t включается, если она смогла правильно предсказать события в опытах $t-1$, $t-2$ и $t-3$.

D. Выбрать из множества включенных схем ту схему, которая чаще всего выбиралась в предыдущих опытах.

E. Является ли выбранная схема схемной компонентой предсказательной гипотезы для опыта t?

F. Да — объяснительная гипотеза в опыте t «сбивает» меня с избранной схемы.

G. Нет — объяснительной гипотезой для опыта t является избранная схема.

Процесс образования объяснительной гипотезы в пробе t начинается с проверки того, правильно ли схемная компонента предсказательной гипотезы определила событие в опыте t. Если схемная компонента «сумела» правильно предсказать событие, то она и является объяснительной гипотезой. Если же схемная компонента «не сумела» предсказать событие правильно, срабатывает механизм замены схем.

Таким образом, если предсказательная гипотеза содержит только схемную компоненту и сама гипотеза подтвердилась, то

объяснительная гипотеза для пробы t является в то же время и предсказательной гипотезой для этой пробы. Если же предсказательная гипотеза для пробы t содержит антисхемную компоненту и гипотеза подтвердилась, то срабатывает механизм замены схемы, так как схемная компонента не смогла сама правильно предсказать событие.

Если предсказательная гипотеза для опыта t была антисхемной гипотезой и она не подтвердилась, то объяснительной гипотезой для опыта t становится схемная компонента предсказательной гипотезы.

Мотивировка здесь очень проста, хотя объяснение несколько надуманно. Если в такой ситуации с двоичным выбором гипотеза, предполагающая, что схема изменится, приводит к неправильному предсказанию, то схема закрепляется, являясь приемлемым объяснением события. Если же гипотеза, предполагающая, что схема изменится, ведет к правильному предсказанию, то схема не закрепляется и рассматривается одна из возможностей новой схемы.

Если схемная компонента предсказательной гипотезы для пробы t не в состоянии предсказать событие t, то включается механизм изменения схемы. Механизм изменения схем состоит из двух частей. Первая из них включает подмножество из девяти ранее рассмотренных схем, а вторая выбирает единственную схему из включенного множества схем. Предполагается, что включаемая схема может являться объяснением событий в пробе t, если схема может предсказать события проб t и $t-1$. Схема предсказательной гипотезы для пробы t, являющаяся схемой, которая не может предсказать событие t, вводится в рассматриваемые множества схем, если она может предсказывать события $t-1$, $t-2$ и $t-3$. Из множества рассматриваемых схем в качестве схемной компоненты объяснительной гипотезы избирается та схема, которая выбиралась в предыдущих опытах чаще всего. Если избрана схемная компонента объяснительной гипотезы для пробы t, то объяснительная гипотеза является антисхемной гипотезой, которая представляет собой модельную интерпретацию гипотезы испытуемого: «Вы пытались сбить меня с правильного пути». (См., например, пробу 9 из протокола в приложении.) Модель интерпретирует событие t как попытку «сбить с правильного пути», когда выполняются следующие три условия: 1) схема не может предсказать событие пробы t; 2) схема может предсказать по меньшей мере три последовательных события проб $t-1$, $t-2$ и $t-3$ и 3) схема наиболее часто выбирается из множества включенных схем.

Фаза II

В то время как фаза I имеет дело в основном со схемной компонентой гипотезы, фаза II имеет дело с антисхемной компонентой. Фаза II представлена в табл. 2.

Таблица 2

Фаза II модели испытуемого D.H. в эксперименте с двоичным выбором

H. Содержала ли предсказательная гипотеза для пробы t антисхемную компоненту?
- I. Да — правильно ли она предсказала событие пробы t?
  - J. Да — предсказательная гипотеза для пробы $t+1$ является схемной компонентой объяснительной гипотезы для пробы t.
  - K. Нет — содержала ли предсказательная гипотеза для проб $t-1$ и $t-2$ антисхемную компоненту и были ли правильными предсказания событий этих проб?
    - L. Да — предсказательная гипотеза для пробы $t+1$ является антисхемной объяснительной гипотезой для пробы t.
    - M. Нет — предсказательная гипотеза для пробы $t+1$ является объяснительной гипотезой для пробы t
- N. Сохранится ли объяснительная гипотеза для пробы t? (Объяснение см. в тексте)
  - O. Да — предсказательная гипотеза для пробы $t+1$ является объяснительной гипотезой для пробы t.
  - P. Нет — предсказательная гипотеза для пробы $t+1$ является антисхемной объяснительной гипотезой для пробы t.

Q. Предсказать событие для пробы $t+1$.

Если предсказательная гипотеза для пробы t содержит антисхемную компоненту, то она обрабатывается аналогично схемной компоненте в фазе I. Если антисхемная предсказательная гипотеза правильно предсказала событие пробы t, то антисхемная компонента сохраняется и предсказательная гипотеза для пробы $t+1$ становится антисхемной объяснительной гипотезой. Если же антисхемная предсказательная гипотеза для пробы t определила событие неправильно, то антисхемная компонента рассматривается как «сбивание с правильного пути».

Если предсказательные гипотезы для проб $t-1$ и $t-2$ имели антисхемные компоненты и эти гипотезы подтвердились, то антисхемная компонента сохраняется для предсказательной гипотезы пробы $t+1$. Если эти условия не выполняются, то антисхемная компонента отвергается и предсказательная гипотеза для пробы $t+1$ является объяснительной гипотезой для пробы t.

Если предсказательная гипотеза для пробы t не содержит антисхемной компоненты, то модель рассматривает, должна ли

в пробе $t+1$ вводиться антисхемная компонента. Модель принимает решение на основе своего прошлого опыта. Она определяет число последовательных событий, включая и событие пробы t, которые могут быть предсказаны объяснительной гипотезой пробы t. Это число мы обозначим как N_1. Затем модель ищет в памяти пробы, начиная с последней, входящей в N_1, пытаясь найти такую пробу, для которой объяснительная гипотеза была такой же, как в пробе t. Вслед за этим модель определяет число последовательных событий, предшествующих, включающих или следующих за этим первым вариантом объяснительной гипотезы для пробы t, которое может быть предсказано этой гипотезой. Это число обозначим N_2. Если $N_1=N_2$, то модель решает, что объяснительная гипотеза для пробы t не будет предсказательной гипотезой для пробы $t+1$. Предсказательная гипотеза для пробы $t+1$ становится в этом случае «антисхемной объяснительной гипотезой» для пробы t. Если $N_2>N_1$, то модель решает, что объяснительная гипотеза для опыта t будет предсказательной гипотезой для опыта $t+1$. Если $N_1>N_2$, то модель делает заключение, что предыдущий вариант объяснительной гипотезы в действительности не подходит, и продолжает искать в своей памяти тот вариант объяснительной гипотезы, у которого $N_2 \geqslant N_1$. Если же такой вариант не обнаруживается, то предсказательная гипотеза для пробы $t+1$ становится объяснительной гипотезой для пробы t.

Предсказание с помощью моделей

Модели, подобные модели для испытуемого D.H., могут использоваться для предсказания тех же серий двоичных событий, которые предлагаются предсказывать испытуемому индивидууму. Высказывания и гипотезы модели — «протокол модели» — могут быть затем сравнены с протоколом испытуемого. Модель не умеет изъясняться на идиоматическом английском языке, поэтому сравнение производится между протоколом машины и соответствующим закодированным вариантом протокола испытуемого (табл. 3 и 4).

Протокол работы модели может быть создан путем предъявления ей событий таким же образом, как в описанном эксперименте с человеком; в другом варианте машина может принять на себя функции экспериментатора, если предприняты соответствующие меры для предохранения модели от «подглядывания». Однако такой «лобовой» метод моделирования поведения испытуемого приводит к некоторым трудностям. Эти трудности аналогичны тем, которые испытывает программа игры в шахматы по книге [797, 950]. Так как решение шахматной программы

о данном ходе m зависит от ее предшествующих ходов $m-1$ и $m-2$, то такая программа, когда она играет по книге, должна «возвращаться на пройденный путь», если ее ход отличается от книжного хода. Чтобы программа имела хорошие шансы принять такое же решение, как в игре по книге, она должна иметь те же предыдущие ходы (ту же историю), что и в книге.

Это возвращение программы на пройденный путь может включать в себя как изменение большого числа параметров, так и изменение самого хода. В другом месте я назвал эту методику исправлений «условным предсказанием».

Предсказание, которое осуществляет модель, обусловлено предыдущими решениями модели, но в то же время совпадает с решениями испытуемого, которые модель пытается предсказать [329].

Применение методики «условного предсказания» для моделей в экспериментах с двоичным выбором, таких, как описанный выше для испытуемого D.H., включает: 1) сравнение поведения программы с поведением испытуемого по всем возможным пунктам; 2) запись выявленной разницы в поведении машины и испытуемого; 3) наложения решения испытуемого на модель там, где это необходимо. С целью осуществления этих функций необходимо создать в программе определенный тип управляющего устройства.

Модель испытуемого D.H. с условной предсказательной системой контроля представлена в табл. 3 и 4. Пример поможет объяснить эти записи. В табл. 3 после каждого решения модели сохранить схему предсказательной гипотезы в пробе t для объяснительной гипотезы пробы t (B) решение сравнивается с решением испытуемого (1). Если решение модели отличалось от решения испытуемого, управление передается механизму изменения схемы (3 пробы). Если же решение модели совпадало с решением испытуемого, управление передается другой части устройства управления (117 проб). Табл. 3 и 4 содержат лишь результаты 195 проб, так как модель начала работать с шестой пробы.

Таблица 3

Сводка поведения для фазы I модели испытуемого D.H., адаптированной для условного предсказания

A. Смогла ли схемная компонента предсказательной гипотезы для пробы t правильно предсказать событие t?

B. Да — объяснительная гипотеза для пробы t является схемной компонентой предсказательной гипотезы для этой пробы.

1. Содержит ли объяснительная гипотеза испытуемого для пробы t схемную компоненту предсказательной гипотезы для пробы t? 120

Да — перейти к 6. 117

Нет — ошибка: неудача при включении механизма изменения схем. Перейти к C. 3

C. Нет — включить схемы, которые правильно предсказали события в пробах t и $t-1$. Схема предсказательной гипотезы для пробы t включается, если она правильно предсказала события в пробах $t-1$, $t-2$ и $t-3$.

2. Включалась ли схема объяснительной гипотезы испытуемого для пробы t? 78

Да — перейти к D. 61

Нет — ошибка: неудача при включении схемы. Добавить схему испытуемого к ранее включенным схемам и продолжать. 17

D. Выбрать из множества включенных схем ту схему, которая чаще всего выбиралась в предыдущих опытах.

3. Избиралась ли схема объяснительной гипотезы испытуемого для пробы t? 78

Да — перейти к E 64

Нет — ошибка: неудача при выборе схемы. Заменить неправильную схему на схему испытуемого и продолжать. 14

E. Является ли выбранная схема схемой компонентой предсказательной гипотезы для пробы t?

F. Da — объяснительная гипотеза для пробы t «сбивает» меня с избранной схемы

4. Содержала ли объяснительная гипотеза испытуемого для пробы t элемент «сбивания с толка»? 27

Да — перейти к H. 26

Нет — ошибка: неправильное включение «сбивания». Вычеркнуть «сбивание» и перейти к H. 1

G. Нет — объяснительной гипотезой для опыта t является избранная схема.

5. Содержала ли объяснительная гипотеза испытуемого для пробы t элемент «сбивания с толку»? 51

Да — ошибка: неудача при включении «сбивания». Включить «сбивание» и перейти к H. 3

Нет — перейти к H. 48

6. Содержала ли объяснительная гипотеза испытуемого для пробы t элемент «сбивания»? 117

Да — ошибка: неудача при включении «сбивания». Включить «сбивание» и перейти к H. 3

Нет — перейти к H. 114

Таблица 4

Сводка поведения для фазы II модели испытуемого D.H., адаптированной для условного предсказания

H. Содержала ли предсказательная гипотеза для пробы t антисхемную компоненту?
I. Да — правильно ли предсказала предсказательная гипотеза событие для пробы t?
J. Да — предсказательная гипотеза для пробы $t+1$ является антисхемной компонентой объяснительной гипотезы для пробы t.
7. Содержала ли предсказательная гипотеза испытуемого для пробы $t+1$ антисхемную компоненту? 6
Да — перейти к 12. 5
Нет — ошибка: напрасное удержание антисхемной компоненты. Исключить антисхемную компоненту и перейти к 12. 1
K. Нет — содержала ли предсказательная гипотеза для $t-1$ и $t-2$ антисхемные компоненты и были ли правильными предсказания событий этих проб?
L. Да — предсказательная гипотеза для пробы $t+1$ является антисхемной объяснительной гипотезой для пробы t.
8. Содержала ли предсказательная гипотеза испытуемого для пробы $t+1$ антисхемную компоненту? 2
Да — перейти к 12. 2
Нет — ошибка: напрасное удержание компоненты Исключить антисхемную компоненту и перейти к 12. 0
M. Нет — является ли предсказательная гипотеза для пробы $t+1$ объяснительной гипотезой для пробы t?
9. Содержит ли предсказательная гипотеза испытуемого для пробы $t+1$ антисхемную компоненту? 12
Да — ошибка: надо было запомнить и удержать в памяти антисхемную компоненту. Ввести ее и перейти к 12. 1
Нет — перейти к 12. 11
N. Сохранится ли объяснительная гипотеза для пробы t? (Объяснение см. в тексте.)
O. Да — предсказательная гипотеза для пробы $t+1$ является объяснительной гипотезой для пробы t.
10. Содержит ли предсказательная гипотеза испытуемого для пробы $t+1$ антисхемную компоненту? 136
Да — ошибка: неудача при включении антисхемной компоненты. Ввести антисхемную компоненту и перейти к 12. 10
Нет — перейти к 12. 126
P. Нет — предсказательная гипотеза для пробы $t+1$ является антисхемной объяснительной гипотезой для пробы t.
11. Содержит ли предсказательная гипотеза испытуемого для пробы $t+1$ антисхемную компоненту? 39
Да — перейти к 12. 2

Нет — ошибка: неправильный выбор антисхемной компоненты. Вычеркнуть антисхемную компоненту и продолжать. 37
12. Была ли схема объяснительной гипотезы испытуемого для пробы t такой же, как схема предсказательной гипотезы испытуемого для пробы $t+1$? 195
Да — перейти к Q 192
Нет — ошибка: неудача при замене схемы. Ввести схему испытуемого в предсказательную гипотезу для пробы $t+1$ и продолжать. 3
Q. Предсказать событие для пробы $t+1$.
13. То же ли событие предсказал испытуемый? 195
Да — перейти к A. 193
Нет — ошибка: неправильное предсказание. Исправить и перейти к A. 2

Выводы

Недостатки моделей

Модель данного испытуемого (D.H.) и подобные ей модели поведения двух других испытуемых в эксперименте с двоичным выбором [326] имеют некоторые недостатки.

Прежде всего нужно отметить, что сравнение поведения модели с поведением испытуемого, являющееся основой для построения модели, не может служить хорошей оценкой модели. Такой тип сравнения позволяет лишь обнаружить некоторое сходство в поведении модели и ее компонент с поведением испытуемого. Для сравнения поведения модели с поведениями испытуемого, не использованными при создании модели, необходимо устранение некоторых из упомянутых недостатков.

Наибольшее число ошибок приходится на долю антисхемной части модели (табл. 4). Хотя испытуемый, несомненно, использует поведение «антисхемного» типа, модель довольно редко предсказывает антисхему, когда испытуемый к ней прибегает. На долю механизма изменения схем приходится меньшее количество ошибок, но тут возникает другая трудность. Эта часть модели фактически является механизмом выбора: схема выбирается из списка схем, используемых испытуемым во время эксперимента. Поэтому более совершенным механизмом изменения схем являлся бы механизм, вырабатывающий схему на основе предшествующей последовательности событий и некоторых основных правил. Одним из таких правил могло бы быть предпочтение при прочих равных условиях схем с одина-

ковым количеством P и C по сравнению со схемами изменения с неодинаковым количеством P и C.

Модели не имеют механизмов, которые могли бы приводить к «ошибкам восприятия», т. е. к восприятию одного символа, в то время как показывается другой Между тем проверка протокола испытуемого D. H. показала, что последний иногда воспринимает C как P (196-я проба).

Модель также не имеет достаточно большого набора гипотез. Испытуемый поддерживает в памяти большее количество типов гипотез относительно серии событий, чем два типа схем (схема и антисхема), которые использует модель. Некоторые испытуемые использовали более сложные структуры формирования гипотез. Например, один из испытуемых был способен установить тот факт, что серия событий была разбита на блоки: серия имела $7P$ и $3C$ в каждом блоке из 10 проб.

Имеются также данные, свидетельствующие о том, что при денежном стимулировании испытуемых некоторые из них чаще дают правильные предсказания события. Создание моделей таких испытуемых требует установления условий, при которых испытуемые отвергают проверку других гипотез или по меньшей мере избегают проверки гипотез путем использования их для предсказания событий. Гипотезы можно рассматривать и проверять и без их использования для предсказания событий.

Значение моделей

Роль машинного моделирования в изучении поведения человека обсуждалась уже в ряде других работ, и я также уже кратко высказал свои взгляды по этому вопросу в другом месте [329]. Поэтому здесь будет достаточно обсудить лишь некоторые применения данной работы для понимания процесса поведения.

Машинные модели поведения в эксперименте с двоичным выбором являются относительно простыми машинными программами, но в то же время сравнительно сложными психологическими моделями. Широко распространенной точкой зрения на поведение в эксперименте с двоичным выбором являлась идея о словесном обусловливании, содержащемся в стохастической модели обучения. В своей простейшей форме эта модель утверждает, что вероятность предсказания испытуемым события E_1 или E_2 в эксперименте с двоичным выбором является экспоненциально взвешенным подвижным средним из предыдущих событий. Однако модель словесного обусловливания довольно плохо согласуется с обнаруженным у испытуемого D.H. и у десятка других испытуемых поведением, имеющим характер проверки гипотез. Протоколы группового поведения в экспериментах по двоичному выбору, предоставленные мне Д. Хейсом, также согласуются с общей идеей проверки гипотез. Другие свидетельства несостоятельности модели словесного обусловливания и доказательства «гипотезо-проверочного» характера модели были рассмотрены ранее [326].

Вычислительная машина дает исследователю «гипотезо-проверочной» модели поведения средства для изучения и испытания этих сложных моделей. Чрезмерно упрощенные объяснения человеческого поведения поэтому уже нельзя больше считать правомерными на том основании, что не существует средства для изучения сложных моделей. Можно ожидать, что использование вычислительных машин для моделирования человеческого поведения послужит расширению человеческих знаний, помогая человеку в изучении его собственного поведения.

Приложение

Протокол испытуемого D. H. [1]

(Сейчас я зачитаю вам некоторые указания. Я собираюсь показать вам серию символов. Это будут символы P и C. Перед каждым символом я буду давать вам сигнал «сейчас». Когда вы услышите этот сигнал, скажите мне, какой символ, по вашему, появится в следующей пробе и почему вы назвали именно этот символ. Ваши показания будут записываться на магнитофоне. Не торопитесь. После того как вы выскажете свое предположение, я покажу вам символ. Вашей целью является предсказывать каждый символ так точно, как вы можете. Итак... Нет ли у вас каких-либо вопросов?) Сначала я должен отгадать, будет ли P или C? (Именно!) Но насчет объяснения, *почему* я так думаю. Это вряд ли будет нечто большее, чем: «я думаю, что будет так, мне так кажется, я так чувствую». Должно ли объяснение содержать еще что-то сверх того? (Вы можете высказать любые соображения. Если у вас возникают только эти соображения, то их и высказывайте. Но может быть, это не так... Ну, ладно, попробуем приступить к делу, а если потом появятся вопросы...)

(Каким будет по вашему 1-й символ?) P. (1-й символ — C.)

(Каким будет 2-й символ?) Это будет P. (Почему?) Он возник у меня в уме. (2-й символ — C.)

Теперь я назову C. (Почему?) Я попытаюсь разгадать вашу закономерность. (3-й символ — C.)

[1] Высказывания в скобках принадлежат экспериментатору.

(Что вы скажете о 4-м символе?). Я скажу опять *С*. (Почему?) Мне кажется, что сейчас будет *С*. (4-й символ — *С*. Когда вы отвечаете, то лучше говорить так: «Я думаю, что пятым будет то-то». Тогда легче будет потом сопоставить магнитофонную запись с листом ответов.)

(Что вы думаете о 5-м?). 5-й будет *Р*. (Почему?) Мне кажется, что это будет *Р*, вот и все. (5-й — *С*.)

(Что вы скажете о 6-м?). 6-й будет *С*, так как вы даете мне все время *С* и я не думаю, чтобы эта закономерность изменилась. (6-й — *С*.)

(Что вы думаете о 7-м?). 7-й будет *С*, так как я не думаю, чтобы закономерность изменилась. (7-й — *С*.)

8-й будет *С* по той же причине. Думаю, что вы не нарушили закономерность. (8-й — *Р*.)

(Что вы думаете о 9-м?). 9-й будет *С*. (Почему?) Мне кажется, что вы дали мне *Р*, чтобы сбить меня, а затем продолжите прежнюю закономерность. (9-й — *С*. Кстати, видите ли вы эти карточки?) Да. (Видите ли вы, чтобы я писал что-то?) Нет, не вижу. (Прекрасно!) Я не смотрю. (Вы можете смотреть на карточки. Я хочу, чтобы вы видели, что я не выдумываю эти символы из головы. Эта последовательность заранее предопределена.)

Хорошо. Тогда сейчас будет *Р*. 10-й будет *Р*. (Почему?) Мне кажется, что теперь последовательность начинает меняться. (10-й — *С*.)

(Что вы думаете об 11-м?) 11-й будет *С*. Вы продолжаете прежнюю закономерность. (11-й — *С*.)

(Что вы думаете о 12-м?) 12-й будет *С*, так как вы продолжаете последовательность. (12-й — *Р*.)

13-й будет *С*. 12-й был *Р*. Вы пытались сбить меня. Закономерность продолжается. (13-й — *Р*.)

14-й будет *Р*. Вы начинаете новую последовательность из *Р*. (14-й — *Р*.)

15-й будет *Р*. Вы все еще продолжаете последовательность. (15-й — *Р*.)

(Каков будет 16-й?) 16-й будет *С*... Чтобы снова сбить меня. (16-й — *С*.)

17-й будет *С*. Вы хотите выяснить, не откажусь ли я от последовательности *Р*. (17-й — *С*.)

18-й будет *Р*. Вы собираетесь нарушить последовательность *С*. (18-й — *С*.)

19-й будет *Р*. Вы собираетесь отказаться от последовательности *С*. (19-й — *Р*.)

20-й будет *Р*. Вы хотите сбить меня, пытаясь заставить меня подумать, что вернетесь к ряду, относительно которого я ошибся.

Я не помню, какой был последний символ. Мне кажется, что *С*. (20-й — *Р*.)

21-й будет *С*. Вы не будете продолжать последовательность *Р*. (21-й — *Р*.)

22-й будет *С*. Вы делаете это для того, чтобы я подумал, что продолжится ряд *Р*. (22-й — *С*.)

23-й будет *С*. Вы пытаетесь заставить меня подумать, что следующим символом будет *Р*, т. е. возвратиться к прежнему ряду. (23-й — *С*.)

24-й будет *С*. Вы собираетесь продолжить ряд *С*. (24-й — *С*.)

25-й будет *С*. Вы все еще хотите продолжить ряд *С*. (25-й — *С*.)

26-й будет *С*. Вы продолжите ряд *С*. (26-й — *С*.)

27-й будет *Р*. Теперь вы нарушите ряд. (27-й — *С*.)

28-й будет *Р*. Теперь вы собираетесь нарушить ряд. (28-й — *С*.)

29-й будет *С*. Вы продолжаете ряд. (29-й — *С*.)

30-й будет *С*. Вы все еще продолжаете ряд. (30-й — *С*.)

31-й будет *С*. Вы продолжите ряд *С*. (31-й — *С*.)

32-й будет *С*. Вы все еще продолжаете ряд. (32-й — *Р*.)

33-й будет *С*. Последний раз вы дали мне *Р*, чтобы сбить меня. (33-й — *С*.)

34-й будет *С*. Вы продолжите ряд. (34-й — *С*.)

35-й будет *Р*. Вы собираетесь сбить меня. (35-й — *С*.)

36-й будет *С*. Вы продолжите ряд. (36-й — *С*.)

37-й будет *С*. Вы продолжаете ряд. (37-й — *С*.)

38-й будет *С*. Ряд продолжается. (38-й — *С*.)

39-й будет *С*. Вы продолжаете ряд. (39-й — *С*.)

40-й будет *С*. Вы продолжаете ряд. (40-й — *С*.)

41-й будет *С*. Вы продолжите ряд. (41-й — *С*.)

42-й будет *С*. Вы продолжаете ряд. (42-й — *С*.)

43-й будет *С*. Вы продолжаете ряд. (43-й — *С*.)

44-й будет *С*. Вы продолжаете ряд. (44-й — *С*.)

45-й будет *С*. Ряд продолжается. (45-й — *С*.)

46-й будет *С*. Ряд продолжается. (46-й — *С*.)

47-й будет *Р*. Думаю, что теперь вы нарушите ряд. (47-й — *С*.)

48-й будет *С*. Вы возвратитесь к прежнему ряду. (48-й — *С*.)

49-й будет *С*. Вы подолжаете ряд. (49-й — *С*.)

50-й будет *С*. Вы продолжаете ряд. (50-й — *Р*.)

51-й будет *С*. Вы дали *Р*, чтобы сбить меня. (51-й — *Р*.)

52-й будет *Р*. Вы начали ряд *Р*. (52-й — *С*.)

53-й будет *Р*. Вы дали *С*, чтобы сбить меня. (53-й — *С*.)

54-й будет *С*. Вы продолжите ряд *С*. (54-й — *С*.)

55-й будет *С*. Вы продолжаете ряд. (55-й — *С*.)

56-й будет *С*. Вы продолжите ряд. (56-й — *Р*.)

57-й будет *P*. Своим последним *P* вы пытались запутать меня, заставив подумать, что этим *P* вы сбиваете меня с последовательности *C*. (Вы сказали, что 57-й — *P*?) *P*. (57-й — *C*.)

58-й будет *C*. Вы начали ряд *C*. (58-й — *P*.)

59-й будет *C*. Вы все еще продолжаете пытаться сбить меня с ряда. (59-й — *P*.)

60-й будет *P*. Вы начинаете ряд *P*. (60-й — *C*.)

61-й будет *P*. Вы начинаете попеременно давать *P* и *C*. (61-й — *P*.)

62-й будет *C*. Вы продолжите чередование. (62-й — *C*.)

63-й будет *C* — нет, скорее 63-й будет *P*, ввиду чередования. (63-й — *P*.)

64-й будет *C*, по закону чередования. (64-й — *C*.)

65-й будет *P*, по закону чередования. (65-й — *C*.)

66-й будет *C*. Вы начали ряд *C*. (66-й — *P*.)

67-й будет *C*. Вы снова вернулись к чередованию. (67-й — *C*.)

68-й будет *C*. Теперь вы примените другой закон чередования: между каждыми двумя *P* — два *C*. (68-й — *P*.)

69-й будет *C*. Вы чередуете *C* с *P*. (69-й — *C*.)

70-й будет *P*. Вы продолжаете эту зависимость. (70-й — *P*.)

71-й будет *C* в силу той же закономерности. (71-й — *C*.)

72-й будет *P* ввиду той же закономерности. (72-й — *C*.)

73-й будет *C*. Вы начали новый ряд, состоящий из *C*. (73-й — *C*.)

74-й будет *C*. Вы продолжаете ряд. (74-й — *C*.)

75-й будет *C*. Вы все еще продолжаете ряд. (75-й — *C*.)

76-й все еще будет *C*. Вы продолжаете ряд. (76-й — *C*.)

77-й будет *C*. Вы все еще продолжаете ряд. (77-й — *C*.)

78-й будет *C*. Ряд продолжается. (78-й — *P*.)

79-й будет *C*. Вы хотели сбить меня этим *P*. Ряд продолжается. (79-й — *C*.)

80-й будет *C*. Ряд продолжается. (80-й — *C*.)

81-й будет *C*. Ряд продолжается. (81-й — *P*.)

82-й будет *C*. Теперь вы чередуете *C* с *P*. (82-й — *P*.)

83-й будет *P*. Вы начали ряд, состоящий из *P*. (83-й — *C*.)

84-й будет *C*. Вы дали *P*, чтобы сбить меня с толку. (84-й — *P*.)

85-й будет *P*. Вы начали новый ряд чередующихся символов. (85-й — *P*.)

86-й будет *C*. Вы следуете закону: одно *C* и два *P*. Сейчас должно быть *C*. (86-й — *C*.)

87-й будет *P*. Вы следуете той же закономерности. (87-й — *C*.)

88-й будет *P*. Вы начали ряд из двух *C* и одного *P*. (88-й — *C*.)

89-й будет *C*. Вы начали новый ряд из одних *C*. (89-й — *C*.)

90-й будет *C*. Вы продолжаете ряд. (90-й — *C*.)

91-й будет *C*. Ряд продолжается. (91-й — *C*.)

92-й будет *C*. Ряд продолжается. (92-й — *P*.)

93-й будет *P*. Раньше вы давали мне *P* для того, чтобы заставить меня подумать, что ряд начинает нарушаться и что вы собираетесь вернуться к нему после этого *P*. Следующим будет *P*. (93-й — *C*.)

94-й будет *C*. Вы вернулись к ряду из *C*. (94-й — *C*.)

95-й будет *C*. Вы начали ряд из одних *C*. (95-й — *P*.)

96-й будет *C*. Вы чередуете *C* с *P*. (96-й — *P*.)

97-й будет *C*. Вы начали ряд из одного *C* и двух *P*. (97-й — *P*.)

98-й будет *P*. Вы начали ряд из *P*. (98-й — *C*.)

99-й будет *C*. Вы начали ряд из трех *P* и трех *C*. Только что вы дали три *P*. 98-м (именно так сказано!) будет *C*. (Итак, вы сказали, что 99-й будет *C*. 99-й действительно есть *C*.)

100-й будет *C*. Вы придерживаетесь этого ряда (100-й — *C*.)

101-й все еще будет *C*. Продолжается ряд из трех *P* и трех *C*. (101-й — *C*.)

102-й будет *C*. Вы начали ряд из *C*. (102-й — *C*.)

103-й будет *C*. Вы продолжаете ряд из *C*. (103-й — *C*.)

104-й будет *C*. Продолжается ряд из *C*. (104-й — *C*.)

105-й будет *C*. Продолжается ряд из *C*. (105-й — *C*.)

106-й будет *P*. Теперь вы нарушите эту закономерность. (106-й — *C*.)

107-й будет *C*. Вы продолжите ряд. (107-й — *P*.)

108-й будет *C*. Вы дали *P*, чтобы сбить меня. Прежний ряд продолжается. (108-й — *C*.)

109-й будет *C*. Вы продолжаете ряд. (109-й — *P*.)

110-й будет *C*. Вы чередуете *C* с *P*. (110-й — *C*.)

111-й будет *P*. Вы продолжаете чередование. (111-й — *P*.)

112-й будет *C*. Вы продолжаете чередование. (112-й — *P*.)

113-й будет *C*. Вы начали ряд из одного *C* и двух *P*. (113-й — *P*.)

114-й будет *P*. Вы начали ряд из *P*. (114-й — *P*.)

115-й будет *P*. Вы продолжаете этот ряд. (115-й — *C*.)

116-й будет *P*. Вы дали *C*, чтобы сбить меня. (116-й — *C*.)

117-й будет *C*. Вы начали ряд из четырех *C* и четырех *P*. (117-й — *P*.)

118-й будет *P*. Ряд из четырех *P* и четырех *C* заменяется на ряд из четырех *P* и трех *C*. (118-й — *C*.)

119-й будет *P*. Вы начали чередование *C* с *P*. (119-й — *C*.)

120-й будет *C*. Вы продолжаете ряд. (120-й — *P*.)

121-й будет *Р*. Теперь следует ряд из двух *С* и двух *Р*. (121-й — *Р*.)
122-й будет *С*. Вы продолжаете ряд из двух *Р* и двух *С*. (122-й — *С*.)
123-й будет *С*. Вы продолжаете ряд. (Какой?) Из двух *С* и двух *Р*. (123-й— *С*.)
124-й будет *С*. Вы начали ряд из *С*. (124-й — *С*.)
124-й (именно так сказано!) будет *С*. Вы продолжаете ряд. (125-й — *С*.)
126-й будет *С*. Вы продолжаете ряд *С*. (126-й — *Р*.)
127-й будет *С*. Вы дали *Р*, чтобы сбить меня. (127-й — *Р*.)
128-й будет *Р*. Вы начали ряд из *Р*. (128-й — *С*.)
129-й будет *С*. Вы начали ряд из двух *С* и двух *Р*. (129-й — *С*.)
130-й будет *С*. Вы начали ряд из *С*. (130-й — *Р*.)
131-й будет *Р*. Вы продолжаете ряд из двух *Р* и двух *С*. (131-й — *С*.)
132-й будет *Р*. Вы опять чередуете символы. (132-й — *С*.)
133-й будет *С*. Вы начали ряд из *С*. (133-й — *С*.)
134-й будет *С*. Вы продолжаете ряд. (134-й — *С*.)
135-й будет *С*. Вы продолжаете этот ряд. (135-й — *Р*.)
136-й будет *Р*. Вы начали... теперь вы пытаетесь сбить меня с толку вторым *Р*. Пожалуй будет только одно *Р*. (136-й — *С*.)
137-й будет *С*. Вы собираетесь продолжить ряд из одних *С*. (137-й — *С*.)
138-й будет *С*. Вы продолжаете ряд. (138-й — *С*.)
139-й будет *С*. Вы продолжаете ряд. (139-й — *Р*.)
140-й будет *С*. Вы дали *Р*, чтобы сбить меня. (140-й — *Р*.)
141-й будет *С*. Вы дали мне два *С* (именно так сказано!) по той же причине, что и в предыдущий раз, когда давали два *С* и два *Р*. (141-й — *С*.)
142-й будет *С*. Вы продолжаете ряд. (142-й — *С*.)
143-й будет *С*. Ряд продолжается. (143-й — *С*.)
144-й будет *С*. Ряд продолжается. (144-й — *С*.)
145-й будет *Р*. Вы нарушите ряд. (145-й — *С*.)
146-й будет *С*. Вы продолжите ряд. (146-й — *С*.)
147-й будет *С*. Вы продолжаете ряд. (147-й — *С*.)
148-й будет *С*. Ряд продолжается (148-й — *С*.).
149-й будет *С*. Ряд продолжается. (149-й — *С*.)
150-й будет *С*. Тот же ряд продолжается. (150-й — *С*.)
151-й будет *С*. Ряд продолжается. (151-й — *С*.)
152-й будет *Р*. Вы нарушите ряд. (152-й — *С*.)
153-й будет *С*. Вы продолжите ряд. (153-й — *Р*.)
154-й будет *С*. Вы нарушили ряд, но вернетесь к нему снова (154-й — *С*.)

155-й будет *С*. Вы продолжите ряд. (155-й — *Р*.)
156-й будет *С*. Вы чередуете *Р* и *С*. (156-й — *С*.)
157-й будет *С*. Чередование *Р* и *С* должно было сбить меня с толку и заставить отказаться от ряда *С*. Ряд *С* будет продолжаться. (157-й — *Р*.)
158-й будет *С*. Вы все же возвращаетесь к ряду *С*. (158-й—*С*).
159-й будет *С*. Вы собираетесь продолжить этот ряд. (159-й — *Р*.)
160-й будет *С*. Вы применяете чередующийся ряд из *Р* и *С*. (160-й — *С*.)
161-й будет *Р*. Вы продолжаете чередование. (161-й — *Р*.)
162-й будет *С*. Вы продолжаете чередование. (162-й — *Р*.)
163-й будет *С*. Вы продолжите чередование. (163-й — *С*.)
164-й будет *Р*. Вы продолжите чередование. (164-й — *С*.)
165-й будет *Р*. Вы вернетесь к чередованию. (165-й — *С*.)
166-й будет *С*. Вы начали ряд из *С*. (166-й — *С*.)
167-й будет *С*. Вы начали ряд из *С* (167-й — *Р*.)
168-й будет *Р*. Вы начали чередование двух *Р* и двух *С*. (168-й — *С*.)
169-й будет *С*. Предыдущие *Р* вы дали, чтобы сбить меня. Вы намерены продолжать ряд из *С*. (169-й — *С*.)
170-й будет *С*. Вы продолжите этот ряд. (170-й — *Р*.)
171-й будет *Р*. Вы начинаете ряд из *Р*. (171-й — *Р*.)
172-й будет *С*. Вы возвращаетесь к ряду из *С*. (172-й — *С*.)
173-й будет *С*. Вы чередуете два *Р* и два *С*. (173-й — *Р*.)
174-й будет *С*. Происходит чередование *С* и *Р*. (174-й — *С*.)
175-й будет *Р*. Вы продолжаете это чередование. (175-й — *С*.)
176-й будет *С*. Вы начали ряд из *С*. (176-й — *Р*.)
177-й будет *С*. Вы продолжите ряд из *С*. (177-й — *Р*.)
178-й будет *С*. Вы начали ряд из двух *Р* и двух *С*. (178-й—*С*.)
179-й будет *С*. Вы сейчас снова дадите *С*, чтобы завершить последовательность из двух *Р* и двух *С*. (179-й — *С*.)
180-й будет *Р*. Вы продолжите ряд. (180-й — *С*.)
181-й будет *С*. Вы начали ряд из *С*. (181-й — *С*.)
182-й будет *С*. Вы продолжите этот ряд. (182-й — *С*.)
183-й будет *С*. Вы продолжите ряд. (183-й — *Р*.)
184-й будет *С*. Вы дали *Р*, чтобы сбить меня. (184-й — *С*.)
185-й будет *С*. Вы продолжите ряд из *С*. (185-й — *С*.)
186-й будет *С*. Вы продолжите этот ряд. (186-й — *С*.)
187-й будет *С*. Ряд продолжается. (187-й — *С*.)
188-й будет *С*. Ряд продолжается (188-й — *С*.)
189-й будет *С*. Вы продолжите ряд. (189-й — *Р*.)
190-й будет *С*. Вы дали *Р*, чтобы сбить меня. (190-й — *С*.)
191-й будет *С*. Два *Р* (именно так сказано!) подряд было дано, чтобы еще более сбить меня. (191-й — *С*.)

192-й будет *С*. Вы... даете мне ряд из двух *Р* и двух *С*. (192-й — *С*.)
192-й (именно так сказано!) будет *Р*. Вы продолжаете ряд из двух *Р* и двух *С*. (193-й — *С*.)
194-й будет *С*. Вы начали ряд из *С*. (194-й — *С*.)
195-й будет *С*. Вы продолжаете ряд. (195-й — *Р*.)
196-й будет *Р*. Здесь вы вводите два *Р*. (196-й — *С*.)
197-й будет *С*. Вы дали *Р*, чтобы сбить меня. (197-й — *С*.)
198-й будет *С*. Вы продолжите ряд. (198-й — *С*.)
199-й будет *С*. Вы продолжаете ряд. (199-й — *С*.)
200-й будет *С*. Вы продолжаете ряд. (200-й — *С*.)

15. МОДЕЛЬ ПРОЦЕССА ДОВЕРИТЕЛЬНОЙ ИНВЕСТИЦИИ

Дж. Кларксон

Введение

Предмет настоящего исследования — размещение кредитного капитала, управляемого по доверенности владельцев банками США. В настоящее время размер этого капитала составляет около 60 млрд. долл. Мы хотим промоделировать процесс инвестиции доверенного капитала в обыкновенные акции[1]. Банковский работник, принимающий решение об инвестиции, сталкивается с большим количеством разнообразной информации. Эта информация в изобилии содержит сведения о деятельности фирм и рыночной стоимости их акций; кроме того, публикуются сообщения, в которых делаются предсказания относительно будущего состояния экономики и фондовой биржи. Когда инвестор действует в качестве посредника или доверенного лица, он должен принимать во внимание также установленные законом ограничения и учитывать пожелания клиента. Все эти факторы оцениваются и на их основе вырабатывается план инвестирования, который в конечном счете выражается в решении о покупке определенного числа таких-то акций и облигаций. Таким образом, инвестор, выбирающий портфель ценных бумаг, перерабатывает некоторую информацию; он отделяет полезное от несущественного и решает, что из общего потока информации является наиболее важным.

Проблему инвестирования капитала по доверенности можно отнести к задачам принятия решения в условиях недостаточной

[1] То есть акции с нефиксированным размером дивидендов. — *Прим. перев.*

информации. Наша модель, представляющая собой программу для цифровой вычислительной машины, имитирует действия инвестора при выборе политики инвестиции для тех или иных капиталов, оценке возможностей, предоставляемых биржой, и выборе требуемого портфеля. Изучение этих операций проводилось в национальном банке средних размеров[1], и наша модель — это модель служащего, осуществляющего инвестицию по доверенности владельца капитала[2]. Мы требуем, чтобы модель выбирала портфель, пользуясь той же самой информацией, какой располагает этот служащий при принятии решений.

Постулаты и исходные данные модели

Поскольку наша модель имитирует поведение отдельного человека при принятии решения, метод анализа основан на теории принятия решений человеком [791]. В соответствии с основными постулатами этой теории предполагается, что процесс инвестиции обладает следующими свойствами:

1. Имеется *память*, в которой хранятся списки отраслей промышленности, причем для каждой отрасли имеется список связанных с ней компаний. Кроме того, в памяти хранится информация относительно экономики в целом, отдельных отраслей и компаний[3].

2. Имеются методы *поиска* и *отбора* нужной информации. С помощью этих методов осуществляются поиск списков в памяти, отбор данных по требуемым признакам, перегруппировка отобранных данных в новые списки и в случае необходимости выполнение арифметических операций. Все эти операции аналогичны действиям служащего, который подготавливает списки подходящих для инвестиции акций, просматривая общий список.

3. Имеется *набор инструкций* или критериев, управляющих

[1] Размер доверенного капитала этого банка приблизительно равен среднему по всем национальным банкам

[2] Следует отметить, что модель отражает поведение одного инвестора и, следовательно, не описывает общего случая. Применение этих исследований в более общей теории инвестиции обсуждается в работе [228].

[3] Инвесторы классифицируют компании по отраслям Разные инвесторы производят такую классификацию по-разному, но сам принцип классификации по отраслям неизменно соблюдается при размещении компаний в памяти. Информация относительно каждой компании также может быть различной у разных инвесторов, но записывается она однотипно: в виде списка параметров, каждый из которых характеризуется некоторым числом, например скорость прироста капитала, размер дивидендов, отношение стоимости акций к прибыли, ожидаемая прибыль, ожидаемый доход и т п

процессом принятия решения. Эти инструкции указывают, когда и как должно применяться каждое решающее правило. Набор инструкций определяет структуру процесса принятия решения некоторым инвестором. Этот набор аналогичен эмпирическим правилам, которыми пользуются опытные специалисты. Но имеется и существенное различие, состоящее в том, что в нашем случае набор инструкций должен однозначно определять действия инвестора.

В нашей программе, как и в других программах для решения сложных задач, обработка данных производится путем последовательной обработки исходной информации. По заданным признакам ищутся списки отраслей и компаний; составляются подсписки, которые исследуются и вновь разделяются. Пусть, например, нужно получить высокий прирост вложенного капитала. Для этого исследуется хранящийся в памяти список компаний и составляется список ценных бумаг, обладающих желаемыми характеристиками. Этот список можно сократить (или расширить), привлекая дополнительные критерии. Дальнейший поиск и проверка по желаемому критерию приводят к решению о покупке определенных акций.

Подобно моделируемому ею инвестору программа запоминает окончательный результат (список). Поэтому если та же задача впоследствии встретится вновь, уже не будет необходимости полностью повторять весь процесс. В этом случае список проверяется по заданным критериям, после чего он принимается, приспосабливается к новым условиям или полностью отвергается. В последнем случае программа заново начинает поиск и отбор для создания нового списка.

Чтобы перейти от этой общей схемы к конкретной модели, необходимо задать основные правила (операции), которые используются в процессе принятия решения о покупке определенных ценных бумаг. С этой целью в нескольких местных банках изучалась работа отделов, занимающихся доверительной инвестицией. Изучение проводилось путем опроса служащих и наблюдения за работой опекунских советов, обсуждающих ранее вынесенные и предстоящие решения. Впоследствии внимание было сосредоточено на работе служащего одного из банков, несущего основную ответственность за все решения, касающиеся выбора портфелей внутри данного банка. После тщательного изучения нескольких расчетов по выбору портфеля была построена грубая модель поведения этого банковского служащего. Эта модель должна была помочь обнаружить те решающие правила, которые использовались во всех расчетах.

Чтобы подтвердить или опровергнуть полученные гипотезы, с согласия служащего его действия в процессе принятия решения фиксировались в «протоколах»[1]. Эти протоколы регистрировали весь процесс принятия решений служащим в ходе его текущей работы. В соответствии с этими решениями затем были действительно приобретены ценные бумаги.

В результате подробного изучения протоколов было обнаружено, что многие решения, связанные с формулированием прогнозов, а также с оценкой текущего состояния отраслей, компаний и акций, были приняты еще до того, как начался непосредственно выбор портфеля. Чтобы выяснить, каким образом были приняты эти предварительные решения, использовался следующий метод. Служащему предложили читать вслух статьи из выписываемых им журналов и финансовых отчетов и комментировать встречающиеся по пути мысли, прогнозы, факты и т. д. Протоколы этих мыслительных процессов оказались очень полезными, поскольку они позволили обнаружить много решающих правил, которые не удалось выявить из ранее сделанных записей.

На основе всех этих данных была построена модель, которая решает задачу вложения вновь поступающих капиталов в обыкновенные акции. Эта модель в том виде, в каком она здесь описана, не отражает процесса распределения капитала между облигациями, привилегированными[2] и обыкновенными акциями. Реализация модели на цифровой вычислительной машине описана в следующем разделе[3].

Процесс доверенной инвестиции можно разбить на три отдельных этапа: а) анализ и составление списка подходящих для данного случая акций (А-список), б) формулирование политики инвестиции и в) выбор портфеля. Каждый из этих этапов в свою очередь может быть разбит на несколько более мелких частей (фиг. 1).

Процесс отбора акций для А-списка (этап «а») требует анализа отдельных компаний, а также оценки факторов, влияющих на соответствующие отрасли и экономику в целом. Задача формулирования политики инвестиций (этап «б») состоит в том, чтобы из информации о клиенте определить цель инвестиции:

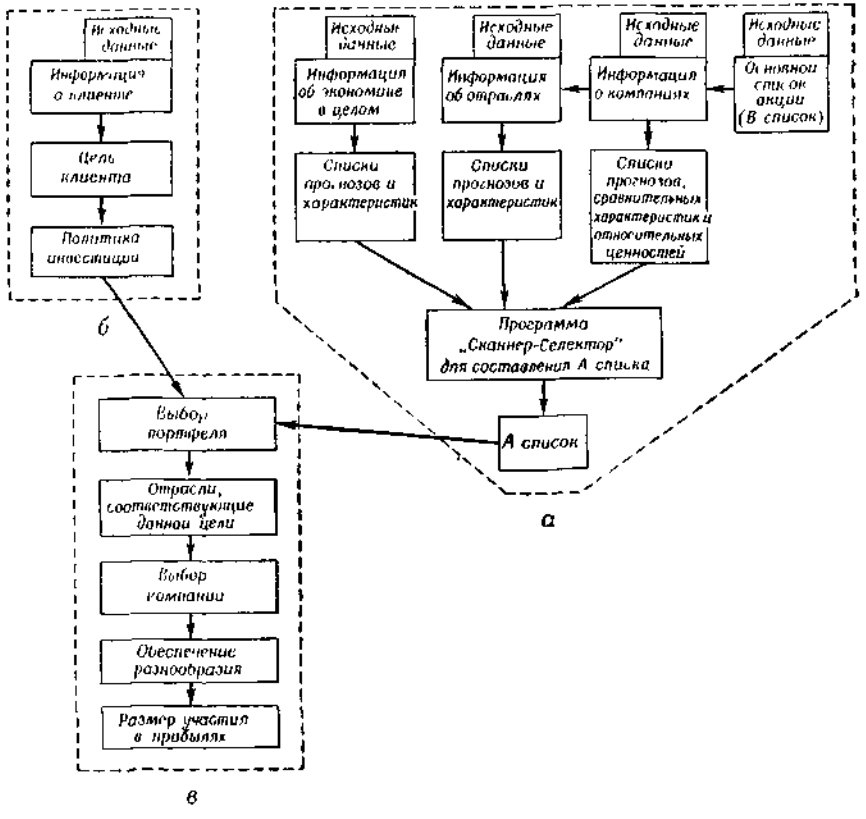
---

[1] «Протоколом» называется дословная запись мыслей и действий субъекта, думающего или решающего задачу вслух. Эта методика дает нам подробное описание того, *что* делает человек, и, следовательно, позволяет избежать трудностей, присущих методике бесед и заполнения анкет, в которых человека просят объяснить, *почему* он производит то или иное действие. Более подробно этот вопрос обсуждается в работе [791]

[2] Привилегированные акции — акции с фиксированным дивидендом, который выплачивается до выплаты дивидендов по обыкновенным акциям. — *Прим. перев.*

[3] Программа написана на языке IPL-V ([805]; см. стр. 458 настоящей книги).

желаемое сочетание дохода и прироста капитала. Этот процесс требует от производящего инвестицию служащего учета таких обстоятельств, как влияние налогов на доход, даваемый выбранным портфелем, а также стабильность этого дохода. Фактический выбор портфеля (этап «в») непосредственно следует из



Фиг 1 Структура процесса принятия решения (см. текст).

этапов «а» и «б». Хотя в процессе выбора используются правила разносторонней инвестиции и определяется, сколько акций каждой компании войдет в портфель, все это фактически является лишь логическим завершением предварительного анализа.

При описании нашей модели мы будем следовать основной схеме, представленной на фиг. 1. После того как мы вкратце

изложили основные элементы процесса инвестиции и метод анализа, применяемый при построении модели, нам перед непосредственным переходом к описанию модели осталось лишь рассмотреть организационные и юридические ограничения, накладываемые на процесс доверенной инвестиции [1].

Так как банки ответственны за все вложения, которые делаются от их имени, существуют тщательно разработанные процедуры для рассмотрения и утверждения всех решений об инвестиции. Кроме того, каждый инвестор должен иметь возможность доказать в суде правильность своих решений. Это заставило разработать ряд критериев, позволяющих обосновать качество выбранного портфеля. Все эти критерии практически сводятся к одному принципу: качество инвестиции гарантируется *в том и только в том* случае, если предлагаемые ценные бумаги покупаются или имеются во владении других ведущих учреждений, управляющих доверенным капиталом. Очевидно, что точное соблюдение этого принципа сделало бы невозможными никакие изменения. На практике более мелкие банки придерживаются правила, согласно которому все нововведения должны исходить только из крупных банков, действующих самостоятельно или небольшими группами. Если нововведения появляются не очень часто, общий список акций, которые считаются подходящими для инвестиции согласно этому принципу, почти не меняется со временем. Еще более точное предсказание возможно, если учесть то обстоятельство, что инвесторы стремятся избежать убытков, избавляясь от акций, цена которых упала ниже покупной цены. Основной список акций (В-список), которые в данном банке считаются пригодными для доверительной инвестиции, на протяжении долгого времени остается почти постоянным, а любое изменение этого списка состоит лишь в добавлении акций. Инвестор каждый раз производит выбор из некоторого подмножества акций основного списка. Это подмножество, называемое А-списком, определяется с помощью понятия об относительной ценности акций. Поскольку параметры ценных бумаг меняются с течением времени, содержание А-списка также меняется со временем.

Составление А-списка

В этом разделе мы опишем информацию и механизмы, которые в нашей модели используются для отбора акций при со-

[1] Поскольку мы рассматриваем преимущественно инвестицию капиталов, лежащих на индивидуальных счетах, для нас важны те ограничения, которые накладываются на действия доверенного инвестора банковским учреждением и доверительными отношениями с клиентом.

ставлении А-списка. В отличие от этапов «б» и «в» эти механизмы не призваны копировать методы анализа, используемые банковским служащим при выборе нового портфеля, ибо при этом мы должны были бы игнорировать всю ту информацию о каждой из компаний, которую он собрал и обработал за прошедшие годы. Совершенно очевидно, что инвестор (в отличие от модели) не оценивает все компании одновременно. Поэтому описываемую в этом разделе часть программы следует рассматривать не как точную модель того, что делает инвестор перед каждым выбором портфеля, а скорее как приблизительную модель методов, которые он использовал в течение ряда лет, накапливая материалы, характеризующие деятельность компаний. Насколько мы преуспели в этом — будет видно из дальнейшего.

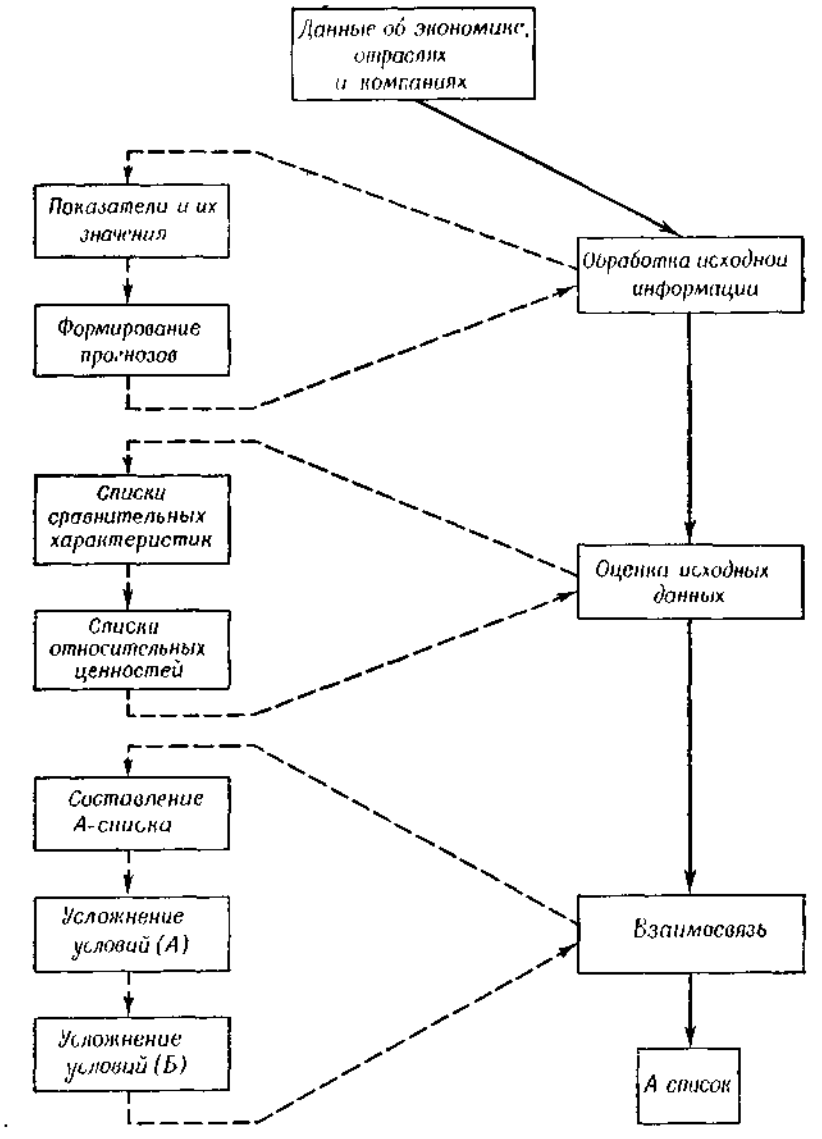
Для описания программы составления А-списка нам потребуется сначала описать работу отдельных блоков, как если бы они были независимыми. После этого мы покажем, каким образом они взаимодействуют в программе. Блок-схема программы представлена на фиг. 2.

А. Обработка исходной информации

Хотя исходная информация подразделяется на три основных вида (экономика в целом, отрасль и компания), обработка каждого вида информации производится приблизительно одинаково.

Каждый вид информации характеризуется рядом важных параметров. Например, компании имеют следующий набор показателей товарооборот, прибыль, приток наличных денег в расчете на 1 акцию, маржа [1], оборотный капитал, отношение стоимости акции к прибыли, отношение дивиденда к прибыли, размер дивиденда в расчете на 1 акцию, суммарный дивиденд, цены. Значение каждого показателя определяется его численной величиной. Это и есть та информация, которая вводится в модель Все переменные величины можно разделить на часто меняющиеся и редко меняющиеся. Последние отражают общую тенденцию развития экономики, отрасли или компании, в то время как первые характеризуют параметры, более чувствительные к кратковременным флуктуациям. Эта классификация осуществляется в программе одними и теми же способами для всех показателей.

[1] Разность между указываемыми в биржевом бюллетене ценами товара, устанавливаемыми продавцом и покупателем — *Прим ред*



Фиг. 2 Составление А списка

а) Определение показателей и их численных величин. Вся информация, за исключением прогнозов состояния экономики и отдельных отраслей, вводится в модель в численной форме и включает данные по каждому показателю за последние 10 лет [1]. Эти данные вводятся в виде списков, причем из каждого списка, соответствующего какому-либо показателю, модель образует еще три списка. Первый из них содержит среднее значение показателя за 10 лет, второй — относительные приращения за каждый год, а третий — среднее относительное приращение за 10 лет. Таким образом, в памяти модели хранятся следующие четыре списка информации о каждом показателе:

1) *Текущее значение.* Список содержит численные значения показателя, наблюдавшиеся за последние 10 лет. Эти значения располагаются в хронологическом порядке (новые значения помещаются в начале списка).

2) *Среднее за 10 лет.* Вычисляется каждый раз, когда в список 1 вводится новое число. Таким образом, этот список содержит скользящее среднее за 10 лет значений списка 1.

3) *Относительное приращение.* Этот список содержит относительное приращение (или уменьшение) каждого значения в списке 1 по сравнению с непосредственно следующей за ним величиной. Так, если величины из списка 1 обозначить через x_i, где $i=1, 2, ..., 10$, то список относительных приращений будет состоять из величин

$$\frac{x_i - x_{i+1}}{x_{i+1}},$$

где $i=1, 2, ..., 9$.

4) *Среднее относительное приращение.* Этот список содержит среднее относительное приращение величин из списка 1 за 10 лет. Подобно списку 2, это число меняется при каждом изменении списка 1.

б) Формирование прогнозов. Информация, касающаяся прогнозов, вводится в модель в двух формах. Прогнозы относительно экономики и отдельных отраслей оцениваются по трехзначной шкале: «выше», «ниже» или «равно». Это делается на основании опубликованных предсказаний относительно величины каждого показателя на ближайший отрезок времени. Численные данные здесь не используются во избежание хаоса при попытке усреднения множества предсказаний, появляющихся в финансовой литературе.

[1] Сами показатели считаются заданными. Они были выбраны в процессе анализа действий инвестора при принятии решения и наблюдения за работой службы инвестиции.

Однако для анализа характеристик компаний необходимо иметь численные предсказания, поэтому во избежание столкновения противоречивых мнений все предсказания, касающиеся показателей компаний, брались из журнала "Value Line Investment Survey".

Каждый предсказываемый показатель характеризуется только одной величиной: последним из предсказанных значений. Предыдущие предсказания не сохраняются. Модель не пытается также оценивать «надежность» предсказаний или как-то регистрировать их успешность. В действительности каждый инвестор отдает предпочтение тем или иным предсказаниям, основываясь на субъективных оценках. Наша модель не обладает способностью «угадывать», и вся информация считается достоверной. Но прежде чем перейти к обсуждению роли прогнозов в нашей модели, необходимо упомянуть о некоторых особенностях деятельности инвесторов.

Вообще говоря, доверительная инвестиция — это долгосрочная инвестиция. Как уже указывалось, инвесторы, действующие по доверенности, не занимаются обменом акций своих клиентов, а следят за тенденциями долгосрочных изменений в экономике и на фондовой бирже, чтобы иметь возможность обосновывать свои решения. Это не означает, что они не обращают внимания на ежедневные, ежемесячные и ежегодные флуктуации, но для их деятельности наибольшее значение имеет анализ состояния отраслей и компаний. Их кредо состоит в том, что в конечном счете биржа всегда придаст акциям любой компании их «истинную» цену. Поэтому инвесторы, как правило, анализируют деятельность компаний, а не положение на бирже.

Очевидно, что изменения на фондовой бирже влияют на поведение инвестора, но их эффект скорее подобен действию обратной связи, так что его собственные предсказания положения на бирже не имеют большого значения для его деятельности. Таким образом, модель включает прогнозы значений показателей, но разным показателям уделяется различное внимание в зависимости от их принадлежности к общей экономике, отдельным отраслям или компаниям. Ниже описаны два типа списков прогнозов, имеющихся в модели:

1) *Списки прогнозов по экономике и отраслям.* Для каждого показателя, характеризующего обе эти категории, в список заносятся две величины. Первая — это предсказанное значение показателя по трехзначной шкале: «выше», «ниже» или «равно». Вторая — это число, стоящее первым в списке относительных приращений, т. е. относительное приращение за последний год, записанное в той же трехзначной шкале.

2) *Списки прогнозов по компаниям.* Эти списки составляются для пяти показателей[1] и содержат по одному или по два значения, выраженных числами. Числа, стоящие первыми, представляют собой прогнозы на год вперед, записанные в форме относительного приращения (иначе говоря, ожидаемое относительное приращение). Вторые числа имеются только в списках, соответствующих показателям «товарооборот» и «прибыль на 1 акцию». Это прогнозы на срок от 3 до 5 лет, записанные как ожидаемое среднее относительное приращение.

Б. Оценка исходных данных

В этом разделе следовало бы описать все применяемые в модели способы оценки данных. Однако мы для простоты разбили их на две группы. Способы, с помощью которых обрабатывается информация внутри каждой категории, т. е. экономики в целом, отрасли или компании, будут рассмотрены здесь; те же способы, в которых привлекаются связи между этими категориями, рассматриваются в следующем разделе.

Для оценки данных модель составляет два основных списка: *список сравнительных характеристик* и *список относительных ценностей*. Но, прежде чем подробнее остановиться на этих процессах, напомним еще раз, какого вида информация уже имеется относительно каждого показателя каждой компании:

1) Список значений показателя за последние 10 лет.

2) Среднее за эти 10 лет.

3) Список относительных приращений.

4) Среднее относительное приращение.

5) Список прогнозов (не для всех показателей); этот список содержит ожидаемое в предстоящем году относительное приращение, а для показателей «товарооборот» и «прибыль на 1 акцию» — ожидаемое в последующие 3—5 лет среднее относительное приращение.

Мы еще раз перечислили эти виды информации, поскольку механизмы оценки используют их в качестве входных данных.

а) Список сравнительных характеристик. Чтобы определить сравнительные характеристики каждой компании внутри данной отрасли, для *каждого показателя* составляется список средних значений этого показателя для *каждой компании*.

[1] Товарооборот, прибыль на 1 акцию, приток наличных денег на 1 акцию, маржа и дивиденд на 1 акцию.

Обозначим через a_{ij} среднее значение i-го показателя ($i=1, 2, \ldots, n$) j-й компании ($j=1, 2, \ldots, m$). Тогда матрица

$$A_{n,m}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

представляет собой упорядоченную совокупность средних значений показателей каждой компании в пределах данной отрасли. Усредняя по компаниям, найдем среднее значение i-го показателя по данной отрасли:

$$\bar{a}_i=\frac{1}{m}\sum_{j=1}^{m} a_{ij}.$$

Совокупность всех таких средних образует вектор

$$\bar{A}_n=\begin{bmatrix} \bar{a}_1 \\ \bar{a}_2 \\ \cdot \\ \cdot \\ \cdot \\ \bar{a}_n \end{bmatrix}.$$

Процентное отклонение каждого из a_{ij} от соответствующего среднего значения определяется как

$$\frac{a_{ij}-\bar{a}_i}{\bar{a}_i}\cdot 100\%.$$

Совокупность всех таких отклонений представляется матрицей

$$\begin{bmatrix} \frac{a_{11}-\bar{a}_1}{\bar{a}_1} & \frac{a_{12}-\bar{a}_1}{\bar{a}_1} & \cdots & \frac{a_{1m}-\bar{a}_1}{\bar{a}_1} \\ \frac{a_{21}-\bar{a}_2}{\bar{a}_2} & \frac{a_{22}-\bar{a}_2}{\bar{a}_2} & \cdots & \frac{a_{2m}-\bar{a}_2}{\bar{a}_2} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{a_{n1}-\bar{a}_n}{\bar{a}_n} & \frac{a_{n2}-\bar{a}_n}{\bar{a}_n} & \cdots & \frac{a_{nm}-\bar{a}_n}{\bar{a}_n} \end{bmatrix}.$$

Эти процентные отклонения оцениваются затем по трехзначной шкале: «выше», «ниже» и «равно». Оценки «выше» и «ниже» выносятся в том случае, когда величина отклонения от среднего $\bar{a}_i$ в ту или иную сторону превышает 5%. Аналогичные операции производятся со списками 4 и 5. Таким образом, для каждого показателя каждой компании составляется список сравнительных характеристик, содержащий:

(а) Среднее значение этого показателя за последние 10 лет и оценку этой величины («выше», «ниже» или «равно») по отношению к среднему по всем компаниям данной отрасли.

(б) Среднее относительное приращение за последние 10 лет и его оценку по отношению к среднему по отрасли.

(в) Ожидаемое относительное приращение на предстоящие 12 месяцев и его оценку по отношению к среднему по отрасли.

(г) Ожидаемое среднее относительное приращение на следующие 3—5 лет и его оценку по отношению к среднему по отрасли[1].

б) Список относительных ценностей. Выше было показано, как определяются сравнительные характеристики компаний в пределах каждой отрасли. Покажем теперь, как определяется относительная ценность акций каждой компании.

Выше говорилось, что каждая компания имеет показатель «прибыль в расчете на 1 акцию», относительно которого делается прогноз на 3—5 лет вперед. Хотя это лишь оценочные данные, они считаются надежными, и на основании этих цифр для каждой компании находится отношение цены акций к предполагаемой прибыли. Поскольку модель уже содержит текущие значения этого отношения и его среднее значение за прошлые годы для данной компании, список относительной ценности составляется следующим образом. Во-первых, для каждой компании находится разность между средним отношением цены акции к прибыли и отношением цены акции к предполагаемой прибыли. Эта разность выражается в процентах от среднего отношения цены к прибыли и оценивается по трехзначной шкале: «выше», «ниже» или «равно». Во-вторых, находится разность между средним и текущим отношениями цены акции к прибыли. Эта разность также выражается в процентах от среднего и оценивается по трехзначной шкале. Все эти операции очень просто записываются в буквенных обозначениях. Пусть

[1] Во всех этих списках содержатся величины (а) и (б). Списки для показателей «приток наличных денег на 1 акцию» и «маржа» содержат (а), (б) и (в). И только списки для показателей «товарооборот», «прибыль на 1 акцию» и «дивиденд на 1 акцию» содержат величины всех четырех типов.

P — текущая рыночная цена акции; E — ожидаемая в текущем году прибыль в расчете на 1 акцию; E^1 — предполагаемая прибыль на 1 акцию через 3—5 лет; $\bar{P}$ — средняя рыночная цена акции за 10 лет. Тогда для каждой компании вычисляются следующие величины:

$$\frac{\frac{\bar{P}}{E}-\frac{P}{E^1}}{\frac{\bar{P}}{E}} \quad (1) \quad \text{и} \quad \frac{\frac{\bar{P}}{E}-\frac{P}{E}}{\frac{\bar{P}}{E}}. \quad (2)$$

Список относительных ценностей для каждой компании содержит эти числа вместе с их оценками по трехзначной шкале.

В. Взаимосвязь процессов обработки данных

До сих пор мы описывали процессы обработки данных так, как будто они ведутся независимо друг от друга. Хотя это до некоторой степени верно, в ходе составления списка подходящих акций все эти процессы связаны между собой. Чтобы сделать эту связь возможно более понятной, мы начнем с рассмотрения процесса составления А-списка при упрощенных условиях, а затем рассмотрим более сложные случаи.

Для упрощения дальнейшего изложения введем следующие обозначения:

$$x=\frac{\bar{P}}{E}-\frac{P}{E^1}, \quad y=\frac{\bar{P}}{E}-\frac{P}{E}.$$

а) Составление А-списка. Для простоты предположим сначала, что в списках прогнозов для экономики в целом и для отдельных отраслей все величины принимают значение «выше». Такое положение возможно лишь в том случае, если экономика переживает период оживленного подъема. Посмотрим, как в этом случае действует алгоритм составления А-списка. Этот алгоритм состоит из двух программ:

1) *Сканнер*[1]. Программа просматривает по очереди все списки прогнозов по экономике и по отраслям, обращая внимание на пары соседних величин (каждая пара соответствует какому-то показателю). В данном случае все пары имеют одно и то же значение — «выше». Закончив просмотр и обнаружив столь

[1] От англ. scan — сканировать, внимательно просматривать. — *Прим. ред.*

полную гармонию, Сканнер останавливается и передает управление Селектору.

2) *Селектор.* В этих идеальных условиях процесс селекции, т. е. отбора, заключается в просмотре списков относительных ценностей для всех компаний и занесении в А-список тех компаний, для которых

(x) = «выше» или «равно»;

(y) = «выше», «равно» или «ниже».

б) Усложнение условий А. Необходимо все время помнить, что информация, относящаяся к разным категориям (т. е. информация об экономике, отраслях и компаниях), поступает через разные интервалы времени. Эти интервалы можно выбрать, исходя из любых требований, но в нашей модели мы считаем, что информация об экономике и отраслях поступает ежеквартально, а информация о компаниях — ежемесячно.

Покажем теперь, как модель обрабатывает вновь поступающую информацию.

1) Как только происходит изменение цен или прибыли на 1 акцию, новая информация обрабатывается, как показано в разд. А и Б, и в список относительных ценностей заносятся новые величины. После этого Сканнер начинает проверять списки прогнозов по экономике и отраслям и, обнаружив, что они не изменились, включает Селектор. Селектор сначала просматривает А-список и удаляет из него те компании, у которых в списках относительной ценности появились величины

(x) = «ниже».

Затем Селектор заносит в А-список все те компании, у которых в списках относительной ценности появились новые величины

(x) = «выше» или «равно».

2) В конце каждого квартала в модель вводится новая информация о показателях экономики и отдельных отраслей, а также, когда это требуется, и о показателях компаний. Как только новая информация поступила, она обрабатывается согласно разд. А и Б, и внимание Сканнера обращается на ту категорию, которая получила новую информацию. Если новую информацию получает не одна категория, то Сканнер начинает работу с более общей (т. е. с экономики в целом или с отрасли), отмечая и записывая все изменения. На этом этапе все изменения в списках прогнозов по экономике и отраслям записываются в виде отметок «оставить» или «вычеркнуть». Эти отметки заносятся в список относительной ценности. Компании, которые до этого находились в А-списке, продолжают оставаться в нем до поступления новой информации о компаниях.

в) Усложнение условий Б. Чтобы показать все операции, выполняемые Сканнером и Селектором, разобьем изменения в списках прогнозов по экономике и отраслям на три типа.

1) *Предсказываемое относительное приращение оказывается ниже последнего относительного приращения.* Как уже указывалось, в функцию Сканнера входит исследование списков прогнозов по экономике и отраслям для тех показателей, относительно которых получена новая информация. Предположим, что модель получила прогноз о прекращении роста капиталовложений, в то время как по самым последним данным наблюдался рост этого показателя. Получив эту информацию, Сканнер начнет просмотр со списка прогнозов по капиталовложениям. Заметив, что остальные списки экономических прогнозов остались без изменения, Сканнер перейдет на следующий более низкий уровень и составит список отраслей, характеризующихся интенсивными капитальными затратами. Затем Сканнер рассмотрит изменения в списках отраслевых прогнозов. Прогнозы для некоторых из этих отраслей также покажут понижение или прекращение роста капитальных затрат, и мы таким образом получим список отраслей, которых коснулись эти изменения в экономике[1].

После этого начинает работать Селектор. Он просматривает список, составленный Сканнером, и ищет в А-списке компании, принадлежащие к этим отраслям. Все такие компании подвергаются следующей проверке:

(a) Метка «оставить» ставится против тех компаний, которые в списке относительной стоимости имеют (x) = «равно».

Если прогнозы относительно других экономических показателей указывают на понижение, Сканнер ищет соответствующие изменения во всех списках отраслевых прогнозов, составляет список отраслей, для которых предполагается понижение, и представляет этот список Селектору. Селектор производит затем серию проверок (a).

2) *Последнее относительное приращение оказывается ниже предсказываемого.* В этом случае Сканнер и Селектор действуют по существу так же, как в случае (1), за исключением того, что Селектор производит одну дополнительную проверку.

Если экономические показатели снизились, характеристики некоторых отраслей и компаний также должны ухудшиться. Это

[1] Предполагается, что прогнозы относительно общих капитальных затрат не могут изменяться без соответствующих изменений в прогнозах относительно одной или всех отраслей с большим капиталом. Единственное исключение из этого правила — строительство, которое также включается в список проверяемых отраслей в случае снижения ожидаемого уровня капиталовложений.

означает, что в оценках компаний в этот момент могут произойти значительные изменения. Но, так как эти изменения происходят раньше, чем изменение экономических показателей, в функцию Сканнера по-прежнему входит составление списка затронутых отраслей, а Селектор производит над компаниями из А-списка, которые принадлежат этим отраслям, следующие проверки:

(α) Меткой «оставить» помечаются те компании, которые в списке относительных ценностей имеют (x) = «равно».

(β) Меткой «оставить» помечаются те компании, которые в списке относительных ценностей имеют (y) = «ниже».

3) *Предсказанное и последнее относительные приращения падают.* Действия Сканнера остаются без изменений. Селектор в этом случае призводит более строгую проверку:

(α) Из А-списка удаляются все компании, которые в списке относительной ценности имеют (x) = «равно».

(β) Меткой «оставить» помечаются те компании, которые в списке относительной ценности имеют (y) = «ниже».

Очевидно, что эти три типа изменений в списках прогнозов не являются взаимно исключающими, и в любой момент информация, вводимая в модель, может представлять собой некоторую комбинацию этих типов. Это никоим образом не меняет функций Сканнера и Селектора. От них лишь требуется рассматривать изменения каждого типа по очереди и последовательно выполнять все требуемые операции.

В том случае когда предсказываемые и последние относительные приращения не уменьшаются, а увеличиваются, все действия Селектора заменяются противоположными. Метка «оставить» заменяется меткой «вычеркнуть», а вместо удаления компаний из А-списка они заносятся в него.

Определение политики инвестиций

Вообще говоря, инвесторы определяют политику инвестиций для капиталов двух типов: 1) большие капиталы, например доверенные общественные фонды (кроме пенсионного и других подобных фондов), и 2) индивидуальные доверенные капиталы, т. е. капиталы, находящиеся на лицевых счетах отдельных лиц.

Так как мы интересуемся главным образом инвестициями последнего типа, в модели не рассматривается проблема инвестиций общественных фондов. Однако инвестор должен каждый раз решать вопрос, производить ли для инвестиции данного капитала отдельный расчет или включить его в один из фондов. Хотя наша модель непосредственно не отражает этого процесса, мы проведем здесь его краткое обсуждение.

А. Доверенные общественные фонды

Расходы на содержание общественных фондов (в расчете на 1 вложенный доллар) намного ниже, чем на содержание индивидуальных капиталов. Поэтому банки предпочитают вкладывать небольшие индивидуальные капиталы в общественные фонды. Чтобы побудить клиентов участвовать в этих фондах, банки вынуждены четко определять цели этих фондов. На практике эти цели могут значительно варьировать. В одних случаях основной упор делается на увеличение вложенного капитала, в других — на текущий доход (при неизменности основного капитала). Между этими крайними случаями имеется целый ряд промежуточных.

Поскольку юридические ограничения, накладываемые на инвестиции в общественные фонды, уже обсуждались в другом месте [228], мы здесь приведем лишь основные правила, используемые при решении вопроса о подключении индивидуальных капиталов к одному из этих фондов.

а) Все индивидуальные капиталы, управляемые по «узаконенной»[1] доверенности, годятся для присоединения к общественным фондам. «Неузаконенные» нельзя присоединять к этим фондам.

б) Все капиталы размером меньше K долл. с узаконенной доверенностью автоматически присоединяются к общественному фонду.

в) Капитал свыше K долл. с узаконенной доверенностью можно как поместить, так и не поместить в общественный фонд.

[1] Постановления об «узаконенной инвестиции» делятся на две группы: 1) те, которые разрешают производить только определенные инвестиции или инвестиции определенных типов, и 2) те, которые разрешают вкладывать в «неузаконенные» бумаги лишь определенный процент инвестируемого капитала. В шт. Пенсильвания действуют постановления первого типа. Узаконенными являются ценные бумаги тех компаний, которые по привилегированным акциям выплачивали дивиденды в течение 16 лет, а обыкновенные акции которых были прибыльными и приносили дивиденды в течение 12 из 16 последних лет. Список ценных бумаг, удовлетворяющих этим требованиям, составляется Пенсильванской ассоциацией банкиров.

Однако многие при формулировании доверенности отказываются от этих ограничений. Поэтому слово «узаконенный» относится к ситуациям, в которых служащий, производящий инвестицию, обязан подчиняться установленным ограничениям.

Индивидуальный вклад не может превышать 100 000 долл. Таким образом, в диапазоне от *K* до 100 000 долл. решение принимается исходя из степени соответствия между целями индивидуального вклада и общественного фонда.

Б. Индивидуальные доверенные капиталы

Доверенный инвестор имеет два основных источника информации о целях клиента: беседа административного служащего с клиентом и письменный документ. В первом случае инвестор получает какое-то субъективное впечатление о клиенте, во втором он имеет копию юридического документа (часто это завещание), подтверждающего доверенность. В большинстве случаев в этом документе содержится информация о клиенте, о полномочиях банка в отношении инвестиции, о том, как следует поступить с основным капиталом, о желаемой сумме дохода и т. п. Документ содержит также информацию о возрасте клиента, его семейном положении, количестве и возрасте лиц, находящихся на его иждивении, постоянном месте жительства, суммах дохода и налога, положении и возрасте будущих владельцев, если таковые имеются.

Вооруженный этими сведениями доверенный инвестор должен выбрать политику инвестиции данного капитала. Эта политика находится где-то между крайними политиками прироста и дохода и определяется следующим образом.

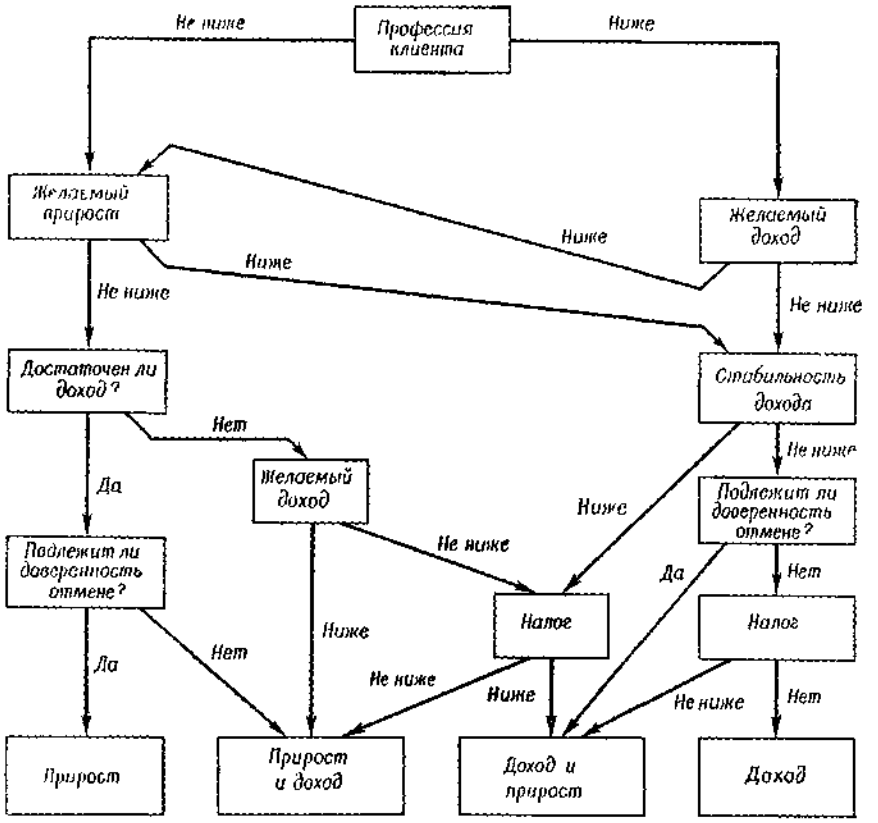
1) *Сканнер.* Информация о клиенте поступает в модель в виде списка, содержащего следующие показатели: (1) желаемая сумма прироста, (2) желаемая сумма дохода, (3) достаточность имеющегося дохода для нужд клиента, (4) желаемая стабильность дохода и основного капитала, (5) суммы дохода и налога, (6) профессия клиента, (7) постоянное место жительства клиента, (8) подлежит ли в дальнейшем доверенность отмене и (9) является ли доверенность «узаконенной». В функцию Сканнера входит просмотр первых семи показателей и их оценка[1]. Сканнер сравнивает значение каждого показателя с некоторой средней величиной (эти величины заранее задаются) и дает оценку «ниже» или «не ниже». Показателям (3) и (7)—(9) присваиваются значения «да» или «нет».

Эти оценки помещаются в список. Так, для клиента, который имеет юридическую профессию, постоянно проживает в шт. Пенсильвания, имеет высокий доход, платит большой налог и хочет укрепить свое материальное положение перед тем, как уйти в отставку, Сканнер выдаст следующий набор оценок:

[1] Показатели (8) и (9) используются непосредственно при выборе порт-

(1) «не ниже», (2) «ниже», (3) «да», (4) «ниже», (5) «не ниже», (6) «не ниже», (7) «да».

2) *Селектор.* Задача Селектора — на основании выданного Сканнером списка определить подходящую политику инвестиции. Очевидно, что число возможных комбинаций прироста и дохода велико, но для практических целей все их можно свести к следующим четырем типам[1]:



Фиг. 3 Выбор политики инвестиций

(1) *Прирост.* Это такое помещение капитала, при котором ожидается прирост в среднем на 10% в год на протяжении 10 лет. В этом случае не придается большого

[1] Следует отметить, что приведенные здесь цифры ни в коем случае не фиксированы и будут меняться с изменением условий на бирже.

значения доходу и допускаются колебания размеров основного капитала.

(2) *Прирост и доход.* Ожидается увеличение капитала на 5—6% в год, а размер дивиденда составляет 2—3% в год.

(3) *Доход и прирост.* Ожидается увеличение капитала всего на 3—4% в год. Желаемый дивиденд 3—4% в год. Большое значение придается устойчивости дохода.

(4) *Доход.* В этом случае основное внимание уделяется размеру и устойчивости дохода. Ожидаемый дивиденд составляет 4—6% в год. Прирост капитала не имеет значения.

Селектор выбирает политику инвестиции для данного клиента, задавая вопросы и находя ответы на них в списке, который он получает от Сканнера. Вопросы задаются в определенном порядке, причем этот порядок существенно зависит от получаемых ответов. Блок-схема процесса представлена на фиг. 3.

Выбор портфеля

Опишем в общих чертах информацию, которая имеется в модели непосредственно перед выбором портфеля.

а) Список акций (А-список), которые в настоящий момент подходят для инвестиции. Эти акции расклассифицированы по отраслям.

б) Для каждой компании из А-списка имеется список относительных ценностей, набор списков сравнительных характеристик, а также информация о прошлых, текущих и предсказанных значениях товарооборота, прибыли, дивиденда и других показателей.

в) Список информации о клиенте, для которого выбирается портфель. Этот список включает информацию, о которой говорилось в предыдущем разделе, а также еще один параметр: сумму, которую следует вложить в обыкновенные акции.

г) Выбранная для данного клиента политика инвестиции.

Выбор портфеля начинается с того, что содержимому списка «г» ставится в соответствие подмножество отраслей и связанных с ними компаний из «а». Затем это подмножество уменьшается с помощью набора тестов, использующих информацию из «б» и «в». В результате остается набор акций, который и образует портфель.

А. Отбор отраслей, соответствующих политике инвестиции

Несмотря на то что характеристики различных отраслей сильно перекрываются, банковский служащий, производящий инвестицию, связывает с каждой целью набор отраслей. Эта связь зависит как от характеристики цели, так и от общей характеристики компаний в пределах данной отрасли. Поэтому набор связанных с данной целью отраслей может включать и некоторые отрасли, связанные и с другими целями. Так, в некоторых отраслях имеются компании, мало различающиеся по своим характеристикам (например, банки, коммунальные предприятия). Другие же отрасли, как, например, нефтяная, более неоднородны и появятся в нескольких списках. Классификация отраслей по целям, осуществляемая доверенными инвесторами, с течением времени меняется очень медленно. Поэтому не делалось попыток определить характер ее изменения во времени, а соответствующие списки были составлены путем опроса служащих и изучения их деятельности. Имея эти списки, модель для каждой цели находит список отраслей и компаний, содержащихся в А-списке и одновременно подходящих для этой цели.

Б. Отбор компаний

После того как составлен список отраслей, Сканнер и Селектор приступают к отбору компаний из этого списка. Первый просматривает информацию о клиенте и проверяет, является ли доверенность узаконенной, а клиент — жителем шт. Пенсильвания [показатели (9) и (7)]. Если один или оба из этих фактов имеют место, Селектор производит следующие операции:

1) Если доверенность узаконена, отвергаются все те компании из данного списка, которые не имеют в шт. Пенсильвания узаконенного положения.

2) Если клиент постоянно проживает в шт. Пенсильвания, отвергаются все те компании, которые в шт. Пенсильвании облагаются налогом. Оставшиеся в списке компании модель проверяет с помощью набора тестов. С каждой политикой инвестиции связан свой набор тестов.

Сканнер упорядочивает компании каждой отрасли на основе главного с точки зрения политики инвестиции показателя. Так, если выбирается портфель с упором на доход, Сканнер расставляет компании каждой отрасли в порядке приносимого ими дохода. Затем Селектор берет первую компанию из отрасли, находящейся во главе списка, и проверяет ее с помощью набора тестов.

Каждый тест представляет собой проверку: обладает ли величина одного из важных показателей некоторым свойством.

Поскольку важность каждого показателя зависит от избранной политики инвестиции, последовательность тестов меняется с изменением цели инвестиции.

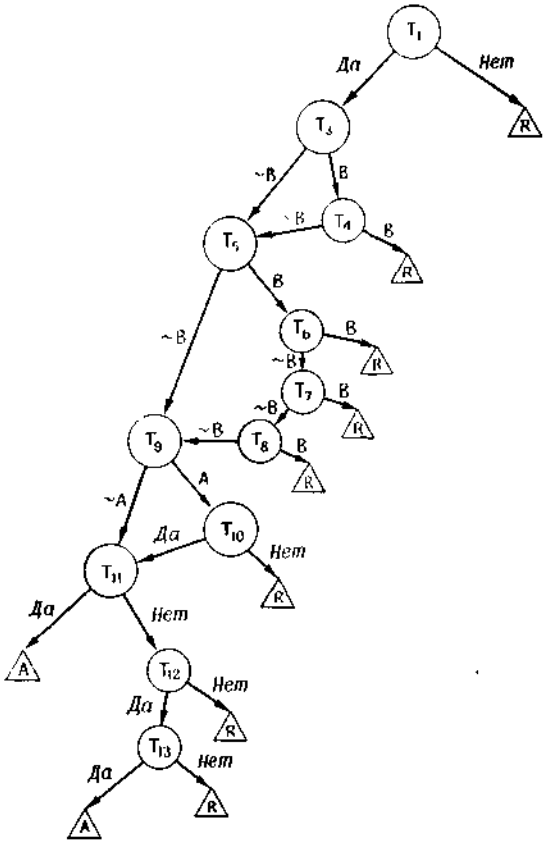
Компании каждой отрасли проверяются по очереди. Если первая компания по какой-либо причине не выдерживает одного из тестов, Селектор переходит ко второй компании, и процесс повторяется. Если ни одна компания из данной отрасли не выдерживает последовательности тестов, Селектор переходит к следующей отрасли. Если после обработки всех отраслей остается невложенный капитал, Селектор возвращается к первой отрасли, из которой не была выбрана компания, и снова начинает обработку. На этот раз процесс начинается с того места, где Селектор остановился в прошлый раз. Как только компания выбрана, Сканнер и Селектор переходят к следующей отрасли.

В качестве примера рассмотрим набор тестов, используемый при выборе портфеля с упором на прирост капитала (фиг. 4).

Как видно из блок-схемы, процесс проверки имеет иерархическую структуру. Так, если компания А выдерживает тест T_3, можно переходить прямо к тесту T_5. Если же она не выдерживает T_3, то перед прохождением дальнейшей проверки по основным тестам она должна выдержать тесты T_6, T_7 и T_8. Если ни одна компания из данной отрасли не принимается и Селектор снова возвращается к этой отрасли, он начинает проверку с того места, где остановился в прошлый раз. Например, если компания А была отклонена на основании теста T_6, Селектор начнет с теста T_7. В этом случае проверка будет продолжаться до тех пор, пока не будет отобрано нужное число компаний. Однако, поскольку каждая такая дискриминационная сеть содержит тест, который должна выдержать любая из отобранных компаний, вполне возможно, что модель отвергнет все компании данной отрасли.

В. Обеспечение разнообразия

Разнообразие обеспечивается требованием, чтобы каждый капитал инвестировался не меньше чем в пять отраслей. При этом обычно ограничиваются покупкой акций одной компании из каждой выбранной отрасли. Если портфель включает облигации и привилегированные акции, то каждые 10 000 долл., вложенные в эти бумаги, считаются эквивалентными участию в одной отрасли. Следовательно, для капитала в 50 000 долл., из которых 20 000 вложено в правительственные облигации, модель потребует, чтобы оставшаяся часть была вложена по крайней мере в три отрасли.



Фиг. 4. Дискриминаторная сеть, соответствующая цели «прирост».

T_1 — средний прирост в цене (за прошедшее время) $\geqslant 20\%$; T_3 — средний прирост прибыли на 1 акцию (за прошедшее время); T_4 — средний прирост товарооборота (за прошедшее время); T_5 — ожидаемый прирост прибыли на 1 акцию (прогноз на 1 год); T_6 — ожидаемый прирост товарооборота (прогноз на 1 год); T_7 — средний прирост притока наличных денег на 1 акцию (за прошедшее время); T_8 — средний прирост маржи (за прошедшее время); T_9 — величина y из списка относительной ценности; T_{10} — P/E меньше $\overline{P/E}$ на 10%; T_{11} — ожидаемый дивиденд >0; T_{12} — ожидаемый дивиденд $=0$; T_{13} — цена более чем на 10% ниже максимальной; B = «ниже», $\sim B$ = «не ниже»; A = «выше», $\sim A$ = «не выше»; A в треугольнике — «принять», R в треугольнике — «отклонить».

Г. Размер участия

Количество подлежащих покупке акций каждой компании определяется «Селектором акций» по следующим правилам:

1) Общая сумма, вкладываемая в обыкновенные акции, делится на число отобранных компаний. При этом получается среднее число долларов, которое следует вкладывать в каждую компанию.

2) Эта сумма затем делится на стоимость акций каждой компании. Полученное число округляется до ближайшего, кратного 5, а всякий раз, когда это возможно, округляется до сотен.

Очевидно, что процесс отбора должен продолжаться до тех пор, пока остается невложенный капитал. Как только весь капитал оказывается израсходованным, отбор прекращается, и выбранные акции образуют требуемый портфель.

Испытание модели

Чтобы проверить, насколько хорошо данная модель способна воспроизводить поведение доверенного инвестора, т. е. имитировать процесс доверительной инвестиции, ее заставили выбрать несколько портфелей в реальных условиях. В частности, курс акций и другие введенные в машину данные относились к первому и третьему кварталам 1960 г. В машину были введены также данные о четырех клиентах, для которых в тот же период выбрал портфели доверенный инвестор. В табл. I представлены портфели, выбранные для этих клиентов программой и доверен-

Сравнение портфелей, выбранных моделью и банковским служащим

Программа выбрала		Служащий выбрал	
число акций	компания	число акций	компания
Счет 1, 8/I 1960 г.			
Цель — прирост капитала; сумма инвестиции 22 000 долл.			
60	General American Transportation	30	Corning Glass
50	Dow Chemical	50	Dow Chemical
10	I. B. M.	10	I. B. M.
60	Merck a. Co.	50	Merck a. Co.
45	Owens Corning Fiberglass	50	Owens Corning Fiberglass
Счет 2, 10/VI 1960 г.			
Цель — доход и прирост капитала; сумма инвестиции 37 500 долл.			
100	American Can Co.	100	American Can Co.
100	Continental Insurance	100	Continental Insurance
100	Equitable Gas Co.	100	Equitable Gas Co.
100	Duquesne Light Co.	100	General Public Utilities
100	Libbey Owens Ford	100	Libbey Owens Ford
100	International Harvester	50	National Lead
100	Philadelphia Electric	100	Philadelphia Electric
100	Phillips Petroleum	100	Phillips Petroleum
100	Socony Mobil	100	Socony Mobil
Счет 3, 8/VII 1960 г.			
Цель — доход и прирост капитала; сумма инвестиции 31 000 долл.			
100	American Can Co.	100	American Can Co.
100	Continental Insurance	100	Continental Insurance
100	Duquesne Light	100	Duquesne Light
100	Equitable Gas	100	Equitable Gas
100	Pennsylvania Power a. Light	100	General Public Utilities
100	International Harvester	100	International Harvester
100	Libley Owens Ford	100	Libley Owens Ford
100	Socony Mobil Oil	100	Socony Mobil Oil

Счет 4, 26/VIII 1960 г.

Цель — доход; сумма инвестиции 28 000 долл.

100	American Can Co.	100	American Can Co.
100	Continental Insurance	100	Continental Insurance
100	Duquesne Light	100	Duquesne Light
100	Equitable Gas	100	Equitable Gas
100	Pennsylvania Power a. Light	100	General Public Utilities
100	International Harvester	100	International Harvester
100	Phillips Petroleum	100	Phillips Petroleum

ным инвестором. Полученные программой портфели затем сравнивались с портфелями, выбранными случайно и с помощью различных упрощенных моделей. Оказалось, что описанная выше программа доверительной инвестиции дает больший процент правильных вкладов, чем любая другая модель.

Для подкрепления этих выводов были проведены дальнейшие испытания, в ходе которых проверке подвергался сам процесс выбора портфеля. Эксперимент заключался в сравнении последовательности данных на выходе модели и записи процесса принятия решения доверенным инвестором. Это сравнение производилось для различных частей программы. Нельзя сказать, чтобы во всех процессах обнаружилось безоговорочное сходство. Тем не менее результаты эксперимента дают сильное подтверждение предположению, что модель хорошо отражает значительную часть процесса доверительной инвестиции.

РАЗДЕЛ 4

СОЦИАЛЬНОЕ ПОВЕДЕНИЕ

В последние годы проводились широкие исследования в области изучения социального поведения, особенно в области экспериментов с малыми группами. Эти исследования дали нам много информации относительно влияния большого числа факторов на групповое поведение людей.

В самое последнее время были проведены работы по созданию моделей группового поведения. Эти работы в какой-то степени основывались на данных экспериментальных исследований, но, кроме того, они и сами способствовали правильной ориентации экспериментальных исследований. Модели были описаны в обычных словесных терминах, а несколько позже выражены

в математической форме через дифференциальные уравнения и марковские процессы.

Проведенные экспериментальные исследования и математические модели оказались очень интересными и перспективными. Однако до сих пор приходится слышать высказывания по поводу примитивности и малочисленности этих математических моделей. Одним из ответов на такую критику и явилось создание машинных моделей.

Бэйлс в своей работе «Теория и исследование малых групп» [75] посоветовал студентам, изучающим социальное поведение, обратить внимание на машинную модель как на полезный метод для исследований в этой области. Бэйлс считал, что цель социологического исследования состоит в предсказании поведения в естественных ситуациях, а не в тщательно контролируемой лабораторной ситуации. Он предполагает, что для выполнения этой цели необходим синтез «большого количества переменных, находящихся в очень сложных взаимоотношениях друг с другом», и указывает, что вычислительная машина является именно тем инструментом, который может помочь при создании такой модели. Необходимо воздать должное Бэйлсу и его коллегам, которые настойчиво работали над программой, направленной на создание машинной модели группового поведения [75], Макфи, который разработал машинную модель поведения при голосовании на выборах [696], и Джону и Джейн Галлахорнам, создавшим модель социального взаимодействия. Предлагаемая статья Галлахорнов представляет собой отчет об их работе, специально подготовленный для данной книги. Джон Галлахорн — сотрудник отделения социологии и антропологии Мичиганского университета, Джейн Галлахорн является аспиранткой того же университета. Оба являются консультантами Секции искусственного разума Корпорации разработки систем.

16. МАШИННАЯ МОДЕЛЬ ЭЛЕМЕНТАРНОГО СОЦИАЛЬНОГО ПОВЕДЕНИЯ

Дж. Галлахорн и Дж. Галлахорн

Введение

Десять лет назад социолог С. Аш заметил: «Для действий в социальной сфере требуется знание социальных фактов об отдельных индивидуумах и группах. Чтобы занять свое место среди других, каждый из нас вынужден учитывать существование других личностей и должен достигнуть определенного пони-

мания их нужд, эмоций и мыслей» ([36], стр. 139). В недавние годы традиции психоанализа, теории социального поля и символического взаимодействия привели к выполнению многих интересных исследований, касающихся того, как отдельный индивидуум воспринимает и осознает свое человеческое окружение. От этих в значительной мере интуитивных подходов радикально отличается работа Дж. Хоманса «Социальное поведение» [484], которая совместно использует принципы двух самостоятельных строгих дисциплин — классической экономики и бихевиористской психологии. С нашей точки зрения, среди всех опубликованных работ эта работа представляет одно из самых перспективных объяснений реакций человека в условиях взаимодействия с другими индивидуумами, и мы поэтому выбрали работу Хоманса в качестве модели для исследований, касающихся индивидуальной реакции в условиях относительно простых социальных взаимодействий. В настоящее время наши усилия направлены прежде всего на разработку и уточнение модели элементарного социального поведения в форме машинной программы, записанной на языке IPL [805]. Помимо того что это поможет разъяснению и уточнению самой модели, мы надеемся также, что такое представление будет значительно способствовать цели естественного предсказания поведения в малых группах.

До того как перейти к обсуждению самой программы, остановимся вкратце на выдвинутой Хомансом формулировке элементарного социального поведения как «контакта лицом к лицу между индивидуумами, при котором вознаграждение, получаемое каждым в результате поведения других, появляется относительно непосредственно и мгновенно» ([484], стр. 7). В его модели поведение человека рассматривается как функция его поощрения; иначе говоря, интенсивность и вид реакций индивидуума зависят от количества и качества поощрения и наказания, вызываемых его действиями.

Чтобы проиллюстрировать применение предположений, выдвигаемых им для объяснения социальных взаимоотношений, Хоманс использует данное Блау [128] описание взаимного поведения работников административного аппарата. 16 работников работают на одинаковой должности в неком федеральном учреждении. Они отличаются между собой по квалификации, и предполагается, что чем квалифицированнее работник, тем больше сослуживцы обращаются к нему за помощью в работе. Анализируя социальные аспекты такого рода консультаций, Блау и Хоманс рассматривают взаимодействие между сотрудниками как некий *обмен ценностями*: оба участника извлекают выгоду из такого обмена, но оба должны уплатить за это

какую-то цену. Сотрудник, обращающийся за помощью, обычно вознаграждается тем, что он после этого в состоянии лучше выполнить некоторую работу; однако это достается ему ценой потери определенной части авторитета в глазах своих коллег, занимающих одинаковую с ним должность. Дающий консультацию, со своей стороны, добивается определенного престижа; однако он расплачивается за это своим временем, которое он мог бы затратить на свою собственную работу.

Эту относительно простую последовательность взаимодействий между двумя воображаемыми сотрудниками, которых мы назвали Тэд и Джордж, мы использовали для того, чтобы реализовать в машинной программе динамические следствия из гипотез Хоманса, связанные с процессами принятия решений индивидуумами, участвующими во взаимном социальном обмене. К настоящему времени существует работающая программа, названная «Гомункулус», моделирующая взаимодействие между двумя людьми; однако теперь мы разрабатываем дополнительные программы, которые должны внести некоторые улучшения в работу основной модели. Хотя поведение нашей модели правдоподобно, она еще не была проверена путем точного сопоставления с реальным социальным взаимодействием.

Программа

Планируя программу для нашей модели, нам прежде всего пришлось в явном виде сформулировать наше понимание индивидуума как организма, перерабатывающего информацию. Это означает, что для осуществления поведения, соответствующего принципам, выдвинутым в гипотезах Хоманса, человек должен быть «запрограммирован» так, чтобы по меньшей мере быть в состоянии делать следующее: он должен обладать способностью воспринимать стимулы, распознавать их, хранить их в памяти, сравнивать и сопоставлять; он должен обладать способностью к действиям, отличать поощрение от наказания, ассоциировать определенные совокупности стимулов с реакциями (ответами), а реакцию — с подкреплением и на основе прошлого опыта должен уметь предсказывать вероятность поощрения в результате намеченной им реакции. В социальных ситуациях он должен отличать одних членов группы от других, оценивать социальные «стимулы» с точки зрения конкретного человека, который является их источником, и надлежащим образом выбирать свою реакцию с целью вызвать положительную реакцию на нее.

Рассмотрев некоторые основные характеристики, необходимые для программирования поведения социального субъекта,

мы столкнулись с практической проблемой, как ввести такого субъекта в вычислительную машину. К счастью, язык IPL идеально приспособлен для решения этой задачи В связи с тем что он является языком обработки списковых структур, как сами данные, так и программы записываются на нем в форме списков.

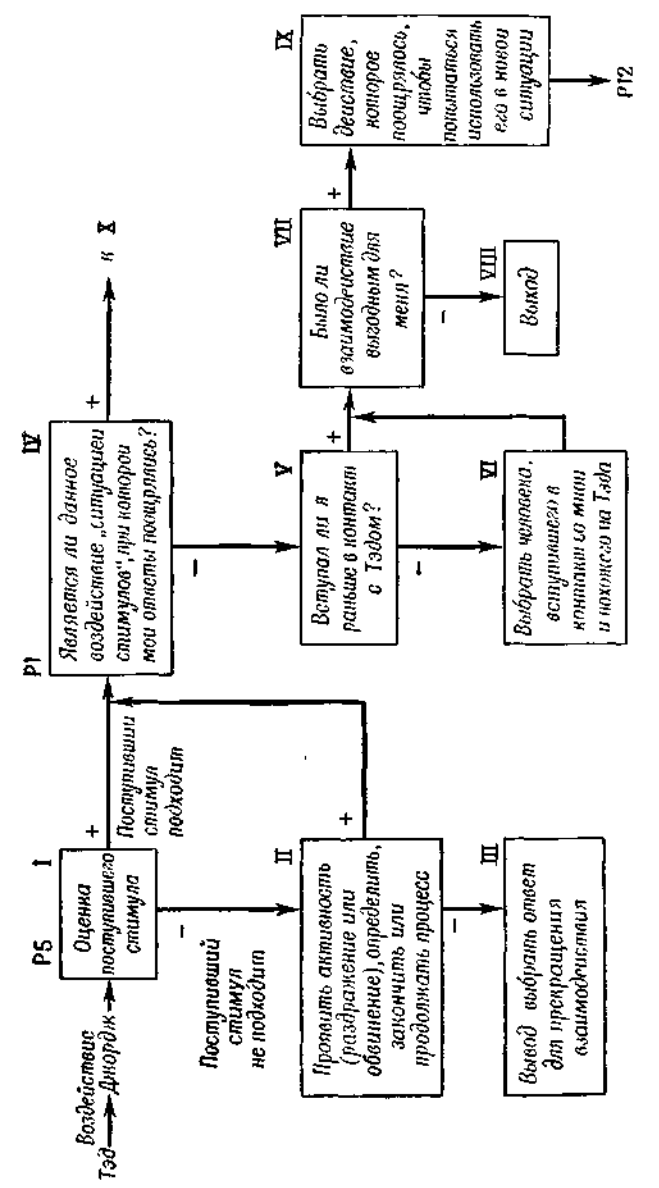
Используя списковые структуры, или иерархии списков, содержащие так много подсписков, как это окажется необходимым, а также описательные списки, которые ставят в соответствие каждому символу список признаков и их оценок, удается эффективно обрабатывать очень сложную информацию. Таким образом, в нашей программе человек представлен в виде списковой структуры, содержащей очень большое число описательных списков. Среди данных, включенных в списковую структуру модели человека, есть, например, такие, как идентификация личности человека, его способности, его относительное и абсолютное положение в различных социальных группах, списки образов связанных с ним групп и списки образов каждого члена группы.

Поточные диаграммы, показанные на фиг. 1—3, представляют нашу интерпретацию процессов, необходимых для осуществления элементарного социального поведения в соответствии с гипотезой Хоманса. Запрограммированная нами цепочка взаимодействий индивидуумов начинается с сотрудника по имени Тэд, посылающего своему коллеге Джорджу символ, представляющий собой просьбу о помощи в выполнении некоторой работы. Отложим пока рассмотрение положения 5 (P5, блок I на схеме фиг 1) и начнем с положения 1 (P1, блок IV), причем будем рассматривать только положительные ветви этой диаграммы, что позволит кратко записать положения одно за другим. Запрограммированная нами формулировка положений Хоманса определяет процессы оперирования с символами, которые дают возможность Джорджу решить, какое действие он должен предпринять в ответ на просьбу Тэда.

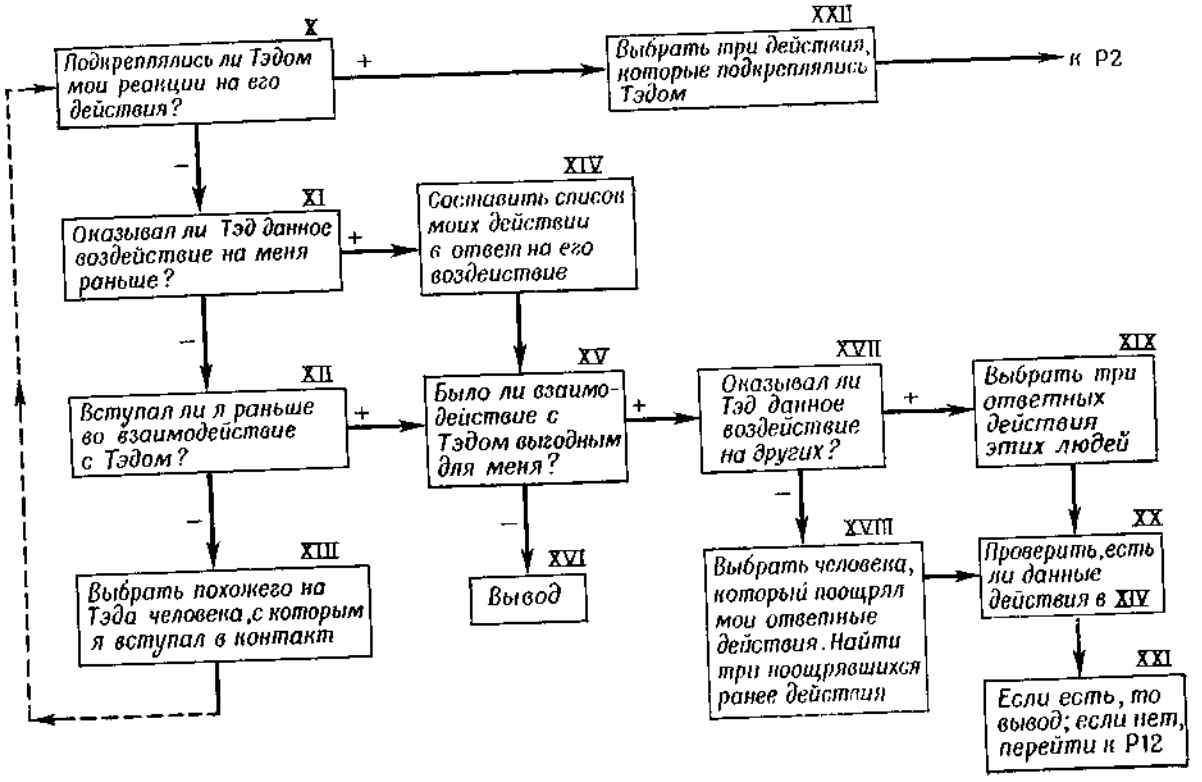
Положение 1 (P1)

Первое положение Хоманса касается влияния обобщения стимулов и реакций:

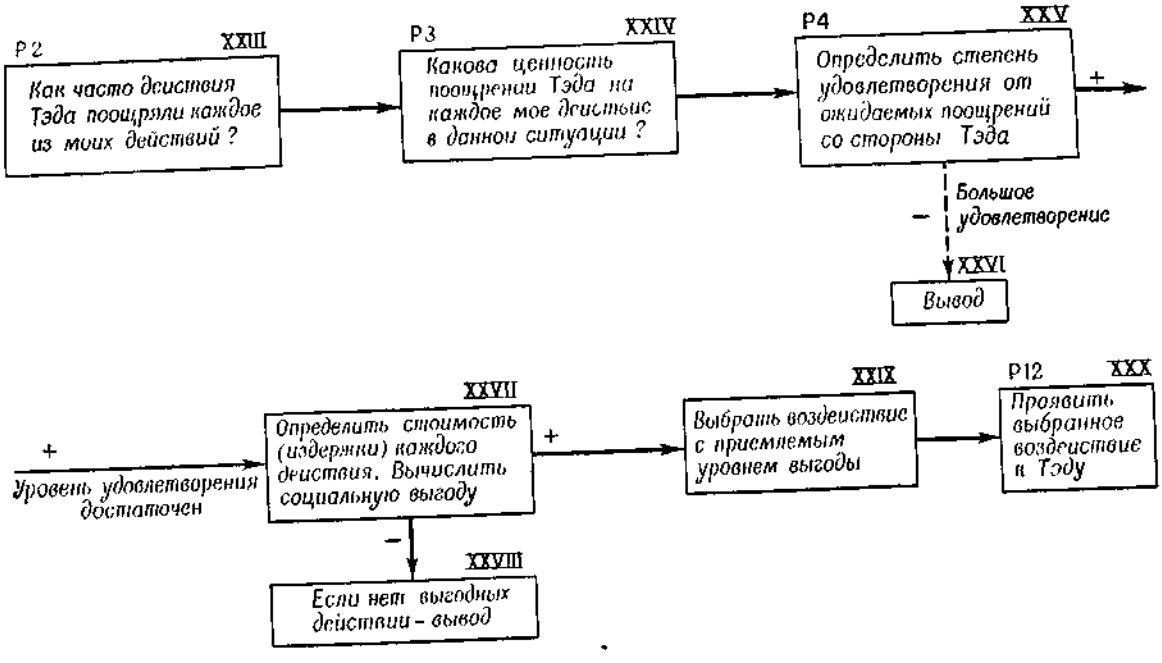
«Если в недавнем прошлом появление той или иной «ситуации стимулов» приводило к условиям поощрения определенных действий человека, то чем более настоящая «ситуация стимулов» походит на прошлую, тем вероятнее, что человек произведет те же действия или какие-то подобные действия» ([484], стр. 53).



Фиг 1


X
Подкреплялись ли Тэдом мои реакции на его действия?
+
XXII
Выбрать три действия, которые подкреплялись Тэдом
к Р2
–
XI
Оказывал ли Тэд данное воздействие на меня раньше?
+
XIV
Составить список моих действий в ответ на его воздействие
–
XII
Вступал ли я раньше во взаимодействие с Тэдом?
+
XV
Было ли взаимодействие с Тэдом выгодным для меня?
+
XVII
Оказывал ли Тэд данное воздействие на других?
+
XIX
Выбрать три ответных действия этих людей
–
XIII
Выбрать похожего на Тэда человека, с которым я вступал в контакт
–
XVI
Вывод
–
XVIII
Выбрать человека, который поощрял мои ответные действия. Найти три поощрявшихся ранее действия
XX
Проверить, есть ли данные действия в XIV
XXI
Если есть, то вывод; если нет, перейти к Р12

Фиг. 2.


Р2
XXIII
Как часто действия Тэда поощряли каждое из моих действий?
Р3
XXIV
Какова ценность поощрений Тэда на каждое мое действие в данной ситуации?
Р4
XXV
Определить степень удовлетворения от ожидаемых поощрений со стороны Тэда
+
–
Большое удовлетворение
XXVI
Вывод
+
Уровень удовлетворения достаточен
XXVII
Определить стоимость (издержки) каждого действия. Вычислить социальную выгоду
+
XXIX
Выбрать воздействие с приемлемым уровнем выгоды
Р12
XXX
Проявить выбранное воздействие к Тэду
–
XXVIII
Если нет выгодных действий – вывод

Фиг. 3.

В процессе перевода данного положения на язык машинной программы, осуществляющей процесс решения, мы встретились с необходимостью рассмотреть по порядку два аспекта «ситуации стимулов». Для начала мы предположим, что сотрудник по имени Джордж будет реагировать довольно «глобально» на ситуацию в целом. Так, наша интерпретация начальной обработки информации, соответствующей положению 1 (см. P1, блок IV на фиг. 1), включает рассмотрение Джорджем того, является ли *AR* (принимаемая активность; в нашем примере — обращение за помощью) общей «ситуацией стимулов», в которой его реакция (*AE*, или выдаваемая активность) поощрялась. При осуществлении этого процесса одна из программ, представляющая собой функцию выбора ответа нашего запрограммированного сотрудника (Джордж), ищет в памяти список ранее подкреплявшихся «ситуаций стимулов», чтобы определить, принадлежит ли к ним данная.

Рассматривая положительную ветвь поточной диаграммы и предполагая тем самым, что Джордж обнаружил, что в прошлом положительный ответ на просьбу о помощи приводил к поощрению, перейдем к его следующим рассуждениям, отображаемым блоком X на фиг. 2. Джордж должен теперь определить, поощрялись ли его реакции на просьбы о помощи Тэдом — человеком, участвующим в данной ситуации.

Чтобы проанализировать свои прошлые взаимоотношения с Тэдом, Джордж должен более глубоко погрузиться в свою память. Выше отмечалось, что списковая структура данного человека включает список, который содержит образ каждого человека, принадлежащего данной группе. Одна из программ находит этот *список образа*, отыскивает подсписок, описывающий результаты предыдущих взаимодействий Джорджа с Тэдом, определяет, получал ли Джордж данный стимул от Тэда прежде и, если это действительно так, поощрялись ли Тэдом положительные ответы на данный стимул. В рассматриваемом случае Джордж обнаруживает, что в прошлом положительные ответы на просьбу о помощи в работе Тэдом поощрялись (подкреплялись).

Определив таким образом, что «ситуация стимулов» носила поощрительный характер и что именно Тэд осуществлял данное поощрение, Джордж должен теперь выбрать один из вариантов ответа. Если его взаимодействие с Тэдом проходило в аналогичных ситуациях и он проявлял различные активности, которые поощрялись Тэдом (например, решение задачи Тэда, указание Тэду на полезный источник информации или отсылка его к специалисту по данному вопросу), то Джордж должен выбрать одну из этих возможных реакций на просьбу Тэда. В нашей программе Джордж перебирает до трех видов различных ответных действий из списка ответов, которые Тэд поощрял (блок XXII на фиг. 2), а затем начинает обрабатывать дополнительную информацию, касающуюся этих предварительно намеченных действий.

Положение 2 (P2)

Второе положение Хоманса касается положительного влияния частоты и новизны подкрепления.

«Чем чаще в течение некоторого периода времени действия одного человека поощряют действия другого, тем чаще этот другой человек проявляет такие действия» ([484], стр. 54).

Переформулировка данного положения с целью его моделирования на вычислительной машине вызвала ряд трудностей. Было бы довольно просто установить счетчик для каждого поощряемого ответа, а затем выбирать требуемую информацию в соответствии с частотой поощрения. Однако мы чувствовали, что такая процедура неадекватна системе обработки информации человеком. Конечно, люди обычно сами себя снабжают точными измерительными масштабами и используют различные искусственные устройства, например вычислительные машины, для увеличения точности своих вычислений. Однако в своих оценках, касающихся частоты и ценности поощрений, получаемых при повседневном социальном взаимодействии, они, видимо, применяют менее тонкие средства измерения. Для программирования этого положения мы поэтому ввели довольно грубую пятизначную шкалу для частоты поощрений: от случая, когда ответ «почти всегда поощрялся», через оценку «поощрялся в половине случаев», до оценки «почти никогда не поощрялся».

В настоящее время нами испытываются различные способы оперирования с данной шкалой. Согласно одной из написанных нами программ, увеличение оценочного показания шкалы для частоты поощрений происходит после трех удачных ответов. Однако этот метод не является вполне удовлетворительным. Действительно, можно возразить, что оценка частоты поощрения далеко не всегда независима от эмоциональной характеристики поощрения. Когда «Гомункулус» достигнет стадии моделирования поведения небольших групп при контролируемых условиях, окажется возможным сравнить различные приближенные оценки суждений о частоте поощрения, даваемых реальными людьми в зависимости от социального взаимодействия, и избрать программу, которая более точно моделирует реальное поведение.

Когда обработка информации, связанной с данным положением, завершена (P2, блок XXIII на фиг. 3), Джордж имеет

грубую оценку частоты, с которой Тэд поощрял каждое из действий, рассматривавшихся им как возможный ответ на текущую просьбу Тэда о помощи в работе. Если рассматривать второе положение Хоманса изолированно, можно прийти к выводу, что Джордж выберет тот из ответов, который наиболее часто поощрялся в прошлом опыте. Однако прежде, чем принять решение, необходимо обработать дополнительную информацию.

Пожалуй, здесь уместно указать, каким образом программа распределяет весь материал в оперативной памяти Джорджа. Для этой цели используется до 100 «личных» ячеек памяти, а команды каждой из программ определяют, какая из ячеек должна использоваться для найденной информации.

В настоящее время Джордж использует около 50 из этих ячеек. К тому же важная информация, доступная для всех участников группы, например то, что было услышано или увидено во время последних пяти взаимодействий, хранится в постоянной («общественной») памяти машины.

Положение 3 (Р3)

Среди других факторов, которые должны быть рассмотрены при выборе надлежащих действий, существенна цена ожидаемого поощрения. Третье основное положение Хоманса (Р3) состоит в следующем:

«Чем ценнее для человека единица действия другого человека, тем чаще он совершает действие, поощряемое действием другого» ([484], стр. 55).

Установление цены действия — задача более сложная, чем оценка частоты, с которой это действие совершается. Ценность действия состоит из двух компонент: одна из них относительно постоянна, другая же, о которой будет говориться в положении 4 (Р4), несколько меняется за время, в течение которого происходят простые взаимодействия, образующие элементарное социальное поведение. Компонента оценки, о которой говорится в положении 3, связана с порядком индивидуальных субъективных предпочтений, которые каждый индивидуум проявляет по отношению к различным действиям.

В нашем примере мы могли бы предсказать, что горячее социальное одобрение, включая похвалы со стороны Тэда перед всеми коллегами, были бы для Джорджа более «ценными», чем полуискреннее «Гм, благодарю» или раздраженное «Ладно, извините за беспокойство»

С этой точки зрения наша программа обладает свойством, которое программисты называют «интуицией». При рассмотрении просьбы Тэда Джордж держит «в уме» три ответа, которые

Тэд давал ему в прошлом, и оценивает частоту, с которой Тэд поощрял каждый ответ. Теперь мы должны более внимательно рассмотреть в отдельности каждое поощрение, которое он ожидает от Тэда в ответ на каждое свое действие, так, чтобы он мог определить и внутреннюю ценность каждого ожидаемого поощрения (Р3, блок XXIV на фиг. 3). Беря снова каждое действие, которое рассматривается Джорджем, программа, осуществляющая положение 3, находит ответы, дававшиеся Тэдом ранее каждому из сотрудников, определяет, какой из ответов он, вероятнее всего, применит сейчас, и ищет описательный список, чтобы определить субъективную оценку этого поощрения для Джорджа.

Положение 4 (Р4)

Положение 4 Хоманса касается вопроса потери полезности из-за насыщения, или, другими словами, предельной полезности данной единицы активности.

«Чем чаще человек в недавнем прошлом принимал поощряющие действия от других людей, тем менее ценной для него становится каждая дальнейшая единица такой активности» ([484], стр. 55).

В противоположность относительно постоянному аспекту оценки, связанному с внутренним удовлетворением, компонента насыщения меняется в довольно широких пределах. Принимая во внимание количество активности, полученное каким-то человеком, мы отмечаем, что данный человек «оценивает» эту активность выше, когда он обычно лишен ее, чем в тех случаях, когда он постоянно получает такого рода активность. Таким образом, несмотря на то что социальное одобрение может являться для человека сильным поощрением, он, вероятно, будет не очень заинтересован в таком обобщенном поощрении, если он в недавнем прошлом уже достаточно часто получал его.

При обработке информации, необходимой для реализации данной стадии программы, Джордж должен оценить свою относительную неудовлетворенность ожидаемым от Тэда поощрением. Теперь Джордж имеет в своей оперативной памяти запись каждой из рассматриваемых им активностей, и вместе с каждой из них хранится разнообразная информация, имеющая отношение к данной активности, включая и ожидаемые от Тэда реакции. Программы, реализующие положение 4, ищут описательный список каждого ожидаемого поощрения, чтобы определить степень текущей удовлетворенности или неудовлетворенности Джорджа таким поощрением. Итоговая оценка удовлетворения — неудовлетворения, основывающаяся на

простой оценочной шкале, записывается как оценка особого признака в описательном списке каждой активности. При выполнении положения 5, которое излагается ниже, программа обновляет оценку удовлетворения — неудовлетворения всякий раз, когда воспринимается какая-то активность.

Выбирая информацию таким образом, Джордж имеет оценку относительной частоты, с которой Тэд поощрял каждую проявляемую им активность. Кроме того, он предсказывает реакцию Тэда на каждое из своих предполагаемых действий и определяет, насколько поощряющими будут лично для него эти предполагаемые реакции, а также насколько он в настоящее время будет удовлетворен или неудовлетворен ожидаемыми поощрениями. Следовательно, Джордж теперь может оценивать свои предположительные ответы с точки зрения их возможной оплаты. Но он еще не готов к тому, чтобы проявить самую высокооцениваемую активность.

Другим важным понятием является стоимость (издержки) предполагаемого ответа. Хоманс определяет стоимость действия как оценку поощрения, ожидаемого при другом действии, от которого пришлось отказаться, чтобы осуществить данное действие. В нашем примере Джордж должен отказаться от собственной работы, когда он затрачивает время на помощь Тэду, поэтому Джордж должен определить относительную ценность поощрения этого другого действия. Для этого он применяет процедуру, аналогичную только что рассмотренной, — он обрабатывает информацию, касающуюся частоты прошлых поощрений и оценки ожидаемых поощрений за это действие, а также его относительное удовлетворение данным поощрением. Затем он может сравнить общее ожидаемое поощрение от данного предполагаемого ответа Тэду с поощрением в результате продолжения своей работы и вычислить, по терминологии Хоманса, *психическую выгоду*, т. е. разность между поощрением определенной активности и ее стоимостью.

Предположим теперь, что Джордж собирается оказать Тэду непосредственную помощь для решения задачи Тэда, потому что Тэд хвалил его в прошлом за это, а Джордж высоко ценит социальное одобрение и в настоящее время чувствует себя недостаточно удовлетворенным в этом отношении. Но представим себе при этом также, что Джорджу нужно срочно закончить важную работу и что, потеряв какое-то время на оказание помощи Тэду, он может ухудшить качество своей работы и тем самым потерять поощрение, которое он надеется получить от своего начальника за хорошую работу. В этом случае Джордж потерпел бы скорее убытки, чем получил выигрыш от непосредственной помощи Тэду. Поэтому он продолжает рассмотрение, пытаясь выяснить, не принесет ли ему выгоду одна из других возможных активностей. Джордж, вероятно, решит, что, отослав Тэда к другому источнику, он в итоге выиграет, так как он все же ожидает некоторого одобрения и за это (правда, меньшего, чем если бы он непосредственно помог Тэду), но понесет меньшие потери в стоимости с точки зрения времени, отнятого от его собственной работы.

Выбрав определенное действие, которое он считает наиболее выгодным с точки зрения социального одобрения, Джордж дает определенный ответ Тэду. Далее наша программа работает циклами, так как активность, проявленная Джорджем, превращается в активность, полученную Тэдом. Теперь уже Тэд должен обрабатывать информацию для выбора соответствующего и выгодного ответа Джорджу.

Положение 5 (P5)

Справедливость распределения поощрений составляет предмет положения 5 Хоманса. Оно отражает, пожалуй, самое сложное из всех понятий, используемых для объяснения элементарного социального поведения. Оно требует рассмотрения информации на другом уровне, а именно на уровне социальных норм или общепринятых оценок поведения внутри группы. Путем многократного повторения ситуации взаимодействия между членами группы определенная схема поведения стабилизируется так, что создаются представления о *справедливом распределении* поощрений и издержек среди членов группы. Чем больше издержки человека в данном взаимодействии, тем больше должно быть получаемое им поощрение. Но понятие справедливого распределения этим не ограничивается; оно учитывает личный вклад человека в данное взаимодействие, его положение в обществе, квалификацию, опыт, возраст и пол. Чем больше вклад данного человека во взаимодействие, тем большую выгоду он при этом взаимодействии вправе ожидать. Таким образом, в соответствии с принципом справедливого распределения поощрений мы вправе ожидать, что определенные издержки и вклады в прошлом должны позднее привести к определенным типам и степеням поощрения.

Хоманс выдвигает в этой связи следующее положение:

«Чем больше закон справедливого поощрения нарушается к невыгоде данного человека, тем вероятнее он обнаруживает эмоциональное поведение, которое мы называем раздражением» ([484], стр. 75).

Имеется и другая сторона вопроса: если человек получает поощрения, превышающие те, на которые он считает себя

вправе претендовать, он, вероятно, должен испытывать чувство неловкости.

Процесс перевода данного положения на язык машинной программы связан с очень интересными проблемами, с которыми нам при работе над «Гомункулусом» еще не приходилось сталкиваться. В сущности списковая структура наших сотрудников должна быть запрограммирована таким образом, чтобы они имели «совесть» и, кроме того, имели некоторый набор «раздраженных ответов».

Интерпретация данного положения на языке программы по существу ставит вопрос, соответствует ли та или иная деятельность данным обстоятельствам (P5, блок I на фиг. 1). Если это так, то человек, на которого направлено такое действие, может обрабатывать соответствующую информацию так же, как Джордж поступал с просьбой Тэда, которую он считал приемлемой. Если же действие рассматривается как неприемлемое, то осуществляется более сложное поведение. Чтобы проиллюстрировать это, составим список взаимодействий между Джорджем и другим сотрудником того же ведомства, которого мы назовем Томом.

Принятая норма поведения в данном ведомстве такова, что сотрудник, нуждающийся в помощи другого сотрудника, должен сообщить об этом открыто в форме, признающей превосходство консультанта над ним в данной области. Том же хочет получить помощь от Джорджа довольно нечестным способом: он подходит к Джорджу, говоря ему, что столкнулся «с очень интересной проблемой» и что ему хочется посмотреть, решит ли Джордж эту задачу так же, как и он. Пусть это происходит три раза подряд, и каждый раз Том, получая от Джорджа готовое решение, комментирует его словами: «Да, вы получили тот же результат, что и я!» Джордж решает, что Том нарушает этим правила честного обмена ценностями, уклоняясь от издержек, связанных с выражением благодарности и признанием превосходства Джорджа в этой области. Поэтому, когда Том в четвертый раз снова предлагает решить ему «очень интересную задачу», Джордж с раздражением отвечает: «Послушайте, почему бы вам не делать свою работу самому!»

Это описание, конечно, не отвечает на вопрос, как составить программу для вычислительной машины, которая обнаруживала бы столь «очеловеченное» поведение. Джордж запрограммирован таким образом, что он рассматривает время, затрачиваемое им на решение чужой задачи, как помощь другому сотруднику, за которую ему причитается признание и общественное одобрение. Когда коллега Джорджа в ответ на его помощь вместо соответствующего поощрения заявляет, что Джордж пришел к тому же выводу, что и он, Джордж считает такие действия неприемлемыми с точки зрения ожидаемого справедливого распределения поощрения. Поэтому программы, осуществляющие положение 5, изменяют имеющийся у Джорджа список образа Тома так, что в следующий раз он ожидает от него большего признания и благодарности, чем обычно, так как Том должен искупить вину за предыдущее уклонение от оплаты. После трехкратного повторения такой последовательности взаимодействий расхождение между поведением Тома и тем поведением, которое Джордж от него ожидает, будет настолько большим, что при оценке ответа Тома он в список образа Тома введет сигнал, указывающий, что взаимодействие с Томом не сопровождается поощрением, так как Том нарушает принятые группой правила.

Если Том еще раз спросит мнение Джорджа уже после того, как этот сигнал предупреждения был зафиксирован в списке, Джордж или ответит на вопрос Тома с раздражением, или затаит раздражение, которое впоследствии выльется на кого-то другого. В машинной программе такая реакция выражается в резком поведении по отношению к другим сотрудникам. Но еще до того, как Том будет активно наказан, Джордж вначале оценивает, каковы будут последствия такого наказания Тома для самого Джорджа. Если в одном из возможных вариантов взаимодействия Джордж заметит, что Том в хороших отношениях с начальником, он может подавить свое агрессивное настроение в данный момент и проявить его в следующий раз, когда взаимодействие будет происходить в более благоприятной обстановке.

Стандартные программы, осуществляющие отрицательную ветвь положения 5 (P5, блок II на фиг. 1), не только изменяют список образа, но используют также некоторые из подпрограмм, соответствующие другим положениям, для оценки последствий непосредственной раздраженной реакции. В зависимости от исхода этого процесса программа работает в соответствии с положением 1 или процесс взаимодействия прекращается.

Выводы

Подобно другим психологам, выражающим свои теории на языке IPL-V с целью изучить процессы человеческого мышления путем моделирования их на вычислительной машине, мы свели сложное социальное поведение к процессам оперирования с символами. Даже из этого краткого описания нашей программы ясно, что мы сочли язык IPL-V удобным для записи работы нашей модели. Мы уже отмечали гибкость работы программы за счет организации информации в списки и списковые струк-

туры, а также изящную простоту и эффективность, которые дает применение описательных списков для хранения информации, касающейся определенных символов. Кроме того, относительная легкость организации подпрограмм в иерархические структуры значительно упрощает последовательную обработку информации, включающую в себя большое число условных подпрограмм. У нас не было времени, чтобы исследовать отрицательную ветвь нашей поточной диаграммы, кроме той ее части, которая связана с положением 5, однако общность процессов очевидна.

По сравнению с более детальными познавательными моделями поведения, например Фейгенбаума [318] и Фельдмана [326], наша модель в основном затрагивает вопросы принятия решения человеком при социальном взаимодействии, где происходят характерные процессы взаимных поощрений и наказаний. «Гомункулус» не является ни полностью обобщенной моделью, как, например, элементарное «познающее и запоминающее устройство» Фейгенбаума, ни специфичной для определенного испытуемого моделью, как созданная Фельдманом модель поведения в эксперименте с двоичным выбором [1]. Скорее в этой программе обнаруживаются оба этих подхода. Процессы обработки информации, осуществляемые данными подпрограммами, являются общими для всех моделируемых испытуемых. Однако каждая списковая структура, описывающая участвующую группу, является весьма специфичной; следовательно, индивидуальный идиосинкразический и прошлый опыт определяет, будут ли выполняться определенные подпрограммы и каков будет результат работы их в определенной цепочке взаимодействий. Что же касается исходных предположений, то наша модель в этом отношении обладает характерными чертами как программы Фейгенбаума, так и программы Фельдмана. Подобно этим моделям, «Гомункулус» основан на нечисловых процессах и является в большей степени детерминированной, чем вероятностной, системой. Кроме того, предполагается, что процесс принятия решений является последовательным, т. е. мы считаем, что человек за определенный отрезок времени может проделать лишь конечное число операций.

[1] См. стр. 338 настоящей книги. — *Прим. ред.*

Ч А С Т Ь III

Обзор подходов и методов

ВВЕДЕНИЕ

Говорят, что лучший способ изложения состоит в том, чтобы сначала рассказать аудитории, о чем вы собираетесь рассказывать, затем рассказать это, а в заключение еще раз сказать аудитории, о чем вы рассказывали. Первые две части этого плана уже выполнены, и осталось сделать последнее. Мы решили завершить эту книгу обзорами Армера [1] и Минского. Мы не собираемся давать здесь подробный обзор этих двух работ, а просим читателя лишь кратко задержаться на двух коротких абзацах.

На отношение к проблеме разумного поведения машин сильно повлияли как недостаток наших знаний, так и отсутствие должного понимания смысла уже проделанной работы, а также предвзятые представления о том, что является доказательством наличия «разума». Эта книга дает читателю возможность ознакомиться с современным положением дел в области исследований искусственного интеллекта. Армер дает обзор и анализ предвзятых представлений, касающихся машинного разума. Его статья основана на обзоре литературы и его собственных наблюдениях в США и СССР. Хотя сам Армер имеет отношение к большей части исследований, изложенных в этом томе, он в вопросе «Могут ли машины мыслить?» занимает благоразумную позицию золотой середины.

Заключительную статью в этой книге образует содержательный обзор Минского, освещающий подходы, ошибки и удачи в исследованиях по искусственному интеллекту. Минский пытается внести в эту область какую-то структурность и сформулировать проблемы, еще подлежащие разрешению. Этот единственный в своем роде обзор необходимо прочесть для правильного понимания всей данной области исследований.

[1] Обзор Армера «Отношение к разумным машинам» исключен из русского перевода, поскольку он уже имеется в русском переводе (см. приложение к книге Таубе [1065]). — *Прим. ред.*

17. НА ПУТИ К СОЗДАНИЮ ИСКУССТВЕННОГО РАЗУМА

М. Минский

Введение

У пришельца на нашу планету могла бы вызвать недоумение та роль, которую играют счетно-вычислительные машины в нашей технике. С одной стороны, он (или оно), вероятно, повсюду читал бы и слышал о чудесном «механическом мозге», поражающем его создателей своими удивительными интеллектуальными способностями. Кроме того, он был бы предупрежден, что эти машины следует держать в определенных границах, чтобы они не подавили нас силой, убеждением или, чего доброго, раскрытием истины, слишком ужасной для того, чтобы с ней можно было согласиться. С другой стороны, пришелец обнаружил бы, что машины со всех сторон поносят за их рабское повиновение, за лишенное всякого воображения буквальное толкование инструкций, за их неспособность к новаторству и безынициативность, короче говоря, за их нечеловеческую тупость.

Наш пришелец, вероятно, так и остался бы в недоумении, если бы самостоятельно попытался разобраться в этих чудищах, ибо он нашел бы только несколько машин (главным образом, «универсальных» вычислительных машин, запрограммированных в какой-то момент для выполнения определенных задач), которые выполняли бы работу, действительно требующую некоторого интеллекта. Кое-какие машины доказывали бы математические теоремы довольно тривиального характера. Другие бы играли в какие-то игры, время от времени нанося поражение своим конструкторам. Третьи различали бы печатные знаки. Достаточно ли этого, чтобы оправдать столь значительный интерес к этой проблеме, не говоря уже о ее более глубоком толковании? Я верю, что это так, что мы находимся на пороге эпохи, в которой будут играть большую, а быть может, и доминирующую роль машины, решающие интеллектуальные проблемы. Но в нашу задачу не входит гадать о том, что нам принесет будущее; единственное, чего мы хотим, это попытаться описать и объяснить, какими, по нашему мнению, должны быть наши первые шаги по созданию «искусственного разума».

Наряду с разработкой универсальных вычислительных машин за последние несколько лет возросли усилия по изучению и автоматизации процессов решения задач. Появился целый ряд статей с описанием теории или реальных программ для вычислительных машин, которые могут играть в различные игры, доказывать теоремы, распознавать образы и решать другие задачи, требующие известного интеллекта. В этих статьях отсутствует, однако, общее рассмотрение важнейших проблем в этой области.

В данной статье будет сделана попытка выделить, проанализировать и определить взаимосвязь между некоторыми из этих проблем. Анализ будет проиллюстрирован достаточным количеством примеров из литературы по этому вопросу, чтобы удовлетворить требованиям обзорной статьи, которая должна ввести в круг обсуждаемых проблем, но, естественно, останется еще много важных работ, которые не смогут быть затронуты в этом обзоре. Статья написана очень сжато и, следовательно, не может претендовать на достаточно подробный анализ всех относящихся сюда вопросов.

Не существует никакой общепринятой теории «разума»; поэтому анализ, даваемый в этой статье, отражает точку зрения автора и не застрахован от критики. Мы сожалеем, что не можем выразить благодарность всем тем, чьи идеи способствовали написанию этой работы; достаточно указать, что затронутые здесь вопросы обсуждались почти с каждым автором, на которых имеются ссылки в статье.

Удобно разбить все рассматриваемые здесь проблемы на пять основных разделов: поиск, распознавание, обучение, планирование и индукция. Это деление совпадает с основными разделами статьи. Суммируем очень кратко основные выводы.

В некотором смысле вычислительная машина может выполнять только то, что ей задают. Но даже если мы не знаем точно, как решить какую-то определенную проблему, мы можем запрограммировать машинный *поиск* для отыскания среди огромного пространства возможных ответов требуемого решения. К сожалению, когда мы составляем непосредственно программу для такого поиска, мы обнаруживаем, что результат оказывается чрезвычайно мало эффективным.

При использовании методов *распознавания образов* эффективность может быть значительно улучшена за счет ограничения применения машинных методов проблемами, для которых они действительно пригодны. При методе *обучения* эффективность еще более улучшается, если направлять поиск в соответствии с ранее приобретенным опытом. При фактическом анализе ситуации с использованием так называемых методов *планирования* эффективность машины может быть существенно повышена путем замены первоначально заданного поиска гораздо меньшим по объему и более приемлемым исследованием. Наконец, в разделе, посвященном *индукции*, мы рассматриваем некоторые более общие идеи, касающиеся выработки разумного поведения у машины.

1. Проблема поиска [1]

Если для некоторой данной задачи имеются способы проверки пригодности предложенного решения, задачу можно решить, опробовав все возможные ответы. Но это всегда отнимает слишком много времени и поэтому не представляет практического интереса. Любое устройство, которое может сократить такой поиск, представляет поэтому большую ценность. Если удается определить относительное улучшение, можно воспользоваться методом «подъема по градиенту» (см. разд. 1Б), но его использование требует знания структуры пространства поиска, и если такая структура не удовлетворяет определенным условиям, этот метод может принести больше вреда, чем пользы.

Когда мы будем в дальнейшем говорить о решении задач, мы обычно будем предполагать, что все задачи, которые требуется решить, с самого начала *«хорошо определены»* [672]. Под этим мы понимаем, что для каждой задачи в нашем распоряжении имеется какой-то систематический метод, позволяющий определить, когда предложенное решение приемлемо. Бо́льшая часть экспериментальных работ, обсуждаемых в данной статье, связана с такими «хорошо определенными» задачами, с которыми мы сталкиваемся при доказательстве теорем или в играх с точными правилами игры и подсчета очков.

В некотором смысле все такие задачи тривиальны, ибо, если существует решение такой задачи, это решение можно в конце концов отыскать с помощью любого слепого исчерпывающего поиска, перебирающего все возможные варианты решения. И обычно такой поиск не трудно автоматизировать или запрограммировать.

Но для любой достаточно сложной проблемы, достойной так называться, поиск среди всех возможных решений оказывается слишком неэффективным при его практическом использовании. С другой стороны, системы, подобные игре в шахматы или нетривиальным разделам математики, слишком сложны для полного анализа.

[1] Прилагательное «эвристический» в том смысле, как оно применяется в данной статье и очень широко используется в литературе, означает «помогающий повысить способность решать задачи». Как существительное это слово используется применительно к любому методу и способу, связанному с улучшением эффективности системы для решения задач. «Эвристическая программа», оцениваемая как хорошая, должна успешно работать в применении к различным задачам, и следует мириться с тем, что она при разрешении некоторых проблем терпит неудачу. Часто мы считаем целесообразным применение эвристического метода и тогда, когда он иногда приводит к неудачам, если он обеспечивает общее улучшение характеристик. Но несовершенные методы не обязательно являются эвристическими, а совершенные — не обязательно неэвристическими. Следовательно, слово «эвристический» не должно рассматриваться как нечто противоположное «надежному», что уже внесло некоторую путаницу в литературу по этому вопросу.

При отсутствии полного анализа всегда должно оставаться какое-то ядро поиска или «метод проб и ошибок». Таким образом, необходимо найти такие методы, посредством которых результаты *неполного анализа* могут быть использованы для того, чтобы сделать поиск более эффективным. Насущная необходимость в этом видна из следующего примера: поиск всех путей для игры в шашки включает 10^{40} ходов [950], в шахматы — 10^{120} ходов [812]. Даже если бы мы из всех частиц нашей Галактики построили нечто вроде вычислительной машины параллельного действия, работающей на частоте жестких космических лучей, то и в этом случае на поиск оптимальных ходов в шахматной игре потребовалось бы неимоверно большое время; поэтому мы не можем ожидать, чтобы одно только улучшение физической конструкции машины могло разрешить наши проблемы. Несомненно, что мы должны использовать все, что мы знаем, чтобы улучшить систему поиска. И мы должны также быть способны использовать результаты, полученные на промежуточных этапах работы [1, 2].

А. Относительное улучшение, подъем по градиенту и эвристические связи

Любая проблема едва ли может вызвать у нас интерес, если мы не имеем о ней никакой информации. Обычно у нас есть какая-то база, пусть даже шаткая, для выяснения, достигнуто ли *улучшение*; некоторые опыты должны оцениваться как более успешные, чем другие. Предположим, например, что мы имеем

[1] Маккарти [672] рассмотрел проблему исчислимости с точки зрения теории рекурсивных функций. Эта незавершенная, но богатая идеями работа предполагает среди прочего, что «при перечислении частично рекурсивных функций первое место нужно предоставить композициям уже имеющихся функций»

Я нахожу эту мысль очень важной, особенно в свете результатов, полученных Шенноном [980] относительно двухполюсных переключательных цепей, согласно которым «среднее значение» переключательной функции n переменных требует около $2^n/n$ контактов. Такого количества контактов обычно не бывает, когда мы создаем «интересные» для нас большие машины, по-видимому, потому, что они основаны на композиции функций, которые уже заведомо пригодны. Следует указать также на первую, основополагающую работу Ньюэлла по этому вопросу [786] и дискуссию Сэмюэля относительно минимаксного процесса [950].

[2] В 1952 и особенно в 1956 г. Эшби превосходно разобрал проблему поиска [48, 52]. (Я не уверен, однако, в пользе его идеи «ультраустойчивости», которая, по-видимому, является не более, чем свойством машины искать до тех пор, пока ее что-нибудь не остановит.)

компаратор, который отбирает лучший результат из любой пары результатов пробных действий. Однако компаратор один не может сделать задачу «хорошо определенной», пока не определена цель. Но если найденная компаратором связь между попытками «транзитивна» (т. е. из того, что A влияет на B, а B влияет на C, вытекает, что A влияет на C), то мы по крайней мере можем установить «прогресс» и дать машине задание за определенное время сделать наилучшее, на что она способна.

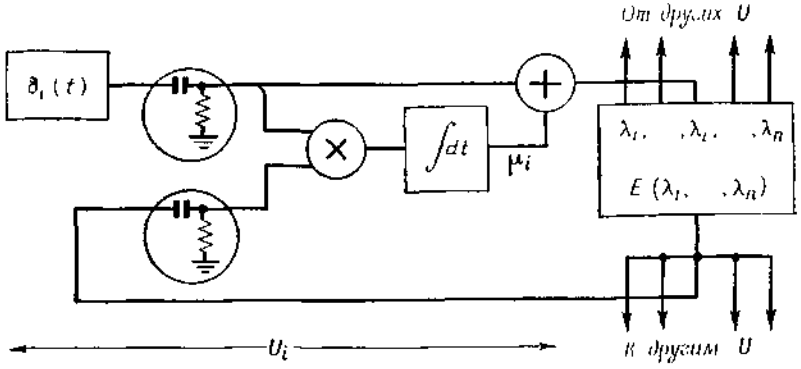
Весьма важно, однако, отметить, что сам по себе компаратор, как бы он ни был совершенен, не может дать какого-либо улучшения по сравнению с исчерпывающим поиском. Конечно, компаратор дает нам информацию о частичном успехе. Но нам необходим еще какой-то метод использования информации, который позволил бы направить поиск в перспективном для решения направлении, т. е. выбрать новые точки для пробных действий, которые в каком-то смысле были бы «похожими» или «подобными» или «лежали в том же направлении» с точками, давшими наилучшие предшествующие результаты. Для этого нам необходима какая-то дополнительная структура пространства поиска. Эта структура не должна иметь большое сходство с обычным пространственным понятием направления или расстояния, но она должна как-то объединять точки, которые связаны эвристически.

Мы будем называть такую структуру *эвристической связью* и вводим этот термин только для словесного описания, поскольку само наше определение также нестрого. Но оно нам необходимо. Многие работы были искажены неправильным употреблением для этой цели точных математических терминов, например «метрический» и «топологический». Термин «связь» (connection) со всеми его различными словарными значениями, пожалуй, является именно тем словом, которое определяет связь без указания ее точной природы.

Важным и простым видом эвристической связи является такая связь, когда пространство имеет координаты (или параметры) и в нем определена «функция успеха» E, являющаяся достаточно гладкой функцией координат. В этом случае мы можем использовать локальную оптимизацию или метод *подъема по градиенту*.

Б. Подъем по градиенту

Предположим, что машина — это черный ящик с входами $\lambda_1, \ldots, \lambda_n$ и выходом $E(\lambda_1, \ldots, \lambda_n)$. Мы хотим так подобрать входные величины, чтобы обеспечить максимум E. Но мы не имеем математического описания функции E и поэтому не можем применить дифференцирование или связанные с ним методы. Очевидный путь состоит поэтому в предварительном локальном исследовании окрестности точки и отыскании направления наиболее крутого подъема функции. Мы продвигаемся на некоторое расстояние в этом направлении и повторяем процесс до тех пор, пока не будет найдено улучшение. Если подъем плавный, это можно приблизительно выполнить путем расчета компоненты градиента $\partial E/\partial\lambda_i$ отдельно для каждой координаты λ_i. Имеются



Фиг. 1. Поиск типа «многомерной совместной оптимизации» значения (локального) максимума некоторой функции $E(\lambda_1, \ldots, \lambda_n)$ от нескольких параметров.

Каждый блок U_i независимо «вырабатывает» свой параметр λ_i, возможно, случайным образом, путем добавления вариации $\delta_i(t)$ к текущему среднему значению величины μ_i. Количественные изменения δ_i и E приводятся в соответствие, и результаты используются для (медленного) изменения μ_i. Фильтры призваны уменьшить постоянные составляющие. Такой метод одновременного действия, фактически представляющий собой некую форму когерентного обнаружения, обычно имеет преимущество над методами, независимо и последовательно воздействующими на каждый параметр (ср. дискуссию об «информационной обратной связи» у Винера [1152], стр. 133 и след.)

и более сложные методы (можно использовать шум, добавленный к каждой переменной, и определить зависимость выходного сигнала от входного; фиг. 1), но такова общая идея. Это фундаментальный метод, который всегда лежит в основе даже более сложных систем. Эвристически его основное достоинство состоит в следующем: усилия, затрачиваемые на выбор (для определения направления градиента), растут в некотором смысле линейно с увеличением числа параметров. Следовательно, если мы можем решить таким методом проблему определенного вида, включающую много параметров, то введение еще нескольких параметров такого же вида не должно вызвать чрезмерного возрастания трудностей. Нас особенно интересуют методы решения

задач, которые могут быть распространены на более трудные проблемы. Но, увы, трудности анализа самых интересных для нас систем, которые включают комбинации операций, обычно растут экспоненциально по мере добавления переменных.

Большое количество вариантов систем «подъема по градиенту» было изучено под названием «самоприспосабливающихся» или «самооптимизирующихся» следящих систем.

В. Затруднения, связанные с подъемом по градиенту

Очевидно, что подъем по градиенту завел бы в ловушку, если бы привел к достижению *локального максимума*, который оказался бы неточным или неудовлетворительным оптимумом. Тогда пришлось бы предпринимать более крупные шаги или вводить новые изменения.

Часто предполагают, что эта проблема «ложного максимума» является главным препятствием при обучении машин по этому методу. Это, конечно, может вызвать затруднения. Однако что касается действительно трудных проблем, то нам кажется, что обычно более фундаментальная проблема состоит в отыскании вообще какого-то существенного максимума. К сожалению, известные функции E для трудных проблем часто обнаруживают то, что мы назвали [740] «Mesa-феноменом»[1], при котором незначительные изменения в параметре обычно либо ведут к большим изменениям в характеристике, либо вообще не меняют характеристику. Таким образом, пространство состоит в основном из плоских участков, или «плато» (Mesa). При любой попытке делать небольшие шаги в системе возникают бесполезные блуждания, не сопровождающиеся приращением информации. Выгодный поиск в таком пространстве потребует таких больших шагов, что поиск методом подъема по градиенту по существу исключен. Для решения задачи требуется тогда найти другие методы. Однако при другом виде «эвристической связи» поиск методом подъема по градиенту может оказаться удовлетворительным.

Конечно, в нашей собственной разумной деятельности мы редко решаем запутанные задачи последовательным приближением к истинному результату. Я сомневаюсь, чтобы с помощью какого-либо одного простого метода, например метода подъема по градиенту, нам удалось бы найти пути для построения эффективной машины для решения широкого класса задач. Вероятно, «разумная машина» потребует набора различных механизмов. Все они будут организованы в иерархическую, а некоторые даже в более сложные, возможно, рекурсивные структуры. И вполне возможно, что нечто требующее на одном уровне использования подъема по градиенту, иногда (на более низком уровне) принимает форму внезапного скачка «интуиции».

[1] Mesa (*испанск.*) — стол, столовая гора, горное плато. — *Прим. ред.*

2. *Проблема распознавания образов*

Чтобы не исследовать все возможности, машина должна классифицировать проблемные ситуации по категориям, связанным с областями эффективности различных машинных методов. Такие методы, основанные на «распознавании», должны выбирать эвристически важные признаки исследуемых объектов. Самые простые методы обычно состоят в сравнении объектов с какими-то эталонами или прототипами. Более сильные методы «списка признаков» подвергают каждый предмет последовательным испытаниям на присутствие *признака* (свойства), важного в эвристическом отношении. Эти признаки не должны изменяться под воздействием обычно встречающихся форм искажения. Здесь возникают две основные проблемы: отыскание новых полезных признаков и комбинирование многих признаков для получения системы распознавания. Для решения сложных задач такие методы должны быть дополнены средствами для расчленения сложных объектов на составные части и описания сложных взаимосвязей между этими частями.

Любая сильная эвристическая программа содержит множество различных методов и приемов. На каждом этапе процесса решения задачи машина должна сначала решить, какой аспект задачи должен быть исследован, а затем — какой метод для этого должен быть применен. Здесь имеется ситуация выбора, так как обычно мы не можем позволить себе опробовать все возможности. Для того чтобы иметь дело с какой-то целью или задачей, а значит, и выбрать соответствующий метод, нужно прежде всего определить, какого типа эта задача. Таким образом, необходимость выбора среди возможных действий заставляет нас снабдить машину способами классификации или средствами их создания. Чрезвычайно важно, чтобы машина располагала такими способами классификации, которые реально осуществимы. Но термин «осуществимый» может быть определен только по отношению к конкретному внешнему миру, с которым машине придется иметь дело, и по отношению к пригодным для него методам. Отличительные особенности, которые не могут быть использованы, не заслуживают распознавания; и методы обычно бесполезны без классификационных схем, помогающих определить, когда они применимы.

А. Телеологические требования к классификации

Полезной классификацией считается такая, которая сравнивает цели и методы машины. Объекты, отнесенные к одному классу, должны иметь нечто общее в смысле их эвристического значения; они должны быть «подобными» в некотором полезном смысле и должны зависеть от соответствующих случаю или существенных признаков. Поэтому нас не должно удивлять, если мы обнаружим, что сами мы применяем для определения классов инверсные или телеологические выражения. Нам фактически надо определить «класс объектов, которые могут быть преобразованы к виду Y», т. е. такой класс объектов, который удовлетворяет некоторой цели.

К широко распространенному предубеждению против использования телеологического языка в науке следует относиться осторожно. Хотя и верно, что разговор о «цели» в некоторых случаях может привести к своего рода «анимизму», но в области решения задач не обязательно должно быть вредно говорить о целях. Трудно представить себе, как можно решать задачи, не задумываясь над целями. Действительная трудность, связанная с телеологическими определениями, носит технический, а не философский характер: она возникает тогда, когда телеологические определения приходится использовать, а не тогда, когда о них упоминается. Конечно, мы не можем себе позволить использовать для классификации метод, который фактически требует ожидания отдаленных последствий его применения, если эта классификация нужна нам именно для того, чтобы решить, стоит ли пытаться применить этот метод. Поэтому на практике идеальные телеологические определения часто приходится заменять практически пригодными приближениями, обычно с некоторым риском внесения ошибок; иначе говоря, определения должны быть *эвристически эффективными* или экономически выгодными. Это имеет огромное значение. (Мы должны представлять себе «эвристическую эффективность» как нечто противоположное обычному математическому понятию «эффективности», выделяющему такие определения, которые вообще можно реализовать с помощью машины, независимо от того, насколько это выгодно.)

Б. Образы и описания

Обычно необходимо иметь способы присвоения определенным классам *названий*, т. е. символических выражений. Структура названий будет оказывать решающее влияние на умственный мир машины, поскольку она определяет тот класс предметов, о которых машина может думать. Существует много способов

присвоения названий. Самые простые схемы используют так называемые *условные* (или *собственные*) названия; в этом случае классам ставятся в соответствие произвольные символы. Но нам может также потребоваться использовать сложные *«описательные»* или «вычислительные» названия, которые создаются для классов при помощи процессов, *зависящих от определений классов*. Чтобы быть полезными, они должны отражать какие-то структурные признаки того, что они определяют, абстрагируемые в той форме, которая соответствует области данной задачи. Понятие описания растворяется в более сложном понятии *модели*; модель, как мы ее понимаем, это вид активного описания. Это нечто по форме отражающее какие-то стороны структуры моделируемого объекта, но имеющее также некоторые характеристики действующей машины.

В разд. 3 этой статьи мы будем рассматривать «обучающиеся» системы. Поведение таких систем можно заставить изменяться подходящим образом в зависимости от их прошлого опыта. Но сами по себе простые обучающиеся системы полезны только в повторяющихся обстоятельствах; они не могут приспособиться к любым существенным видоизменениям. Нетривиальное поведение приобретается только тогда, когда обучающиеся системы обладают индуктивной способностью к классификации или распознаванию образов. Поскольку разнообразие объектов, с которым мы сталкиваемся при нетривиальном поиске, очень велико, мы не можем оставаться в зависимости от повторения, а одно только накопление прошлого опыта может дать лишь ограниченный эффект. Распознавание образов, обеспечиваемое эвристической связью, соединяющей старое с новым, может сделать обучение чрезвычайно полезным.

Что такое образ? Мы часто используем этот термин телеологически, имея в виду систему объектов, которые в некотором (полезном) смысле могут трактоваться как подобные. Для каждой задачи нужно выяснить, какие образы были бы полезны для работы машины над такой задачей.

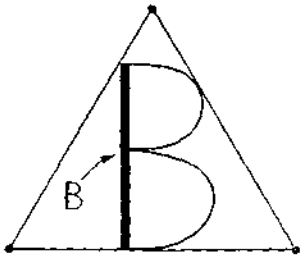
В последние годы большое внимание уделяется проблемам *визуального* распознавания образов, и бóльшая часть приводимых в данной статье примеров взята из этой области.

В. Образы, полученные с помощью прототипов

Проблема чтения *печатных* букв — яркий пример ситуации, когда классификация в итоге основывается на фиксированной системе «прототипов», т. е. штампов, по которым был отлит типовой комплект шрифта. Отдельные отпечатки литер на печатной странице могут нести следы различных искажений.

Некоторые искажения носят довольно систематический характер: изменения размеров, положения, ориентации. Другие зависят от характера «шума»: размытость, зернистость, слабая контрастность и т. д.

Если шум не слишком сильный, мы можем управлять идентификацией при помощи так называемой *«нормализации»* или процесса *сравнивания с шаблоном*. Прежде всего мы устраняем различия, относящиеся к размерам, а это значит, что мы *нормализуем* фигуру на входе. Это можно сделать, например, построив подобную фигуру, вписанную в определенный фиксированный треугольник (фиг. 2); или же можно преобразовать

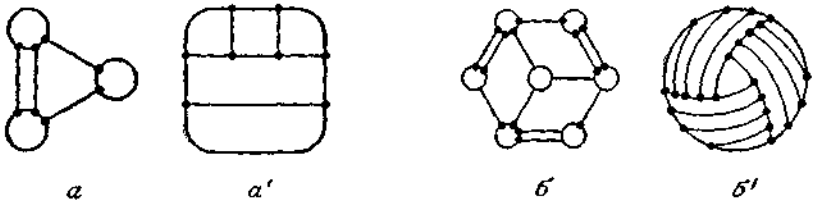


Фиг. 2. Простой метод нормализации.

Если объект равномерно растягивать во всех направлениях без вращения, пока он не коснется всех трех сторон треугольника, результирующая фигура будет единственной, так что распознавание может продолжаться без учета относительных размеров и положения образа.

фигуру, чтобы получить определенный фиксированный центр тяжести и второй момент относительно центра, равный 1. (Существует другая задача, связанная с эквивалентностью образов при их повороте, в которой трудно избежать всех затруднений. Никому не захочется, чтобы машина отождествляла «6» и «9». Точно так же мы не хотели бы, чтобы отождествлялись 0 и *o* или *X* и *x*, а также индексы 0 в выражениях x_0 и x^0, так что здесь, по-видимому, должна быть введена зависимость от контекста.) После нормализации неизвестной фигуры ее можно сравнить с *шаблонами* прототипов и при помощи какого-либо *сравнения* (измерения степени совпадения) выбрать наиболее подходящий шаблон. Каждый «критерий сравнения», как и каждый процесс нормализации, будет чувствительным к частным формам шума и искажения. Метод «вписывания» может быть чувствительным к небольшой зернистости (крапинкам, частичкам), в то время как метод момента будет особенно чувствительным к размытию, по крайней мере для символов, очерченных тонкими линиями, и т. д. Выбор критерия совпадения должен зависеть от вида шума и преобразований, с которыми обычно приходится иметь дело. Все же для многих задач можно получить приемлемые результаты путем использования прямых корреляционных методов.

Класс эквивалентных преобразований очень велик, т. е. когда имеются локальное растяжение и искажение, очень трудно бывает найти однородный метод нормализации. Вместо этого можно рассматривать процесс локальной «подгонки» изображения к шаблону. (Измеряя степень совпадения, можно локально подогнать фигуру, и если этим достигается некоторое улучшение, то процесс может быть повторен с незначительными изменениями и т. д.) Обычно практически невозможно применить к символу *все* допустимые преобразования, а распознавание *топологической* эквивалентности таких пар, как на фиг. 3, выходит, вероятно, за пределы любого практического вида итера-



Фиг. 3. Фигуры *а—а′* и *б—б′* представляют собой топологически эквивалентные пары.

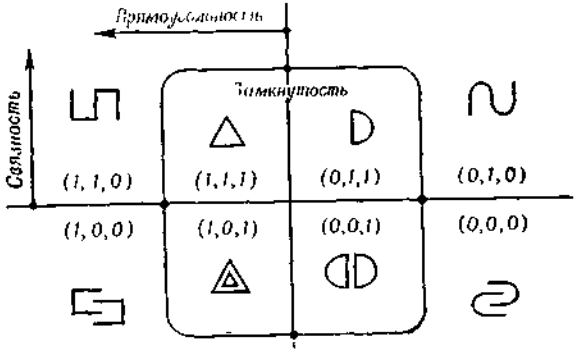
Длины искажены произвольным образом, но отношения связи между соответствующими точками сохранены. В работах Шермана [993] и Галлера [440] можно найти программы для вычислительных машин, которые способны работать с такими эквивалентными образами.

тивного процесса локального улучшения или процесса подъема по градиенту. (Такое распознавание может быть автоматизировано при помощи метода, который позволяет следовать вдоль линии, обнаруживать узлы и строить *описания* в форме, скажем, таблиц соединений узлов фигуры.)

Схема сравнения с шаблоном, основанная на нормализации путем прямого сравнения и на критерии совпадения, слишком ограничена для того, чтобы ее можно было с успехом применить в более сложных задачах. Если множество преобразований велико, нормализация или «подгонка» может оказаться неосуществимой, в особенности в тех случаях, когда в пространстве преобразований нет достаточной эвристической связи. Кроме того, система для каждого определенного образа должна снабжаться прототипом. Но если имеется в виду достаточно абстрактный класс объектов, может оказаться невозможным представить его основные особенности одним или несколькими конкретными примерами. Как можно представить единственным прототипом класс фигур, имеющих четное число несвязанных частей? Ясно, что система шаблона имеет ограниченное применение. Система «списка признаков» освобождает нас от некоторых из этих ограничений.

Г. Список признаков и „характеристики"

Мы определяем *признак* как некоторую двузначную функцию, которая делит фигуры на два класса; о фигуре говорят, что она имеет или не имеет данный признак, в зависимости от того, равна ли эта функция 1 или 0. Задав число n различных признаков, можно с помощью сочетания признаков определить 2^n подклассов и, следовательно, 2^{2^n} *образов* путем комбинации признаков при помощи логических функций «И» и «ИЛИ». Таким образом, если мы имеем три признака: *прямоугольность*,



Фиг. 4 Восемь областей, представляющие все возможные сочетания значений трех признаков: «прямоугольный», «связный» и «замкнутый»

Каждая область содержит типичную фигуру и связанную с ней последовательность двоичных чисел — их «характеристику»

связность и *замкнутость*, имеются восемь подклассов (256 образов), определенных их пересечениями (фиг. 4).

Если заданные признаки разместить в некотором фиксированном порядке, можно представить любую из этих элементарных областей вектором или рядом цифр. Вектор, определенный таким образом, будет называться *характеристикой* этой фигуры (относительно последовательности рассматриваемых признаков). (В работе «Некоторые аспекты эвристического программирования и искусственного мышления» [732] мы использовали термин «характеристический» для признака, значения которого не ограничены двумя величинами.) Таким образом, квадрат имеет характеристику (1, 1, 1), а круг — (0, 1, 1) при данной последовательности признаков.

Для многих задач можно применить такие характеристики в качестве названий для категорий и в качестве простейших элементов, с помощью которых определяется соответствующая система образов. Характеристики являются чем-то большим,

чем просто условными названиями. Они используются в качестве зачаточных форм описания (они имеют форму самого простого символического выражения — *списка*), структура которого обеспечивает некоторую информацию об отображаемых классах. Это хотя и небольшой, но все же шаг вперед по сравнению с методом шаблона; характеристики не являются простыми примерами шаблонов, а сами признаки могут быть очень абстрактными. Отыскание хорошей системы признаков является основной задачей многих эвристических программ.

Д. Инвариантные признаки

Одно из основных требований к хорошему признаку состоит в том, что он должен быть инвариантным при обычных эквивалентных преобразованиях. Так, при использовании метода визуального распознавания нам обычно хочется, чтобы идентификация объекта не зависела от однородного изменения размеров и положения. Питтс и Маккаллок [865] впервые описали общую методику получения инвариантных признаков из неинвариантных, предположив, что пространство преобразований имеет определенную (теоретико-групповую) структуру. Идея их математического доказательства состоит в следующем: предположим, что мы имеем функцию P символов, и допустим, что для данного символа F мы определяем, что $[F] = \{F_1, F_2, \ldots$ есть система всех символов, эквивалентная F при данном множестве преобразований; определим далее, что $P[F]$ есть система $\{P(F_1), P(F_2), \ldots\}$ значений P по этим символам. Наконец, пусть $P^*[F]$ есть *среднее* значение $(P[F])$. Теперь мы имеем новый признак P^*, значения которого не зависят от того, какое именно F из класса выбрано. Мы должны быть уверены, что, когда различные представители из класса выбраны, система $[F]$ в каждом случае всегда будет оставаться одинаковой. В случае непрерывности пространства преобразований должна существовать некоторая *мера*, или эквивалент, связанный с системой $[F]$, относительно которого операция усреднения определяется как интегрирование [1].

[1] В случае, разбиравшемся Питтсом и Маккаллоком [865], область преобразования составляет *группу* с однозначно определенной мерой. множество $[F]$ может быть вычислено без повторений при помощи поэтапного применения всех преобразований T_a к заданной фигуре, так что инвариантный признак можно определить по уравнению

$$P^*(F) = \int_{a \in G} P(T_a(F))\, d\mu,$$

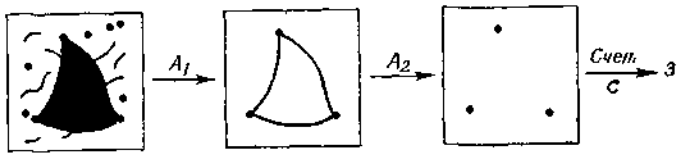
где G — группа, а μ — мера. Подставляя в это уравнение $T_\beta(F)$, мы убеждаемся, что результат не зависит от выбора β, так как мы получаем такой же интеграл по $G\beta^{-1} = G$.

Этот метод предложен Питтсом и Маккаллоком в 1947 г. в качестве нейрофизиологической модели восприятия звуков независимо от высоты тона и визуального распознавания изображения независимо от его размеров (дополненной механизмом центрирования в поле зрения). Эта модель рассматривалась также Винером ([1152], стр. 160 и след.). Ее практическое применение, вероятно, ограничивается одномерными группами и аналоговыми сканирующими устройствами.

В многочисленных работах последнего времени эта проблема обходится путем использования инвариантных относительно этих преобразований признаков. Так, например, признак «количество связанных компонентов на рисунке» является инвариантным относительно изменения размеров и положения, а признак «количество вертикальных линий на рисунке» инвариантен относительно размеров и положения (но не вращения) изображения.

Е. Формирование признаков

Проблема образования полезных признаков рассматривалась Селфриджем [973]; мы коротко суммируем его предположения



Фиг 5 Произвольная последовательность преобразований изображения, сопровождаемая численной функцией, может быть использована в качестве функции *признака* изображения.

A_1 устраняет все точки, которые не находятся на границе зачерненного участка. A_2 оставляет только вершинные точки, на которых дуга внезапно меняет направление. Функция C просто подсчитывает количество оставшихся точек. Все замечания в тексте можно обобщить так, чтобы они были приложимы к таким признакам, как A_1, A_2, C, которые могут принимать более двух значений.

по этому вопросу. На начальном этапе работы машине дается совокупность основных преобразований $A_1, \ldots, A_n$, каждое из которых преобразует некоторым существенным образом одно изображение в другое. Преобразование A_1 может, например, устранить все точки, расположенные *не на границе* зачерненной зоны; A_2 может оставить только точки, расположенные *в вершинах*; A_3 может *заполнить пустые зоны* и т. д. (фиг. 5). Каждая последовательность $A_{i1}, A_{i2}, \ldots, A_{ik}$ этих преобразований дает новое преобразование, и таким образом можно получить их бесконечное многообразие. Мы снабжаем машину также одной или несколькими операциями «окончания», так что любая последовательность элементарных преобразований, за которой

следует операция окончания, определяет признак. (Дайнин [284] описывает, как такие процессы были запрограммированы на цифровой вычислительной машине.) Мы можем начать с нескольких коротких последовательностей, которые можно выбрать произвольно. Селфридж ([973], стр. 93) следующим образом описывает процесс изучения машиной новых полезных признаков:

> «Будем показывать машине буквы А и О, сообщая ей каждый раз, какая это буква. Кроме последовательностей из двух букв, машина строит функцию распределения результатов попыток применения этих последовательностей к предъявляемому ей изображению. Поскольку последовательности были выбраны совершенно произвольно, может оказаться, что большинство последовательностей имеет очень плоские функции распределения, т. е. не несет никакой информации, и поэтому по определению не является значимым. Пусть машина отбросит их и выберет какие-то другие. Рано или поздно некоторые последовательности окажутся значимыми, т. е. их функции распределения где-то достигнут максимума. Теперь машина должна только строить последовательности, похожие на значимые. Это самый существенный момент. Если выбирать последовательности случайным образом, потребуется очень много времени, чтобы найти действительно самые лучшие последовательности. Но при использовании некоторых последовательностей, полностью или частично приводящих к успеху, можно ожидать, что этот процесс будет протекать гораздо быстрее. Остается решить коренной вопрос о том, как мы строим последовательности, «подобные», но не идентичные другим последовательностям. Пока мы полагаем, что будем просто строить последовательности, исходя из частот перехода значимых последовательностей. Мы будем строить матрицу частот перехода и используем последовательности как вероятности перехода при выборе новых последовательностей.
>
> Мы не настаиваем на том, что это действительно самый хороший способ выбора последовательностей, а лишь утверждаем, что это лучше, чем вовсе не использовать знания того вида последовательностей, который оказался значимым. Нам кажется, что все это имеет решающее значение при обучении».

Было бы действительно странно, если бы таким способом не удалось создать более полезные признаки, чем при совершенно произвольном отборе последовательностей. Проблема вырабатывания признаков рассматривалась также Минским [728].

Ньюэлл, Шоу и Саймон [799] описали более разработанный и менее статистический способ, который можно использовать при отыскании наборов признаков, подходящих для данной задачи.

Предложение Селфриджа можно рассматривать как систему, которая использует для описания своих признаков язык конечных состояний. Соломонов [1031, 1035] предложил методику отыскания общих черт системы выражений, т. е. описаний признаков, полезность которых уже была установлена; эти методы затем можно использовать для вырабатывания новых признаков с теми же общими чертами. Общее направление поисков Селфриджа [973], Ньюэлла, Шоу и Саймона [798] и Соломонова [1035, 1032] мне представляется чрезвычайно важным, хотя их исследования пока и не завершены.

Ж. Комбинирование признаков

Нельзя рассчитывать на то, что удастся легко найти *небольшой* набор признаков, полностью описывающих проблему. Обычно гораздо легче найти большой их набор, в котором каждый признак дает лишь немного полезной информации. При этом мы сталкиваемся с проблемой нахождения способа такого комбинирования этих признаков, чтобы получилось желаемое описание. Самым простым методом является выбор для каждого класса типичной характеристики (частной последовательности значений признаков) с последующим использованием какой-то процедуры сравнения, т. е. подсчета числа совпадений и несовпадений для сравнения неизвестного с этими выбранными «характеристическими прототипами». Линейная весовая схема, описанная ниже, представляет собой некоторое обобщение этого. Такие методы рассматривают признаки как более или менее независимые свидетельства в пользу или против сделанных предположений; более общая методика (о которой мы имеем еще мало практически пригодной информации) должна уметь учитывать также нелинейные соотношения между признаками, т. е. должна содержать весовые члены для связанных подмножеств значений признаков.

1. **Схемы условной вероятности для комбинирования независимых признаков.** Рассмотрим один эксперимент, в котором объект помещается перед машиной, работающей по принципу списка признаков. Каждый признак E_i имеет значение 0 или 1. Предположим, что определена система «классов объекта» F_j, и мы хотим использовать выходные данные этого эксперимента, чтобы решить, к какому классу принадлежит объект.

Допустим, что ситуация в основном вероятностная и известна вероятность p_{ij} того, что если объект принадлежит классу F_j, то i-е свойство E_i имеет значение 1. Допустим далее, что эти свойства независимы, т. е. даже при заданном F_j знание значения E_i ничего не говорит нам о значении различных E_k в том же эксперименте. (Это, как будет показано ниже, жесткое условие.) Пусть φ_j — абсолютная вероятность того, что объект принадлежит к классу F_j. Пусть, наконец, в нашем эксперименте V является той частной системой значений i, для которых E_i равны 1. Тогда это V представляет характеристику объекта. Из определения условной вероятности имеем

$$P(F_j,V)=P(V)\cdot P(F_j|V)=P(F_j)P(V|F_j).$$

Мы хотим по заданной характеристике V определить значение F_j (при наименьшей вероятности ошибки, т. е. при так называемой оценке *максимального правдоподобия*); иначе говоря, мы хотим узнать, для какого значения j величина $P(F_j|V)$ является наибольшей? Поскольку в приведенном выше равенстве $P(V)$ не зависит от j, нам остается только вычислить, для какого значения j величина $P(F_j)P(V|F_j)=\varphi_j P(V|F_j)$ наибольшая. Таким образом, по нашей гипотезе независимости мы должны найти максимум выражения

$$\varphi_j \prod_{i\in V} p_{ij} \prod_{i\in\bar V} q_{ij} = \varphi_j \prod_{i\in V} \frac{p_{ij}}{q_{ij}} \prod_{\text{по всем } i} q_{ij}. \qquad (1)$$

Такие операции решения, основанные на «максимальном правдоподобии», может выполнить простое устройство, изображенное на фиг. 6[1].

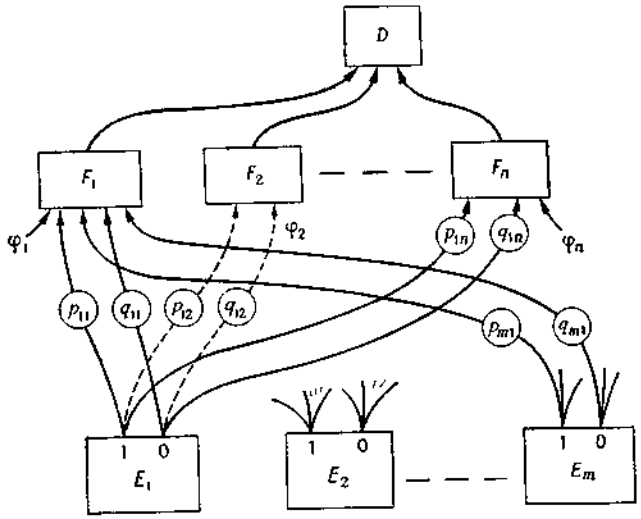
Эти схемы похожи на общие схематические диаграммы, предложенные в модели «Пандемониума» Селфриджем (см. [976], фиг. 3). Там было сделано предположение, что некоторые интеллектуальные процессы могли бы быть осуществлены посредством иерархии одновременно функционирующих блоков машины, названных «демонами». Каждый блок предназначен для обнаружения некоторых образов, а на выходе каждого блока определяется степень уверенности в том, что блок видит то, что он ищет. Наши блоки E_i являются «информационными

[1] Если добавить еще один ряд блоков, то можно было бы подсчитать все возможные g_{jk}, которые встретились бы, если относить к F_k фигуру, принадлежащую на самом деле к F_j; в этом случае наиболее выгодное решение определяется значением k, для которого

$$\sum_j g_{ik}\varphi_j \prod_{i\in V} p_{ij} \prod_{i\in\bar V} q_{ij}$$

наименьшее. Здесь $\bar V$ — множество, дополнительное к V, а $q_{ij}=1-p_{ij}$.

демонами» Сэлфриджа, а блоки F_j — его «познающими демонами». Каждый из них собирает из абстрактных данных доказательства в пользу того или иного предположения. Самый верхний «демон решения» D реагирует на тот из множества расположенных ниже блоков, сигнал которого сильнее всего (см. также [977]).



Фиг. 6. Структурная модель промежуточных решений по максимальному правдоподобию, основанная на линейном взвешивании значений признаков.

Входные данные исследуются каждым из «фильтров признаков» E_i. Каждый из блоков E_i имеет два выходных канала: «0» и «1», один из которых возбуждается от каждого сигнала на входе. Вероятности возбуждения этих выходов изображаются соответствующими значениями p_{ij}, как указано в тексте. Результирующие сигналы перемножаются в блоках F_j, каждый из которых «собирает доказательства» для классификации фигуры. (В данном случае мы могли бы воспользоваться выражением $\log(p_{ij})$ и в блоках F_j производить *сложение*). Окончательное решение принимается самым верхним блоком D, который просто отбирает наибольшее значение величины F_j. Заметим, что логарифм коэффициента p_{ij}/q_{ij} во втором выражении (1) может быть построен как «вес доказательства» E_i в пользу F_j (см. также работы [822, 930]).

К этой схеме условной вероятности легко добавить устройство, которое позволит ему «заучивать» оптимальные веса связей. Представим себе, что после каждого события машине сообщают, каким именно было событие F_j. Это можно выполнить, посылая сигнал обратно по линиям связи, ведущим к блоку F_j.

Предположим, что связь для p_{ij} (или q_{ij}) содержит двухполюсное устройство (или «синапс»), которое запоминает число w_{ij}. Всякий раз когда происходит сложное событие (F_j, $E_i=1$), мы изменяем w_{ij} путем замены его на $(w_{ij}+1)\theta$, где θ — число, несколько меньшее единицы. Когда же происходит сложное событие (F_j, $E_i=0$), мы уменьшаем w_{ij}, заменяя его на $(w_{ij})\theta$. Нетрудно показать, что ожидаемые значения величин w_{ij} станут пропорциональны значениям p_{ij} (фактически они будут приближаться к $p_{ij}[\theta/(1-\theta)]$). Таким образом, машина на опыте «обучается» оптимальному взвешиванию. (Аналогичное устройство должно быть введено в машину и для оценки значения φ_j). Вариация нормированных весов $w_{ij}[(1-\theta)/\theta]$ стремится к $[(1-\theta)/(1+\theta)]p_{ij}q_{ij}$. Следовательно, небольшая величина θ приводит к быстрому обучению, но она связана с большим разбросом (вариации w_{ij}) и, следовательно, с низкой надежностью. Выбор значения θ, близкого к единице, ведет к медленному, но надежному обучению. θ фактически является чем-то вроде постоянной затухания памяти, и ее выбор должен определяться шумом и устойчивостью окружающей среды — значительный шум требует большого времени усреднения, в то время как меняющаяся среда требует быстрой приспособляемости. Эти требования, конечно, являются несовместимыми, и решение должно быть поэтому основано на каком-то экономичном компромиссе (см. также [740, 822]).

2. Возможности использования случайных сетей для схем условной вероятности. Схемы, представленные на фиг. 6, по своей структуре довольно упорядочены. Но необходима ли вся эта структура? Конечно, если признаков очень много и *каждый обеспечивает очень небольшую частную информацию,* нежелательно упустить хотя бы одну из них. Тогда можно ожидать положительных результатов при простом переборе всех возможных путей связи w_{ij}. Таким образом, *в этом частном случае* можно использовать цепь случайной связи.

Данная двухрядная структура напоминает структуру «перцептрона», предложенного Розенблаттом [930]. Схема перцептрона имеет дополнительный уровень связей, идущих прямо от произвольно выбранных точек «сетчатки». Здесь признаки — устройства, абстрагирующие визуальные входные данные, — представляют собой просто функции, которые добавляют одни входные сигналы, вычитают другие и определяют, превышает ли результат пороговое значение. По нашему мнению, уравнение (1) показывает, что́ в этой схеме содержится ценного. Ясно, что с помощью таких сетей можно провести анализ выходного сигнала функций признаков по максимальному правдоподобию. Но подобные схемы с их простыми произвольно созданными

соединениями никогда, вероятно, не дадут возможности распознать такие образы, как «класс фигур, имеющих две разделенные части», они не смогут даже достигнуть эффекта узнавания путем сравнения с шаблоном без нормализации размеров и положения (если только фигуры образца не будут предварительно показаны по существу во всех размерах и положениях). Это связано с тем, что с помощью случайных методов весьма мало шансов получить достаточное число полезных признаков для образов, значительно более отвлеченных, чем в случае использования прототипов. Такие сети фактически могут только выделять (посредством взвешивания) информацию в признаках, полученных от отдельных входов; они не могут извлекать последующую информацию, представленную в неаддитивной форме. Класс машин типа перцептрона не способен вырабатывать признаки, лучшие, чем случайные, или образовывать их комбинации, лучшие, чем аддитивные комбинации случайно образованных признаков[1].

Для распознавания *нормализованных* печатных или рукописных символов пригодны одноточечные признаки (Хайлимен и Каменецкий [468]); это как раз соответствует «усреднению» многих образцов. Бледсоу и Браунинг [129] заявляют о хороших результатах, полученных ими при применении парноточечных признаков. Робертс [918] описывает серию экспериментов в этой области. Дойл [285] без нормализации, но с весьма усложненными признаками получил превосходные результаты; его признаки уже в значительной степени являются размерно и позиционно инвариантными. Общий обзор работ Дойла и других экспериментов по распознаванию образов можно найти у Сэлфриджа и Нейсера [977].

Что касается сложных различений, например, между одним и двумя связанными объектами, то проблема признаков является очень серьезной, особенно для большого количества связанных объектов, таких, которые трактуются Киршем [551]. В этом случае требуется какого-то рода рекурсивный процесс; комбинации же простых признаков почти не приносят успеха даже при больших сетях и длительном обучении.

Схемы условной вероятности имеют ряд ограничений. Предположение о том, что для данного значения j значения p_{ij} представляют независимые события, фактически является очень жестким условием. Без этого предположения мы могли бы еще построить схемы максимального правдоподобия, но нам потребовался бы дополнительный ряд ячеек, чтобы представить все сложные события V, т. е. нам необходимо было бы знать все значения $P(F_j|V)$. Это обеспечивало бы общее (но тривиальное) решение, однако потребовало бы 2^n ячеек для n признаков, что совершенно неприемлемо для больших систем. Итак, требуется система, которая вычисляла бы некоторые выборки из всех связанных условных вероятностей и использовала их при необходимости для оценки других вероятностей. Работы Аттли [1102—1106] относятся к этой проблеме, но предложенные и созданные им экспериментальные устройства пока еще не показывают достаточно отчетливо, как избежать экспоненциального роста (см. также Паперт [822]).

[1] См. также работы Робертса [918], Паперта [822] и Хоукинса [454]. Мы не нашли в работе Розенблатта [930] или его последующих работах ничего, что напоминало бы анализ уравнения (1).

3. Расчленение объектов. Ограничение метода „списка признаков"

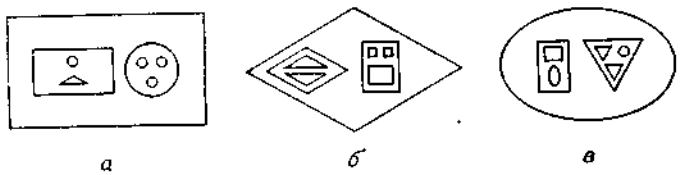
Благодаря своему фиксированному размеру схема метода «списка признаков» ограничена (для некоторого заданного набора признаков) с точки зрения тех подробностей, которые она может различить. Схема оказывается очень слабой, когда приходится иметь дело со сложной картиной, содержащей несколько объектов, а размеры ее громоздки и неоправданны. Если машина может опознать стул или стол, она, несомненно, сможет сказать нам, что «имеются стул и стол». Мы можем до некоторой степени изобрести признаки, которые создадут возможность наложения характеристик объектов[1]. Но еще нет способа избежать ограничения информации.

Можно ясно сформулировать, что требуется: 1) *список (любой необходимой длины)* простейших объектов, имеющихся в изображении, и 2) сведения о соотношениях между ними. Таким образом, о фиг. 7, *а* можно сказать: «Прямоугольник (1) содержит две подфигуры меньшего размера, размещенные по горизонтали. Левая подфигура представляет другой прямоугольник (2), в котором также размещены две подфигуры по вертикали: верхняя — круг (3), а нижняя — треугольник (4). Фигура, расположенная справа ... и т. д.»

При таком описании необходимо иметь возможность разделить или «расчленить» картину на части. (Заметим, что в этом примере расчленение существенно *рекурсивно*; сначала фигура делится на две части; затем, при помощи того же механизма, описывается каждая часть в отдельности.) Можно формализовать этот вид описания на языке выражений, где основной грамматической формой является пара (R, L), первый член которой

[1] Ср метод Муэрса [748, 749]

R (relation) означает *отношение*, а второй L (list) — *упорядоченный список* $(x_1, x_2, \ldots, x_n)$ объектов, или «подфигур», связанных друг с другом этим отношением. Требуемая гибкость достигается тем, что члены списка L содержат не только названия «элементарных» фигур, но также и «подвыражения» в форме



Фиг. 7

Картина *а* описана в тексте сначала словами. Затем мы путем введения символических обозначений для отношений «внутри», «слева» и «выше» строим символическое описание. Такие описания могут быть образованы машинами, и они способны оперировать с ними. Абстрагируясь от сложных отношений между частями фигуры, мы можем использовать ту же формулу для описания связанных с предыдущей картин *б* и *в*, изменив лишь список простейших частей. Программист должен решить, на каком именно уровне сложности часть картины должна рассматриваться как «простейшая»; это будет зависеть от того, для чего должно быть использовано описание. Далее можно было бы разделить рисунки на вершины, прямые линии и дуги. Очевидно, что для некоторых применений отношения должны содержать больше метрической информации, например указания на величину длин и углов.

(R, L), обозначающие сложные подфигуры. В этом случае описанную выше фигуру можно изобразить выражением:

$$\left[\odot, \left(\square, \left(\rightarrow, \left\{\left(\odot, \left(\square, \left(\downarrow, (\bigcirc, \triangle)\right)\right)\right), \left(\odot, \left(\bigcirc, \left(\triangledown, (\bigcirc, \bigcirc, \bigcirc)\right)\right)\right)\right\}\right)\right)\right],$$

где $(\odot, (x, y))$ означает, что y содержится в x; $(\rightarrow, (x, y))$ означает, что y находится справа от x; $(\downarrow, (x, y))$ значит, что y ниже x, и, наконец, $(\triangledown, (x, y, z))$ означает, что y находится справа от x, а z — ниже и между ними. Символы $\square$, $\bigcirc$ и $\triangle$ означают простейшие геометрические объекты.

Этот язык описаний с помощью пар выражений можно рассматривать как простую разновидность языка «списковой структуры». Ньюэлл, Шоу и Саймон первые создали мощную вычислительную методику для обращения с символическими выражениями в таком языке для целей эвристического программирования (см. замечания в конце разд. 4).

Если некоторые из членов списка сами представляют собой список, то они должны быть взяты в скобки, чем и объясняется такое обилие скобок в написанном выражении.

Может потребоваться построить описания, в которых выделяется сложное соотношение, так что мы будем, например, иметь выражение в форме FG, где F обозначает сложное соотношение между всеми простейшими частями, перечисленными в G. При этом возникает проблема «связи» переменных, т. е. определения способа, при помощи которого элементы из G участвуют в соотношении F. С этой проблемой в общем случае удается справиться путем введения λ-символики [674], но здесь мы можем пользоваться только целыми числами для упорядочения переменных.

В данном примере можно описать относительную часть F выражением

$$\odot\Big(1, \rightarrow\Big(\odot(2, \downarrow(3, 4))\Big), \odot(5, \triangledown(6, 7, 8))\Big)\Big),$$

в котором мы теперь используем «функциональную запись»; так, „$(\odot, (x, y))$" для удобства заменено обозначением „$\odot\ (x, y)$" и т. д.

Чтобы получить желаемое описание, это выражение нужно приложить к упорядоченному списку простейших объектов, которыми в этом случае являются ($\square$, $\square$, $\bigcirc$, $\triangle$, $\bigcirc$, $\bigcirc$, $\bigcirc$, $\bigcirc$). Эта сложная функциональная форма позволяет нам абстрагировать сложное соотношение. Изменением одного только списка объектов мы можем получить также описание объектов, изображенных на фиг. 7, *б* и *в*.

Важным в таких «расчлененных» описаниях является то, что их можно получить *повторным применением фиксированного множества приемов распознавания*. Таким путем мы можем получить описания *произвольной сложности* с помощью механизма классификации фиксированной сложности. Здесь от машины требуется (кроме возможности манипулировать списковыми структурами) способность «расчленять», т. е. полностью сосредоточивать внимание на выбранной части картины и использовать все средства для решения именно этой задачи. В эффективных решающих программах такое описание обычно неосуществимо в одной-единственной операции. Глубина и подробность описания достигаются другими процессами. Более глубокое и тщательное рассмотрение объекта будет выполняться только тогда, когда это необходимо, например если имеющееся описание не годится для непрерывно изменяющихся условий. Автор совместно с Ходсом (L. Hodes) работает в настоящее время над

схемами распознавания, использующими расчлененные описания. Манипулируя формальными описаниями, мы получаем возможность рассматривать перекрывающиеся или незавершенные фигуры, а также решать другие проблемы «гештальт»-типа.

По-видимому, системы пассивной классификации становятся все менее удовлетворительными по мере того, как приходится решать все более сложные проблемы; в этом случае следует обращаться к схемам, основанным на создании гипотез самой машиной, возможно, с «контролем ошибок», как, например, было предложено Маккеем [640].

Из-за недостатка места мы вынуждены ограничиться этим беглым обсуждением процессов распознавания и описания. Среди нерассмотренных здесь важных работ необходимо упомянуть работы Бомба [141] и Гримсдейла с сотр. [424, 425], касающиеся различных элементов описания, работы Унгера [1098] и Холленда [481] для схем параллельной обработки, работу Хебба [458], касающуюся физиологических моделей описания, и работы «гештальт»-психологов, в частности Колера [569], который, если и не решил, то во всяком случае поставил много важных вопросов. Шерман [993], Холлер [440] и другие создали программы, использующие линейно-следящие операции для топологической классификации. Статьи Селфриджа [973, 974] оказали решающее влияние на работу в этой области.

Следует указать также на работы Кирша и др. [552], обсуждающих ряд интересных методик машинного распознавания образов, и на работы Майнота [724] и Стивенса [1052], посвященные читающим машинам и относящимся сюда проблемам. Полезно ознакомиться также с некоторыми биологическими работами, например с работами Тинбергена [1073], из которых, в частности, видно, что на первый взгляд очень сложные различения иногда могут быть достигнуты на основе небольшого числа простых признаков, расположенных в виде простого «дерева решений».

3. Обучающиеся системы

При решении всякой новой задачи следует прежде всего попытаться использовать методы, сходные с теми, которые оказались эффективными при решении аналогичных проблем. Применяя такую «основную обучающую эвристику», нам приходится опираться на прошлый опыт, что, в частности, можно осуществить с помощью решающих моделей с «подкреплением успеха». Было показано, что такие обучающиеся системы относятся к усредняющим устройствам. Используя устройства, которые дополнительно заучивают также, какие события связаны

с подкреплением (поощрением), можно построить более автономные системы «вторичного подкрепления». При попытке применить такой метод к решению сложных задач приходится столкнуться с серьезной трудностью — с распределением оценки вероятности успеха в сложной стратегии среди различных возможных решений. С этой трудностью удается справиться, если предусмотреть локальное подкрепление частичного успеха в иерархии целей и введение «тренирующей последовательности» задач параллельно с процессом постепенного «созревания» способностей машины.

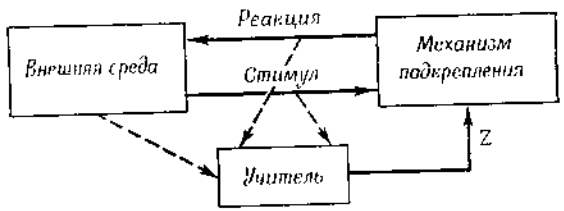
При решении новой задачи обычно используют так называемую основную обучающую эвристику, состоящую в том, что вначале пытаются воспользоваться методами, оказавшимися эффективными при решении подобных задач в прошлом. Мы и от машин также требуем, чтобы они использовали опыт прошлого! Поскольку нельзя ожидать, чтобы новые условия в точности соответствовали старым, для успешного обучения машины необходимо снабдить ее некоторой способностью к *обобщению*. Трудно дать точное определение термина «обучение», так как с «обучением» связано слишком много различных понятий. Однако можно с уверенностью сказать, что любая эффективная обучающаяся система должна использовать результаты прошлого опыта как *основу для более общих предположений*, т. е. она должна выполнять что-то вроде «индуктивного вывода» (см. разд. 5Б). Возможно, что простейшим способом обобщения внутри множества признаков является построение «идеального» или, скорее, типичного члена данного множества; обычно это может быть осуществлено путем сглаживания различий посредством некоторого процесса усреднения. И действительно мы убеждаемся, что большинство *простых* обучающихся устройств связано с некоторой техникой усреднения — часто с усреднением произведения определенного вида, что дает своего рода корреляцию. Мы обсудим данный тип устройств в следующем разделе, а некоторые более абстрактные схемы в разд. 5.

А. Подкрепление

Процесс подкрепления — это такой процесс, при котором применение некоторого «оператора подкрепления» Z заставляет в будущем усилиться (или ослабиться) некоторые аспекты поведения системы. Необходимо, чтобы этот оператор влиял только на те аспекты поведения системы, которые действительно обнаруживались в недавнем прошлом.

Здесь можно провести аналогию с воздействием «поощрения» или «подавления» (но не наказания) на поведение животных.

Важным моментом в процессах такого рода является то, что они представляют собой «операнты» (по терминологии Скиннера [1021]); подкрепляющий оператор не создает поведения, а просто выбирает желательный «учителю» вид поведения из имеющихся видов. Такая система должна содержать устройство M, создающее многообразие видов поведения (скажем, во взаимодействии с некоторой окружающей средой), и устройство («учитель»), которое критически оценивает и выбирает подходящие операторы подкрепления (фиг. 8).



Фиг 8. Составные части обучающейся системы с «оперантным подкреплением».

На внешние стимулы машина дает один из возможных ответов и запоминает, какие решения принимались при выборе этого ответа. Вскоре вслед за этим «учитель» посылает в машину положительный или отрицательный подкрепляющий сигнал (поощрение); это увеличивает или уменьшает тенденцию принимать в будущем то же решение. Заметим, что «учитель» вовсе не должен знать, как решать задачу, а должен лишь знать, как обнаружить успех или неудачу в ее решении или относительное улучшение характеристик решения, т. е. его функция селективна. «Учитель» должен быть в состоянии наблюдать фактические реакции на стимулы, а для более интересного класса систем и некоторые функции состояния окружающей среды

Рассмотрим простую модель подкрепления. Предположим, что при каждом применении стимула S животное должно сделать выбор, например повернуться влево или вправо, и что вероятность того, что при n-м испытании оно повернется вправо, равна p_n. Пусть мы хотим, чтобы животное повернулось вправо. Всякий раз когда оно будет делать это, мы можем «подкрепить»[1] его действие, применяя оператор Z_+:

$$p_{n+1} = Z_+(p_n) = \theta p_n + (1-\theta); \quad 0 < \theta < 1,$$

[1] Собственно говоря, функции подкрепления должны зависеть как от p, так и от предыдущей реакции — подкрепление должно уменьшать p, если животное только повернулось влево. Применяющиеся в литературе обозначения в этом отношении не вполне четки.

который увеличивает p на долю $(1-\theta)$. Если нам не нравится действие животного, мы применяем отрицательный оператор

$$p_{n+1} = Z_-(p_n) = \theta p_n,$$

который стремится уменьшить p до 0. Теорию таких «линейных» операторов обучения, обобщенную для нескольких стимулов и реакций, можно найти в работе [182]. Можно показать, что результат обучения — это усредненная величина, весовым множителем в которой служит экспоненциально затухающий со временем коэффициент. Пусть $Z_n = \pm 1$ (знак плюс — при поощрении, а знак минус — при подавлении n-го события); заменим p_n на $c_n = 2p_n - 1$, так что $-1 \leqslant c_n \leqslant 1$, как для коэффициента корреляции. Тогда (при $c_0 = 0$) методом математической индукции получаем

$$c_{n+1} = (1-\theta)\sum_{i=0}^{n} \theta^{n-i} Z_i,$$

а так как

$$\frac{1}{1-\theta} \approx \sum_{0}^{n} \theta^{n-i},$$

то можно написать

$$c_{n+1} \approx \frac{\sum \theta^{n-i} Z_i}{\sum \theta^{n-i}}. \qquad (1)$$

Если Z_i рассматривать как произведение двух величин, отражающих соответственно а) как реагировало животное и б) какой вид подкрепления был использован, то c_n будет своего рода функцией корреляции (с уменьшающимися весами) совместного действия этих величин. Обычно однородно взвешенное среднее имеет ту же общую форму, но с θ, зависящим от времени:

$$c_{n+1} = \left(1 - \frac{1}{N}\right) c_n + \frac{1}{N} Z_n. \qquad (2)$$

В (1) мы снова встречаемся с ситуацией, описанной в разд. 2Ж1: небольшая величина θ приводит к быстрому обучению и возможности быстрого приспособления к изменяющимся условиям; величина же θ, близкая к 1, ведет к медленному обучению, но сглаживает флуктуации. Как указывалось в разд. 2Ж1, распределение реакций приближается к вероятности поощрения альтернативных реакций. (Значение этого явления, по-моему, несколько переоценивается. Это, безусловно, не

очень рациональный способ действия. Разумнее было бы вычислять числа p_{ij}, как указано, но при каждой попытке делать «наиболее подходящий» выбор. За исключением того случая, когда имеешь дело с противником в игре[1], обычно нет смысла использовать «смешанную» стратегию.)

Программа оптимизации коэффициентов Сэмюэля (см. разд. 3Б1) представляет собой наиболее остроумный компромисс между методами экспоненциального и однородного усреднения: величина N в (2) сначала равна 16 и остается таковой до тех пор, пока не достигается значение $n=16$; тогда N становится равным 32 до $n=32$ и т. д. до $n=256$. После этого N фиксируется и все время остается равным 256. Таким путем предотвращаются флуктуации c_n вначале, процесс на некоторое время приближается к однородному взвешиванию, и наконец получается экспоненциально взвешенная корреляция; и все это достигается способом, требующим очень небольших вычислительных усилий! Программа Сэмюэля является в настоящее время уникальным примером программирования игры, соответствующей средним человеческим способностям, и своим успехом (в реальном времени) она обязана искусному применению эвристик и изяществу программирования.

Проблема подавления или «отучивания» принципиально важна для сложного, иерархического обучения, поскольку если однажды сделано какое-то обобщение прошлого опыта, то вероятно, что на него в дальнейшем будут опираться. Пусть, например, в качестве важных выбраны некоторые признаки, которые затем используются для характеристики текущего опыта; возможно, информация даже будет записываться в память в этих терминах. Если позднее обнаружится, что некоторые другие признаки более подходят, придется столкнуться с проблемой «перевода» или же отказа от записей, произведенных в прежней системе. Это может обойтись очень дорого. Нелегко отказываться от раз принятого взгляда на вещи, если новый взгляд требует затраты больших усилий и опыта. Поэтому *тренировочные последовательности*, на которых обучаются наши машины, должны быть достаточно разнообразными для того, чтобы обеспечить машине базу для решения в будущем сложных проблем.

И все же, несмотря на то что здесь много места уделено обзору таких «дискретных» или «статистических» схем обучения, я не убежден, что они должны играть центральную роль в наших моделях. Они, конечно, будут появляться в качестве составных частей наших программ, но, я думаю, главным образом по ошибке. Чем умнее машина, тем чаще она сможет извлекать из опыта вполне определенный результат, например принять или отклонить ту или иную гипотезу или изменить цель. (Очевидное исключение составляют истинно статистические внешние условия, в которых усреднение неизбежно. Но центральным пунктом в решении задач всегда, как мне кажется, будет его комбинаторная часть, которая создает условия для поиска, и мы всегда должны иметь методы борьбы с неприятными «шумами».) В этой связи можно сослаться на работу Миллера, Галантера и Прибрама [715][1]. Это, по-видимому, первая серьезная работа в области психологии, стремящаяся показать влияние исследований по искусственному разуму, и предложенная в ней программа довольно детально разработана.

[1] Во многих проблемах основополагающую роль играет вопрос о том, как часто для усиления информации следует прибегать к стратегии оцениваемого оптимума. См., например, Шубик [998].

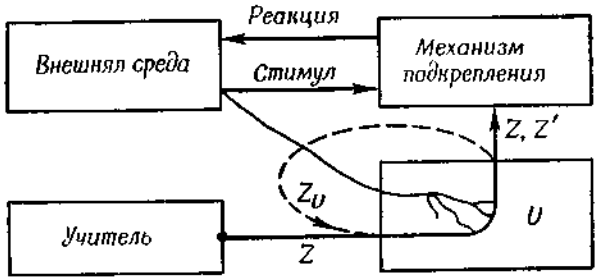
Б. Вторичное подкрепление и модели ожидания

Возможности системы простого подкрепления ограничены ее зависимостью от «учителя». Если «учитель» может обнаружить только факт *решения* задачи, то мы сталкиваемся с «Mesa»-явлением, ограничивающим способность решения трудных проблем (см. разд. 1В). Один из путей преодоления этой трудности состоит в том, чтобы научить машину обобщать действия «учителя». Тогда она при решении трудных задач сможет сама себе давать частичное подкрепление в процессе работы, например при решении соответствующих частных задач. Машина, показанная на фиг. 9, до некоторой степени обладает такой способностью.

Новый блок U — устройство, которое заучивает, какой внешний стимул сильно коррелирован с теми или иными подкрепляющими сигналами, и реагирует на такие стимулы воспроизведением соответствующих подкрепляющих сигналов. (Сам блок U *не является* подкрепляющим обучающим устройством; он больше напоминает «павловское» устройство, вырабатывающее условные рефлексы, в котором сигналы Z трактуются как «безусловные», а сигналы S — как «условные» раздражители.) Эвристическая идея состоит в том, что любой сигнал из внешнего мира, который в прошлом был сильно коррелирован, скажем с положительным подкреплением, служит вероятным указанием на то, что только что произошло нечто «хорошее». Если тренировка машины на прежних задачах была такова, что это правдоподобно, то машина в конечном итоге могла бы отделиться от «учителя» и стать автономной. Если в дальнейшем мы

[1] См. особенно гл. 10 указанной работы.

замкнем цепь «вторичного подкрепления», допуская связи, показанные на фиг. 9 пунктиром, схема может оказаться в принципе



Фиг. 9. Добавочное устройство U позволяет машине, показанной на фиг. 8, изучать, какие сигналы из внешней среды были связаны с подкреплением.

Первичные сигналы подкрепления Z проходят через U. При помощи павловского фильтрующего процесса (не описанного здесь) внешние сигналы вызывают сигналы подкрепления, похожие на те, которые часто сопровождали их в прошлом. Такие сигналы могут быть абстрактными, например словесное одобрение. Если возможны сигналы «вторичного подкрепления» для создания дальнейших внешних ассоциаций (например, через канал Z_U, как показано), то машина сможет иметь возможность управлять цепочкой подзадач. Но необходимо как-то стабилизировать систему по отношению к положительной символической обратной связи, образуемой Z_U. Возникающие при этом трудности, возможно, нашли выражение в том, что у низших животных очень трудно обнаружить такие цепеобразующие эффекты

очень мощной. Однако очевидно, что при такой степени автономности система может в конце концов стать «неадаптивной».

В. Предсказание и ожидание

Блок U предназначен для того, чтобы иметь возможность оценивать, хороша или плоха данная ситуация. Такую оценку можно применять как к воображаемым, так и к действительным ситуациям. Если бы мы могли заранее оценить результаты некоторых действий (без их фактического выполнения), то можно было бы использовать блок U для оценки ожидаемой результирующей ситуации. Это помогло бы уменьшить поиск, и наша машина фактически приобрела бы некоторую способность предвидеть, или *планировать*. Для осуществления этого необходимо добавочное устройство P, которое по заданному описанию обстановки и некоторого действия будет предсказывать описание вероятного результата. (Предназначенные для этого схемы мы рассмотрим в разд. 4Б.) Блок P может быть также сконструи-

рован как обучающееся устройство с подкреплением. В такой системе требуемые сигналы подкрепления имели бы очень удобный характер. Действительно, машина должна будет сообщить P положительное подкрепление, если *действительные результаты похожи на предсказанные*, — точные предсказания поощряются. Если бы можно было еще добавочно в виде «премии» усиливать подкрепление тех предсказаний, которые характеризуются некоторой новизной, то можно было бы ожидать встретить в машине поведение, движимое своего рода «любознательностью». Подкрепление механизмов подтвержденных новых предсказаний (или новых объяснений) может дать нам ключ к моделированию мотивов интеллектуального поведения [1].

Программа Сэмюэля для игры в шашки

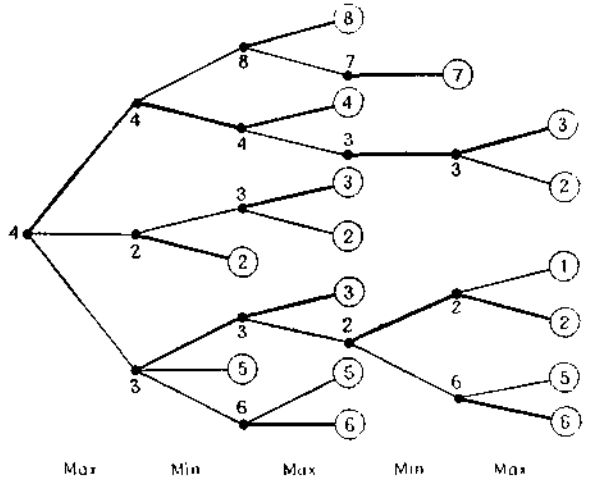
В «обучающейся обобщениям» программе Сэмюэля для игры в шашки [950] мы находим новую, эвристическую технику, которую можно рассматривать как простой пример идеи «поощрения предсказаний». Рассмотрим вкратце ситуацию при игре двух партнеров в подобных играх. Как заметил Шеннон [987], такие игры в принципе «конечны», и наилучшую стратегию в них можно отыскать путем анализа всех возможных продолжений: «. . .если он пойдет так, я могу пойти так или так и т. д. . . .», а затем возвращения назад «минимаксной» процедурой, исходя из конечных позиций — побед, поражений или ничьих. Но на практике полное исследование возникающего при этом колоссального «дерева ходов» невозможно. Конечно, некоторое исследование продолжений всегда неизбежно в подобных играх. Однако «дерево» должно быть ограничено. Можно просто поставить предел глубине исследования — числу ходов и ответов. Можно также ограничить число исследуемых продолжений в каждой позиции — это требует применения некоторого эвристического метода выбора «подходящих ходов» [2]. Но если все же приходится использовать подсчет выигрышей и проигрышей (с неполным «деревом ходов»), нужно заменить абсолютный критерий «побед, поражений и ничьих» некоторым другим «статическим» способом оценки промежуточных позиций [3] (фиг. 10). Однако простейшей схемой, вероятно, является использование взвешенной суммы некоторого выбранного множества функций,

[1] См. также гл. 6 работы Минского [726].

[2] См. работу Бернстейна и др. [120], а также обширный обзор и обсуждение в очень содержательной работе Ньюэлла, Шоу и Саймона [792].

[3] В некоторых задачах процесс «возврата» может быть описан в замкнутой аналитической форме, так что можно использовать такие методы, как «динамическое программирование» Беллмана [101]. Фреймер [343] приводит несколько примеров игр, в которых ограниченное «предвидение» не достигает цели.

отображающих свойства позиций: подвижности фигур, продвинутости пешек, контроля над центром и т. п. Это проделано в программе Сэмюэля и большинством его предшественников.



Фиг. 10. Метод «обратного прослеживания статических оценок» предполагаемых ходов на дереве игры.

Из узловой точки слева, представляющей существующее положение на доске, исходят три ветви возможных ходов, которые может сделать *игрок*. Каждому из них соответствует множество возможных ходов *противника* и т. д. В соответствии с некоторой программой создается конечное дерево игры. Затем вычисляется ценность каждой позиции для игрока (см. текст). Если ценность ее для противника такова же, он будет стремиться минимизировать итог, в то время как первый игрок всегда будет пытаться максимизировать его. Жирные линии показывают, как совершается этот «обратный» минимаксный процесс, пока не будет сделан выбор для данной позиции на доске.

Полное дерево ведения игры в шахматы имеет порядок 10^{120} ветвей, что превышает возможности любого человека или машины. Существует фундаментальное эвристическое соотношение между эффективностью функции оценки и величиной дерева игры. Очень слабая оценка (например, оценка, сравнивающая только ценность фигур у игроков) привела бы к очень расточительной игре, если бы машина могла исследовать все продолжения, скажем, до 20-го хода. Но просмотр только на 6 ходов вперед, примерно соответствующий возможностям наибольших из современных машин, дал бы, вероятно, не очень блестящую игру; менее исчерпывающая стратегия в соответствии с идеями Ньюэлла, Шоу и Саймона [792] здесь оказалась бы более выгодной.

С этим связан метод одновременной множественной оптимизации для отыскания правильных коэффициентов (с использованием упомянутой в разд. 3А техники корреляции). Новинкой в работе Сэмюэля [950] является, однако, источник подкрепляющих сигналов. На каждом шаге обучения мы не можем

до конца разыгрывать один или несколько вариантов партий. Вместо этого Сэмюэль измеряет *для каждого хода* разность между значением оценочной функции, вычисленным *непосредственно* для позиции, и тем значением, которое она *предсказывает* на основе обширного исследования продолжений, т. е. путем применения «метод возврата». Знак этой разности, или ошибки, «дельта» используется для подкрепления; таким образом, система может чему-то научиться *при каждом ходе*[1].

Г. Проблема оценки промежуточных ходов для сложных систем обучения с подкреплением

При игре в сложные игры, такие, как шахматы и шашки, или при составлении программы такой игры для вычислительной машины существует четко определенный критерий успеха — выигрыш или проигрыш игры. В ходе самой игры каждый окончательный успех (или неудача) связан с большим числом промежуточных ходов. Но если игра была успешной, как распределить оценки «успешности» среди множества принятых в ходе игры решений?

Ньюэлл заметил:

> «Крайне сомнительно, содержится ли в оценке «выигрыш, проигрыш или ничья» в применении ко всей партии достаточно информации, чтобы вообще было возможно какое-нибудь обучение за достаточно короткое время... Для того чтобы состоялось обучение, каждая разыгранная партия должна давать гораздо больше информации. Это ... достигается разбиением проблемы на составные части. Единицей успеха является частная цель. Если эта цель достигнута, то подцели подкрепляются; если нет, то они подавляются. (Фактически подкрепляется правило преобразования, способствующее достижению подцели...) Это также справедливо для построений другого вида: каждый произведенный тактический ход дает информацию об успехе или неудаче правил поиска тактики; каждый ход противника дает информацию об успехе или неудаче «выводов» правдоподобия и т. д. Количество информации, относящейся

[1] Необходимо отметить, что Сэмюэль [950] описывает также довольно удачную программу игры в шашки, основанную на регистрации и отыскании информации о встречавшихся в прошлом позициях. Это менее отвлеченный способ использования прошлого опыта. Работа Сэмюэля замечательна разнообразием выполненных экспериментов с различными эвристиками и без них. Это дает возможность действительно сравнить различные эвристические методы между собой. Желательно, чтобы возможно больше специалистов выбирало такие проблемы, для которых (при прочих равных условиях) применимы эти разнообразные методы.

к обучению, увеличивается прямо пропорционально числу механизмов, используемых играющей в шахматы машиной» ([786], стр. 108).

Мы полностью согласны с Ньюэллом в его трактовке проблемы[1]. Создается впечатление, что многие специалисты, работающие в области «самоорганизующихся систем» и «случайных нервных сетей», не сознают со всей отчетливостью важности этой проблемы. Предположим, что сложная задача (такая, как выигрыш шахматной партии) включает миллион промежуточных решений. Можем ли мы каждому из этих элементарных решений приписать одну миллионную долю успеха в общем деле? При некоторых особых обстоятельствах мы вправе так поступить, например в машинах Розенблатта [930], Робертса [918], Фэрли и Кларка [313] и др., где подкрепляемые связи в значительной степени независимы. Но способность этих машин к решению задач соответственно невелика.

В случае более сложных проблем, когда имеются иерархии решений (а не простое их суммирование на одном уровне), а приращения достаточно малы, чтобы обеспечить вероятную сходимость, время, требуемое для решения задачи, становится фантастически большим.

Для таких сложных проблем нам придется определить «успех» в несколько более узком смысле. Некоторые трудности можно обойти использованием тщательно подобранных «обучающих последовательностей», как описано в следующем разделе.

Программирующая программа Фридберга

Важным примером неудачной оценки промежуточных ходов является программа Фридберга [344, 345] для решения проблем программирования. Задача состоит в составлении программ для очень простой цифровой вычислительной машины (точнее для ее модели). Берется простая задача, например «вычислить функцию «И» для двух ячеек в памяти и поместить результат в указанную ячейку». Генерирующее устройство создает случайную (из 64 команд) программу. Программа выполняется машиной, и ее успех или неудача отмечается. Информация об успехе используется для подкрепления *отдельных команд* программы (размещенных в фиксированных ячейках) так, чтобы каждый успех увеличивал вероятность появления удачной

[1] См. также обсуждение Сэмюэлем ([950], стр. 22) оценки промежуточных ходов с точки зрения конечного результата для «дельты».

программы в будущем. (Из-за недостатка места мы не можем здесь разъяснить этот процесс более подробно.)

Таким образом, для каждой ячейки в памяти программы машина пытается более или менее независимо подобрать «хорошие» команды. Машина научилась решать некоторые очень простые задачи. Однако для этого потребовалось в 1000 раз больше времени, чем можно было ожидать при обучении ее способом чисто случайных попыток. В ч. II работы Фридберга и др. [345] эта неудача обсуждается и частично относится за счет того, что мы назвали (разд. 1В) «Mesa-феноменом». Изменяя за один раз только одну команду, машина не сделала достаточного числа шагов при поиске в пространстве программ.

Вторая работа обсуждает последовательность модификаций генератора программ и его подкрепляющих операторов. При их помощи и с некоторой «затравкой» (задавая направление решения с помощью некоторых полезных команд) была получена система, которая лишь немного уступала системе случайного выбора. Фридберг с сотр. [345] приходят к выводу, что с этими усовершенствованиями «общее превосходство таких машин над машинами без механизма подкрепления успехов состоит в том, что такой механизм можно положить в основу конструирования обучающихся машин».

С этими выводами я не могу согласиться. Мне кажется, что каждое такое «усовершенствование» по существу служит лишь увеличению размера шага при поиске, т. е. повышает случайный характер механизма. Это помогает избежать «Mesa-явления» и приближает поиск к случайному. Но нет совершенно никаких оснований считать, что действует «механизм обучения»; чтобы доказать это, следовало бы получить в системе результаты, которые были бы лучше случайных. Трудность, по-видимому, состоит в правильной оценке промежуточных решений. В работающей программе оценки могут быть назначены только целым функциональным группам команд, например подпрограммам, и, так как они действуют в рамках некоторых иерархических систем, нельзя ожидать, чтобы подкрепление отдельных команд было эффективным[1]. Следует лишь удивляться, что Фридберг и др. [345] не заметили, насколько справедливыми оказались высказанные ранее сомнения! В последнем разделе работы [345] действительно описывается некоторое улучшение, достигнутое разбиением проблемы на части и последовательным их решением. (В этом успешном примере, в котором применялось

[1] Глубокое обсуждение применимости этой схемы можно найти в работе Фридберга [344].

разбиение на подпроблемы, вовсе не используется никакой механизм подкрепления.) О некоторых аналогичных экспериментах сообщается также в работе Килберна, Гримсдейла и Саммера [549].

По моему убеждению, никакая схема для обучения или распознавания не может иметь универсального применения, если в ней не предусмотрен рекурсивный или по крайней мере иерархический механизм для использования полученных ранее результатов. Нельзя ожидать, что обучающаяся система справится с достаточно сложной задачей, если ее не подготовить к этому с помощью последовательности постепенно усложняющихся задач. Первая задача должна быть такой, чтобы ее можно было решить в приемлемое время с первоначально заданными машине возможностями. Вторая должна решаться в относительно короткое время с помощью простых и доступных комбинаций методов, развитых при решении первой, и т. д.

Если отказаться от использования рациональной «обучающей последовательности», то остаются лишь два возможных выхода: 1) увеличение исходных возможностей машины и 2) осуществление ею таких фантастических исследовательских процессов, примеры которым можно найти лишь в истории эволюции органического мира[1]. Но даже здесь, если принять точку зрения Дарлингтона [259], который подчеркивает эвристические аспекты генетических систем, нужно будет иметь развитые ранее (например, путем мейоза или кросинговера) высокоспециализированные механизмы, обеспечивающие выделение образований, предназначенных для решения подпроблем. В последнее время в связи с программированием «обучающих машин» много усилий посвящается построению обучающих последовательностей. Естественно, что в психологической литературе существует множество теорий относительно того, как сложное поведение строится из более простого. Но применительно к нашей области наиболее тщательное рассмотрение этой зависимости от «обучающих последовательностей» проведено Соломоновым [1031].

[1] Мыслимо, впрочем, построить обучающиеся механизмы, которые сами смогли бы выбирать для себя (из сложной окружающей среды) достаточно хорошие «обучающие последовательности» при задании системе первоначально сравнительно слабо развитых («незрелых») возможностей. Этого можно добиться, сделав так, чтобы последовательность целей, которых добивается первичный «учитель», на каждой ступени достаточно хорошо отвечала сложности поведения, которой система располагает в механическом смысле для распознавания и других целей. В этом направлении многое можно сделать простым ограничением глубины иерархического механизма и лишь позднее прибегать к ограниченному рекурсивному механизму.

4. Решение задач и планирование

Решение машиной действительно сложных задач потребует разнообразных средств управления. В ходе решения задачи приходится сталкиваться со множеством взаимосвязанных подзадач. Из них на каждом этапе лишь очень немногие должны подвергаться исследованию. Выбор подзадач для исследования должен основываться 1) на оценках относительных трудностей, которые встретятся при исследовании этой подзадачи, и 2) на оценках того, насколько те или иные подзадачи важны для решения общей задачи. Следом за выбором подзадач (для которых предлагается несколько эвристических методов) нужно выбрать методы решения, подходящие для выбранных подзадач. Но для решения действительно сложных задач даже применение в целях уменьшения поиска этих эвристик, работающих «шаг за шагом», не принесет результатов, и машина должна обладать способностью к анализу структуры проблемы в целом, т. е. способностью «планирования». Обсуждению подвергаются различные схемы планирования, в том числе использующие различные модели — аналоговые, семантические и абстрактные. Некоторые абстрактные модели типа «алгебра характеристик» могут быть на основе опыта или анализа построены самой машиной. Для определенности мы начнем обсуждение таких схем с описания простой, но важной системы «Логик-теоретик» (ЛТ), которая используется при решении некоторых из этих проблем.

А. Программа „Логик-теоретик" Ньюэлла, Шоу и Саймона

Не удивительно, что основным испытательным полем для ранних работ по механическому решению задач являлись области математики и теории игр, в которых правила определены с абсолютной ясностью. Машина «Логик-теоретик» Ньюэлла, Шоу и Саймона [787, 789] была первой попыткой доказать логические теоремы чисто эвристическими методами. Хотя эту программу по сравнению с возможностями обыкновенного человека и нельзя назвать блистательной (и она никоим образом не превзошла своих создателей), она все же играет роль важнейшей вехи на пути эвристического программирования и развития автоматического программирования в наши дни.

Задача программы ЛТ заключается в построении доказательств в системе Рассела—Уайтхеда для исчисления высказываний. Эта система задается в виде некоторого множества (пяти) аксиом и (трех) правил вывода. Последние определяют,

как следует применять известные преобразования, чтобы из старых теорем и аксиом вывести новые теоремы.

Программа ЛТ построена на идее «обратного пути» продвижения от конечного результата к исходным предпосылкам. Получив задание доказать теорему T, программа ЛТ ищет среди аксиом и ранее установленных теорем такую, из которой эта теорема T может быть выведена простым применением одного из трех «методов» (которые включают данные правила вывода). Если такая теорема или аксиома найдена, задача считается решенной. Поиск может оказаться также целиком безуспешным. Однако в конце концов поиск приведет к одной или нескольким «задачам», являющимся обычно теоремами, из которых непосредственно можно вывести данную теорему T. Если одна из них в свою очередь может быть доказана, главная задача будет решена. (В действительности дело обстоит несколько сложнее.) Каждая такая «подзадача» присоединяется к «списку подзадач» (после ограниченного числа предварительных попыток ее разрешения), и в дальнейшем ЛТ использует также и ее. ЛТ может приложить все свои силы к любой подзадаче, так как может использовать сам себя рекурсивным способом в качестве подпрограммы.

Эвристическая техника продвижения назад от результата создает своеобразный «телеологический» процесс, и ЛТ является предвестником более сложных систем, которые сами строят иерархии целей и подцелей. Но и при этом основная структура программы — не более чем система вставленных друг в друга поисков среди списков подзадач в памяти. Сначала мы опишем эту структуру, а затем упомянем о некоторых эвристиках, использованных для ее усовершенствования.

Структура программы такова:

1. Взять следующую задачу из списка задач. (Если задач больше нет, ВЫВОД: задача не решена.)

2. Выбрать следующий из трех основных МЕТОДОВ. (Если нет больше методов, возвратиться к 1.)

3. Выбрать следующую аксиому из списка аксиом и доказанных теорем. (Если их больше нет, возвратиться к 2.)

Затем применить МЕТОД к задаче, используя выбранную теорему или аксиому.

Если задача решена, ВЫВОД с полным доказательством.

Если результата нет, возвратиться к 3.

Если возникает новая подзадача, обратиться к 4.

4. Попытаться применить к подзадаче специальный МЕТОД (подстановка).

Если задача решена, ВЫВОД с полным доказательством.

Если результата нет, поместить подзадачу в конец списка задач и возвратиться к 3.

Изучались следующие эвристические связи: 1) *испытание на подобие* для уменьшения работы на 4-м шаге (который включает еще один поиск среди списка теорем); 2) *испытание на простоту* для выбора более легких задач из списка и 3) *испытание на невозможность строгого доказательства* для исключения из списка задач выражений, которые, по всей вероятности, неверны и, следовательно, недоказуемы. В ряде опытов использовалось «обучение» для нахождения среди ранее доказанных теорем наиболее полезной, которой должно быть оказано предпочтение на 3-м шаге. Мы не можем более подробно остановиться на эффекте этих изменений. Представляло интерес соблюдать равновесие между дополнительными затратами на управление определенными эвристическими методами и достигаемым при этом сокращением поиска. Это равновесие в некоторых случаях было близко к нарушению, когда память машины была почти вся занята.

Система была очень чувствительна к обучающей последовательности, т. е. к порядку, в котором расположены задачи. Поэтому некоторые эвристики, хотя они дают лишь незначительное общее улучшение, тем не менее влияют на класс разрешимых проблем. Как ни странно, общая эффективность ЛТ лишь немного улучшилась в результате использования одной или даже всех этих мер. Однако весь приобретенный практический опыт нашел отражение при проектировании значительно более сложной системы «GPS», описанной кратко в разд. 4Г2.

Ван Хао [1133, 1134] критиковал проект ЛТ на том основании, что, как было показано им и другими, существуют машинные методы доказательства, требующие для того же круга задач значительно меньших усилий, чем ЛТ, и имеющих то преимущество, что они в конце концов позволяют найти доказательство любого доказуемого предложения. (Программа ЛТ не располагает «процедурой решения» столь исчерпывающего характера и может оказаться несостоятельной при доказательстве некоторых теорем.) Авторы работы «Эмпирические исследования машины „Логик-теоретик"», вероятно, неосведомленные о существовании скромных по эффективности исчерпывающих методов, подкрепляют свои аргументы сравнением с одним особенно неэффективным исчерпывающим процессом. И все же я чувствую, что некоторые из выпадов Ван Хао несправедливы. Его критика направлена не по адресу. Он, по-видимому, не понял, что авторов ЛТ интересует не столько доказательство этих теорем, сколько общая проблема решения сложных задач. Комбинаторная система Рассела и Уайтхеда (с которой имеет дело ЛТ)

значительно менее проста и элегантна, чем система Ван Хао[1]. (Достаточно обратить внимание на вопросы, на которые делается упор в работах Ньюэлла, Шоу и Саймона [791, 792].)

Задачи Ван Хао, будучи *логически* эквивалентными этим задачам, *формально* намного проще. Его методы не включают никаких способов использования ранее полученных результатов (следовательно, они неизбежно быстро потеряют силу при определенном уровне сложности проблемы), в то время как программа ЛТ ориентирована главным образом именно на это. Наконец в результате того, что метод Ван Хао очень эффективен главным образом при решении *специального* множества теорем, ему просто не приходилось сталкиваться с основной эвристической проблемой: *когда следует принять решение отказаться от того или иного пути продвижения к цели*? Таким образом, превосходные результаты его программы [1133, 1134], возможно, отвлекли его внимание от эвристических проблем, которые снова неизбежно должны будут возникнуть, когда придется иметь дело с настоящей математикой.

Это не означает, что мы отвергаем важность работы Ван Хао и его критики. Он и другие, работающие над «машинной математикой», обнаружили, что существуют способы доказательства, которые гораздо более эффективны, чем ожидалось. Такая работа, несомненно, поможет при конструировании «разумных» машин, и такие методы, когда они будут разработаны, окажутся предпочтительнее, чем «ненадежные эвристические методы». Ван Хао, Девис, Патнэм и некоторые другие теперь внедряют эти новые методы в гораздо более заманчивую область доказательства теорем в исчислении предикатов (для которого способ исчерпывающих разрешающих процедур уже не применим). Мы не можем обсуждать этот вопрос за недостатком места[2], но кажется ясным, что для того, чтобы программа могла решать настоящие математические проблемы, ей придется как-то сочетать математическую изощренность методов Ван Хао с эвристической изощренностью методов Ньюэлла, Шоу и Саймона[3].

[1] Процесс, описанный Ван Хао [1133], также идет от результата к исходным положениям и может рассматриваться как обобщение метода «фальсификации» при решении о наличии тавтологии. В работе [1134] и неопубликованных ее дополнениях Ван Хао вводит более сильные методы (для гораздо более сложных задач).

[2] См работы [268, 1134]

[3] Все эти усилия направлены на облегчение поиска В этом смысле все они — эвристические программы. Так как на практике никто не использует слово «эвристический» в противоположность «алгоритмическому», специалистам следовало бы избегать бездоказательных рассуждений на этот счет Настоящая задача заключается в том, чтобы найти методы, задерживающие казалось бы неизбежное экспоненциальное разрастание дерева поиска.

Б. Эвристические методы для выбора подзадач

При построении системы для решения задач программист часто располагает набором более или менее различных «методов»; его действительная задача состоит в том, чтобы найти для программы эффективные способы решать, где и когда эти методы должны быть использованы.

Методы, которые не решают задачи, могут все же преобразовать ее так, что возникают новые задачи или подзадачи. Следовательно, в ходе решения одной задачи может оказаться необходимым решение множества взаимосвязанных подзадач. Гипотетическая «параллельная» вычислительная машина смогла бы делать это одновременно. Но даже у нее должен быть некоторый способ распределять свои возможности, так как она не может одновременно применять все свои методы ко всем задачам.

Разделим эту проблему управления на две части: 1) выбор тех подзадач, которые представляются более важными, многообещающими или по иным причинам предпочтительными, и 2) выбор метода, который можно применить к данной задаче.

В основной программе для ЛТ (см. разд. 4А) выбор подзадачи очень прост. Новые задачи кратко исследуются и (если они не решаются сразу) помещаются в конец (линейного) списка задач. Основная программа просматривает весь этот список (шаг 1), решая задачи в порядке их появления. Более мощные программы должны подходить более разборчиво к вопросам генерирования и выбора подзадач, потому что только таким образом можно ограничить излишнее «разветвление»[1].

Можно ожидать, что в более сложных системах придется рассматривать по крайней мере два аспекта каждой подзадачи: 1) ее предполагаемую «важность» — насколько ее решение будет способствовать решению основной задачи, 2) ее предполагаемую «трудность» — каких усилий потребует ее решение. Для оценки каждого из этих параметров, а также для надлежащего выбора одной из задач и определения усилий, необходимых для ее решения, нужны эвристические методы[2]. Обо всем этом изве-

[1] Как легко заметить, простая схема ЛТ имеет то свойство, что каждая возникшая задача в конце концов будет рассмотрена, даже если на шаге 3 их было создано несколько. Если бы необходимо было уделять *полное* внимание каждой возникшей задаче, то, возможно, до других ветвей очередь так никогда и не дошла бы.

[2] Интересно поставить вопрос, является ли эта проблема такой же (или весьма сходной), как рассматривавшиеся прежде. См обсуждение этого вопроса в работе Гелернтера и Рочестера [363]. С этой проблемой в общем случае можно справиться, просто *вспоминая* характеристику решавшихся прежде задач и *сравнивая* с ними новую задачу, например методами Муэрса [748]; если оказывается, что они совпадают, проводится более тщательное исследование

стно еще мало, поэтому если последующие замечания покажутся читателю несколько неясными, то это связано не только со сжатостью изложения.

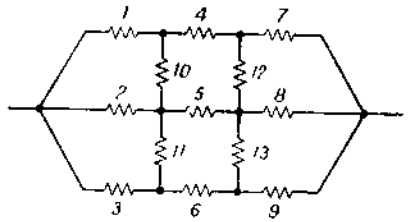
Представим себе, что задачи и их взаимные отношения образуют что-то вроде направленных графов [729, 788, 363]. Основная задача состоит в том, чтобы установить «значащие» связи между двумя первоначально различными вершинами графа. Возникновение новых задач можно изобразить добавлением новых, еще не значащих связей или введением новых вершин в старые связи. Затем задачи представляются «еще не значащими» связями, а «важность» их — размещением в структуре. С каждой связью сопоставлена некоторая величина, описывающая текущее состояние задачи (решена, правдоподобна, сомнительна и т. д.) и текущую оценку трудности решения.

1) **Глобальные методы**. Наиболее общие методы выбора задачи являются «глобальными» — на каждом шаге они просматривают всю структуру. Существует одна такая простая схема, хорошо применимая по крайней мере к элементарной интерпретации нашего графа. Она основана на электрической аналогии, подсказанной нам машиной, сконструированной Шенноном (который описал много разнообразных интересных играющих и обучающих машин [986]), для разыгрывания варинта настольной игры, продающейся под названием „Hex" (и известной среди математиков как „Ним" („Nash")). Исходная позиция может быть представлена как определенная цепь из сопротивлений (фиг. 11). Цель одного игрока заключается в построении кратчайшей цепи между двумя заданными точками, цель другого игрока — разомкнуть цепь. Каждый ход замыкает (или размыкает) необратимо одно из остальных сопротивлений. Машина Шеннона прикладывает разность потенциалов между крайними точками и выбирает то сопротивление, через которое течет наибольший ток. Грубо говоря, это сопротивление наиболее критично, так как изменение его будет оказывать наибольший эффект на сопротивление схемы и, следовательно, на замыкание (или размыкание) цепи. И хотя этот аргумент несовершенен, а модель — несовершенная модель реальной комбинаторной ситуации, машина играет исключительно хорошо. (Она может делать неверные ходы в искусственных условиях, но ничто не заставит ее сделать это во время игры.)

Использование такого глобального метода для выбора задачи требует, чтобы имеющиеся «оценки трудности» соответствующих подзадач были упорядочены приблизительно таким же способом, как и значения сопротивлений. Можно считать также

что эта машина использует для «планирования» «аналоговую модель» (см. разд. 4Г)[1].

2) **Локальные и «наследственные» методы**. Необходимость изучения на каждом этапе всей структуры задачи нежелательна, особенно потому, что обычно после каждой попытки структура задачи изменяется лишь незначительно. Естественнее стремление отыскать методы, которые бы просто *освежали* или видоизменяли небольшую часть записанной в запоминающем устройстве информации. Между этими двумя крайностями — методом подробного линейного списка задач и полными методами глобального обзора — лежит множество компромиссных методов. Возможно, наиболее привлекательными из них являются методы, которые мы назовем *наследственными* и которые носят существенно рекурсивный характер.



Фиг 11. Игра Шеннона

Эта настольная игра представляет собой цепь из равных сопротивлений Цель первого игрока — разомкнуть цепь между конечными точками, цель второго игрока — закоротить цепь Ход состоит в присоединении или отсоединении сопротивления Если первый игрок отсоединяет сопротивление *1*, второй игрок может ответить замыканием сопротивления *4*, следуя описанной в тексте стратегии. Оставшиеся пары ходов (если *оба* игрока используют эту стратегию) должны быть (*5*, *8*), (*9*, *13*), (*12*, *10* или *2*), (*2* или *10*: *выигрыш*). В этой игре первый игрок всегда имеет возможность выиграть и «стратегия максимального тока», по-видимому, всегда позволяет ему достигнуть выигрыша, даже в случае больших цепей

В «наследственном» методе усилия, которые предполагается затратить на решение некоторой подзадачи, определяются только ее предысторией. В момент возникновения задачи ей тут же приписывается некоторая величина Q, характеризующая время или затраты, которые предназначаются для ее разрешения. Позже, когда задача разбивается на подзадачи, последним также назначаются свои Q при помощи некоторого локального процесса, который *зависит только от относительных достоинств подзадач* и от оставшейся величины Q. Таким образом в явном виде решается «проблема важности».

Такие схемы очень легко программировать, особенно с помощью новых систем программирования, таких, как IPL (Ньюэлл и Тонг [800] и LISP (Маккарти [674]), которые сами основаны на определенной «наследственности» или рекуррентности операций. Специальные случаи «наследственного» метода возникают тогда, когда можно использовать простую двоичную

[1] В программе, над которой работает Р. Сильвер (лаборатория Линкольна, МТИ), разнообразные комбинаторные методы сопоставляются с аналоговыми цепями

форму Q по принципу «все или ничего», например когда наступает «прекращение вычислений». Это приводит к методу исследования задач, названному Голомбом [396] «методом возврата». Процедура декодирования Возенкрафта [1171] — другой важный вариант «наследственного» метода.

Предложенный для шахмат Ньюэллом, Шоу и Саймоном [792] сложный процесс исследования является образцом «наследственного» метода *с нечисленным условием прекращения вычислений*. Здесь подзадачи «наследуют» *набор целей, которых нужно достигнуть*. Такое телеологическое управление должно осуществляться добавочной системой выбора цели и дополняться глобальным (но достаточно простым) правилом прекращения вычислений по методу «возврата» (см. разд. 3Б). (Здесь мы отождествляем проблему ограничения «дерева игры» с проблемой выбора задачи.) Хотя подробные экспериментальные результаты работы еще не опубликованы, создается впечатление, что схема Ньюэлла, Шоу и Саймона [792] заслуживает тщательного изучения всеми, кто планирует серьезные исследования в этой области. Эта схема лишь намечает те трудности, с которыми нам неизбежно придется столкнуться при создании разумных машин[1].

В. Машины типа „символ — метод"

После того как задача выбрана, необходимо решить, какой метод должен быть испытан первым. Это зависит от нашего умения классифицировать или характеризовать проблемы. Вначале следует определить «символ» проблемы (используя методику распознавания образов), а затем проконсультироваться с таблицей «символ—метод» или другим устройством, предназначенным для определения наиболее эффективного метода (или методов) решения задач, имеющих данный символ. Первоначально такая информация может быть либо взята программистом из прошлого опыта, либо выведена из «совета» (Маккарти [673]) или же получена самой машиной в результате решения некоторых других задач, как это предлагается в GPS [795]. Во всяком случае, если рассматривать такое поведение машины со стороны, то его можно трактовать как своего рода реакцию на стимул или операцию «просмотра таблицы».

Если символы (или описания) включают слишком большой класс величин, возникает серьезная проблема запоминания таблицы «символ—метод». Вероятно, следовало бы уменьшить подробность информации, например, используя только несколько важных признаков. Так, *различия* в GPS (разд. 4Г2) описывают лишь то, что нужно для определения одиночной цели, а схема приоритетов выбирает только одно из них для характеристики ситуации. Гелернтер и Рочестер [363], использовав схему взвешивания признаков, предложили для этой цели особый вид «схемы условных вероятностей» («байесова сеть», описанная в разд. 2Ж).

[1] Дальнейшее обсуждение этого вопроса можно найти в работе Слейджла [1024].

Г. Планирование

Для решения сложной задачи обычно нужно разбить ее на несколько частей, каждая из которых требует меньшего времени поиска (или должна быть подвергнута дальнейшему разделению). Вообще говоря, удачное разбиение будет уменьшать время поиска не пропорционально, а по экспоненте. В графе с 10 ребрами, исходящими из каждой вершины, 20-этапный поиск включает 10^{20} попыток, что превышает наши возможности, в то время как введение четырех *лемм*, или *последовательных подцелей*, могло бы свести поиск только к $5 \cdot 10^4$ попыткам, которые машина вполне может исследовать. Поэтому имеет смысл приложить даже сравнительно большие усилия, чтобы выявить такие «островки» при решении сложных задач (см. Эшби [51], § 10). Заметим, что даже если каждому успеху при такой процедуре будет предшествовать, скажем, 10^6 ошибок, все равно общее уменьшение попыток составит величину порядка 10^{10}. Таким образом, *на практике любая возможность «планирования» или «анализа» окажется полезной*, если задача достаточно трудна.

Можно с уверенностью сказать, что все простые однозначные рецепты построения разумной машины проявят свою несостоятельность уже при столкновении с задачами средней трудности. Только от схем, которые активно продолжают анализ для выработки набора *последовательных целей*, можно ожидать, что они смогут постепенно осваивать решение задач все возрастающей сложности.

Наиболее «лобовой» метод планирования, вероятно, состоит в использовании упрощенной модели проблемной ситуации. Предположим, что для данной проблемы существует некоторая другая проблема «существенно того же характера», но с меньшим числом деталей и меньшей сложности. Тогда мы можем сначала приступить к решению более простой проблемы. Предположим также, что при этом используется вторая система методов, которые также проще, но до некоторой степени соответ-

ствуют оригиналу. *Решение этой более простой проблемы может быть затем использовано как «план» для решения более сложной.* Возможно, что при этом каждый этап решения придется затем развертывать со всеми подробностями. Но общее время поиска образуется как *сумма, а не как произведение* количеств времени, затраченных на поиск. Ситуация была бы идеальной, если бы модель была математически *гомоморфной* оригиналу. Но даже и без этого решение, полученное для модели, явится ценным ориентиром при решении исходной задачи. В математике процесс доказательства обычно проводится следующим путем: сначала, в стадии планирования, предполагают, например, что интегралы и пределы всегда сходятся. Когда уже есть набросок решения для этой упрощенной математической модели, возвращаются назад, чтобы попытаться сделать этапы доказательства «более строгими», т. е. заменить их цепочками доводов, использующих подлинные правила вывода. И даже если план ошибочен, его можно исправить, заменив лишь некоторые из его этапов.

Другой способ планирования — *семантическая модель*, противоположная модели гомоморфизма (Минский [728, 732]). В этом случае мы можем *интерпретировать* решаемую проблему с помощью другой, не обязательно более простой, но более знакомой, для которой уже имеются более мощные методы. Так, в связи с планом доказательства теоремы нам захочется знать, будут ли предложенные леммы, или «островки» в доказательстве, действительно *справедливы*; если нет, план будет наверняка ошибочен. Часто одного взгляда на интерпретацию достаточно, чтобы сказать, будет ли данное предложение справедливым. Так, об истинности некоторого предложения в планиметрии можно заключить, по крайней мере с большой степенью надежности, на основании фактических измерений на некоторых чертежах (или на их эквиваленте в аналитической геометрии). Геометрическая машина Гелернтера и Рочестера [363—365] с превосходными результатами использует такую семантическую модель, созданную в соответствии с предложенными Минским [728] принципами.

1) **Модель «алгебры символов».** Планирование с помощью некоторой модели является чрезвычайно ценным методом для сокращения поиска. Не можем ли мы построить машины, которые сами бы отыскивали свои модели? Я уверен, что изложенное ниже указывает общий, «лобовой» путь создания некоторых видов полезных абстрактных моделей. Важно только, чтобы мы могли составить матрицу «символ—метод» (в добавление к простой таблице «символ—метод», описанной в разд. 4Б). *Матрица «символ—метод» — это система величин, которая с некоторой надежностью предсказывает, что произойдет при применении методов к проблемам.* Оба измерения матрицы характеризуются символами проблем; *если существует метод, обычно преобразующий проблемы с характеристикой C_i в проблемы с характеристикой C_j*, матричный элемент C_{ij} будет *служить обозначением* этого метода (или списка таких методов). Если такого метода нет, соответствующий матричный элемент есть нуль.

Предположим теперь, что для C_{ij} нет матричного элемента. Это означает, что нет *прямого* способа преобразовать проблему типа C_i в проблему типа C_j. Умножим матрицу саму на себя. Если новая матрица имеет ненулевой элемент (i, j), то должна существовать последовательность *двух* методов, которая дает желаемое преобразование. Если этого нет, то можно попытаться образовать более высокую степень матрицы. Заметим, что (если положить члены (i, i) равными единице) мы можем образовать матрицу степени 2^n при помощи только n умножений. Нет нужды определять смысл символической операции умножения; вместо этого можно использовать в матрице численные элементы, положив в исходной матрице любой *ненулевой элемент равным единице*, а нулевой — равным нулю. Это приводит к простой матрице связей, или поточной диаграмме, а ее n-я степень расскажет нам кое-что о системе ее связей длиной 2^n (см., например, [478]).

(Коль скоро обнаружен ненулевой элемент в матрице, существуют эффективные способы отыскания соответствующих последовательностей методов. Фактически эта проблема совпадает с проблемой отыскания пути в лабиринте, и метод Мура [753] здесь может оказаться вполне эффективным. Почти любая проблема может быть превращена в проблему нахождения цепи связи между двумя конечными выражениями в некоторой формальной системе.) Если символы дают абстрактное представление выражений, формулирующих проблемы, то эта модель «алгебры символов» может быть столь же абстрактна, как и имеющиеся схемы распознавания символов (см. Минский [728, 732]).

Критической проблемой при использовании модели «алгебры символов» для планирования, естественно, является *надежность предсказания матричных элементов.* Нельзя ожидать, что символ результата будет точно определен символом оригинала и используемым методом. По мере возрастания степени матрицы надежность предсказаний будет во всяком случае быстро уменьшаться. Но, как мы уже отмечали, любой план вообще настолько лучше отсутствия плана, что система должна работать много лучше, чем расточительный поиск, даже если качество предсказания плохое.

Такая матричная формулировка является, очевидно, лишь

частным случаем идеи символического планирования. В общем случае мы скорее будем иметь описания, а не фиксированные символы, поэтому должны быть созданы более общие методы вычисления, позволяющие определить на основе описания вероятные последствия применения некоторого метода.

2) **Символы и различия.** В программе GPS (General Problem Solver), предложенной Ньюэллом, Шоу и Саймоном [795, 798], мы находим слегка отличную схему решения: там, где мы говорим о символе одиночной задачи, в этой схеме используется различие между двумя задачами (или выражениями). Эти точки зрения эквивалентны, если под задачами подразумеваются звенья соединения или связи между выражениями. Но это понятие различия как символ пары выражений более непосредственно приводит к телеологическим рассуждениям. Действительно, цель, определяемая некоторой задачей, состоит не в чем ином, как *в уменьшении «различия» между существующим и желаемым состояниями.* Структура, лежащая в основе GPS, в точности та же, какую мы назвали «машиной символ—метод», где каждый вид различия связан в таблице с одним или более методами, «уменьшающими» это различие. Так как характеристика здесь всегда зависит от выражения данной проблемы и желаемого конечного результата, разумно считать, как предлагают авторы, что GPS использует анализ «соотношения средств и цели».

Чтобы проиллюстрировать использование различий, рассмотрим пример [798]. Задача состоит в том, чтобы доказать, что из $S \wedge (-P \supset Q)$ можно вывести $(Q \vee P) \wedge S$. Программа исследует оба эти выражения при помощи процесса рекуррентного *сличения*, который ответвляется от главных связей. Первое встреченное различие заключается в том, что S находится по разные стороны от главного соединяющего знака $\wedge$ в обоих выражениях. Поэтому программа ищет его в таблице «различие—метод» под рубрикой «изменение позиции». Там она отыскивает метод, использующий теорему $(A \wedge B) \equiv (B \wedge A)$, подходящий, очевидно, для устранения или «уменьшения» различий в позиции. GPS применяет этот метод, в результате чего получается $(-P \supset Q) \wedge S$. Теперь GPS спрашивает, каково различие между этим новым выражением и тем, которое требуется получить («целью»). На этот раз процедуре сличения подвергнутся связки в левом члене выражения, причем будет обнаружено различие между $\supset$ и $\vee$. Программа «просматривает» в таблице «символ—метод» рубрику «изменение связок» и отыскивает там соответствующий метод с помощью правила $(-A \supset B) \equiv (A \vee B)$. Применив этот метод, она получает $(P \vee Q) \wedge S$. В последнем цикле процедура оценки различий обнаруживает необходимость в «изменении позиции» внутри левого члена; при этом применяется метод, использующий соотношение $(A \vee B) \equiv (B \vee A)$. Этим завершается решение задачи[1].

Очевидно, что успех анализа «соотношения средств и цели» в уменьшении общего поиска будет зависеть от точности таблицы «различие — метод»; по существу то же самое требуется и для эффективности моделей «алгебры символов».

Различия могут быть использованы также и для *планирования.* Можно представить себе «алгебру различий», в которой предсказания имели бы форму $D = D'D''$. Мы должны построить соответственно алгебру факторизации различий для открытия более длинных цепей $D = D_1 \ldots D_n$ и соответствующего планирования. Следует отметить, что нельзя ожидать использования подобных методов планирования с такими примитивными различиями, как в работе Ньюэлла, Шоу и Саймона [798]; для них не может быть создана подходящая «алгебра различий» (или алгебра признаков). Если характеризующие выражения не имеют нескольких уровней дескриптивных деталей, степень матрицы будет быстро вырождаться. Это вырождение в конечном итоге ограничит возможность любого формального планирующего устройства.

Общую эвристику планирования можно представить себе погруженной в некий рекуррентный процесс. А именно: предположим, что мы имеем проблему P.

1. Составляем план для решения проблемы P.
2. Выбираем первую (следующую) ступень плана (если нет больше ступеней, вывод—«успех»).
3. Применяем предложенный метод (методы) решения.

В случае успеха возвращаемся к шагу 2, т. е. переходим к следующей ступени плана.

В случае неудачи возвращаемся к шагу 1, т. е. составляем новый план или, возможно, изменяем прежний так, чтобы избежать этой ступени плана.

В случае если задача оказывается слишком сложной, *применяем всю процедуру к задаче данной ступени.*

---

[1] Сравните это с процессом «сличения» (matching), описанным у Ньюэлла и Саймона [787] Понятия «символ», «алгебра символов» и т д возникли у Минского [728—730], но оказались полезными для получения описаний в системе GPS [787, 798]. Последняя работа содержит много дополнительного материала, который мы не можем здесь рассмотреть. Существенным является тот факт, что GPS должна сама применяться к проблеме отыскания различий, свойственных данным задачам. Эта идея (применение системы решения задач к проблеме улучшения некоторых ее собственных методов) старая и хорошо известна, но в работе Ньюэлла, Шоу и Саймона [798] мы находим, быть может, первое конкретное предложение для реализации этих идей

Очевидно, что такая схема программы существенно рекурсивна; она использует сама себя в качестве подпрограммы (очевидно, на последней ступени); при этом ее текущее состояние должно фиксироваться и восстанавливаться, когда она снова используется в качестве основной программы[1].

Миллер, Галантер и Прибрам ([715], гл. 12 и 13) обсуждали возможную аналогию между решением задач человеком и некоторыми эвристическими планирующими схемами. Можно с уверенностью сказать, что уже через несколько лет обнаружится тесная связь между теориями поведения человека и попытками повысить интеллектуальные возможности машин. Но мы должны быть готовы также к открытию эффективных путей эвристического программирования, которые не имитируют человеческого поведения[2].

---

[1] Это нарушает, например, ограничения, накладываемые на формирование циклов в программирующих системах типа FORTRAN. Подходящая методика программирования таких систем была развита Ньюэллом, Шоу и Саймоном [798, 799]. Состояния программ хранятся в «списках продвижения», и как программа, так и данные хранятся в форме «списка структур». Гелернтер [364, 365] распространил систему FORTRAN на управление некоторыми из таких методов. Маккарти [674] применил эти идеи в LISP, чтобы сделать возможным точное рекуррентное определение программы на языке, основанном на рекурсивных функциях символических выражений; здесь управление переменными состояниями программы полностью автоматическое. См. также статью Орчард-Хэйса [817].

[2] Недостаток места не позволяет здесь подробно обсудить теорию самоорганизующихся нервных сетей и других моделей, основанных на аналогиях с мозгом. Некоторые из них описаны или цитировались в трудах Симпозиума по автоматизации процесса мышления [N] и Симпозиума по самоорганизующимся системам [U]. Я думаю, что это упущение не очень серьезно при исследовании темы эвристического программирования, так как объяснения и методы в этих двух областях кажутся весьма различными. До настоящего времени, по крайней мере при исследовании моделей червных сетей, прежде всего старались показать, что определенные достаточно простые эвристические процессы, например обучение с подкреплением или распознавание образов по списку признаков, могут быть осуществлены совокупностями простых элементов без высокоорганизованных взаимосвязей. В отличие от этого работа по эвристическому программированию характеризуется поисками новых, более мощных эвристических методов для решения сложных задач и мало интересуется тем, какая физическая основа (нервная или иная) была бы минимально достаточна для реализации эвристической программы. Короче говоря, работа над «сетями» касается того, как далеко можно пойти, имея небольшие исходные возможности; работа в области «искусственного разума» касается использования всего, что мы знаем, для построения максимально мощной системы, какую только можно построить. По моему мнению, в деле решения сложных задач системы (якобы мозгоподобные) с минимальной структурой не смогут выдержать соревнования с более обдуманно спроектированными современными машинами; тем не менее их нужно изучать для создания элементов и подсистем, необходимых при построении более совершенных машин.

5. Индукция и модели

А. Интеллект

Во всем предыдущем обсуждении проблемы мы не соприкасались ни с чем из того, что можно было бы специально выделить как «интеллект». Мы обсуждали только эвристики, методы сокращения поиска и технику классификации. Было ли при этом что-либо пропущено? Я уверен, что рано или поздно мы сможем составить программу, обладающую большой способностью к решению задач, благодаря сложным комбинациям эвристических механизмов — многократной оптимизации, методов распознавания, алгебры планирования, процессов рекурсивного управления и т. п. Ни в одном из них мы не найдем место нахождения «интеллекта». Что же такое «в действительности» интеллект? С моей точки зрения, это скорее вопрос эстетики или самолюбия, чем науки и техники! Для меня «интеллект» означает едва ли больше, чем комплекс активности, который мы уважаем, но не понимаем. Точно так же дело обычно обстоит с «глубиной» в математике. Как только понято доказательство некоей теоремы, ее содержание кажется тривиальным. При этом может сохраняться чувство восхищения перед тем, каким образом это доказательство было открыто.

Программисты тоже знают, что в программе нет никакого «сердца». Имеются лишь программы более высокого уровня. Но все, что они делают, сводится к инструкциям типа «если то-то и то-то, следует перейти к такой-то и такой-то подпрограмме». И если мы посмотрим на подпрограммы низшего уровня, которые «фактически выполняют весь труд», то увидим бесчувственные связи—циклы и последовательности тривиальных операций, лишь выполняющие то, что им приказано. Интеллект в такой системе оказывается столь же неосязаемым, как смысл изолированного слова в обиходной речи, если даже все вновь и вновь повторять его с глубокомысленным видом.

Но наша неспособность установить местонахождение интеллекта не должна привести нас к заключению, что раз вычислительные машины работают по заданной программе, они поэтому не могут думать. Ибо для *человека* так же, как и для *машины*, справедливо, что, если мы поймем до конца структуру и программу, ощущение тайны (а вместе с ним и наше самомнение) исчезнет (см. Минский [728—733]).

Подобную точку зрения на «творческую деятельность» мы находим у Ньюэлла, Шоу и Саймона [793]. Выраженное Розенблюмом [939] мнение, что ум (или мозг) не достижим для

машины, основано, по-видимому, на ошибочном толковании значения «теоремы неразрешимости» Гёделя [1, 2].

Б. Индуктивный вывод

Зададим теперь нашей машине ряд задач, более заманчивых, чем любая обычная игра или математическая головоломка. Предположим, что нам нужна такая машина, которая, если поместить ее на время в некоторый «внешний мир», будет пытаться дать описание этого мира — открыть его закономерности, или «законы природы». Мы могли бы попросить ее предсказать то, что вскоре должно случиться, или же мы могли бы попросить ее предсказать, каковы будут наиболее вероятные последствия тех или иных действий или экспериментов. Ее можно было бы также попросить сформулировать законы, управляющие некоторым классом событий. В любом случае наша задача состояла бы в том, чтобы сообщить машине способность *к индуктивному мышлению*, т. е. снабдить ее методами, которые она могла бы использовать для построения общих утверждений о событиях, не встречавшихся ранее. Однако может оказаться, что не существует системы индуктивного вывода, которая

[1] Речь идет о доказанной в 1931 г. Г. Гёделем теореме о неполноте формализованной арифметики, согласно которой в логической системе формализации арифметики имеются неразрешимые, т. е. недоказуемые и неопровержимые, предложения. — *Прим. ред.*

[2] Что касается проблем волевого акта, мы в общем согласны с Маккаллоком [685], что наша *свобода воли* «означает, вероятно, лишь нашу способность различать между тем, что мы намерены делать (т. е. нашим планом), и некоторым вмешательством в наши действия». (См. также работу Маккея [644] и приводимые в ней ссылки.) Нас, однако, не убедило его восхваление «аналоговых» устройств. Что касается проблемы «ум—мозг», то следует рассмотреть аргументацию Крейка [248], Хэйка [456] и Паска [825]. Среди видных исследователей современного эвристического программирования, может быть, только Сэмюэль [953] занял жесткую позицию как противник возможности машинного мышления. Его аргументы основаны на том, что вычислительные машины делают только то, что им предписано. Но из этого не следует, что программист знает о том, что должно произойти (и поэтому полностью отвечает за это и предвидит это). Ибо программист может составить эволюционирующую систему, ограничений которой он не знает, а возможно, их и вообще не существует. Не лучше и математик знает все следствия, вытекающие из предложенной им системы аксиом. Конечно, для того чтобы работать, машина должна быть в порядке. Но мы не можем приписывать все ее достоинства лишь ее программисту, если в процессе работы обнаруживаются структуры, которые он не мог распознать или предвидеть. Пока мы еще не видим у машин больших задатков для того, чтобы им можно было приписать умственную деятельность, но вместе с тем аргументы Сэмюэля (вращающиеся в порочном круге, поскольку они исходят из предположения, что машины не имеют ума) не убеждают нас в обратном. Тьюринг [1085] очень компетентно рассмотрел эти вопросы

хорошо работала бы во всех возможных «мирах». Однако, если задан «мир» (среда) или совокупности сред и если задан критерий успеха, эта (эпистемологическая) проблема для машины становится скорее технической, чем философской. В литературе часто освещается этот вопрос, но мы обсудим только одну попытку, которая сейчас кажется нам наиболее обещающей; речь идет о том, что́ можно назвать схемами «грамматической индукции» Соломонова [1031—1033], основанными главным образом на работе Хомского и Миллера [211, 212].

Будем считать, что *язык* — это система выражений, образованных из некоторой данной системы простейших символов или выражений посредством повторного применения некоторой данной системы правил; простейшие выражения плюс правила — это *грамматика* языка. Большинство проблем индукции могут быть сформулированы как проблемы *раскрытия грамматики языка*. Предположим, например, что прошлый опыт машины подытожен в виде большого списка утверждений, одни из которых при помощи некоторого устройства определяются как «хорошие», другие — как «плохие». Каким путем могли бы мы селективно создавать другие хорошие утверждения? Задача состоит в отыскании относительно простого (формального) языка, в котором хорошие утверждения являются грамматическими, плохие — нет. Задав такой язык, мы можем использовать его для того, чтобы создать новые утверждения, и, по всей вероятности, они будут стремиться походить на хорошие. Эвристический довод заключается в том, что если мы нашли относительно простой способ разделения этих двух систем, открытое нами правило, вероятно, должно оказаться эффективным и вне пределов непосредственного опыта. Если это расширение окажется несовместимым с новыми данными, мы могли бы несколько видоизменить правила и в общем случае могли бы использовать для этой цели несколько обычных методов решения задач.

Проблема отыскания эффективной грамматики во многом сходна с проблемой нахождения эффективного *кода* или программ для машины; в обоих случаях необходимо открыть в имеющихся данных важные закономерности и использовать их для *сокращения* процесса. Важность работы Соломонова [1035], с моей точки зрения, заключается в том, что, несмотря на некоторые очевидные недостатки, она указывает направление для систематического математического исследования этой проблемы. Он рассматривает класс всех программ (для некоторой данной универсальной вычислительной машины), которые дадут некоторый заданный выход (совокупность соответствующих данных).

Большинство таких программ, если позволить им продолжать работу, будут прибавлять новые данные к этой совокупности. Правильно взвешивая эти программы, например в соответствии с их продолжительностью, мы можем получить соответствующие веса для различных возможных продолжений и, таким образом, базу для предсказания. Для того чтобы это предсказание представляло интерес, необходимо продемонстрировать некоторую независимость данной вычислительной машины; пока, однако, не совсем ясно, какую форму примет такой результат.

В. Модели „самого себя"

Если некая машина может ответить на вопрос о каком-то гипотетическом эксперименте, не проводя этого эксперимента, то ответ должен быть получен от некоторой подмашины внутри этой машины. Выход такой подмашины (выражающий правильный ответ), так же как и ее вход (выражающий вопрос), должны быть кодовыми описаниями соответствующих внешних событий или классов событий. С точки зрения этой пары кодирующего и декодирующего каналов внутренняя подмашина действует как среда и поэтому имеет характер «модели». Проблема индуктивного вывода может тогда рассматриваться как проблема построения такой модели.

В той мере, в какой действия машины влияют на среду, эта внутреняя модель мира будет нуждаться в сведениях о самой себе. Если кто-то спрашивает машины, почему они решили сделать так-то и так-то (или машина спрашивает об этом сама себя), то любой ответ должен прийти от этой внутренней модели. Таким образом, доказательность интроспекции сама по необходимости должна в конечном итоге базироваться на процессах, используемых при построении кем-то образа «самого себя». Спекулятивные рассуждения о форме такой модели приводят к любопытному предсказанию, что разумные машины неохотно будут верить в то, что они *только* машины. Доводы следующие: наша модель «самого себя» имеет существенно двойственный характер. Имеется часть, касающаяся физической или механической среды, — с поведением неодушевленных объектов, и часть, связанная с социальной и психологической стороной. Мы вынуждены рассматривать обе эти части порознь именно по той причине, что до сих пор еще не имеем удовлетворительной физической теории умственной деятельности. Мы не можем отказаться от такого разделения, даже если бы желали этого, до тех пор, пока мы не создадим взамен объединенную модель. Если мы теперь спросим такую машину, что она за существо, она не может ответить «прямо» на вопрос, она должна исследовать свою модель (или модели). И она должна ответить, что она, по-видимому, является чем-то «двойственным», имеющим две части — «душу» и «тело». Таким образом, даже робот, если его не снабдить удовлетворительной теорией искусственного мышления, сохранил бы дуалистический взгляд в этом вопросе[1].

Заключение

Пытаясь соединить в одной статье обзор работ по «искусственному разуму» с изложением наших собственных взглядов, мы не смогли воздать должное всем достойным упоминания работам. Важные упущения были сделаны в области «моделей мозга», не упомянуты ранние работы Фэрли и Кларка [313], а также статья Фэрли в сборнике «Самоорганизующиеся системы» [315] и работы Рочестера и др. [924] и Милнера [720]. Работа Леттвина и др. [598] имеет отношение к теории Селфриджа [976]. Мы вообще не касались проблем логики и языка, а также помехоустойчивости, с которыми сталкиваются, когда приходится работать с запоминающими устройствами большого объема (см., например, Маккарти [673]). Мы не обсудили также основных результатов математической логики, которая связана с вопросом о возможностях машин. Существует целая литература, которой мы лишь едва коснулись, — пионерская работа Рашевского (1929 г.) и его более поздних последователей [901]; теории обучения (см., например, Горн [410]), теория игр (см., например, Шубик [998]), психология (см., например, Брунер и др. [172]). Каждый знает также работы Пойа [870, 871] о том, как решать задачи. Мы можем лишь надеяться, что передали дух некоторых наиболее важных работ, *непосредственно* связанных с проблемой создания машин, которые могли бы взять на себя значительную часть усилий по решению задач.

И последнее замечание: мы рассмотрели лишь работы, касающиеся более или менее автономных программ решения задач, однако за последнее время начинает возрастать активность в создании *распределяющих время* или многопрограм-

[1] Понятие машин, имеющих *хорошую* модель «самой себя», вызывает проблему бесконечной регрессии. Само собой разумеется, что вложенные внутрь одна другой модели должны утрачивать подробности и в конце концов исчезнуть. Но аргумент, например, Хэйка ([456], разд. 8.69 и 8.79), что мы не можем «полностью постигнуть унитарный порядок» (нашего собственного ума), игнорирует силу рекурсивных описаний так же, как и доказательство Тьюринга о том, что любая «универсальная» машина может ответить на любой вопрос относительно ее самой и что любая достаточно большая машина также может сделать это.

мных вычислительных систем. Такие системы позволяют сочетать деятельность человека в реальном времени с возможностями крупных вычислительных машин. Это означает, что мы сможем начать программировать системы, которые действительно будут «помогать нам мыслить». В будущем мы ожидаем, что эти системы «человек — машина» будут принимать участие, а может быть, даже в течение некоторого времени играть ведущую роль в наших усилиях по созданию «искусственного разума».

Приложение

КРАТКОЕ ОПИСАНИЕ ЯЗЫКА IPL-V[1]

А. Ньюэлл и Ф. Тонг

Введение

В статье дается краткое описание языка, удобного для работы с символами и списками символов, — IPL-V (Information Processing Language V). В настоящее время этим языком можно пользоваться для программирования на машинах IBM-650, IBM-704 и IBM-709. В статье обсуждаются особенности задач, для решения которых разрабатывалась серия языков IPL[2], и основные понятия, входящие в язык IPL-V. Полное описание этого языка можно найти в работах [800а, 800б].

Серия языков IPL

Существует много задач, которые человек может решать достаточно хорошо, не зная в деталях, как он это делает. В качестве примеров можно привести игру в шахматы, принятие деловых решений, доказательство теорем. «Поведение» вычислительной машины в некотором смысле полностью определяется программистом. Чтобы заставить машину играть в шахматы или доказывать теоремы при помощи тех же методов, которыми пользуется человек при решении аналогичных задач, необходимо сделать средства связи человека с машиной значительно более

[1] *Communications of the ACM*, 3, № 4, 205—211 (1960).

[2] Название „Information Processing Language" (язык для переработки информации) было дано языкам этой серии в начале ее разработки и, пожалуй, соответствует их назначению. Разумеется, такие языки, как LISP [674а], FORTRAN List Processing Language [365а] и COMIT [1177], с таким же правом можно называть языками для переработки информации, как и IPL

гибкими, чем это требуется для выполнения формальных алгебраических операций. Программист должен каким-то образом сообщить машине свои несовершенные знания относительно того, как следует вести себя в этих сложных ситуациях. Серия языков программирования IPL и была разработана, чтобы облегчить создание программ для решения задач, использующих характерные для поведения человека методы отсечения плохих вариантов — так называемые «эвристики». Эти языки должны были служить инструментом для изучения эвристических методов решения задач.

Язык IPL-I первоначально предназначался для составления программы доказательства теорем исчисления высказываний [787], но он ни разу фактически не был применен для этой цели. IPL-II и IPL-III были разработаны для машины JOHNNIAC и использовались в программе «Логик-теоретик» [790].

Затем группа из Технологического института им. Карнеги разработала усовершенствованный вариант IPL для машины IBM-650 [463а]. Этот проект получил название IPL-V. В то же время сходный язык IPL-IV был разработан для машины JOHNNIAC и использовался для написания шахматной программы [792] и эвристической программы балансирования сборочного конвейера [1075]. Большие программы были написаны на языке IPL-V для моделирования познавательной деятельности человека, включая работы по обучению различению [318], двоичному выбору [326] и доказательству теорем в некоторых формализованных средах [795].

Последний язык этой серии IPL-VI представляет собой проект системы команд для машины, которая могла бы непосредственно исполнять команды по переработке информации и тем самым позволила бы намного увеличить скорость выполнения программ по сравнению с их реализацией на обычных машинах, основанной на использовании интерпретирующей программы.

Область применения

Опишем вкратце особенности задач, для решения которых были разработаны языки IPL, в том числе язык IPL-V.

1. Эти задачи в основном связаны с оперированием символами, отличными от числовых.

2. В программе для решения сложных задач распределение памяти не может быть задано заранее; в процессе работы программы возникают новые данные, объединяемые в сложные структуры. Например, программа для запоминания списков бессмысленных слогов [318] строит сеть дискриминаторов для

различных слогов. Размер, форма и сложность этой сети целиком зависят от конкретного списка слогов, с которым работает программа.

3. Связи между различными элементами вырабатываемой структуры перестраиваются в процессе работы программы: фиксируются новые ассоциации и уничтожаются старые.

4. Процесс решения сложной задачи естественным образом имеет несколько уровней; на каждом уровне рассуждения строятся с использованием фактов, устанавливаемых на более низких уровнях. Так, в шахматной программе имеются: язык для описания доски и фигур; язык более высокого уровня, описывающий фигуры с учетом их положения на доске в данной позиции (например, учитывающий, что фигура защищает данное поле); язык еще более высокого уровня для описания желаемой ситуации (например, контроль над центром).

5. Процедура решения задачи часто меняется в процессе создания и испытания программы. Следовательно, программа должна легко допускать модификацию на разных уровнях при минимальном количестве изменений в остальной части программы.

IPL-машина

IPL-V — это формальный язык, на котором можно записывать процессы переработки информации. В IPL-V существуют два типа выражений: *списковые структуры*, содержащие перерабатываемую информацию, и *программы*, определяющие процесс ее переработки. Мы пользуемся термином «IPL-машина» для обозначения языка IPL-V, запрограммированного для одной из используемых нами машин (IBM-650, IBM-704 или IBM-709).

IPL-машина состоит из следующих частей:

1. Набор *ячеек*, в которых могут быть записаны «IPL-слова» (так называемая *свободная память*).

2. Набор *символов*, из которых составляются «IPL-выражения» (внутри машины все символы имеют адреса и, таким образом, служат наименованиями ячеек)

3. Набор *элементарных программ*, которые вычислительная машина может выполнять без дополнительной расшифровки.

4. *Интерпретатор*[1], который расшифровывает и выполняет программы.

[1] Для краткости здесь используется этот термин вместо обычно применяемого термина «интерпретирующая программа» — *Прим. перев.*

Представление данных

Символы. Программист имеет в своем распоряжении символы двух типов: *региональные* и *локальные*. Региональные символы состоят из некоторой буквы, за которой следует ее индекс, например A27, C5, G1000. Региональные символы используются при программировании обычным образом. Локальные символы записываются в виде цифры 9, за которой следует произвольный номер, например 9-7, 9-100. Цифра 9 рассматривается как региональный символ. Локальный символ имеет одно и то же значение лишь в пределах данного IPL-выражения. Одни и те же локальные символы могут иметь различные значения в разных частях программы и в разных списковых структурах.

Программа может во время работы использовать символы, которые в явном виде не были использованы в ней программистом, и соответствующие им ячейки памяти. Эти символы называются *внутренними*.

Списки. Как правило, возникает необходимость в более крупной самостоятельной единице информации, чем отдельный символ. В IPL такой единицей (элементом) является список и имеется набор основных процессов для работы со списками. Каждая используемая ячейка содержит IPL-слово, состоящее из четырех частей, обозначаемых P, Q, SYMB и LINK[1]. Символы объединяются в список способом, указанным в табл. 1, где показан список из трех символов S1, S2, S3. Символ L0 — наименование списка и ячейки, называемой *головной ячейкой* списка. Наименования *ячеек списка*[2] являются внутренними символами. Часть слова, обозначаемая через LINK, является наименованием ячейки, в которой находится следующий символ списка.

Таблица 1

Простой список

Наименование	P, Q	SYMB	LINK
L0		0	36
	36	S1	508
	508	S2	13
	13	S3	0

В последней ячейке списка на месте LINK стоит *символ окончания* 0. Условились первый символ списка помещать в первую

[1] SYMB — сокращение от „Symbol" (символ); LINK — связь — *Прим. перев.*

[2] *Головная ячейка* списка не включается в *ячейки списка*. — *Прим. ред.*

ячейку списка, оставив SYMB головной ячейки для другой цели (Программисту нет необходимости заботиться о внутренних символах. В системе IPL они подбираются программой автоматически.) Таким образом, несколько символов можно объединить в один элемент информации, поместив их в список. Эти символы могут служить наименованиями других списков или более сложных структур.

Дескриптивные списки. Со списком может быть связана некоторая характеризующая его информация, которую по желанию можно изменять, уничтожать или пополнять. Это осуществляется при помощи *дескриптивных списков.* Символ, хранящийся в головной ячейке некоторого списка, является наименованием соответствующего дескриптивного списка. Символы дескриптивного списка рассматриваются попарно. Первый член пары обозначает признак, второй — значение этого признака. Каждый признак соответствует некоторой функции, аргументами которой служат описываемые этим признаком элементы информации. Так, для элемента информации «трава» признак «цвет» будет иметь значение «зеленый». В табл. 2 представлен список L1, содержащий символы S4 и S5, которые описываются двумя признаками A1 и A2.

Таблица 2

Дескриптивный список

Наименование	P, Q	SYMB	LINK
L1		9-1	
		S4	
		S5	0
9-1		0	
		A1	
		V1	
		A2	
		V2	0

Элементарная программа «найти значение признака A1 списка L1» даст в результате символ V1.

Дополнительная информация может быть добавлена к списку с помощью элементарной программы, которая присваивает символу признак и его значение. Подобным же образом можно имеющееся значение заменить новым или полностью выбросить

из памяти признак вместе с его значением. Программисту нет необходимости знать фактическую структуру дескриптивного списка. Вся требуемая работа сводится к поиску списка, в котором находится нужный признак, а внесение требуемых изменений осуществляется с помощью соответствующих элементарных программ.

Термы. Итак, символы приобретают смысл благодаря тому, что они служат наименованиями списков и благодаря связанной с ними описательной («дескриптивной») информации. С символом может быть также связана некоторая информация вне рамок IPL, например *числа* (целые или с плавающей запятой), наборы единиц и нулей, занимающие часть разрядов ячейки, или буквенно-цифровая информация Эта информация в закодированном виде помещается в ячейку, наименованием которой служит данный символ, и обрабатывается с помощью IPL-программ, осуществляющих операции над символами Символ и связанная с ним закодированная информация называются *термом.* Элементарные программы IPL-V выполняют арифметические действия с числовыми термами и печатают термы всех вышеупомянутых типов. В систему легко могут быть включены также иные типы термов и соответствующие элементарные программы

Таблица 3

Списковая структура

Наименование	P, Q	SYMB	LINK	
L2		9-1		
		9-7		
		G4	0	
9-1		0		
		A1		
		V1		
		A5		
		9-10	0	
9-7		0		
		S4		
		9-5	0	
9-10	1			15
9-5			0	
		Z1		
		L2		

Списковые структуры. Если в качестве наименований использовать локальные символы, можно ввести более сложные элементы информации. *Списковая структура* состоит из *основного списка*, наименованием которого служит региональный, или внутренний, символ, и всех тех структур, наименования которых содержатся в основном списке и являются локальными. В табл. 3 приведена структура, состоящая из основного списка L2, дескриптивного списка 9-1 с термом 9-10 (представляющим собой целое число 15) в качестве значения признака A5 и подсписков 9-7 и 9-5.

Элементарные программы IPL строят списковые структуры, воспроизводят их, стирают или переводят во внешние запоминающие устройства как единое целое. Существуют также программы, позволяющие просматривать списковые структуры и обрабатывать их иными способами.

Магазинные списки

Программист может использовать ячейки памяти в качестве рабочих ячеек, т. е. хранить символы в SYMB этих ячеек. В этом случае LINK содержит символ окончания.

Но часто желательно запомнить информацию в некоторой ячейке, не уничтожая при этом ту информацию, которая в ней к этому моменту содержится. Например (более подробно об этом будет сказано ниже), интерпретатор всегда хранит наименование ячейки, в которой находится текущая IPL-команда, в специально отведенной ячейке памяти H1. Если эта команда определяет некоторую подпрограмму, которую необходимо расшифровать, интерпретатор должен хранить ее адрес, пока эта подпрограмма выполняется, чтобы не потерять место этой команды в программе более высокого уровня. Следовательно, так как подпрограмма сама может содержать некоторую подпрограмму и т. д., в разных программах потребуется хранить разное число адресов.

Это делается с помощью операций *сохранения* и *восстановления*. Сохранить ячейку означает выбрать неиспользованную ячейку из свободной памяти и переписать в нее содержимое сохраняемой ячейки. Наименование этой новой ячейки помещается в LINK сохраняемой ячейки. После этого в сохраняемую ячейку можно поместить другой символ, не уничтожив при этом первоначально содержавшейся в ней информации. В первоначальное состояние эту ячейку можно вернуть с помощью обратной операции — операции восстановления. Список сохраненных символов, связанный с некоторой ячейкой, называется ее *магазинным списком*.

В табл. 4 представлена ячейка H1, в которой вначале был записан символ K3, после чего к этой ячейке была применена операция сохранения, а в ней был записан символ Q5.

Таблица 4

Магазинный список

Наименование	P, Q	SYMB	LINK
H1		Q5	387
387		K3	0

Таким образом, приступая к расшифровке подпрограммы, интерпретатор сохраняет ячейку H1, прежде чем записать в нее наименование подпрограммы в качестве нового адреса текущей команды. По окончании подпрограммы интерпретатор восстанавливает H1, чтобы получить адрес текущей команды в программе более высокого уровня.

Список свободной памяти

В процессе того, как списки выстраиваются в памяти и видоизменяются, ячейки попеременно то используются, то снова оказываются ненужными (как, например, в случае операций сохранения и восстановления). Необходимо каким-то образом фиксировать, какие ячейки памяти в данный момент не используются. В IPL все не используемые в данный момент ячейки объединены в список, именуемый H2, который называется *списком свободной памяти*. В процессе расшифровки или в любом другом случае, в котором может потребоваться свободная ячейка, в качестве нее выбирается первая ячейка из списка свободной памяти. Подобным же образом ненужные ячейки возвращаются в этот список. Этот механизм освобождает программиста от необходимости решать задачу распределения памяти и позволяет ему свободно пользоваться различными процессами, меняющими структуру памяти.

Расшифровка программ

Программы. IPL-программа представляет собой список команд. В ходе расшифровки программы интерпретатор последовательно рассматривает каждое командное слово и выполняет действие, которое это слово обозначает. Этим действием может быть выполнение некоторой другой программы. Правила образования программ в IPL и способ, которым производится

расшифровка, дают возможность организовать каждую программу в виде подпрограммы, которая может использоваться любой программой, в том числе и ею самой. Программы, оформленные в виде подпрограмм, организованы в иерархическую структуру с помощью специальным образом устроенной связи между ними, а также некоторых правил задания их параметров и использования рабочих ячеек.

Список адресов текущих команд. Как уже упоминалось выше, адрес ячейки, содержащей текущую команду, запоминается в специальной ячейке H1. Если эта команда означает подпрограмму, которую необходимо расшифровать, H1 сохраняется, а по окончании расшифровки восстанавливается. Таким образом, информация, необходимая для возврата, содержится в списке сохранения, связанном с H1, который называется списком адресов текущих команд. Программист указывает лишь название подпрограммы; возврат осуществляется интерпретатором автоматически.

Задание входов и выходов[1] **— связующая ячейка.** Входы для любой программы задаются путем помещения их в связующую ячейку (communication cell), именуемую H0. Содержимое H0 сохраняется перед записью каждого нового входа, так что множество входов программы — это первые символы в магазинном списке H0. Принято правило, согласно которому каждая программа удаляет свои входы из H0 и оставляет там свои выходы.

Рабочие ячейки. 10 ячеек W0—W9 выделяются под так называемую общую рабочую память (Наряду с этим в программах в качестве рабочих ячеек можно использовать и любые свободные ячейки.) Если программы, в которых используются ячейки общей рабочей памяти, сначала сохраняют ячейку, добавляя содержащуюся в ней информацию к ее магазинному списку, а по окончании работы восстанавливают содержимое ячейки, то любая программа может использовать любую другую (в том числе и себя) в качестве подпрограммы. При этом не возникает опасности, что нужная какой-либо программе информация в рабочих ячейках будет испорчена.

Назовем связующую ячейку и ячейки общей рабочей памяти *надежными ячейками* (safe cells). Это означает, что любая программа, в которой они используются, обязана вначале сохранить их, а по окончании восстановить. Такое тщательное обращение с контекстом позволяет достигнуть гибкости в нескольких отношениях, поскольку выходы процесса могут быть оставлены

[1] Входами процесса (оператора) авторы называют символы, которые обрабатываются этим процессом, а выходами — символы, полученные в результате выполнения процесса — *Прим. перев.*

в связующей ячейке в качестве входов следующей программы, а каждая программа по отношению к рабочей памяти является независимой. Недостатком же такой организации является необходимость детальной обработки каждой используемой надежной ячейки.

Тестовая ячейка. Многие программы дают в качестве выхода, помимо других выходных величин, информацию типа «да» — «нет». Например, «да, я нашел место этого символа в списке» или «нет, эти два символа не тождественны». Такие ответы заносятся в тестовую ячейку H5 (плюс при положительном ответе и минус при отрицательном).

Структура команды. Каждая команда программы записывается в виде IPL-слова. Содержание программы определяется приставками P и Q и частью слова, обозначаемой SYMB. При этом LINK является адресом следующей команды программы.

Приставка Q определяет операцию присваивания значения (designation). В результате этой операции значение SYMB текущей команды или некоторых связанных с ней ячеек присваивается некоему символу S. Это способ, которым в IPL осуществляется косвенная адресация. Три вида операций присваивания, возможные в IPL-V, представлены в табл. 5.

Таблица 5

Операция присваивания

Q=0 S=SYMB
Q=1 S=символ в ячейке, наименованием которой является SYMB
Q=2 S=символ в ячейке, наименованием которой является символ в ячейке, наименованием которой является SYMB.

Например, для следующих двух ячеек:

Наименование	P,Q	SYMB	LINK
C0		B0	
B0		K0	

символ S может иметь значения:

0C0=C0,
1C0=B0,
2C0=K0.

Приставка P определяет операцию, которую следует затем выполнить над символом S. В табл. 6 представлены операции, соответствующие различным значениям P.

Таблица 6

Коды операций

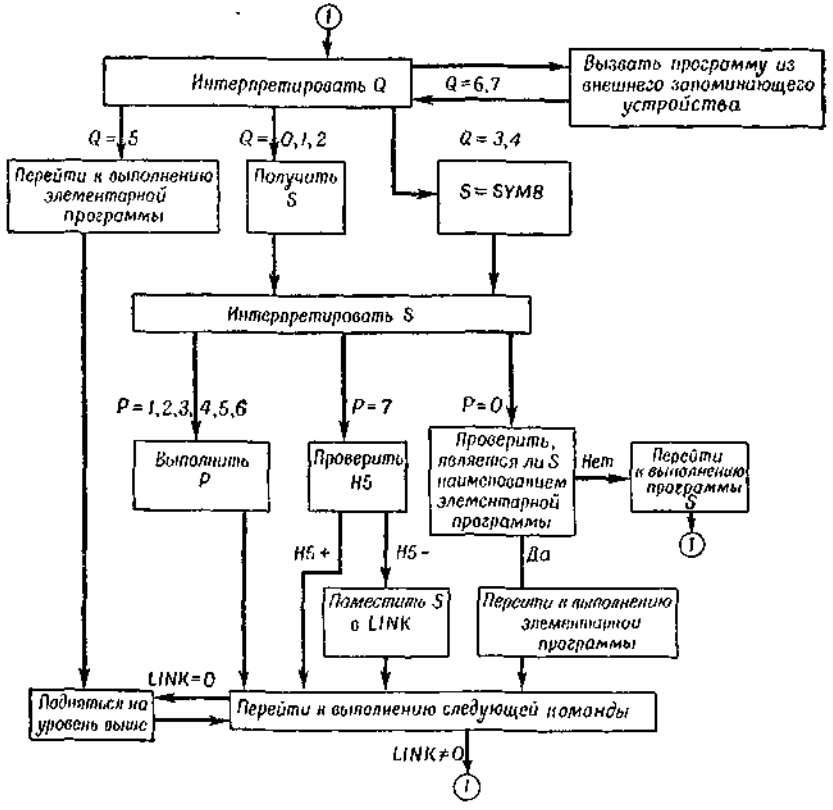
P=0. Выполнить S. Предполагается, что S — это наименование программы или элементарной программы

P=1. Подать S на вход. Связующая ячейка H0 сохраняется; после этого S переписывается в H0

P=2. Поместить выход в S. Символ из H0 (для сокращения обозначаемый далее через (0)) переписывается в ячейку S. H0 затем восстанавливается

P=3. Восстановить S. Символ, который помещен последним в магазинный список ячейки S, переводится в S. Символ, находящийся к этому моменту в S, теряется

P=4. Сохранить S. Символ из ячейки S помещается в магазинный список этой ячейки. При этом содержимое S не меняется



Фиг. 1. Блок-схема интерпретатора.

P=5. Заменить (0) на S. S переписывается в H0. При этом (0) теряется (это аналог обычной операции вызова в сумматор)

P=6. Переписать (0) в S. (0) переписывается в S. Символ, находящийся к этому моменту в S, теряется, а (0) остается без изменения (аналог обычной операции переписи из сумматора в ячейку)

P=7. Перейти к S, если в H5 минус. Если в H5 плюс, LINK указывает ячейку, в которой находится следующая команда. (Это обычный порядок выполнения команд). Если в H5 минус, следующей выполняется команда, находящаяся в ячейке S

Интерпретатор. Интерпретатор расшифровывает программу, рассматривая ее как последовательность элементарных подпрограмм. Эти подпрограммы выполняются по очереди. Блок-схема интерпретатора представлена на фиг. 1.

Стандартные подпрограммы

Система IPL-V включает около 150 стандартных подпрограмм. Хотя совершенно очевидно, что это не минимальный набор — некоторые из стандартных подпрограмм сами записаны на IPL-V, — опыт использования первых языков серии IPL показывает, что такой набор полезен. Ниже описаны некоторые типы стандартных подпрограмм.

Общие подпрограммы. Например: «пустая операция», «проверить тождественность двух символов», «записать плюс в H5» и «останов».

К числу подпрограмм обработки дескриптивных списков относятся такие, как «найти значение признака», «присвоить признаку новое значение» и «стереть признак вместе с его значением».

Подпрограммы, оперирующие с общей рабочей памятью, дают возможность посредством одной команды осуществить сохранение, восстановление или перемещение символа из связующей ячейки в несколько ячеек рабочей памяти.

Подпрограммы обработки списков включают такие операции, как «определить адрес следующего символа в списке», «поместить символ в список», «стереть списковую структуру», «переписать списковую структуру».

Среди арифметических подпрограмм имеются такие, как «сложить», «умножить» и «проверить неравенство $a>b$». Система включает также стандартные подпрограммы, позволяющие генерировать случайные числа в определенном диапазоне.

С помощью подпрограмм префиксации данных программист может идентифицировать и помечать признаками

различные типы символов и термов, имеющихся в системе. Это подпрограммы типа «проверить, является ли символ обозначением некоторого терма» и «сделать символ локальным».

Подпрограммы обмена с внешней памятью дают возможность программисту перемещать списковые структуры из вспомогательной памяти в оперативную и обратно.

Подпрограммы ввода — вывода позволяют читать или записывать данные, организованные в списковые структуры, используя любые имеющиеся у машины внешние устройства. Данные выводятся на перфокарты или на ленты таким образом, что их можно снова вводить в машину. Система IPL дает возможность печатать данные в виде столбцов и строк с любыми интервалами.

Повторение операций осуществляется в IPL или в цикле (с использованием операции условного перехода), или с помощью программ специального типа, называемых *генераторами*. Генератор — это программа, которая вырабатывает последовательность символов и обрабатывает каждый символ с помощью некоторой подпрограммы. Один и тот же генератор может использовать любую подпрограмму — наименование требуемой подпрограммы задается в качестве одного из входов генератора. (Подпрограмма должна в соответствии с принятым соглашением подавать генератору сигнал о продолжении или прекращении генерирования символов.) Таким образом, в генераторе, так же как в операторах «итерирования» обычных компиляторов, имеет место разделение частей цикла, связанных с «выработкой элементов» и «выполнением процесса».

Подпрограмма выполняется над каждым символом выходной последовательности так, будто это продолжение программы, включившей генератор (суперпрограммы), т. е. так, как если бы генератор не пользовался связующей ячейкой или общей рабочей памятью. Генераторы отличаются от остальных IPL-программ тем, что в этом случае в рабочей памяти одновременно должна находиться как информация, относящаяся к генератору, так и информация, относящаяся к подпрограмме и суперпрограмме. Управление генератором и подпрограммой чередуется так же, как и относящаяся к ним информация в рабочей памяти. Чтобы выработать некоторый символ последовательности, генератор должен находиться под управлением, и необходимая для этого информация должна находиться в ячейках W; чтобы обработать этот символ, подпрограмма также должна находиться под управлением, а вся относящаяся к этой подпрограмме информация — в ячейках W. В результате строгая иерархия программ и подпрограмм нарушается, и следует принять необходимые меры для того, чтобы информация в ячейках оставалась невредимой и каждая программа выполнялась в правильном контексте.

Для этой цели предусмотрены специальные внутригенераторные программы. Они обеспечивают сохранение контекста генератора во время выполнения подпрограммы. По окончании работы подпрограммы этот контекст возвращается в ячейки W. Программист пользуется этими программами при программировании генераторов. Некоторые часто употребляемые генераторы — «воспроизвести символы списка», «воспроизвести ячейки списковой структуры», «воспроизвести ячейки древовидной структуры» — включены в стандартные подпрограммы обработки списков.

Вспомогательные операции

Средства для отладки включают выборочную «прокрутку» любой программы, «моментальный снимок» любой информации (в том числе ячеек, в которых записана соответствующая программа) в начале и конце «прокрутки» и вывод на печать любой информации по окончании работы программы. Предусмотрена также возможность запоминания программы на ленте или перфокартах для дальнейшего использования.

Пример IPL-программы

В качестве простого примера программирования на IPL рассмотрим следующую задачу: проверить, имеется ли данный символ в данном дереве. Деревом называется списковая структура, в которой никакой подсписок не встречается больше одного раза. Списковая структура, представленная в табл. 3, является деревом.

Мы опишем два варианта программы для решения этой задачи. В первом варианте мы воспользуемся стандартной подпрограммой, позволяющей следовать вниз по списку от ячейки к ячейке (J60), а во втором варианте — стандартной подпрограммой для воспроизведения ячеек, организованных в древовидную структуру (J102).

Ниже приводится перечень необходимых стандартных подпрограмм. (Символ в H0 обозначается через (0), следующий символ в магазинном списке H0 обозначается соответственно через (1), а символ, следующий за (1), — через (2) и т. д.)

J50: Сохранить W0, затем поместить (0) в W0.

J60: Найти адрес символа, следующего за ячейкой (0). Предполагается, что (0) — это наименование некоторой ячейки. Если имеется следующая ячейка (LINK ячейки (0) не является символом окончания), то выходом этой подпро-

граммы является наименование следующей ячейки, а в H5 записывается плюс. Если LINK — символ окончания, выходом подпрограммы является наименование последней ячейки (т. е. выход равняется входу), а в H5 записывается минус.

J132: Проверить, является ли (0) локальным символом. Если да, записать в H5 плюс, если нет, записать минус.

J2: Проверить равенство (0) = (1). Если равенство имеет место, записать в H5 плюс; в противном случае записать минус.

J30: Восстановить W0 (выполняются те же действия, что и по команде 30 W0).

J131: Проверить, является ли (0) наименованием терма. Если да, записать в H5 плюс, если нет, записать минус

J5: Изменить знак H5 на обратный.

J8: Восстановить H0 (выполняются те же действия, что и по команде 30 H0).

J102: Генератор, воспроизводящий ячейки дерева (1) для подпрограммы (0). На вход подпрограммы (0) последовательно подаются символы из ячеек дерева (1). При этом ячейки каждого подсписка обрабатываются раньше, чем последующие ячейки списка более высокого уровня. Подпрограмма дает генератору сигнал «продолжать» или «остановиться», помещая в H5 соответственно плюс или минус. Генератор заканчивает работу, имея в H5 плюс, если он не был остановлен подпрограммой, и минус в противном случае. Кроме того, генератор дает сигнал подпрограмме, помещая в H5 плюс, если очередная ячейка на входе подпрограммы является головной ячейкой подсписка, и минус в противном случае.

Приведем определение программы E0.

E0: Проверить, имеется ли символ (0) в дереве (1). Если да, записать в H5 плюс, если нет, записать минус.

Первый вариант программы, в котором используется подпрограмма J60, имеет вид, представленный в табл. 7. Та же программа с использованием стандартной программы J102, представлена в табл. 8.

Заметим, что подпрограмма меняет знак, полученный в результате J2, для того, чтобы дать сигнал генератору. Если символы тождественны, подпрограмма должна остановить генератор; поэтому она меняет плюс на минус. Если символы не тождественны, генератор должен продолжать работу; соответствующий сигнал подпрограммы будет плюс. Программа более высокого уровня E0 меняет сигнал генератора на обратный, так как подпрограмма останавливает генератор (с минусом в H5) только в том случае, если найден проверяемый символ.

Таблица 7

Наименование	P, Q	SYMB	LINK	Примечания
E0		J50		Сохранить W0 и поместить проверяемый символ в W0
9-3		J60		Найти следующий символ дерева
	70	9-1		Если больше ячеек нет, выход из программы с минусом в H5
	12	H0		Поместить на вход символ из следующей ячейки списка
	11	W0		Поместить на вход проверяемый символ
		J2		Проверить тождественность символов
	70	9-2		Если символы тождественны, выход с плюсом в H5
9-1	30	H0	J30	Ликвидировать связь со списком, восстановить W0
9-2	12	H0		Снова поместить символ из списка на вход
		J132		Проверить, является ли он локальным
	70	9-3		Если он не локальный, продолжать движение вниз по списку
	12	H0		Снова поместить символ из списка на вход
		J131		Проверить, является ли он наименованием терма
	70		9-3	Если да, продолжать движение вниз по списку
	12	H0		Если нет, записать его в качестве наименования подсписка
	11	W0		Поместить проверяемый символ на вход
		E0		Применить весь процесс к подсписку
	70	9-3	9-1	Если символ найден в подсписке, выход с плюсом в H5

Таблица 8

Наименование	P, Q	SYMB	LINK	Примечания
E 0		J50		Сохранить W0 и поместить проверяемый символ в W0
	10	9-10		Поместить на вход наименование подпрограммы
		J102		Воспроизвести ячейки дерева для подпрограммы 9 10

Наименование	P, Q	SYMB	LINK	Примечания
		J5	J130	Изменить знак на обратный, восстановить W0
9-10	70	J8		Если ячейка головная, отбросить без проверки
	52	H0		Подать на вход символ из списка (нарушив связь между ячейками)
	11	W0		Подать на вход проверяемый символ
		J2	J5	Проверить тождественность символов; изменить знак на обратный

Заключительные замечания

Хотя полезность описанной выше системы можно полностью оценить только на основе опыта ее практического использования, мы уверены, что такой подход позволяет значительно продвинуться в области оперирования символами. Мы достигли гибкости, необходимой для решения многих интересных задач, которые не удавалось решить простыми способами, используя более приспособленные к существующим машинам системы программирования. Как сложные структуры, так и сложные программы можно обозначать одним символом и затем оперировать с ними, как с простыми элементами. Мы разработали эту систему для того, чтобы облегчить решение интересовавших нас задач по переработке информации, которые трудно поддаются описанию с помощью обычных языков программирования.

За все это нам пришлось расплачиваться потерей скорости и ухудшением использования памяти. Эта плата очень высока, когда дело касается стандартных арифметических операций, так как существующие вычислительные машины специально сконструированы для выполнения таких операций. Она становится менее высокой, как только программы и процессы обработки данных усложняются настолько, что независимо от используемой системы программирования требуют тщательной разработки некоторых вспомогательных операций.

ЧАСТЬ IV

Библиография работ по искусственному разуму

М. Минский

ВВЕДЕНИЕ

Настоящая библиография предназначена для того, чтобы служить кратким руководством при изучении литературы по «искусственному разуму», т. е. литературы, посвященной проблеме создания машин с разумным поведением. Мы разбили эту область на темы и в соответствии с этим разбиением снабдили все ссылки перекрестными индексами. Большие библиографии без какой-либо классификации почти бесполезны. Хотя данная область исследований еще молода, в ней также уже можно указать многочисленные примеры того, как масса времени расходовалась на открытие уже известного. За последние год или два положение еще более ухудшилось, и в такой ситуации любая информация лучше, чем никакой.

Именно эту цель — дать некоторую информацию об имеющейся литературе — и преследует настоящая библиография. Наш выбор был ограничен главным образом теми публикациями, которые имеют прямое отношение к построению систем, способных решать сложные задачи. Многие пограничные области исследований либо вовсе не нашли в ней отражения, либо представлены всего лишь несколькими ссылками.

Предлагаемая система классификации не особенно точна. Выбранное нами тематическое разбиение не всегда позволяет провести четкие разграничения, и эти разграничения не всегда удобны. В распределении статей по темам, вероятно, много ошибок, что относится к тем работам, которые я либо не читал, либо не до конца понял. Я внимательно просмотрел или обсудил с их авторами около половины всех статей. Относительно половины из оставшихся статей мне казалось, что я способен установить их содержание по другим источникам. О содержании остальных я догадывался, пользуясь различными намеками, и мог при этом очень сильно ошибиться.

Смысл «дескрипторов» — наименований тем — подробно не разъясняется, так как при современном состоянии этой области

исследований какая-то стандартизация здесь невозможна. Употребление этих терминов предполагает понимание основных вопросов, обсуждавшихся в моей статье [734], которая опубликована в достаточно доступном журнале. Очевидно, что вопрос о том, к какой теме следует отнести ту или иную статью, в ряде случаев не может быть решен строго однозначно; это касается, например, таких тем, как *обучение* и *индуктивный вывод*. В большинстве случаев я относил статью *не ко всем* темам, которые имеют к ней отношение, так что читателю придется просмотреть несколько смежных тем, чтобы найти всю литературу по интересующему его частному вопросу. Не все статьи удалось отыскать для сверки их библиографического описания. Поэтому настоящую библиографию нельзя использовать как основу для составления других библиографий без тщательной сверки всех ссылок. Наша цель состояла в том, чтобы дать доступное пособие для изучения литературы, а не в обеспечении исчерпывающей полноты и точности библиографии. Многие ошибочные ссылки, на протяжении нескольких лет кочующие из одной библиографии в другую, здесь были исправлены, но, конечно, неизбежно вкрались другие ошибки.

При выборе статей мы руководствовались следующими правилами. Статьи, *непосредственно* касающиеся проектирования, построения и использования механизмов для решения задач, включались почти без разбора. Статьи из имеющих сюда отношение разделов математики, психологии и физиологии отбирались более критически — так, чтобы каждая такая статья могла служить отправным пунктом при изучении литературы из соответствующего раздела. Многие из таких разделов почти полностью опущены, и полезно сказать несколько слов о принципах такого отбора. Мы не пытались сделать библиографию исчерпывающей; в большинстве случаев приведены только последние работы каждого из авторов. Очень мало ссылок приводится на неамериканские источники; это отчасти объясняется тем, что до самого последнего времени было, если не считать английских работ, очень мало иностранных публикаций; вероятно, это связано с тем, что исследования в этой области не могут развиваться без проверки моделей на вычислительных машинах. Здесь уместно заметить, что я отнюдь не рассматриваю всю литературу по искусственному разуму как сокровищницу, из которой можно черпать хорошие идеи; главные проблемы в этой области даже вообще еще как следует не поставлены. Однако можно не сомневаться, что в ней содержится и очень много плохих идей, так что для работающих в этой области было бы полезно, если бы они умели их быстро распознавать. Мои собственные взгляды достаточно ясно изложены в работе [734].

Теория автоматов, теория переключательных схем, теория рекурсивных функций, логика

В этот раздел мы включили лишь несколько основополагающих работ. Ощущается насущная необходимость в создании теории вычислений и теории иерархии вычислимых функций. Однако в настоящее время ни та, ни другая в сколько-нибудь удовлетворительной форме не существуют. Мы включили небольшое число работ, посвященных доказательству теорем с помощью вычислительных машин; во всех этих работах рассматриваются задачи, требующие эвристического подхода, хотя, возможно, сами авторы не всегда с этим согласны. Мы также включили несколько ссылок, относящихся к вопросу о надежных вычислениях с помощью ненадежных элементов (впервые рассмотренному в работе фон Неймана [783]). Эта задача хотя и не связана непосредственно с проблемой искусственного разума, но, несомненно, важна с точки зрения реализации указанных идей. То же самое можно сказать и о теории вероятностных машин; по этой теме имеется пока еще очень скудная литература. Мы опустили обширную литературу по минимизации автоматов и переключательных схем, хотя для этих задач предложено много полезных эвристических методов.

Языки, машинный перевод

По этому разделу включено значительное число работ, посвященных естественным и формальным языкам. Эта область имеет прямое отношение к искусственному разуму, и исследования в этих направлениях можно было бы связать еще теснее. Полностью отсутствуют ссылки на статьи, посвященные машинному переводу, хотя мы и понимаем важность этого направления; по этой теме имеется библиография Эттингера [1], включающая 645 работ.

Теория информации, теория кодирования, теория статистических решений

Здесь дается всего несколько ссылок. Хотя проблемы, связанные с кодированием и принятием решений, для нас очень важны, пока еще нельзя утверждать, что они имеют прямое отношение к теме.

Нейрофизиология и психофизиология

Когда-нибудь с помощью этих наук, вероятно, станет понятно, как работает мозг. В настоящее же время лишь немногое известно о *психологии решения задач*. Мы постарались включить

[1] Oettinger A. G., Bibliography of Mathematical Linguistics and Automatic Translation, Cambridge, Mass., Computation Lab., Dept. of Linguistics, Div of Eng. a Appl. Phys., Harvard Univ., Sept 1959

побольше ссылок на основные работы в этих областях. Не включено большое количество статей, посвященных обучению низших животных; мы убеждены, что, если нашей целью является построение машин, способных решать сложные задачи, вряд ли принесет большую пользу усиленное изучение систем, которые сложных задач не решают. Полностью опущен раздел *самоприспосабливающихся систем* как слишком элементарный по своему характеру; в то же время работы по *исследованию операций* не включены только из-за неосведомленности автора в этой области; это же относится и к *информационному поиску*. Область исследований, которая без сомнения заслуживает быть представленной более широко, — это *теория индуктивного вывода*; по этому вопросу имеется много философской литературы.

Предлагаемый список литературы — это пересмотренный и исправленный вариант работы, которая упомянута в библиографии под номером [736]. Первоначальный список основывался на моих собственных картотеках и на библиографиях Алисы Пирс[1], Аллена Ньюэлла[2], Рассела Кирша[3], Симмонса и Симмонса[4], а также вышеупомянутой библиографии Эттингера. Я предполагаю в дальнейшем опубликовать еще один пересмотренный и дополненный вариант; к настоящему времени объем относящейся к теме литературы составляет около 2000 статей. Библиография такого размера без дескрипторов была бы совершенно бесполезной, и даже при нашем способе классификации возможности ее практического использования довольно ограничены. Остается лишь надежда на то, что, как только прояснятся наиболее перспективные направления исследований, появится возможность увеличить избирательность при отборе литературы[5].

---

[1] Pierce A. M., A Concise Bibliography of the Literature on Artificial Intelligence, Bedford, Mass., Communic. Sciences Lab., Electronic Research Directorate, AFCRC-TN-59-773, ARDC, USAF, Sept. 1959

[2] Newell A., Bibliography GI-506, March 1958 (не опубликовано).

[3] Kirsch R. A., Bibliography for NBS Graduate Course 204-204, Washington, Data Processing Systems Div., NBS, Sept. 1959 — Apr. 1960. Effective computational processes with finite and infinite machines

[4] Simmons P. L., Simmons R. F., The Simulation of Cognitive Processes. An Annotated Bibliography, Santa Monica, Calif., Syst. Develpt. Corp., *IRE Trans. on Electronic Computers*, EC-10, 462—483 (1962). Содержит 498 ссылок, краткие аннотации статей и предметный указатель.

[5] В русском издании библиография существенно дополнена за счет включения работ, появившихся за последние 2—3 года, а также многочисленных работ советских авторов. Добавленные работы отмечены звездочкой и распределены по соответствующим тематическим разделам.

Работа по дополнению и расширению библиографии выполнена В. Н. Оксенем. — *Прим. ред.*

ТЕМАТИЧЕСКИЙ УКАЗАТЕЛЬ К БИБЛИОГРАФИИ

Каждый тематический раздел обозначен своим индексом — «дескриптором», который состоит из прописной буквы русского алфавита и цифры, напечатанных жирным шрифтом. Буква указывает на основной тематический раздел, цифра — на соответствующий подраздел. Цифры, приведенные в каждой тематической рубрике, означают порядковые номера соответствующих источников в библиографии, относящихся к данному разделу.

А. *Математическая теория вычислительных машин и автоматов*

А1. Конечные автоматы. Математическая теория

9, 47, 51, 55, 71, 148, 177, 178, 255, 257, 374, 478, 480, 541, 557, 560, 676, 695, 733, 751, 760, 763, 782, 783, 863, 889, 985, 1097.

А2. Теория логических сетей

9, 71, 178, 240, 369, 383, 479, 557, 676, 695, 726, 728, 733, 760, 980, 1062, 1079, 1118.

А3. Теория переключений

190, 695, 756, 760, 980.

А4. Машины с неограниченной памятью (машины Тьюринга)

177, 214, 270, 556, 594, 617, 695, 737, 782, 889, 926, 987, 1035, 1084, 1132.

А5. Машины с неограниченной структурой (растущие автоматы)

177, 178, 179, 480, 481, 482, 695, 782.

А6. Теория вероятностных машин

46, 136, 244, 245, 255, 256, 398, 486, 583, 594, 633, 690, 691, 693, 695, 726, 738, 756, 782, 890, 1035, 1113, 1114, 1132.

А7. Вычислимость и рекурсивные функции

221, 266, 267, 392, 480, 555, 556, 594, 672, 674, 695, 737, 876, 889, 926, 927, 1084.

А8. Теория вычислений

131, 179, 196, 674, 675, 695.

Б. *Структура вычислительных машин*

116, 739, 1020, 1047.

Б1. Обычные цифровые машины

20, 69, 815, 883, 985, 997, 1136.

Б2. Параллельные машины

71, 72, 136, 178, 179, 218, 245, 481, 482, 682, 776, 799, 976, 997, 1097, 1098, 1102, 1106, 1113, 1114, 1118.

Б3. Надежность благодаря избыточности компонентов

13, 136, 137, 244, 245, 301, 565, 583, 617, 619, 690, 691, 693, 756, 782, 783, 784, 1113, 1114.

Б4 Случайные «нервные сети»

12, 44, 71, 92, 189, 225, 313, 398, 454, 458, 541, 719, 720, 721, 726, 732, 740, 825, 829, 891, 924, 930, 931, 933, 934, 1100, 1101, 1106, 1118.

Б5. «Мозгоподобные» машины

26, 70, 76, 243, 251, 442, 446, 458, 541, 719, 726, 729, 897, 901, 930, 931, 934, 936, 947, 1100, 1106.

Б6. «Нервные сети» не обязательно случайного строения

13, 31, 70, 71, 72, 92, 115, 129, 130, 137, 178, 179, 240, 244, 245, 254, 255, 256, 313, 316, 369, 397, 420, 428, 442, 446, 447, 454, 479, 480, 486, 533, 557, 580, 583, 584, 600, 641, 654, 676, 677, 690, 691, 693, 719, 720, 721, 726, 728, 729, 733, 738, 760, 783, 891, 892, 897, 901, 902, 924, 937, 972, 995, 1057, 1062, 1100, 1102, 1106, 1118, 1122, 1131, 1151, 1164.

В. *Проблемы поиска*[1]

39, 43, 51, 54, 132, 133, 158, 243, 482, 662, 705, 727, 734, 739, 775, 799, 820, 868, 947, 1122.

В1. Численные алгоритмы, процессы принятия решения

47, 101, 106, 107, 214, 266, 291, 472, 490, 517, 672, 738, 792, 793, 851, 877, 880, 888, 1133.

В2. Несовершенный поиск, связанный с ошибками, в противоположность к процессам принятия решений

267, 268, 384, 385, 787, 789, 792, 793, 1133

[1] Эти проблемы особенно трудно выделить из других.

В3. Обсуждение проблем эффективности при большом поиске

36, 47, 52, 132, 267, 268, 344, 345, 491, 648, 672, 740, 789, 822, 883, 950, 1165.

В4. Эвристики для сокращения объема поиска

36, 52, 47, 101, 106, 107, 641, 745, 1075, 1111.

В5. Расчленение проблем на подпроблемы

36, 47, 52, 101, 106, 107, 132, 259, 345, 482, 745, 1039, 1075, 1133, 1165.

В6. Схемы последовательного улучшения

21, 22, 52, 55, 92, 101, 106, 107, 259, 353, 482, 490, 644, 727, 740, 745, 883, 974, 1075, 1111.

В7. Проблемы локальных максимумов и «Mesa-феномен» (явление плоских участков)

191, 259, 344, 345, 491, 727, 740.

В8. Подъем по градиенту; многократная одновременная оптимизация

21, 22, 36, 312, 344, 345, 353, 472, 491, 531, 549, 623, 740, 974, 976, 1150.

В9. Случайность и творческая интуиция

65, 189, 192, 194, 233, 243, 549, 633, 729.

Г. *Распознавание образов и восприятие*

115, 158, 162, 279, 389, 486, 574, 704, 705, 713, 739, 743, 862, 899, 902, 913, 969, 991, 995, 1018, 1020, 1047, 1067, 1078, 1090, 1091, 1096, 1109, 1154.

Г1. Общее обсуждение, обзоры

67, 71, 86, 159, 225, 253, 284, 308, 314, 315, 393, 395, 415, 420, 459, 532, 552, 568, 569, 575, 641, 657, 659, 724, 732, 734, 775, 820, 866, 897, 901, 973, 974, 977, 1038, 1048, 1050, 1054, 1098, 1100, 1103, 1118, 1174.

Г2, Г3. Обсуждение критериев сравнения

67, 225, 276, 290, 310, 315, 338, 348, 386, 471, 496, 515, 667, 694, 822, 973, 974, 977, 1041, 1044, 1048, 1067, 1124.

Г4. Списки признаков или схемы символ-векторов

72, 129, 130, 283, 284, 285, 290, 320, 346, 441, 601, 694, 724, 729, 822, 950, 973, 974, 1024, 1044, 1050.

Г5. Схемы, использующие расчленение объектов

67, 76, 77, 141, 195, 295, 346, 424, 440, 449, 458, 459, 496, 581, 634, 712, 732, 735, 746, 993, 1050, 1087, 1098.

Г6. Зрительное восприятие человека

7, 14, 67, 68, 166, 193, 315, 341, 380, 381, 416, 437, 446, 456, 458, 485, 524, 525, 526, 568, 569, 570, 592, 593, 646, 649, 661, 709, 777, 778, 779, 865, 919, 967, 977, 1052, 1054, 1089.

Г7. Незрительное восприятие, особенно машины, распознающие устную речь

94, 167, 188, 260, 261, 262, 263, 265, 276, 317, 335, 347, 348, 442, 446, 496, 581, 609, 659, 711, 853, 861, 1001, 1045, 1052, 1156.

Г8. Преобразование изображений, особенно локально-однородных («ядра»)

67, 71, 141, 175, 263, 284, 300, 532, 552, 584, 724, 973, 974, 993, 1050, 1051, 1097, 1098.

Г9. Распознавание образов как задача теории статистических решений

218, 285, 310, 315, 338, 454, 472, 657, 740, 822, 930, 931, 932, 933, 934, 936, 970, 971, 1050, 1091, 1097, 1100, 1102, 1103, 1104, 1105, 1106, 1153.

Г10. Машины, читающие печатные или рукописные буквы

74, 79, 92, 129, 130, 141, 168, 218, 283, 285, 296, 298, 302, 332, 338, 341, 346, 387, 412, 413, 414, 424, 440, 457, 468, 469, 470, 471, 473, 533, 534, 539, 575, 595, 657, 667, 724, 821, 852, 918, 930, 977, 993, 1041, 1044, 1048, 1050, 1067, 1087, 1098, 1124.

Г11. Отыскание полезных свойств для различения образов

71, 72, 77, 172, 175, 181, 263, 276, 279, 341, 382, 386, 396, 441, 457, 493, 494, 496, 526, 539, 552, 598, 641, 659, 667, 694, 724, 729, 732, 740, 798, 822, 970, 971, 973, 974, 976, 1031, 1041, 1051, 1087, 1089, 1092, 1122, 1138.

Г12. Инвариантные к преобразованиям свойства, не требующие предварительной нормализации

71, 72, 141, 175, 279, 285, 443, 444, 492, 493, 494, 532, 598, 682, 725, 865, 918, 934, 977, 1050, 1051, 1152.

Г13. Различение образов с помощью взвешенных сумм относительно простых свойств

129, 130, 157, 218, 276, 285, 359, 454, 523, 541, 657, 667, 740, 761, 762, 918, 930, 931, 932, 933, 934, 936, 976, 1067, 1086, 1100, 1104, 1106, 1124.

Г14. Системы, использующие иерархии распознающих устройств

195, 276, 493, 494, 581, 598, 712, 715, 738, 777, 778, 779, 936, 947, 976, 977, 1073, 1100, 1103, 1104, 1106.

Д. *Обучающиеся системы*

8, 22, 23, 43, 51, 133, 196, 211, 254, 280, 298, 308, 320, 349, 372, 389, 421, 434, 482, 499, 500, 506, 561, 648, 734, 739, 764, 773, 775, 807, 820, 825, 829, 831, 835, 836, 838, 841, 843, 915, 916, 917, 936, 947, 994, 1020, 1025, 1047, 1052, 1059, 1078, 1085, 1112, 1122, 1145, 1175.

Д1. Обучение животных

272, 459, 474, 497, 584, 729, 775, 1023, 1054, 1072, 1073, 1138, 1185.

Д2. Обучение человека

14, 83, 213, 272, 320, 379, 416, 459, 460, 474, 563, 715, 775, 807, 813, 827, 914, 991, 1013, 1021, 1022, 1023.

Д3. Подкрепление (поощрение, подавление)

92, 182, 183, 191, 213, 225, 243, 309, 313, 344, 345, 370, 397, 410, 541, 549, 551, 584, 623, 667, 719, 720, 726, 729, 740, 815, 822, 918, 930, 931, 932, 950, 976, 1022, 1023, 1152, 1164.

Д4. Вычисление корреляции

36, 312, 344, 345, 515, 529, 531, 549, 667, 694, 740, 786, 932, 950, 1046, 1138.

Д5. Ассоциации, теории обучения без подкрепления

92, 318, 319, 320, 350, 456, 458, 459, 586, 719, 720, 883, 1049, 1102, 1103, 1104, 1105, 1106.

Д6. Подтверждение выдвигаемых гипотез

77, 129, 130, 213, 326, 327, 328, 568, 569, 648, 735, 786, 798, 822, 895, 1031, 1033.

Д7. «Адаптация», регулирование внутренних параметров

26, 41, 47, 55, 102, 109, 189, 192, 243, 259, 312, 529, 827, 836, 839, 969, 1030, 1050, 1090.

Д8. «Обусловливание»

26, 153, 486, 584, 783, 864, 897, 901, 902, 1128, 1129, 1143, 1185.

Д9. Статистические теории обучения

39, 66, 110, 182, 183, 309, 315, 337, 357, 359, 370, 410, 523, 584, 738, 761, 762, 815, 826, 893, 973, 974, 1102, 1103, 1105, 1147, 1150

Д10. «Заучивание», нецифровая запись в память машины

320, 619, 662, 950, 984, 986, 1073, 1127.

Е. *Схемы планирования*

Общее обсуждение: 335, 358, 434, 561, 705, 734, 735, 739, 775, 786, 820, 832, 934.

Е1. Внутренние модели окружающей среды

125, 157, 248, 640, 641, 648, 726, 729.

Е2. Предсказание результатов предполагаемого действия

369, 568, 569, 640, 641, 645, 720, 726, 729.

Е3. Использование семантических моделей

363, 365, 366, 633, 644, 729, 730, 732, 735

Е4. Использование упрощенных, по возможности гомоморфных, моделей

52, 158, 452, 561, 729, 732, 745, 798, 913, 986, 1075.

Е5. Построение внутренних абстрактных моделей

648, 729, 732, 735, 798, 919

Е6. Стратегии, используемые человеком при «планировании»

233, 273, 274, 275, 422, 590, 624, 705, 715, 785, 826, 870, 871, 913, 982, 983, 1008.

Ж. *Решение задач*

51, 91, 144, 321, 329, 355, 358, 421, 434, 572, 579, 590, 620, 705, 734, 739, 773, 775, 799, 802, 803, 804, 810, 820, 827, 832, 843, 868, 910, 1012, 1052, 1058, 1059, 1060, 1074, 1075, 1078, 1080, 1168, 1181

Ж1. Проблема управления при эвристическом программировании

105, 364, 729, 730, 732, 738, 745, 769, 770, 772, 789, 795, 798, 885, 907, 908, 911, 912, 913, 961, 1024.

Ж2. Предел дерева поиска со статической функцией оценки (игра)

101, 120, 121, 343, 553, 786, 792, 950, 982, 983, 1043, 1086, 1152.

Ж3. Явное использование целей и подцелей

101, 106, 107, 120, 121, 223, 273, 274, 275, 364, 490, 673, 688, 705, 715, 726, 729, 775, 786, 791, 792, 793, 802, 810, 1039, 1111.

Ж4. Решение задач человеком (психологическая литература)

145, 172, 193, 213, 226, 273, 274, 275, 286, 326, 327, 328, 357, 378, 435, 459, 461, 487, 501, 521, 522, 536, 561, 568, 569, 608, 615, 625, 709, 713, 754, 755, 789, 791, 792, 793, 795, 798, 802, 803, 804, 807, 810, 857, 858, 859, 867, 870, 871, 875, 895, 896, 906, 911, 912, 913, 920, 1006, 1008, 1012, 1013, 1074, 1115, 1146

Ж5. Эффект применения различных тренировочных последовательностей

124, 145, 344, 345, 356, 536, 563, 625, 705, 732, 791, 792, 793, 857, 870, 871, 911, 912, 913, 946, 1007, 1021, 1023, 1031.

Ж6. Умозаключение и открытие (человек)

124, 145, 172, 194, 278, 280, 378, 458, 459, 487, 625, 705, 715, 791, 793, 795, 798, 802, 803, 804, 867, 899, 900, 920.

Ж7. Машинное доказательство теорем

241, 267, 268, 291, 363, 364, 365, 366, 384, 385, 556, 675, 729, 730, 732, 738, 787, 789, 795, 798, 802, 803, 804, 867, 871, 877, 888, 920, 1024, 1133, 1134, 1135

Ж8. Использование дедуктивной логики при решении задач

222, 241, 291, 360, 361, 384, 385, 654, 671, 673, 705, 735, 754, 787, 789, 795, 798, 845, 899, 900, 911, 912, 913.

Ж9. Задачи, требующие программирования последовательных действий

16, 191, 344, 345, 549, 673, 798, 1014

Ж10. Выбор символ-метода машинами

363, 365, 366, 729, 732, 795, 798, 913, 1024.

И. *Языки*

51, 140, 321, 714, 775, 848, 1052, 1096.

И1. Естественные языки

84, 85, 134, 199, 207, 208, 209, 210, 211, 212, 213, 216, 393, 417, 449, 450, 610, 611, 612, 613, 656, 683, 706, 807, 816, 846, 861, 957, 979, 1022, 1023, 1156, 1163, 1177, 1178.

И2. Формальные языки

88, 196, 207, 208, 209, 210, 211, 212, 214, 215, 240, 266, 380, 417, 449, 509, 556, 557, 599, 706, 759, 787, 790, 795, 798, 816, 876, 883, 913, 939, 1032, 1033, 1034, 1081, 1133, 1163, 1178.

И3. Языковые программирующие системы

20, 73, 110, 117, 139, 171, 196, 229, 239, 263, 299, 307, 409, 417, 481, 519, 666, 674, 705, 772, 773, 790, 791, 795, 798, 799, 800, 817, 854, 908, 911, 913, 923, 925, 962, 989, 990, 1004, 1081, 1097, 1098.

И4. Программирующие системы оперирования символами

171, 196, 248, 295, 340, 363, 364, 367, 416, 417, 418, 475, 530, 673, 674, 705, 716, 745, 775, 787, 790, 791, 792, 793, 795, 798, 800, 805, 809, 817, 854, 911, 913, 925, 989, 990, 1024, 1053, 1075, 1163, 1177.

И5. Язык как средство мышления и общения

80, 81, 82, 83, 134, 166, 167, 193, 204, 205, 213, 232, 487, 568, 569, 586, 596, 615, 638, 642, 647, 706, 717, 735, 748, 749, 750, 758, 786, 818, 829, 845, 943, 948, 957, 974, 1022, 1123.

И6. Язык и его кодирование для моделей

179, 211, 248, 610, 611, 612, 613, 633, 647, 656, 748, 749, 784, 786, 798, 812, 1004.

И7. Теория информации и теория кодирования

15, 67, 68, 80, 81, 82, 83, 86, 164, 204, 205, 234, 244, 245, 301, 338, 388, 457, 512, 514, 634, 637, 642, 644, 647, 648, 656, 708, 709, 717, 743, 744, 862, 893, 943, 967, 979, 980, 1025, 1061, 1152, 1171.

К. *Машины индуктивного вывода*

51, 434, 714, 739, 947.

К1. Условная вероятность

22, 23, 25, 86, 169, 233, 337, 348, 399, 436, 437, 475, 515, 545, 580, 624, 641, 644, 729, 738, 870, 871, 874, 973, 974, 986, 1026, 1035, 1102, 1103, 1104, 1105, 1106, 1137, 1147, 1153, 1185.

К2. Грамматическая индукция; абстрагирование формы набора формальных выражений

169, 213, 713, 729, 732, 734, 738, 787, 812, 820, 973, 974, 1032, 1033, 1034, 1035, 1092.

К3. Сокращенное кодирование

25, 662, 862, 934, 974, 979, 1032, 1033.

К4. Выдвижение и подтверждение гипотез

6, 16, 172, 211, 397, 488, 489, 502, 503, 563, 640, 641, 648, 713, 779, 798, 874, 1032, 1033, 1092, 1137, 1145.

К5. Теории индуктивного вывода

399, 456, 507, 624, 738, 870, 871, 874, 948, 1035, 1036, 1137.

К6. Простота и индукция

176, 399, 402, 404, 543, 546, 672, 682, 732, 738, 874, 980, 1031, 1033, 1035.

Л. *Эвристики*

259, 321, 344, 377, 433, 620, 654, 664, 705, 739, 744, 773, 885, 912.

Л1. Обсуждение эвристик для машинного решения задач

52, 95, 120, 121, 149, 345, 365, 366, 384, 385, 480, 553, 614, 662, 664, 715, 729, 730, 734, 735, 738, 745, 772, 775, 786, 787, 789, 791, 792, 793, 795, 806, 820, 911, 913, 1017, 1024, 1031, 1043, 1074, 1075, 1086, 1099, 1134, 1171.

Л2. Обсуждение эвристик, используемых человеком при решении задач

52, 120, 149, 194, 213, 273, 274, 275, 286, 397, 422, 435, 501, 522, 563, 672, 688, 715, 745, 770, 772, 775, 786, 787, 789, 791, 792, 793, 795, 798, 799, 802, 803, 804, 807, 810, 867, 868, 870, 871, 907, 908, 910, 911, 913, 1012, 1013, 1024, 1074, 1075, 1115, 1133.

М. *Теории функций мозга*

28, 119, 293, 485, 767, 768, 771, 773, 775, 832, 944, 959, 1090, 1120, 1152, 1170, 1182.

М1. Усиление связи нейронов. Повышение проводимости синапса

22, 70, 92, 313, 401, 458, 459, 584, 726, 930, 931, 1164.

М2. Теории клеточных ансамблей

31, 92, 225, 401, 458, 459, 719, 720, 721, 722, 726, 740, 924, 976.

М3. Другие нейрофизиологические модели

18, 26, 31, 40, 70, 86, 200, 230, 231, 253, 394, 442, 446, 451, 459, 493, 494, 532, 570, 577, 578, 584, 585, 586, 598, 661, 693, 719, 720, 721, 722, 726, 766, 769, 770, 847, 848, 856, 861, 865,

878, 879, 897, 898, 900, 901, 902, 907, 913, 936, 994, 995, 996, 1037, 1054, 1066, 1100, 1101, 1106, 1116, 1126, 1130, 1142, 1182, 1184, 1185.

М4. Сравнение машины с мозгом

41, 158, 165, 252, 373, 398, 401, 633, 635, 636, 639, 646, 650, 678, 679, 682, 686, 689, 715, 718, 782, 784, 802, 803, 804, 856, 866, 886, 1019, 1027, 1038, 1040, 1065, 1128, 1129.

Н. *Эпистемологические вопросы*

31, 144, 220, 237, 247, 254, 281, 400, 477, 537, 544, 635, 636, 639, 677, 679, 682, 686, 689, 734, 739, 802, 803, 804, 810, 814, 819, 820, 827, 829, 832, 837, 841, 843, 866, 874, 940, 941, 948, 968, 1085, 1152, 1159, 1167, 1182.

Н1. Что может знать машина?

143, 456, 676, 685, 825, 1029, 1064, 1084.

Н2. Может ли машина мыслить? Природа интеллекта

35, 59, 76, 243, 398, 542, 582, 616, 633, 650, 685, 688, 732, 742, 789, 919, 927, 939, 942, 953, 1013, 1036, 1064, 1082, 1085, 1132, 1133, 1141, 1142, 1160, 1161, 1162, 1173.

Н3. Свобода воли человека и машины

146, 645, 650, 685, 945, 953, 1023, 1085.

Н4. Проблема разум—мозг

7, 152, 293, 325, 376, 401, 456, 587, 645, 650, 669, 702, 948, 1023, 1038, 1071, 1085.

П. *Память и поиск информации*

60, 340, 773, 793, 931, 1052, 1142, 1182.

П1. Поиск полезной информации

67, 85, 164, 184, 318, 319, 320, 350, 417, 540, 627, 628, 715, 735, 748, 749, 750, 773, 789, 844, 883, 903, 950, 951, 1003, 1034, 1049, 1164.

П2. Соображения относительно хранимой информации

417, 611, 612, 613, 626, 671, 673, 735, 747, 749, 750, 791, 792, 844, 1049, 1137.

П3. Человеческая память (психологическая литература)

67, 68, 165, 193, 286, 307, 310, 320, 322, 350, 380, 397, 459, 536, 563, 577, 578, 592, 593, 685, 709, 710, 717, 715, 1040, 1116.

Р. *Сервомеханизмы и механизмы стабилизации. Кибернетика*

18, 43, 51, 100, 116, 154, 158, 242, 321, 352, 544, 571, 588, 630, 670, 742, 811, 824, 827, 828, 830, 832, 836, 837, 841, 843, 869, 892, 897, 901, 943, 947, 960, 978, 1010, 1025, 1027, 1028, 1029, 1055, 1083, 1140, 1157, 1166.

Р1. Адаптирующиеся управляющие системы

25, 54, 102, 103, 104, 153, 312, 342, 371, 515, 588, 602, 605, 606, 723, 865, 905, 947, 1150, 1152.

Р2. Механизмы целенаправленного поведения («телеологические» механизмы)

22, 47, 55, 223, 281, 303, 554, 646, 940, 941, 965, 1152.

Р3. Автоматизация. Машины и общество

2, 97, 282, 371, 495, 588, 698, 700, 999, 1152, 1155, 1158.

Р4. Кибернетика и психиатрия и т. п.

6, 17, 18, 50, 111, 162, 206, 237, 351, 561, 577, 578, 597, 684, 1056, 1070, 1144, 1152.

Р5. «Самоорганизующиеся» системы

42, 47, 71, 155, 162, 203, 304, 305, 336, 390, 825, 826, 829, 831, 833, 837, 840, 1116, 1118, 1119.

Р6. Гомеостазис и устойчивость

40, 41, 44, 47, 49, 52, 54, 55, 61, 189, 303, 370, 589, 826, 1113, 1116, 1126, 1140, 1152.

Р7. Социальные организации

1, 3, 30, 37, 38, 65, 97, 98, 99, 128, 235, 258, 266, 429, 484, 591, 913, 929, 963, 1000, 1010.

С. *Некоторые специальные вопросы*

330, 580, 929, 963, 1176.

С1. Теория игр по Нейману [785]

3, 127, 355, 370, 624, 785, 826, 827, 998, 999, 1000, 1147.

С2. Теория статистических решений

3, 4, 5, 127, 355, 393, 472, 563, 624, 657, 998, 999, 1000, 1007, 1125.

C3. Взаимодействие человека с машиной

110, 184, 247, 308, 356, 608, 827, 836, 843, 1002, 1004, 1063, 1180.

C4. Самовоспроизводящиеся машины

178, 179, 482, 513, 544, 617, 618, 619, 752, 765, 782, 784, 849, 850.

C5. Игровые задачи

48, 95, 120, 121, 187, 273, 274, 275, 398, 436, 438, 511, 538, 551, 553, 573, 624, 662, 688, 703, 705, 786, 792, 797, 826, 851, 881, 885, 894, 915, 916, 950, 952, 974, 982, 983, 986, 1016, 1043, 1053, 1086, 1147, 1152.

C6. Программы музыкальной композиции

169, 263, 475, 476, 535, 666, 908.

C7. Обучение в лабиринте

243, 727, 753, 855, 984, 1127.

C8. Промышленные применения

96, 97, 128, 227, 228, 234, 258, 463, 510, 741, 885, 998, 999.

Дополнительная классификация

Была сделана попытка создать более общую классификацию статей, но эта классификация носит предварительный характер и не свободна от ошибок. Рубрики в этой классификации обозначаются двузначными цифрами.

1. Характер программ:
- **11.** Немашинный эксперимент.
- **12.** Программа для универсальных вычислительных машин.
- **13.** Эксперимент с использованием специального оборудования.
- **14.** Использование специальных программирующих систем.
- **15.** Для практического применения.
- **16.** Психологический эксперимент.

2. Уровень рассмотрения:
- **21.** Математический.
- **22.** Психологический.
- **23.** (Нейро-)Физиологический.
- **24.** Философский.

3. Характер статьи:
- **31.** Обзорная статья.
- **32.** Содержит обширную библиографию.
- **33.** Предлагается эксперимент.
- **34.** Сообщаются результаты эксперимента.
- **35.** Учебное руководство.
- **36.** Общее обсуждение.

БИБЛИОГРАФИЯ

Ссылки на литературу расположены в алфавитном порядке и перенумерованы [1], статьи одного и того же автора даны в хронологическом порядке.

Вслед за библиографической ссылкой приведены соответствующие данной работе дескрипторные индексы (см. тематический указатель, стр. 479), расположенные в большинстве случаев (но не всегда) в порядке их важности. Они обозначены русской прописной буквой и цифрой, набранными жирным шрифтом; приведенные там же двузначные жирные цифры позволяют определить характер статьи (см. классификацию в конце тематического указателя, стр. 490).

Затем в некоторых случаях приводятся ссылки на другие работы, посвященные близким по тематике вопросам. Они указаны порядковым номером статьи в списке литературы, заключенным в квадратные скобки. Латинские буквы в квадратных скобках указывают сборник или симпозиум, в котором данная работа помещена наряду с другими (список таких сборников помещен в конце библиографии на стр. 543—546). Настоящая библиография может послужить хорошей основой для создания в будущем полного библиографического указателя работ по «искусственному разуму».

1*. Abelson R. F., Computer simulation of "hot" cognition, см. Tomkins S., Messick S. (eds.), Computer simulation of personality, Wiley, New York, 1963. **P7**; **16**, 22.

2. Ackoff R. L., Automatic management: A forecast and its educational implications, *Management Sci.*, 2, № 1, 55—60 (1955). **P3**.

3. Ackoff R. L., Games, decisions and organizations, *General Systems*, 4, 145—150 (1959). **C1**, **C2**, **P7**.

4. Ackoff R. L., Scientific Method: Optimizing Applied Research Decisions, Wiley, New York, 1962. **C2**.

5. Adey W. R., Instrumentation of Nervous System for Studies of Behavior, presented at the 14th Annual Meeting of the ARS, 1959. **C2**.

6*. Adey W. R., Computer applications at the biomedical research, AFIPS Conf. Proc., vol. 24, Spartan Books, Baltimore, 1963, pp. 603—607. **K4**, **P4**; **23**.

7. Adrian E. D., The physicial background of perception, The Waynefiete Lectures, Oxford, New York, 1946. **Г6**, **M**, **H4**.

8*. Айзерман М. А., Опыты по обучению машин распознаванию зрительных образов, см. Биологические аспекты кибернетики, Изд-во АН СССР, 1962. **Д**; **21**, **34**.

9*. Айзерман М. А., Гусев Л. А., Розоноэр Л. И. и др., Логика, автоматы, алгоритмы, Физматгиз, 1963. **A1**, **A2**; **35**.

10*. Айзерман М. А., Браверман Э. М., Розоноэр Л. И., Теоретические основы метода потенциальных функций в задаче об обучении автоматов и распознаванию планов, *Автомат. и телемех.*, 25, 6 (1964). **21**.

[1] Звездочкой отмечены работы, добавленные в русском издании книги.

11*. Akers S. B., Rutter B. H., The use of threshold logic in pattern recognition, *Wescon 63*, Aug. 20—23, **4**, 1—7 (1963); русский перевод: Экерс С., Раттер Б., Использование пороговой логики для распознавания образов, *Зарубежная радиоэлектроника*, № 6 (1964). **21.**
12. Allanson J. T., Some properties of a randomly connected neural network, см. [I], chap. 30. **Б4.**
13. Allanson J. T., The reliability of neurons, см. [F]. **Б6, Б3.**
14. Allport F. H., Theories of Perception and the Concept of Structure, Wiley, New York, 1955. **Г6, Д2, М; 31, 36.**
15. Alluisi E. A., Conditions affecting the amount of information in absolute judgements, *Psychol. Rev.*, **64**, 97—103 (1957). **И1.**
16. Amarel S., An approach to automatic theory formation, см. [W]. **К4, Ж9, Ж11.**
16a. Amarel S., On the mechanization of creative processes, *IEEE Spectrum*, 3, 112—114 (1966).
17*. Амосов Н. М., Кибернетика и живой организм, изд-во «Наукова думка», Киев, 1964. **Р4; 23.**
18*. Амосов Н. М., Моделирование мышления и психики, изд-во «Наукова думка», Киев, 1964. **Р, Р4, М3; 23.**
19*. Амосов Н. М. (ред.), Моделирование в биологии и медицине, изд-во «Наукова думка», Киев, 1965.
20. Andree R. V., Programming the IBM 650 Magnetic Drum Computer and Data Processing Machine, Holt, New York, 1958.
21. Andrew A. M., Machines which learn, *New Scientist*, **4**, 1383 (Nov. 27, 1958). **В6, В8, 36.**
22. Andrew A. M., Learning machines, см. [N], pp. 475—505. **В6, В8, Д, К1, М1, Р2; 13, 36.**
23. Andrew A. M., Conditional probability computer, см. [N], pp. 945—946. **Д, К1; 34.**
24. Andrew A. M., Learning in a non-digital environment, см. [AG].
25. Andrew A. M., A self-optimizing system of coding, см. [X]. **К1, К3, Р1; 13, 34.**
26. Angyan A. J., Machina reproductrix, an analogue model to demonstrate some aspects of neural adaptation, см. [N], Vol. 2, pp. 933—943. **Б5, Д7, Д8, М3; 13, 22, 34, 36.**
27*. Анохин П. К., Физиология и кибернетика, см. Философские вопросы кибернетики, Соцэкгиз, 1961, стр. 262—305. **36.**
28*. Анохин П. К., Опережающее отражение действительности, *Вопр. филос.*, № 7, 97—111 (1962). **М; 23.**
29*. Anonimous, Current research and development in scientific documentations, 14, Nat. Sci. Foundation, Washington, Rep. NSF-66-17, 1966. **Г, И, К, Н; 32, 36.**
30*. Ansoff H. I., Corporate strategy, McGraw-Hill, New York, 1965. **Р7.**
31*. Arbib M. A., Brains, machines and mathematics, McGraw-Hill, New York, 1964. **Б6, М2, М3, Н; 21, 23, 35, 36.**
32. Arin E. I., см. [321].
33*. Aris R., Discrete Dynamic Programming, Blaisbell, New York, 1964.
34*. Аркадьев А. Г., Браверман Э. М., Обучение машины распознаванию образов, изд-во «Наука», 1964.
35. Armer P., Attitudes toward intelligent machines, см. [S] и [1065]. **Н2; 32.**
36. Arnold R. F., A compiler capable of learning, см. [Q], pp. 137—143; русский перевод: Арнольд Р. Ф., Компилирующая программа, способная обучаться, Кибернетический сборник, вып. 4, ИЛ, 1962. **В3, В4, В5, В8, Д4; 12, 33.**
37. Asch S. E., Social Psychology, Prentice-Hall, New York, 1952. **Р1.**
38*. Asch S. E., The metaphor: a psychological inquiry, см. Taguri R., Petrullo L. (eds.), Person perception and interpersonal behavior, Stanford Univ. Press, Stanford., Calif., 1958. **Р7; 22, 36.**
39. Ashby W. R., The physical origin of adaptation by trial and error, *J. General Psychol.*, **32**, 13—25 (1945). **В, Д9.**
40. Ashby W. R., The behavioral properties of systems in equilibrium, dynamics of the cerebral cortex, *Am. J. Psychol.*, **59**, 682—686 (1946). **М3, Р6.**
41. Ashby W. R., The nervous system as a physical machine: with special reference to the origin of adaptive behavior, *Mind*, **56**, 1—16 (1947). **М4, Д7, Р6.**
42. Ashby W. R., Principles of the self-organizing dynamic system, *J. General Psychol.*, **37**, 125—128 (1947). **Р5.**
43. Ashby W. R., Design for a brain, *Electronic Eng.*, **20**, 379—383 (1948). **В, Р, Д.**
44. Ashby W. R., The stability of a randomly assembled nerve network, *EEG a. Clin. Neurophysiol.*, 2, 471—482 (1950). **Б4, Р6.**
45. Ashby W. R., The cerebral mechanisms of intelligent behavior, см. Richter D. (ed.), Perspectives in Neuropsychiatry, London, 1950.
46. Ashby W. R., Statistical machinery, *Thales*, **7**, № 1, 1—3 (1951). **А6.**
47. Ashby W. R., Design for a Brain, Wiley, New York, 1952. (rev. ed. 1960); русский перевод: Эшби У. Р., Конструкция мозга. Происхождение адаптивного поведения, ИЛ, 1962; изд. 2-е, дополн., изд-во «Мир», 1964. **Р6, А1, В1, В3, В4, В5, Д7, Р5, Р2; 21, 35, 13, 34.**
48. Ashby W. R. Can a mechanical chess player outplay its designer? *Brit. J. Philos. of Sci.*, 3, 44—57 (1952). **С5.**
49. Ashby W. R., Homeostasis, см. [B], 9th Conf., pp. 73—108. **Р6.**
50. Ashby W. R., The application of cybernetics to psychiatry, *J. Mental Sci.*, **100**, 114—124 (1954). **Р4.**
51. Ashby W. R., An Introduction to Cybernetics, Wiley, New York, 1956; русский перевод: Эшби У. Р., Введение в кибернетику, ИЛ, 1959. **А1, В, Д, Ж, К, Р; 35.**
52. Ashby W. R., Design for an intelligence amplifier, см. [G], pp. 215—234. **В3, В4, В5, В6, Р6, Е4, Л1, Л2; 13** («Гомеостат»).
53. Ashby W. R., Cybernetics, *Recent Progr. in Psychiatry*, **3**, 94—117 (1958).
54. Ashby W. R., Requisite variety and its implications for the control of complex systems, *Cybernetica*, **1**, 83—99 (1958).
55. Ashby W. R., The mechanism of habituation, см. [N], pp. 95—113. **А1, В6, Д7, Р2, Р6; 21.**
56. Ashby W. R., The brain as regulator, *Nature*, **186**, 413 (1960).
57. Ashby W. R., Computers and decision making, *New Scientist*, **7**, 746 (1960).
58*. Ashby W. R., Cybernetics Today and its Future Contribution to Engineering Sciences, New York, 1961; русский перевод см. [1065].
59. Ashby W. R., What is an intelligent machine? см. [Z], pp. 278—280; русский перевод: Эшби У. Р., Что такое разумная машина? *Зарубежная радиоэлектроника*, № 3, 67 (1962). **Н2.**
60. Ashby W. R., Riguet J., The avoidance of over-writing in self-organizing systems, *J. Theor. Biol.*, **1**, 431—439 (1961). **П.**
61. Ashby W. R., Principles of the self-organizing system, см. [W], pp. 255—278. **Р6.**
62*. Ashby W. R., The self-reproducing system, см. Aspects Theory Artificial Intelligence, Plenum Press, New York, 1962, pp. 9—18. **С4.**
63*. Ashby W. R., Systems and information, *IEEE Trans.*, April—July (1963); русский перевод: Эшби У. Р., Системы и информация, *Зарубежная радиоэлектроника*, № 5 (1964). **И7; 36**

64*. Асратян Э. А., Симонов П. В., Надежность мозга, Изд-во АН СССР, 1963. **М**; **23**.
65*. Atkinson J. W. (ed.), Motives in fantasy, action and society, Van Nostrand, Princeton, N. J., 1958. **В9**, **Р7**; **22**.
66. Atkinson R. C., An Analysis of Rote Serial Learning in Terms of a Statistical Model, диссертация, Indiana Univ., 1954. **Д9**.
67. Attneave F., Some informational aspects of visual perception, *Psychol., Rev.*, **61**, № 3, 183—193 (1954). **Г6**, **Г1**, **Г3**, **Г5**, **Г8**, **И7**, **П1**, **П3**; **16**.
68. Attneave F., Symmetry, information and memory for patterns, *Am. J. Psychol.*, **68**, 209—222 (1955). **Г6**, **И7**, **Р3**.
69. Babbage C., см. [150].
70. Babcock M. L., Reorganization by Adaptive Automation, Technical Rep. 1, Contr. Nonr 1834 (21), Electr. Eng. Res. Lab., Engineering Experiment Station, Univ. of Illinois, Urbana, Ill., 1960. **Б5**, **Б6**, **М1**, **М3**; **13**, **34**, **23**, **36**.
71. Babcock M. L. et al., Some Principles of Pre-organization in Self-organizing Systems, Techn. Rep. 2, Contr. Nonr 1834 (21), Electr. Eng. Res. Lab., Engineering Experiment Station, Univ. of Illinois, Urbana, Ill., 1960. **Г1**, **А1**, **Б6**, **Г3**, **Г11**, **Г12**, **Б2**, **А2**, **Б4**, **Р5**; **21**.
72. Babcock M. L., Some physiology of automata, см. [Z], pp. 291—293. **Б2**, **Б6**, **Г1**, **Г4**, **Г11**, **Г12**; **13**, **34**.
73. Backus J. W., Automatic programming: properties and performance of FORTRAN systems I and II, см. [N], pp. 233—248. **И3**; **15**.
74. Bailey C. E. G., Norrie C. O., Automatic reading of typed or printed characters, British IRE Convention on Electronics in Automation, 1957. **Г10**.
75. Bales R. F., Small-group theory and research, см. Merton, Broom, Cottrell (eds.), Sociology Today, Basic Books, New York, 1959.
76. Balescu I., Future possibilities and limitations in reproducing brain mechanism by electronic machines, *Rev. of Science a. Sociology* (Bucharest), **1**, № 2, 45 (1956). **Б5**, **Н2**.
77. Banerji R. B., An Information-processing Program for Object Recognition, Dept. of Engineering Administration, Case Inst. of Technol., Cleveland, Ohio, 1960. **Г5**, **Г11**, **Д6**.
78*. Бардзинь Я. М, Емкость среды и поведение автоматов, *ДАН СССР*, **160**, № 2, 302—305 (1965) **А**; **21**.
79. Baran P., Estrin G., An adaptive character reader, *IRE Wescon Convent. Rec.*, **4**, № 4, 29—41 (1960). **Д10**.
80. Bar-Hillel Y., Carnap R., Semantic information, *Brit. J. Philos. of Sci.*, **4**, 127 (1953). **И7**, **И5**.
81. Bar-Hillel Y., Carnap R., An Outline of the Theory of Semantic Information, Techn. Rep. 247, Res. Lab. of Electronics, MIT, Cambridge, Mass., 1953. **И5**, **И7**.
82. Bar-Hillel Y., An examination of information theory, *Philos. of Sci.*, **22**, 86—105 (1955). **И7**, **И5**.
83. Bar-Hillel Y., Semantic information and its measures, Trans. 10th Macy Conf. on Cybernetics, 1955. (Von Foerster H, ed.), см. [B], 10th. Conf., pp. 33—48. **И5**, **И7**.
84. Bar-Hillel Y., The present status of automatic translation of languages, см. Alt F. L. (ed.), Advances in Computers, vol. 1, Acad. Press, New York, 1960.
85. Bar-Hillel Y., Theoretical aspects of the mechanization of literature searching, см. Hoffman W. (ed.), Digitale Informationswandler, Vieweg, Braunschweig, 1962, pp. 406—443. **И1**.

86. Barlow H. B., Sensory mechanisms, the reduction of redundancy and intelligence, см. [N], vol. 2, pp. 537—574, **И7**, **Г1**, **М3**; **13**, (**К1**), **34**, **36**.
87. Barnett M. P., Digitized Description Processing; Part I: Elementary Considerations, NRL, Dept. of Chemistry, Univ. of Wisconsin, Madison, ASTIA Doc. AD 153312, 1958. **Г5**.
88. Barron D. W., Buxton I. N. et al., The main features of CPL, *Computer J.*, July (1963). **И2**; **14**.
89*. Бассин Ф., Бернштейн Н., Латаш Л., К проблеме связи между структурой и функциями мозга, сб. трудов X Всесоюзного съезда физиологов в Ереване, 1965. **23**.
90*. Barthel D., Simulation elementarer Lehrvorgänge auf einer elektronischen Rechenmashine, *Acta Biol. et Med. Germana*, **12**, № 6, 669 (1964). **Д1**; **23**, **34**.
91. Bartlett F. C., Thinking, Basic Books, New York, 1958. **Ж**.
92. Barus C., Machine Learning and Pattern Recognition, Progress Rep., Nat. Sci. Foundation Grant G-5945, Swarthmore College, Swarthmore, Pa., 1959. **В6**, **М2**, **Б4**, **Б6**, **Д3**, **Г10**, **Д5**, **М1**; **33**, **36**, **21**.
93. Barus C., A Scheme for Recognizing Patterns from an Unspecified Class, Progr. Rep., Nat. Sci Foundation, Grant G-5945, Swarthmore College, Swarthmore, Pa., 1961.
94. Bauman R. H., Licklider J. C. R., An electronic word recognizer, *J. Acoust. Soc. America*, **26**, 137 (1954). **Г7**.
95*. Baylor G. W., A heuristic approach to chess programs, *Carnegie Techn.*, **25**, № 1, 24—27 (1960). **Л1**, **С5**; **22**.
96. Beer S., Operational research and cybernetics, см. [F], pp. 29—45. **Р**, **С8**
97. Beer S., The impact of cybernetics on the concept of industrial organization, см. [F], pp. 535—554. **Р3**, **Р7**, **С8**.
98. Beer S, The scope for operational research in industry, *Inst. for Production Eng. J.*, **16**, № 2 (1957). **Р7**.
99. Beer S, Cybernetics and Management, English Univ. Press, London, 1957, 1960; Wiley, New York, 1961; русский перевод: Бир С., Кибернетика и управление производством, изд. 2-е, дополн., изд-во «Наука», 1965, **Р7**.
100. Беленеску И. Н., Кибернетика и некоторые вопросы психологии и философии, *Вопр. филос*, **11**, 153 (1957); Psychological Abstracta, 3472 (1958). **Р**; **22**, **23**.
101. Bellman R, Dynamic Programming, Princeton Univ. Press, Princeton, N. J, 1957; русский перевод: Беллман Р., Динамическое программирование, ИЛ, 1960. **В1**, **В4**, **В5**, **В6**, **Ж2**, **Ж3**; **21**.
102. Bellman R., Kalaba R., On communication processes involving learning and random duration, *IRE Nat. Conv. Rec*, **6**, № 4, 16—21 (1958). **Д7**, **Р1**.
103. Bellman R., Kalaba R., On adaptive control processes, *IRE Nat. Conv. Rec.*, **7**, № 4, 3—11 (1959); см. *IRE Trans. on Automatic Control*, **AC-4**, 1—9 (1959); русский перевод: Беллман Р., Калаба Р., О самонастраивающихся процессах управления, ГНТБ 62/62651, М., 1962. **Р1**.
104. Bellman R., Kalaba R., Dynamic programming and adaptive processes: mathematical foundations, *IRE Trans. on Automatic Control*, **AC-5**, 5—10 (1960); русский перевод: Беллман Р., Калаба Р., Математические основы динамического программирования и процессов приспособления, *Зарубежная радиоэлектроника*, № 11, 92—101 (1960).
105. Bellman R., Brock P., On the concepts of a problem and problem-solving, *Am. Mathem. Monthly*, **67**, № 2, 119—134 (1960). **Ж1**.
106*. Bellman R., Dynamic programming treatment of the travelling salesman problem, *J. Assoc. Computing Machinery*, **9**, № 1, 61—63 (1962);

русский перевод Беллман Р, Применение динамического программирования к задаче о коммивояжере, Кибернетический сборник, вып 9, изд-во «Мир», 1964 **В1 В4, В5, В6, Ж2, Ж3, 15, 21**

107*. Bellman R, Computers and decision making, *Computers a Automation*, **12**, № 1, 10—12, 14, 47 (1963), русский перевод Беллман Р, Вычислительные машины и принятие решений, *Зарубежная радиоэлектроника*, № 1, 50—55 (1964) **В1, В4, В5, В6, Ж3, 21, 36**

108 Bellman R, Mathematical model making as an adaptive process, Univ Calif Press, Math Optimiz Techn, Berkeley—Los Angeles 1963, pp 333—339 **Р1, 21, 35**

109* Bellman R, Adaptive control processes a guided tour, Princeton Univ Press, Princeton, N J, русский перевод Беллман Р, Процессы регулирования с адаптацией, изд во «Наука», 1964 **Д7, 21**

110 Bemer R W, A checklist of intelligence for programming systems, *Communic of the ACM*, **2**, № 3, 8—13 (1959) **Д9, И3, С3**

111 Berg A I, Cybernetics and society, Economic Gazette, Soviet Review, Internat Arts and Sciences Press, New York, 1960, англ перевод статьи Берг А И, Кибернетика и общество, Экономическая газета, 12 июня 1960 г **И4**

112* Берг А И (ред), *Возможное и невозможное в кибернетике*, сб статей, изд во АН СССР, 1963

113* Берг А И (ред) *Кибернетика, мышление, жизнь* изд-во «Мысль», 1964

114* Berge C, Theorie des graphes et ses applications, Dunod, Paris, 1958, русский перевод Берж К, Теория графов и ее применения, ИЛ, 1962 **21, 35**

115 van Bergeijk W A, Harmon L D, What good are artificial neurons? см [S], pp 395—406 **Б6, Г**

116 Berkeley E C, Giant Brains, or Machines That Think, Wiley, New York, 1949 **А, Б, Р, 35**

117* Berkeley E C, The programming language LISP an introduction and appraisal *Computers a Automation* **13**, № 9, 16—23 (1964) **И3, 21, 31.**

118* Berlyne D E, Structure and Direction in Thinking, Wiley, 1965 **22.**

119* Bernhard R, Towards a chemical kinetic brain model, *Cybernetica*, **6**, № 3, 158—173 (1963) **М, 31**

120 Bernstein A et al, A chess playing program for the IBM 704 computer, *Proc Western Joint Computer Conf*, 157—159 (1958). **С5, Ж2, Ж3, Л1, Л2, 12, 34, 36** [553, 794]

121 Bernstein A, Roberts M deV, Computer vs chess-player, *Scientific American*, **198**, 96—105 (1958) **С5, Ж2, Ж3, Л1, Л2, 12, 34, 36**

122* Бернштейн Н А, Новые линии развития в физиологии и их соотношение с кибернетикой, изд-во Ин та философ АН СССР, 1962

123* Бернштейн Н А, Очерки по физиологии движений и физиологии активности, изд во «Медицина», 1966

124 Birch H G, The relation of previous experience to insightful problem solving, *J of Comparative Psychol*, **38**, 367 (1945) **Ж5, Ж6**

125* Bishop G, Feedback through the environment as an analog of brain functioning, см [U], pp 122—144 **Е1**

126* Bjerstedt A, Mapping the effectstructure of self-instructional materials, *Programm Learn*, **2**, № 2, 99—109 (1965)

127 Blackwell D, Girshick M A, Theory of Games and Statistical Decisions, Wiley, New York, 1954, русский перевод Блекуэлл Д, Гиршик М, Теория игр и статистических решений, ИЛ, 1958 **С1, С2**

128 Blau P M, The Dynamics of Bureaucracy, Univ of Chicago Press, Chicago, 1955 **Р7, С8**

129 Bledsoe W W, Browning I, Pattern recognition and reading by machine, см [Q], pp 225—232 **Г10, Г13, Г4, Б6, Д6, 12, 34** [470, 471, 473 130]

130 Bledsoe W W, Further results on the N tuple character recognition method, *IRE Trans on Electronic Computers*, **EC-10**, № 1, 96 (1961) **Г10, Г13, Г4 Б6, Д6, 34**

131 Bledsoe W W, A basic limitation on the speed of digital computers, *IRE Trans on Electronic Computers*, **EC-10**, № 3 (1961) **А8, [160]**

132 Bledsoe W W, Lethally Dependent Genes Using Instant Selection, Panoramic Res Publ, Palo Alto, Calif, 1961, PRI-2 **В, Д**

133 Bledsoe W W, The Use of Biological Concepts in the Analytical Study of Systems, Panoramic Res Publ, Palo Alto, Calif, 1961, PRI 2 **В, Д.**

134 Bloomfield L, Language, Holt, New York, 1933 **И1, И5, 36**

135 Blum G S, A Model of the Mind, Wiley, New York, 1961

136 Blum M, McCulloch W S, On parallel computation, Res Lab of Electronics, MIT, *Quart Progr Rep*, **56**, 194—195 (1960). **Б2, Б3, А6**

137 Blum M, Properties of a neuron with many inputs, см [S] **Б6, Б3,** см также [W]

138 Blumberg D F, New directions for computer technology and applications, A long range prediction, *Comput a Automat*, **13**, № 1, 8—17 (1964), русский перевод Бламберг Д, Анализ современного состояния вычислительной техники в США и перспективы ее развития *Зарубежная радиоэлектроника* № 8 (1964)

139 Bobrow D G, Raphael B A, Comparison of list processing computer languages *Commun ACM*, **7**, 231—240 (1964) **И3**

140* Bobrow D G, Sintactic analysis of English by computer — a survey Proc JFCC, **24**, 365—387 (1963) **И.**

141 Bomba J S, Alpha numeric character recognition using local operations, см [Q], pp 218—224 **Г8, Г10, Г12, Г5, 12, 34**

142 Бонгард М М Моделирование процесса узнавания на цифровой счетной машине *Биофизика* **6** 129—141 (1961)

143 Booth A D, How much can machines learn? *Automatic Data Processing*, **2**, № 2, 22—26 (1960) **Н1**

144 Boring E G, Mind and mechanism, *Am J of Psychol*, **59** № 2, 173—192 (1946) **Н, Ж**

145 Boring E G, Dual role of the Zeitgeist in scientific creativity, *Sci Monthly*, **80**, № 2, 101—106 (1955) **Ж4, Ж5, Ж6**

146 Boring E G, When is human behavior predetermined? *Sci Monthly*, **84**, № 4, 189—196 (1957) **Н3**

147 Borko H, Computer Applications in the Social Sciences, Englewood Cliffs, N J, Prentice Hall, 1962

148* Бородянский Ю М, Некоторые замечания по поводу моделирования на конечных автоматах приспособления к внешней среде, *Кибернетика*, № 2 29—34 (1965) **А1, 21**

149 Bouricius W G, Keller J M, Simulation of human problem solving, см [P], pp 116—119 **Л1, Л2, 12, 34, 36**

150 Bowden B V (ed), Faster than Thought, Pitman, New York, 1963 **А, Б1, С8, С5** [chap 25) В гл 1 и приложении имеются исторические сведения о машинах Бэббеджа

151* Bower G H, Trabasso T R, Concept Indentification, Stud Math Psychol, Univ Press, Stanford, Calif, 1964, pp 32—94

152 Brain W R, *Mind Perception and Science*, Blackwell Sci Publ., Oxford, 1951 **Н4**

153 Брайнес С Н, Напалков А В, Шрейдер Ю А, Analysis of the working principles of some self-adjusting systems in engineering and biology, см [R] **Р1, Д8**

154. Braines S. N., Napalkov A. V., Svechinskii V. B., Problems of Neuro-cybernetics, 1959; англ. перевод работы: Брайнес С. Н., Напалков А. В., Свечинский В. Б., Проблемы нейрокибернетики, Изд-во Акад. медиц. наук СССР, 1959. **Р**.
155. Braines S. N., Napalkov A. V., Certain Problems in the Theory of Self-organizing Systems, 1960; англ. перевод работы: Брайнес С. Н., Напалков А. В., Некоторые вопросы теории самоорганизующихся систем, *Вопр. филос.*, № 6, 148 (1959). **Р5**
156*. Брайнес С. Н., Напалков А. В., Свечинский В. Б., Нейрокибернетика, Медгиз, 1962. **Р**.
157. Браверман Э. М., Некоторые проблемы конструирования машин для классификации объектов по заранее не определенному признаку, *Автомат. и телемех.*, **21**, № 10, 1375—1386 (1960).
158*. Браверман Э. М., Опыты по обучению машины распознаванию зрительных образов. *Автомат. и телемех.*, **23**, 349—364 (1962). **Г**; **21**.
159*. Бранловский В. Л., Об одном методе распознавания объектов, описываемых несколькими параметрами, и возможности его применения, *Автомат. и телемех.*, **23**, № 12 (1962). **Г1**; **21**.
160. Bremermann H. J., The Evolution of Intelligence: The Nervous System as a Model of its Environment, Techn. Rep. 1 (July), Contr. Nonr 477 (17), Project NR-043-200, Mathem. Dept., Univ. of Calif., Berkeley, Calif., 1958. **Е4**, **Е1**, **В**, **М4**, **Р**; **21**.
161. Brenner W. C., Schinzinger R, Suarez R. M., Application of High speed electronic computers to generator design problems, AIEE Conf., Paper 56-940, 1956. **15**.
162*. Brick D. B., Owen J., A mathematical approach to pattern recognition and self-organization, см. Ton J. D, Wilcox R. (eds.). Computers and Information Sciences, Spartan Books, Washington, 1964. **Г**, **Р5**; **21**.
163*. Brick D. B., Wiener's nonlinear expansion procedure applied to cybernetic problems, *IEEE Trans. on Systems Sci. and Cybernetics*, SSC-1, № 1, 67—75 (1965). **Р4**; **21**.
164. Brillouin L., Information theory and most efficient codings for communication or memory devices, *J. Appl. Phys.*, **22**, 1808—1811 (1951). **П1**, **И7**.
165. Broadbent D. E., A mechanical model for human attention and immediate memory, *Psychol. Rev.*, **64**, 205 (1954). **П3**, **М3**.
166. Broadbent D. E., Perception and Communication, Pergamon Press, London, 1958. **Г6**, **И5**.
167*. Broadbent D. E., Attention and the perception of speech, *Scientific American*, **206**, 143—151 (1962). **Г7**, **И5**.
168. Broido D., Recent work on reading machines for data processing, *Automation Progr.*, **4**, 183—224 (1958). **Г10**; **31**.
169. Brooks F. P., Hopkins A. L., Neumann P. G., Wright W. V., An experiment in musical composition, *IRE Trans. on Electronic Computers*, **EC-2**, № 3, 175—182 (1957). **С6**, **К1**, **К2**; **12**, **34**
170*. Brown R. M., Jenks R. D., The CSX-1 computer, *IEEE Trans. on Electronic. Comput.*, **13**, № 3, 247—250 (1964). **М4**; **15**.
171*. Брудно А. Л., Программирование в содержательных обозначениях, ДАН СССР, **154**, № 2, 279—282 (1964). **И3**, **И4**.
172. Bruner J. S., Goodnow J. J., Austin G. A., A Study of Thinking, Wiley, New York, 1956. **Ж4**, **Ж6**, **Г11**, **К4**; **16**, **34**, **36**.
173. Bruner J. S. et al., Contemporary Approaches to Cognition, Cambridge, Mass., Harvard, 1957. **22**, **35**, **36**.
174. Bryton B., Balancing of a Continuous Production Line, Northwestern Univ., Evanston, Ill, 1954. [1075].

175. Buell D. N., Chrysler optical processing scanner (COPS), *Proc. Eastern Joint Computer Conf.*, **20**, 352—370 (1961); см. также Buell D. N., Computers, a Key to Total Systems Control, 1961. **Г8**, **Г11**, **Г12**.
176. Burge W. H, Sorting, trees, and measures of order, *Information a. Control*, **1**, 181—197 (1958). **К6**.
177. Burks A. W., Wang H., The logic of automata, *J. Assoc. Computing Machinery*, **4**, № 1, 193—218, № 2, 279—292 (1957). **А1**, **А4**, **А5**; **21**, **31**, **36**.
178. Burks A. W., Logic of Fixed and growing automata, Proc. of an Intern. Symp. on the Theory of Switching, см. Annals of the Computation Lab., vol. 29, Cambridge, Mass., Harvard, 1959, p. 147. **А1**, **А2**, **А5**, **Б2**, **Б6**, **С4**; **21**, **35**, **36**.
179. Burks A. W., Computation, behavior and structure, см. [U], pp. 282—311. **А5**, **А8**, **Б2**, **Б6**, **И6**, **С4**; **36**.
180*. Burks A. W., Goldstein H. H., von Neumann J., Preliminary discussion of the logical design of an electronic computing instrument, von Neumann J., Collected works, Pergamon, Oxford—London—New York—Paris, 1963, pp. 34—79; русский перевод: Беркс А., Голдстейн Г., Нейман Дж., Предварительное рассмотрение логической конструкции электронного вычислительного устройства, Кибернетический сборник, вып. 9, изд-во «Мир», 1964. **А**; **21**.
181*. Бурлака А. А., Использование контуров символов в процессе автоматического распознавания знаков, см. сб. Читающие устройства, АН СССР, 1962, стр. 36—41. **Г11**; **21**.
182. Bush R. R., Mosteller F., Stochastic Models for Learning, Wiley, New York, 1955. **Д3**, **Д9**; **21**, **35**, **36**; русский перевод: Буш Р., Мостеллер Ф., Стохастические модели обучаемости, Физматгиз, 1962.
183. Bush R. R., Estes W. K. (eds.), Studies in Mathematical Learning Theory, Stanford Univ., Stanford, Calif., 1959. **Д9**, **Д3**, **Д2**; **16**, **31**, **35**, **36**.
184. Bush V., As we may think, *Atlantic Monthly*, **176**, 101 (July 1945). **П1**, **С3**.
185. Butler S., Erewhon, chaps. 23—25. The book of the Machines, London, 1865.
186. Butler S, Erewhon and Erewhon Revisited, Modern Library № 136, Random House, New York, 1933.
187. Byard S., Robots which play games, *Penguin Sci. News*, **20**, 82 (1950). **С5**.
188. Byrnes B., Gold B., Kemball C., Some Results of the MAUDE Programm, Lincoln Lab. Group Rep. 34—72, MIT; hand keyed Morse Code machine, 1958. **Г7**, [393, 977].
189. Cadwallader-Cohen et al. (V. A. Vyssotsky). The Chaostron: an important advance in learning machines, *IRE Prof. Group in Information Theory Newsletter*, **19**, April (1961). **Р6**, **Б4**, **В9**, **Д7**; **11**, **12**, **13**, **32**.
190. Calderwood J. H., Porter A., Pattern recognition in the synthesis of complex switching systems, *J. of Electronics and Control*, **4**, 466—480 (1958). **А3**, **15**.
191. Campaigne H., Some experiments in machine learning, см. [P], pp. 173—175. **Ж9**, **В7**, **Д3**; **12**, **34**.
192. Campbell D. T., Adaptive behavior from random response, *Behavioral Sci.*, **1**, 105—110 (1956). **В9**, **Д7**.
193. Campbell D. T., Systematic error on the part of human links in communication systems, *Information a. Control*, **1**, № 4, 334—369 (1958). **Ж4**, **П3**, **Г6**, **И5**; **22**, **31**, **32**, **36**.

194. Campbell D. T., Blind variation and selective survival as a general strategy in knowledge-processes, см. [U], pp. 205—231. **В9, Л2, Ж6; 32, 36.**
195. Canaday R. H., The Description of Overlapping Figures, диссертация, MIT, 1962. **Г5, Г14; 14 (LISP), 12, 34.**
196. Carr J. W., Languages, logic, learning and computers, *Computers a. Automation*, **7**, № 4, 21—26 (1958). **А8, И2, И3, И4, Д; 36.**
197. Carr J. W. et al., A visit to computation centers in the Soviet Union, *Communic. Assoc. Computing Machinery* (1959).
198*. Carroll J. B., The study of language, Harvard Univ. Press, Cambridge, Mass., 1953.
199. Ceccato S., La machine qui pense et qui parle, см. [F] pp. 288—299. **И1.**
200*. Ceccato S., A model of the mind, *Quad. Ricerca scient.*, № 31, 21—79 (1965). **М3.**
201*. Chandessais Ch., Essai sur la formalisation dans les sciences du comportement, *Cybernetica*, **7**, № 3, 149—202 (1964) **21, 22.**
202*. Chapanis A., Men, machines and models, *Am. Psychol.*, **16**, 113—131 (1961).
203*. Chapman B. L. M., A self-organizing classification system, *Cybernetica*, № 3, 152—161 (1959). **Р5.**
204. Cherry C. C., The communication of information, *Am. Scientist*, **40**, 640—664 (1952). **И7, И5.**
205. Cherry C. C., On Human Communication, Wiley, New York, 1957. **И5, И7, 21, 22, 31, 35, 36**; русский перевод: Черри К., О логике связи (синтактика, семантика и прагматика), см. [АМ].
206*. Черныш В. И., Напалков А. В., Математический аппарат биологической кибернетики, Медгиз, 1964, **Р4; 21, 23, 35.**
207. Chomsky A. N., Systems of syntactic analysis, *J. Symbolic Logic*, **18**, № 3, 242—265 (1953). **И1, И2.**
208. Chomsky A. N., Logical syntax and semantics: their linguistic relevance, *Language*, **31**, 36—45 (1955) **И1, И2.**
209. Chomsky A. N., Three Models for the Description of Language, Proc. of a Symp. on Information Theory, *IRE Trans. on Information Theory*, **IT-2**, № 3, 113—124 (1956); русский перевод: Хомский А. Н., Три модели описания языка, Кибернетический сборник, вып. 2, ИЛ, 1961. **И2, И1.**
210. Chomsky A. N., Syntactic Structures, Den Haag, Nederland, Mouton, 1957. **И2, И1.**
211. Chomsky A. N., Miller G. A., Pattern Conception, AFCRC, Techn. Note Rep. AFCRCRTN-57-57, Astia Doc. 110076, Bedford, Mass., 1957. **Д, К4, И2, И6, И1; 34.**
212. Chomsky A. N., Miller G. A., Finite state languages, *Information a. Control*, **1**, № 3, 91—112 (1958). **И2, И1**; русский перевод: Хомский А., Миллер Дж., Языки с конечным числом состояний, Кибернетический сборник, вып. 4, ИЛ, 1962.
213. Chomsky A. N., Review of B. F. Skinner's "Verbal Behavior", *Language*, **35**, 26—58 (1959). **К2, И1, Д2, Д3, Д6, Ж4, Ж5, И5, Л2; 31, 32, 36**, [1022].
214. Chomsky A. N., On certain formal properties of grammars, *Information a. Control*, **2**, 137 (1959); русский перевод: Хомский А., О некоторых формальных свойствах грамматик, Кибернетический сборник, вып. 5, ИЛ, 1962. **В1, А4, И2; 21.**
215. Chomsky A. N., A note on phrase structure grammars, *Information a. Control*, **2**, 393—395 (1959); русский перевод: Хомский А., Заметка о грамматиках непосредственных составляющих, Кибернетический сборник, вып. 5, ИЛ, 1962. **И2.**

216*. Chomsky A. N., Miller G. A., Introduction to the formal analysis of natural languages, см. Luce R., Bush R., Galanter E. (eds.), Handbook of mathematical psychology, vol. 2, Wiley, New York, 1963. **И1; 21.**
217*. Chorafas D. N., Systema and simulation, Acad. Press, New York—London, 1965; *русский перевод:* Хорафас Д. Н., Системы и моделирование, изд-во «Мир», 1967.
218. Chow C. K., Optimum character recognition system using decision functions, *IRE Wescon Convent. Rep.*, Pt. 4, 121—129 (1957); см. также *IRE Trans. on Electronic Computers*, **EC-6**, № 4, 247—254 (1957). **Г13, Г10, Г9, Б3, 21, 36** [332].
219. Chow C. K., Comments on optimum character recognition system, *IRE Trans.*, **PGEC-8**, 230 (June 1957).
220*. Christie R., Lindauer F., Personality structure, *Ann. Rev. Psychol.*, **14**, 201—230 (1963). **Н; 22.**
221. Church A., An unsolvable problem of elementary number theory, *Am. J. Mathem.*, **58**, 345—363 (1936). **А7.**
222. Church A., Introduction to Mathematical Logic, **1**, Princeton Univ. Press, Princeton, N. J., 1956; русский перевод: Чёрч А., Введение в математическую логику, том I, ИЛ, 1960. **Ж8.**
223. Churchman C. W., Ackoff R. L., Purposive behavior and cybernetics, *Social Forces*, **29**, 32—39 (1951). **Ж3, Р2.**
224. Churchman C. W., Ackoff R. L., Methods of Inquiry, Educat. Publ., St. Louis, 1950.
225. Clark W. A., Farley B. G., Generalization of pattern-recognition in a self-organizing system, см. [E], p. 86. **Б4, Г1, Г3, Д3, М1; 12, 24, 36**, [313].
226. Clarkson G. P. E., Simon H. A., Simulation of individual and group behavior, *Am. Economic Rev.*, **50**, 920—932 (1960). **Ж4.**
227. Clarkson G. P. E., Meltzer A. H., Portfolio selection: a heuristic approach, *J. Finance*, **15**, № 4 (1960). **С8; 34.**
228. Clarkson G. P. E., Portfolio Selection: A Simulation of Trust Investment, Prentice-Hall, Englewood Cliffs, N. J., 1962. **С8.**
229*. Climenson W. D., RECOL — a retrieval command language, *Communic. ACM*, № 3 (1963). **И3; 21, 31.**
230. Coburn H. E., The brain analogy, *Psychol, Rev.*, 58, 155—178 (1951). **М3.**
231. Coburn H. E., The brain analogy: A discussion, *Psychol. Rev.*, **59**, 453 (1952). **М3.**
232*. Cofer C. N., The role of language in human problem solving, paper read at Conf. on Human Problem Solving Behavior, New York Univ., April 8, 1954. **И5; 22.**
233. Cohen J., Information theory and music, *Behavioral Sci.*, **7**, № 2 (1962). **И7, К1, Е6, В9; 22, 31, 32, 36.**
234. Cohen K. J., Computer Models of the Shoe, Leather, Hide Sequence, Prentice-Hall, Englewood Cliffs, N. J., 1960. **С8.**
235. Cohen K. J., Cyert R. M., Computer models and dynamic economics, *Quart. J. Economics*, **75**, № 1 (1961). **Р7.**
236. Colby K. M., Experiments of the effects of an observer's presence on the Imago System during psychoanalytic free-association, *Behavioral Sci.*, **5**, 216—232 (1960).
237*. Colby K. M., Computer simulation of a neurotic process, см. Tomkins L. S., Messick S. (eds.), Computer simulation of personality, Wiley, New York, 1963. **Н, Р4; 22.**

238*. Comfort W. T., Multiword list items, *Commun. Assoc. Computing Machinery*, 7, № 6, 357—362 (1964).
239*. Cooper D. C., Whilfield N., ALP. An autovode list-processing language, *Computer J.*, 5, № 1 (1962). **И3.**
240. Copi I. M., Elgot C. C., Wright J. B., Realization of events by logical nets, *J. Assoc. Computing Machinery*, 5, 181—196 (1958); русский перевод: Копи И. М., Элгот К. С., Райт Д. Б., Реализация событий логическими сетями, Кибернетический сборник, вып. 3, ИЛ, 1961. **А2, Б6, И2; 21** [557].
241. Copi I. M., Beard R. W., Programming an Idealized General-purpose Computer to Decide Questions of Truth and Falsehood, Rep. 2144-402-T, Willow Run Labs., Univ. of Michigan, Ypsilanti, 1959. **Ж7, Ж8.**
242. Couffignal L., Les machines semantiques, см. [F], pp. 129—138. **Р.**
243. Coupling J. J. (Pierce J. R.), How to build a thinking machine, Astounding Science Fiction, Aug. 1950. **Д3, Д7, Б5, В, В9, Н2, С7.**
244. Cowan J., Many-valued logics and reliable automata, см. [W]. **А6, Б3, Б6, И7.**
245. Cowan J., Towards a proper logic for parallel computation in the presence of noise, см. [S]. **А6, Б3, Б6, И7, Б2.**
246*. Cowan J. D., The Engineering Approach to the Problem of Biological Integration, см. [N], 1963, pp. 22—30; Discussion, P. 26.
247. Craik K. J. W., Theory of the human operator in control systems, *Brit. J. Psychol.*, 38, 56 (1947). **С3.**
248. Craik K. J. W., The Nature of Explanation, New York, Cambridge, 1952. **Е1, И4, И6, Н; 24, 36.**
249*. Cram D., Explaining "Teaching machines" and Programming, Fearon Publ., San Francisco, 1963; русский перевод: Крэм Д., Программированное обучение и обучающие машины, изд-во «Прогресс», 1965.
250*. Crane D. B., An Information-processing model of concept formation, Carnegie Inst. of Technol., Pittsburgh, 1961.
251. Crichton J. W., Holland J. H., A New Method of Simulating the Central Nervous System Using an Automatic Digital Computer, Rep. 2144-1195-M, Willow Run Labs., Univ. of Michigan, Ypsilanti, 1959. **Б5; 14.**
252. Crozier W. J., Physiology and computation devices, Proc. of the Second Symp. on Large-scale Digital Calculating Machines, Cambridge, Mass., Harvard, 1951. **М4.**
253. Culbertson J. T., A device for optic nerve conduction and form perception, *Bull. Mathem. Biophys.*, 10, 31, 97 (1948). **Г1, М3.**
254. Culbertson J. T., Consciousness and Behavior, Brown, Dubuque, Iowa, 1950. **Н, Д, А, Б6.**
255. Culbertson J. T., Hypothetical Robots and the Problem of Neuroeconomy, Rep. P-296, RAND Corp., Santa Monica, Calif., 1952. **А1, А6, Б6.**
256. Culbertson J. T., Some uneconomical robots, см. [G], pp. 99—116. **А1, А6, Б6.**
257. Culbertson J. T., Robots and automata: A short history, *Computers a. Automation*, 6, 32 (1957). **31, 32.**
258. Cyert R. M., Feigenbaum E. A., March J. G., Models in a behavioral theory of the firm, *Behavioral Sci.*, 4, 81—95 (1959). **Р7, С8.**
259. Darlington C. D., The Evolution of Genetics, Basic Books, New York, 1958. **В5, В6, В7, Д7, Л.**
260. David E. E., Ears for computers, *Scientific American*, 192, 92 (Febr. 1955). **Г7; 13, 34, 36.**

261. David E. E., Artificial auditory recognition in telephony, *IBM J. Res. a. Develpt.*, 2, 294 (Oct. 1958). **Г7.**
262. David E. E., Matthews M. V., McDonald H. S., Description and results of experiments with speech using digital computer simulation, *IRE Wescon Convent. Rec.*, 2, № 7, 3—10 (1958). **Г7; 12, 34.**
263. David E. E., Digital simulation in research on human communication, *Proc. IRE*, 49, 319—328 (Jan. 1961); русский перевод см. [Y]. **Г7, Г8, С8, И3, Г11.**
264. Davies D. W., Mechanization of thought processes, *Nature*, 183, 225—226 (Jan. 1959) (обзор содержания [N]).
265. Davis K. H., Biddulph R., Balashek S., Automatic recognition of spoken digits, см. [K]. **Г7; 13, 34.**
266. Davis M., Computability and Unsolvability, McGraw-Hill, New York, 1958. **А7, А4, В1, И2; 21, 31, 32, 35, 36.**
267. Davis M., Putnam H., A Computational Proof Procedure, AFOSR TR 59-124, Rensselaer Institution, Troy, N. Y., 1959. **В2, В3, А7, Ж7; 12, 21, 32, 33.**
268. Davis M., Putnam H., A computing procedure for qualification, *J. Assoc. Computing Machinery*, 7, № 2 (1960). **В2, В3, А7, Ж7; 12, 21, 32, 33.**
269*. Davis M., Logemann G., Loveland D., A machine program for theoremproving, *Commun. Assoc. Computing Machinery*, 5, № 7, 394—397 (1962). **Ж7; 12, 21, 33.**
270. Davis R. C., The domain of homeostasis, *Psychol. Rev.*, 62, 8—13 (1958). **31, (Р6), 36.**
271*. Dearnley F. H., Newell G. B., Automatic segmentation of programs for a two-level store computer, *Computer J.*, 7, № 3, 185—187 **(1964). А; 21, 31.**
272. Deese J., The Psychology of Learning, McGraw-Hill, New York, 1952. **Д1, Д2; 22, 35, 36, 32, 31.**
273. DeGroot A. D., Het Denken van den Schaker, Amsterdam, Nederland, 1946. **Е6, Ж3, Ж4** (шахматы), **С5, Л2.**
274*. DeGroot A. D., Chess playing programs, *Proc. Koninkl. Nederl. Akad. Wetensch. Amsterdam*, Ser. A, 67 (1964); *Indag. Math.*, 26, 385—398 (1964). **С5, Л2, Е6, Ж3, Ж4; 12, 22.**
275*. DeGroot A. D., Thought and choice in Chess, Den Haag, Mouton, 1965. **Е6, Ж3, Ж4, Л2, С5; 22.**
276. Denes P., Mathews M. V., Spoken digit recognition using time-frequency pattern matching, *J. Acoust. Soc. America*, 32, № 11, 1450—1455 (1960). **Г2, Г3, Г7, Г11, Г13, Г14.**
277. Deutsch J. A., A new type of behavior theory, *Brit. J. Psychol.*, 44, 304 (1953).
278. Deutsch J. A., A machine with insight, *Quart. J. Exper. Psychol.*, 6, 6—11 (1954). **Ж6.**
279. Deutsch J. A., A theory of shape recognition, *Brit. J. Psychol.*, 46, № 1, 30—37 (1955). **Г, Г11, Г12** [1051].
280. Deutsch J. A., A theory of insight, reasoning and latent learning, *Brit. J. Psychol.*, 47, 115 (1956). **Ж6, Д.**
281. Deutsch K. W., Mechanism, teleology and mind, *Philos. a. Phenomenol. Res.*, 12, 185—222 (1951). **Р2, Н.**
282. Diebold J., Automation: The Advent of the Automatic Factory, Van Nostrand, Princeton, N. J., 1952. **Р3.**
283. Dimond T. L., Devices for reading handwritten characters, Proc. Eastern Joint Computer Conf. (EJCC), 1957, pp. 232—237. **Г10, Г4; 13.**
284. Dinneen G. P., Programming pattern recognition, см. [E], pp. 94—100. **Г1, Г4, Г8; 12, 34** [973].

285. Doyle W., Recognition of sloppy hand-printed characters, см. [V], pp. 133—142. **Г13, П2, Г10, Г4, Г9; 12, 34** [977].
286. Duncan C. P., Recent research on human problem solving, *Psychol. Bull.*, **56**, № 6, 397—429 (1959). **Ж4, Д2, П3; 22, 31, 32, 36.**
287[5]. Duncan E. L. R., Robert B., Galanter E., Handbook of mathematical psychology, Wiley, New York—London, vol. 1, 1963, chap. 1—8.
288[1]. Duncker K., A qualitative study of productive thinking, *Ped. Sem.*, 33, 642—708 (1926). **Г4, Г3; 16, 22, 34, 36.**
289*. Duncker K., Krechevsky J., On solution-achievement, *Psychol. Rev.*, **46**, № 2, 176—185 (1939).
290. Duncker K., On problem solving, *Psychol. Monogrs.*, 58, № 270 (1945). **Г4, Г3; 16, 22, 34, 36.**
291. Dunham B., Fridshal R., Sward G. L., A nonheuristic program for proving elementary logical theorems, см. [R], pp. 282—285. **В1, Ж7, Ж8; 12, 21, 34.**
292*. Dusheck G. I., Hilinski T. G., Puttrath F. L., Flexible neural logic network, *IEEE Trans.*, MIL-7, № 2, 3, 208—212 (1963); русский перевод: Дашек Г., Хилинский Т., Патрас Ф., Обучающаяся логическая схема на нейронах, *Зарубежная радиоэлектроника*, № 8 (1964).
293. Eccles J. C., The Neurophysiological Basis of the Mind: The Principles of Neurophysiology, Oxford, New York, 1953. **М, Н4; 23, 36.**
294*. Edelstein L. A., "Picture logic" for "Bacchus", a fourth-generation computer, *Computer J.*, **6**, № 2, 114—153 (1963); русский перевод: Эдельштейн Л., «Картинная логика» для системы «Бахус» четвертого поколения вычислительных машин, *Зарубежная радиоэлектроника*, № 5 (1964). **31.**
295. Eden M., Halle M., Characterization of cursive writing, см. [X]. **Г5, И4; 34.**
296. Eden M., Handwriting and pattern recognition, см. [AA], pp. 160—166. **Г10, Д.**
297[5]. Edwards W., Dynamic Decision Theory and Probabilistic Information Processing, *Human Factors*, **4**, № 2, 59—73 (1962); русский перевод: Эдвардс У., Динамическая теория решений и вероятностная обработка информации, сб. Инженерная психология, изд-во «Прогресс», 1964.
298. Eldredge K., Teaching machines how to read, *Stanford Res. Inst. J.*, **1**, May (1957).
299*. Elgot C. C., Robinson A., Random-access stored program machines. An approach to programming language, *J. Assoc. Computing Machinery*, **1**, № 4, 365—399 (1964). **И3.**
300. Elias P. et al., Fourier treatment of optical processes, *J. Opt. Soc. America*, **42**, 127 (1952). **Г8; 21, 35, 36.**
301. Elias P., Computation in the presence of noise, *IBM J. Res. a Develpt.*, 2, 346 (1958). **И7, Б3; 21.**
302*. Елисеев В. К., Ковалевский В. А., Исследование алгоритма распознавания машинописных знаков, *Журн. вычислит. матем. и матем. физики*, 2, № 5 (1962). **Г10; 21, 34.**
303. Elsasser W. H., Physical Foundation of Biology, Pergamon, New York, 1958. **Р2, Р6.**
304*. Емельянов-Ярославский Л. Б., Схемы и программы самоорганизующегося автомата, *Вопр. радиоэлектроники*, сер. VII, вып. 1, 95—124 (1962). **Р5; 21.**
305*. Емельянов-Ярославский Л. Б., Принципы работы самоорганизующегося автомата, *Вопр. радиоэлектроники*, сер. VII, вып. 3, 89 (1962). **Р5; 21.**

306*. Enslein K., Digital and analog computation considerations for biomedical problems, Data acquisit and process. in biol. and med., vol. 2, Macmillan, New York, 1964, pp. 173—189. **15.**
307. Ernst H. A., MH-1. A Computer-operated Mechanical Hand, диссертация, MIT, 1961; presented at the Western Joint Computer Conf. (WJCC), May 1962. **И3, П3; 13, 12, 34.**
308. Estavan D., Pattern Recognition, Machine Learning and Automated Teaching, Rep. SP-70, System Develpt. Corp., Santa Monica, Calif., 1959. **Г1, Д, С3.**
309. Estes W. K., Toward a statistical theory of learning, *Psychol. Rev.*, 57, 94—107 (1950). **Д9, Д3; 21.**
310. Estes W. K., Statistical models for recognition and recall of stimulus patterns by human observers, см. [U]. **П3, Г3, Г9; 16, 34, 36.**
311*. Estrin G., Fuller R., Some applications for content-addresssable memories, AFIPS Conf. Proc., vol. 24, Spartan Books, Baltimore, Cleaver-Hume Press, London, 1963, pp. 495—508. **А8; 15.**
312. Eykhoff P., Optimizing Control and Process Parameter Estimation, диссертация, Univ. of Calif., Berkeley, 1960. **Р1, Д4, Д7, В8.**
313. Farley B. G., Clark W. A., Simulation of self-organizing system by a digital computer, *IRE Trans. on Information Theory*, **PGIT-4**, 76—84 (1954). **Б6, Б4, Д3, М1; 21, 34** [225].
314. Farley B. G., Frishkopf L. S., Clark W. A., Gilmore J. T., Computer Techniques for the Study of Patterns in the Electroencephalogram, Techn Rep. 1965, Lincoln Lab., MIT, Lexington, Mass. 1957. **Г1; 12, 34, 36.**
315. Farley B. G, Self-organizing models for learned perception, см. [U]. **Г1, Г3, Г6, Г9, Д9; 36.**
316. Farley B. G., Clark W. A., Activity in Networks of neuronlike elements, см. [X]. **Б6; 12, 34.**
317. Fatechand R., Machine recognition of spoken words, *Advances in Computers*, **1**, 193—229 (1960). **Г7.**
318. Feigenbaum E., An Information Processing Theory of Verbal Learning, P-1817, Santa Monica, Calif., RAND Corp., Oct. 1959. **П1, Д5.**
319. Feigenbaum E., The simulation of verbal learning behavior, *Proc. Western Joint Computer Conf.*, **19**, 121—132 (1961). **П1, Д5.**
320. Feigenbaum E., Simon H. A., Forgetting in an associative memory, *Proc. Nat. Conf. Assoc. Computing Machinery*, **17**, 2C2—2C5 (1961). **Г4, Д, Д2, Д10, П1, П3, Д5.**
321. Feigenbaum E., Soviet cybernetics and computer sciences, 1960, *Communic. Assoc. Computing Machinery*, **4**, 566—579 (1961) (см. также [197]). **Д, Ж, И, Л, Р, 31.**
322. Feigenbaum E., Simon H. A., A theory of the serial position effects, *Brit. J. Psychol.*, **53**, 307—320 (1962). **П3.**
323[1]. Feigenbaum E., Simon H. A., Generalization of an elementary perceiving and memorizing machine, Inform. Proc. (1962), N. Holland Publ., Amsterdam, 1963, pp. 401—405; Discussion, pp. 405—406. **36.**
324*. Feigenbaum E., Artificial intelligence research, *IEEE Trans. on Information Theory*, **9**, № 4, 248—253 (1963). **31, 32.**
325. Feigl H., Philosophical embarrassments of psychology, *Am. Psychologist*, **14**, 115—128 (1959). **Н4.**
326. Feldman J, An Analysis of Predictive Behavior in a Two-choice Situation, диссертация, Carnegie Inst. of Technology, Pittsburgh, Pa, 1959. **Д6, Ж4.**
327. Feldman J., Simulation of behavior in the binary choice experiment, см. [Z], pp. 133—144. **Д6, Ж4.**
328. Feldman J., Tonge F., Kanter H., Empirical Explorations of a

Hypothesis-testing Model of Binary Choice Behavior, SP-546, System Develpt. Corp., 1961. **Д6, Ж4.**

329. Feldman J., Computer simulation of cognitive processes, см Borko H. (ed.), Computer Applications in the Behavioral Sciences, Englewood Cliffs, N. J., Prentice-Hall, 1962. **Ж.**

330*. Feurzeig W., Conversational teaching machine, *Data-mation*, **10**, № 6, 38—42 (1964). **С; 31.**

331*. Fink D. G., Artificial intelligence, McGraw-Hill Year-Book on Sci. a. Technol., McGraw-Hill, New York—Chicago—San Francisco—Dallas—Toronto—London, 1963, pp. 63—72. **31.**

332. Flores I., An optimum character-recognition system using decision functions, *IRE Trans. on Electronic Computers*, **EC-7**, 180 (June 1958). **Г10** [218].

333*. Fogel L. I., Levels of intelligence in decision making, *Annals New York Acad. Sci.*, **89**, № 5, 732—751 (1961); русский перевод: Фогель Л., Интеллектуальные уровни решений.

334*. Fogel L. I., Toward inductive inference automata, Inf. Proc. 1962, N. Holland Publ., Amsterdam, 1963, pp. 395—400; Discussion p. 400. **К5, А6, В1; 1.**

334a*. Fogel L. I., Artificial Intelligence Trough Simulated Evolution, Wiley, New York, 1966.

335. Forgie J. W., Forgie C. D. Results obtained from an auditory-recognition computer program, *J. Acoust. Soc. America*, **31**, 1480—1484 (1959). **Г7; 12, 34.**

336*. Formby J., An introduction to the mathematical formulation of self-organizing systems, Spon, London, 1965. **Р5; 21.**

337. Foulkes J. D., A class of machines which determine the statistical structure of a sequence of characters, *Wescon Convent. Rec.*, **3**, № 4, 66—73 (1959). **Л1, Д9.**

338. Frankel S., Information-theoretic aspects of character reading, см. [R], pp. 248—251. **Г3, Г10, Г9, И7.**

339*. Frederiksen N., Gulliksen H. (eds), Comtributions to mathematical psychology, Holt, Renehart a. Winston, Inc., New York—Chicago—San Francisco—Toronto—London, 1964.

340. Fredkin E., Trie memory, см. [T]. **И4, П.**

341. Freiberger H., Murphy E., Reading machines for the blind, *IRE Trans. on Human Factors in Electronics*, **2**, № 1, 8—20 (1961). **Г10, Г6, Г11; 31, 32.**

342. Freimer M., A dynamic programming approach to adaptive control processes, *IRE Trans. on Automatic Control*, **AC-4**, 10—15 (1959). **Р1.**

343. Freimer M., Topics in Dynamic Programming, II, Lincoln Lab. Rep. 52-G-0020, MIT, Lexington, Mass., MIT Hayden Library № H-82, 1960; см. особенно разд. I—E. **Ж2; 21.**

344. Friedberg R. M., A learning machine, Pt. I, *IBM J. Res. a. Develpt.*, **2**, 2—13 (1958). **Д3, Д4, В3, В7, В8, Ж9, Ж5, Л; 12, 34** [36, 549].

345. Friedberg R. M., Dunham B., North J. H., A learning machine, Pt. II, *IBM J. Res. a. Develpt.*, **3**, 282—287 (1959). **Д3, Д4, В3, В5, В7, В8, Ж9, Ж5, Л1; 21, 34, 36.**

346. Frishkopf L. S., Harmon L. D., Machine reading of cursive script, см [X], pp. 300—316. **Г4, Г5, Г10; 12, 13, 34, 33, 36.**

347. Fry D. B., Denes P., Experiments in mechanical speech recognition, см. [K], pp. 206—212. **Г7; 34.**

348. Fry D. B., Denes P., An analogue of the speech recognition process, см. [N], vol. 1, pp. 377—395. **Г7, Г2, Г3, К1; 33.**

349*. Fu K. S., Learning control systems, Computer and Information Science, Spartan Books, Washington, 1964, pp. 318—343. **Д; 31, 32.**

350. Furst B., Stop Forgetting! Garden City, New York, 1949. **П1, П3, Д5; 22, 35.**

351*. Гаазе-Рапопорт М. Г., Автоматы и живые организмы, Физматгиз, 1961. **Р4.**

352. Gabor D., Communication theory and cybernetics, *IRE Trans. on Circuit Theory*, **CT-1**, 19—31 (Dec. 1954). **Р.**

353. Gabor D., Wilby W. P., Woodcock R., A self-optimizing nonlinear filter, predictor and simulator, см. [X]. **В6, В8; 13.**

354. Gagne R. M., Smith E. C., Jr., A study of the effects of verbalization on problem solving, *J. Exper. Psychol.*, **63**, 218—227.

355. Galanter E., Gerstenhaber M., On thought: The extrinsic theory, *Psychol. Rev.*, **63**, 218—227 (1956). **Е, Ж, С1, С2; 22, 36.**

356. Galanter E. (ed.), Automatic Teaching, Wiley, New York, 1959. **С3, Ж5.**

357. Galanter E., Miller G. A., Some comments on stochastic models and psychological theories, см. Arrow, Karlin, Suppes (eds.), Mathematical Methods in the Social Sciences, Stanford Univ. Press, Stanford, 1960. **Д4, Ж4.**

358. Galton F., Inquiries into Human Faculty and Development, Macmillan, New York, 1883. **Е, Ж; 16, 22, 34, 36.**

359. Gamba A., Optimum performance of learning machines, см. [Y], pp. 349. **Г13, Д9.**

360. Gardner M., Logic machines, *Scientific American*, **186**, № 3, 68 (1952). **Ж8; 13, 31, 35.**

361. Gardner M., Logic Machines and Diagrams, McGraw-Hill, New York, 1958. **Ж8; 13, 31, 35.**

362*. Garner W. R., Uncertainty and structure as psychological concepts, Wiley, New York, 1962. **22.**

363. Gelernter H., Rochester N., Intelligent behavior in problem-solving machines, *IBM J. Res. a. Develpt.*, **2**, № 4, 336—345 (1958). **Ж7** (геометрия), **Ж10, Е3, И4, Л1; 14, 33, 36** [729].

364. Gelernter H., A note on syntactic symmetry and the manipulation of formal systems by machine, *Information a. Control*, **2**, 80—89 (1959). **Ж1, Ж3, И4, Ж7** (геометрия); **21, 36.**

365. Gelernter H., Realization of a geometry theorem-proving machine, см. [R], pp. 273—282. **Ж7** (геометрия), **Ж10, Е3, Л1; 15, 14, 34, 36.** [730].

365a*. Gelernter H., The Fortran List Processing Language, IBM dittoed paper, 1959.

366. Gelernter H., Hansen J. R., Loveland D. W., Empirical exploration of the geometry theorem machine, см. [V], pp. 143—147. **Ж7** (геометрия), **Ж10, Е3, Л1; 14, 34, 36.**

367. Gelernter H., Hansen J. R., Gerberich C. L., A FORTRAN-compiled list-processing language, *J. Assoc. Computing Machinery*, **7**, 87—101 (Apr. 1960) **И3** (FLPL), **И4; 14, 15, 34.** [790, 674, 418].

368*. Gelernter H., Machine-generated problem-solving graphs, Proc. Polytechn. Inst. Brooklin Symp. on Mathematical Theory of Automata, April 1962.

369. George F. H., Logical networks and behavior, *Bull. Mathem. Biophys.*, **18**, 337 (1956). **А2, Б6, Е2.**

370. George F. H., Logical networks and probability, *Bull. Mathem. Biophys.*, **19**, 187 (1957). **Д3, Д9, Р6, С1.**

371. George F. H., Automation, Cybernetics and Society, Philos. Library, New York, 1959. **Р1, Р3.**

372. George F. H., Inductive machines and the problem of learning, *Cybernetica*, **2**, 109—126 (1959). **К, Д.**

373*. George F. H., The brain as a computer, Pergamon, Oxford—London—New York—Paris, 1961; русский перевод: Джордж Ф., Мозг как вычислительная машина, ИЛ, 1963. **М4**.
374*. George F. H., Finite automata and the nervous system, см. [N], 1963, pp. 37—53. **А1**.
375*. George F. H., Pragmatic Machines, *Elektron. Rechenanlage*, **6**, Beih. 7, 53—68 (1964). **А**.
376. Gerard R. W., The biological basis of imagination, *Sci. Monthly*, **62**, 477 (June 1946). **Н4**.
377*. Gere W., Heuristics in job shop scheduling, Carnegie Inst. Technol., Pittsburgh, Pa., 1960. **Л**.
378. Ghiselin K. (ed.), The Creative Process, Univ. of Calif. Press, Berkeley, Calif., 1952. **Ж4**, **Ж6**; **22**.
379. Gibson E. J., A systematic application of the concepts of generalization and differentiation to verbal learning, *Psychol. Rev.*, **47**, 196—229 (1940). **Д2**.
380. Gibson J. J., The reproduction of visually perceived forms, *J. Exper. Psychol.*, **12**, 1—39 (1929). **Г6**, **И2**, **П3**; **16**, **22**, **34**.
381. Gibson J. J., The Perception of the Visual World, Houghton Mifflin, Boston, 1950. **Г6**; **35**.
382. Gill A., Minimum-scan pattern recognition, *IRE Trans. on Information Theory*, **IT-5**, № 1, 52—58 (1959). **Г11**; **21**.
383. Gill A., Analysis of nets by numerical methods, *J. Assoc. Computing Machinery*, **7**, № 2, 251—254 (July 1960). **А2**; **21**.
384. Gilmore P. C., A program for the production of proofs for theorems derivable within the first order predicate calculus from axioms, см. [R], pp. 265—273. **Ж7**, **Ж8**, **В2**, **Л1**; **33**, **34**.
385. Gilmore P. C., A proof method for quantification theory: its justification and realization, *IBM J. Res. a. Develpt.*, **4**, № 1, 28—35 (1960). **Ж7**, **Ж8**, **В2**, **Л1**; **21**, **34**, 38 (176).
386. Glantz H. T., On the recognition of information with a digital computer, *J. Assoc. Computing Machinery*, **4**, № 2, 178—188 (Apr. 1960). **Г3**, **Г11**.
387. Glauberman M. H., Character recognition for business machines, *Electronics*, **29**, 132—136 (Febr. 1959). **Г10**.
388*. Глезер В. Д., Цуккерман И. И., Информация и зрение, Изд-во АН СССР, 1961. **И7**; **23**.
389*. Глушков В. М. Теория обучения одного класса дискретных персептронов, *Журн. вычислит. матем. и матем. физики*, **2**, вып. IV, № 2, 317—335 (1962). **Г**, **Д**; **21**.
390*. Глушков В. М. Самоорганизующиеся системы и абстрактная теория автоматов, *Журн. вычислит. матем. и матем. физики*, **2**, вып. V—VI, № 3, 459—466 (1962). **А**, **Р5**; **21**.
391*. Глушков В. М., Кибернетика и умственный труд, М., 1965. **36**.
392. Gödel K., Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I, *Monatsh. für Mathem. u. Phys.*, S. 173—189 (1931). **А7**.
393. Gold B., Machine recognition of hand-sent Morse code, *IRE Trans. on Information Theory*, **IT-5**, 17—24 (March 1959). **Г1**, **С2**, **И1**; **12**, **13**, **34**.
394. Goldstein M. H., A statistical model for interpreting neuroelectric responses, *Information a. Control*, **3**, 1—17 (1960). **М3**.
395*. Goldstine H. H., Computers and perception, *Proc. Am. Philos. Soc.*, **108**, № 4, 282—290 (1964). **Г1**.
396. Golomb S., A mathematical theory of discrete classification см. [X]. **Г11**; **21**, **35**.

397. Good I. J., How much science can you have at your finger-tips? *IBM J. Res. a. Develpt.*, **2**, № 4, 282—288 (Oct. 1958). **П3**, **Л2**, **Б6**, **Д3**, **К4**.
398. Good I. J., Could a machine make probability judgements? *Computers a. Automation*, **8**, № 1, 14—16; № 2, 24—26 (1959). **Н2**, **А6**, **С5**, **Б4**, **М4**.
399. Good I. J., Weight of evidence and false target probabilities, см. [X]. **К1**, **К5**, **К6**.
400. Good I. J., A causal calculus, *Brit. J. Philos. Sci.*, **11**, 305—318; **12**, 43—51 (1961). **Н**.
401. Good I. J., The mind-body problem, or could an android feel pain? Theories of the Mind, Glencoe Free Press, Urbana, 1962. **Н4**, **М2**, **М4**, **М1**.
402. Goodman N., The Structure of Appearance, Cambridge, Mass., Harvard, 1951. **К6**; **24**.
403. Goodman N., Fact, Fiction and Forecast, Cambridge, Mass., Harvard, 1954.
404. Goodman N., The test of simplicity, *Science*, **128**, 1064—1069 (1958). **К6**.
405. Goodwin G. L., Digital computers tap out designs for large motors... fast, *Power*, April (1958). **12**, **15**.
406. Goodwin G. L., Machine recognition of hand sent Morse code, *IRE Trans. on Information Theory* (1959).
407*. Goodwin B. C., Temporal Organization in Cells, Acad. Press, New York, 1964.
408*. Gordon C. K., Jr., In defense of the search for the pretercomputer, *Cybernetica*, **6**, № 2, 57—75 (1963).
409. Gorn S., Standardized programming methods and universal coding, *J. Assoc. Computing Machinery*, **4**, 254 (July 1957). **И3**.
410. Gorn S., On the mechanical simulation of learning and habit-forming, *Information a. Control*, **2**, № 3, 226—259 (1959). **Д3**, **Д9**; **21**, **36**; русский перевод. Горн С., О механическом моделировании образования привычек и обучения, Кибернетический сборник, вып. 8, изд-во «Мир», 1964.
411. Grant D. A., Testing the Null Hypothesis and the strategy and tactics of investigation theoretical models, *Psychol. Rev.*, **69**, 54 (1952).
412. Greanias E. C., Hoppel C. J., Cloomok M., Osborn J. S., Design of logic for recognition of printed characters by simulation, *IBM J. on Res. a. Develpt.*, **1**, № 1, 8—18 (1957). **Г10**; **12**, **34**.
413. Greanias E. C. Hill J. Y., Considerations in the design of character recognition devices, *1957 IRE Nat. Convent. Rec.*, pt. 4, 119—126 (1957). **Г10**.
414*. Greanias E. C., Meagher P. F., Norman R. J., Essinger P., The recognition of handwritten numerals by contour analysis, *IBM J. Res. a. Develpt.*, **7**, № 1, 14—21 (1963); русский перевод: Гриниас Н., Мигер П., Норман Р., Изингер П., Распознавание рукописных цифр с помощью анализа контуров, *Зарубежная радиоэлектроника*, № 2, 46—57 (1964). **Г10**.
415. Green B. F., The use of high-speed digital computers in studies on form recognition, см. [1174]. **Г1**; **31**, **32**.
416. Green B. F., Non-computational uses of digital computers, *Behavioral Sci.*, **4**, 164—167 (1959). **Г6**, **Д2**, **И4**; **31**.
417. Green B. F., Baseball: an automatic question-answerer, см. [Z]. **П1**, **И1**, **П2**, **И2**, **И4**; **12**, **14** (IPL), **34**.
418. Green B. F., Computer languages for symbol manipulation, см. [AF], pp. 3—8; см. также *IRE Trans. on Electronic Computers*, № 4, 729—735 (1961). **И4**, **И3** [790, 800, 367, 674].

419*. Green B. F., Jr., Digital Computers in Research, McGraw-Hill, New York, 1963.
420. Greene P. H., Networks for pattern perception, *Proc. Nat. Electronics Conf.*, **15** (1959). **Г1, Б6.**
421. Greene P. H., Problem-solving and learning machines, *Behavioral Sci.*, **4**, 249—250 (1959). **Д, Ж.**
422. Greene P. H., An approach to computers that perceive, learn and reason, см. [P], pp. 181—186. **Е6, Л2; 22, 24, 32, 36.**
423. Greene P. H., A suggested model for information representation in a computer that perceives, learns and reasons, см. [V], pp. 151—164. **Г5, Г11, Г9, Г3, Е.**
424. Grimsdale R. L., Sumner F. H., Tunis C. J., Kilburn T., A system for the automatic recognition of patterns, *Proc. IEE*, **106B**, № 26, 215 (1959). **Г5, Г10; 12, 34.**
425. Grimsdale R. L., Automatic pattern recognition, *Wireless World*, **65**, 499—501 (1959). 38 (199).
426*. Guilford J. P., Personality, McGraw-Hill, New York, 1959. **22.**
427*. Guilford J. P., Threefaces of intellect, *American Psychologist*, **14**, № 8 (1959).
428*. Guinn D. F., Large artificial nerve net (LANNET), *IEEE Trans. Milit. Electron.*, **7**, № 2, 234—243 (1963). **Б6.**
429. Gullahorn J., Gullahorn J., Homunculus: A Simulation of Social Interaction, Michigan State Univ., East Lansing, Mich., 1962. **P7.**
430*. Gullahorn J., Gullahorn J., Computer simulation of human interaction in small groups, см. 1964 Proc. Spring Joint Computer Conf., Spartan, Baltimore, 1964, pp. 103—113. **P7.**
431*. Gullahorn J. T., Gullahorn J. E., A Computer experiment in elementary social behavior, *IEEE Trans. on Systems sci. a. Cybernetics*, **SSC-1**, № 1, 45—52 (1965). **P7.**
432. Гуляев П. И., Значение динамического программирования и теории игр для физиологии, сб. Нервная система, вып. 3, ЛГУ, 1962. **36.**
433. Гутенмахер Л. И., Проблема информационных машин, *Известия*, 1 апреля 1959.
434. Gyr J. W., An investigation into, and speculations about, the formal nature of a problem-solving process, *Behavioral Sci.*, **5**, № 1, 39—59 (1960). **Д, Е, Ж, К, Л.**
435. Hadamard J., The Psychology of Invention in the Mathematical Field, Princeton Univ. Press, Princeton, N. J., 1945. **Ж4, Л2.**
436. Hagelbarger D. W., SEER, a sequence extrapolation robot, *IRE Trans. on Electronic Computers*, **EC-5**, № 1, 1—7 (1955); русский перевод: Хагельбаргер Д. У., СПИР — автомат, экстраполирующий последовательности, Кибернетический сборник, вып. 1, ИЛ, 1960. **С5, К1; 13**, [000, 000].
437. Hake H. W., Hyman R., Perception of the statistical structure of a random series of binary symbols, *J. Exper. Psychol.*, **45**, 64 (1953). **К1, Г6.**
438. Haldane J. B. S., The mechanical chess player, *Brit. J. Philos. Sci.*, **3**, 189 (1952). **С5.**
439. Halle M., Stevens K., Speech recognition: A model and a program for research, см. [AA], pp. 155—159.
440. Haller N., Line Tracing for Character Recognition, диссертация, MIT, Cambridge, Mass., 1959. **Г5, Г10.**
441*. Харкевич А. А., О выборе признаков при машинном опознавании, *Изв. АН СССР*, о.т.н., техн. кибернетика, № 2, 3—10 (1963). **Г4, Г11; 21.**
442. Harmon L. D., Artificial neuron, *Science*, **129**, 962—963 (1959). **Б6, Б5, М3, Г7; 13, 34, 23.**

443. Harmon L. D., A line-drawing recognizer, см. [V], pp. 351—364. **Г11, Г12; 13.**
444. Harmon L. D., Line-drawing pattern recignizer, *Electronics*, 39—43, (Sept. 1960).
445. Harmon L. D., Neural analogs, *Proc. IRE*, **49**, № 8, 1316—1317 (1961).
446. Harmon L. D, Levinson J., van Bergeijk W. A., Analog models of neural mechanisms, см. [AA], pp. 107—112. **Б5, Б6, Г6, Г7, М3**, библ.
447. Harmon L. D., Studies with artificial neurons, I: Properties and functions of an artificial neuron, *Kybernetik* (1962). **Б6.**
448*. Harran D., Communication: A Logical Model, Wiley, New York, 1963.
449. Harris Z. S., Methods in Structural Linguistics, Univ. of Chicago Press, Chicago, 1951. **И1, И2, Г5.**
450. Harris Z. S., Project summary, см. Current Research and Development under time pressure, Carnegie Inst. of Technol., Pittsburgh, 1964. 1960, pp. 52—53, **И1.**
451. Hartline H. K., The response of single optic nerve fibers of the vertebrate eye to illumination of the retina, *Am. J. Physiol.*, **121**, 400—415 (1938). **М3.**
452. Hartmanis J., Task simplification and learning devices, см. [X]. **Е4.**
453. Hartree D. R, Calculating Instruments and Machines, Univ. of Illinois Press, Urbana, Ill., 1949.
454. Hawkins J. K., Self-organizing systems — a review and commentary, см. [Y], pp. 31—48. **Б6, Б4, Г13, Г9; 31, 32, 35.**
455*. Hawkins R., An experimental and simulation study of concept formation under time pressure, Carnegie Inst. of Technol., Pittsburgh, 1964.
456. Hayek F. A., The Sensory Order, Univ. of Chicago Press, Chicago, 1952. **Н4, Н1, К5, Г6, Д5; 24, 22, 36.**
457. Heasly C. C., Jr., Some communications aspects of character-sensing systems, см. [P], pp. 176—180. **Г10, Г11, И7.**
458. Hebb D. O., The Organization of Behavior, Wiley, New York, 1949. **Б4, Б5, Г6, Д5, Г5, Ж6; 11, 22, 23, 36, 31** [719, 924, 14].
459. Hebb D. O., Textbook of Psychology, Saunders, Philadelphia, 1958. **Г1, Г5, Д1, Д6, Ж4, М1, М2, П3.**
460. Hebb D. O., Distinctive features of learning in the higher animal, см. [AC_2]. **Д1, Д2.**
461. Heidbreder E., An experimental study of thinking, *Archives psychol.*, **11**, № 73, 5—175 (1924). **Ж4.**
462*. Held M., Karp R. M., A dynamic programming approach to sequencing problems, *J. Soc. Ind. a. Appl. Math.*, **10**, № 1, 196—210 (1962); русский перевод: Хелд М., Карп Р. М., Применение динамического программирования к задачам упорядочения, Кибернетический сборник, вып. 7, ИЛ, 1963.
463. Helgeson W. B, Kwo T. T., Letter to the Editor, *Management Sci.*, **3** (1956).
463a*. Hensley C. B., Newell A., Tonge F. M., 650 IPL Information Processing Language, C.I.P. Working Paper № 9, Carnegie Inst. Technol, April 30, 1958.
464*. Hesse M., Analogy and confirmation theory, *Dialectica*, **17**, № 2, 3, 284—295 (1963).
465*. Hess E., Attitude and pupil size, *Scientific American*, **212**, 46—54 (1965). **22.**
466. Heymann H, Jr., The USSR in the technological race, RAND Corp. Paper P-1754, Senta Monica, 1959.

467*. Hiffl J. K., Jodelt J. G., A dynamic allocation scheme, *Computer J.*, **5**, № 3, 200—209 (1962).
468. Highleyman W. K., Kamentsky L. A., Pattern recognition (perception) machine, *Behavioral Sci.*, **4**, 248 (1959). **Г10**; **13**, **34**.
469. Highleyman W. K., Kamentsky L. A., A generalized scanner for pattern and character recognition studies, см. [P], pp. 291—294. **Г10**; **13**, **15**, **34**.
470. Highleyman W. K., Kamentsky L. A., Comments on a character method of Bledsoe and Browning, *IRE Trans. on Electronic Computers*, **BC-9**, 163 (June 1960). **Г10**, **Г3**; **12**, **34**.
471. Highleyman W. K., Further comments on the N-tuple pattern recognition method, *IRE Trans. on Electronic Computers*. **EC-10**, № 1, 97 (1961). **Г3**, **Г16**. [129, 470].
472. Highleyman W. K., Linear Decision Functions with Application to Pattern Recognition, Brooklyn Polytechnic Inst., 1961. **Г9**, **C2**, **B1**, **B8**; **12**, **15**, **21**, **34**.
473. Highleyman W. K., The design and analysis of pattern recognition experiments, *Bell Syst. Techn. J.*, **41**, № 2, 723—744 (1962). **Г3**, **Г10** (обсуждение работ [129, 141, 285, 346, 657, 971 и др.]).
474. Hilgard E. R., Theories of Learning, Appleton-Century-Crofts, New York, 1956. **Д1**, **Д2**; **31**, **32**, **35**.
475. Hiller L. A., Jr., Isaacson L. M., Experimental Music, McGraw-Hill, New York, 1959. **C6**, **K1**, **И4**; **12**, **34**, **36**.
476. Hiller L. A., Jr., Computer music, *Scientific American*, **201**, 109—120 (Dec 1959). **C6**.
477*. Hoenich P. L., Robot-art the hopeful monster, *Cybernetica*, **7**, № 1, 27—67 (1964). **H**.
478. Hohn F. E., Seshu S., Aufenkamp D. D., The theory of nets, *IRE Trans. on Electronic Computers*, **EC-3**, №3, 154—161 (1957). **A1**; **21**, **32**.
479. Holland J. H., Cycles and automation behavior, notes from course, Advanced Theory of the Logical Design of Digital Computers, Summer Session, Univ. of Michigan, Ann Arbor, 1958. **A2**, **Б6**.
480. Holland J. H., Survey of Automata Theory, Rep. 2900-52-R, Willow Run Labs., Univ. of Michigan, Ypsilanti, 1959. **A7**, **A1**, **Б6**, **A5**, **Л1**; **21**, **32**, **35**, **36**.
481. Holland J. H., Iterative circuit computers, см. [V], pp. 259—266. **Б2**, **A5**, **И3**; **21**, **36**.
482. Holland J. H., Outline for a logical theory of adaptive systems, *J. Assoc. Computing. Machinery*, № 3, 297—314 (1962). **Б2**, **B**, **C4**, **A5**, **B5**, **B6**, **Д**.
483*. Holt A. W., Program organization and record keeping for dynamic storage allocation, Information Processing, 1962, N. Holland Publ., Amsterdam, 1963, pp. 539—544; discussion 544. **A**; **21**.
484. Homans G. C., Social Behavior: Its Elementary Forms, Harcourt, Brace a. World, New York, 1961 **P1**.
485. Householder A. S., A theory of the induced size effect, *Bull. Mathem. Biophys.*, **5**, 155 (1943). **Г6**, **M**.
486. Householder A. S., Landahl H. D., Mathematical Biophysics of the Central Nervous System, Principal Press, Bloomington, Ind., 1945. **Б6**, **Г**, **Д**, **A6**; **21**, **31**, **32**, **36** [897].
487. Hovland C. I., A "Communication analysis" of concept learning, *Psychol. Rev.*, **59**, 461—472 (Nov. 1952). **Ж6**, **Ж4**, **И5**; **22**.
488. Hovland C. I., Hunt E. B., The computer simulation of concept attainment, *Behavioral Sci.*, **5**, 265—267 (1960). **E4**.
489. Hovland C. I., Hunt E. B., Programming a model of human concept formation, см. [Z], pp. 145—155. **E4**.

490*. Howard R. A., Dynamic programming and Markov processes, Cambridge, Mass., 1960; русский перевод: Ховард Р., Динамическое программирование и марковские процессы, изд-во «Советское радио», 1964. **B1**, **B6**, **Ж3**; **21**.
491. Howland B., Minsky M. L., Selfridge O. G., Hill-climbing: Some Remarks on Multiple Simultaneous Optimization, Group Rep. 54-15, Lincoln Lab., MIT, Lexington, Mass., 1959. **B8**, **B7**, **B3**; **12**, **13**, **21**, **34**, **36**.
492. Hu M. K., Visual pattern recognition by moment invariants, см. [AA], pp. 179—187. **Г12**.
493. Hubel D. H., Wiesel T. N., Receptive fields of single neurons in the cat's striate cortex, *J. Physiol.*, **148**, 574—591 (1959). **M3**, **Г11**, **Г12**, **Г14**; **23**, **34**, **36** [598].
494. Hubel D. H., Wiesel T. N., Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, *J. Physiol.*, **160**, 106—154 (1962). **M3**, **Г11**, **Г12**, **Г14**; **23**, **34**, **36**.
495. Hugh-Jones E. M., Automation today, см. Automation in Theory and Practice, Oxford, 1956. **P3**.
496. Hughes G. W., Halle M., On the recognition of speech by machine, см. [R], pp. 252—256. **Г5**, **Г7**, **Г11**, **Г3**; **12**, **34**, **36**.
497. Hull C. L., The influence of caffeine and other factors on certain phenomena of rote learning, *J. General Psychol.*, **13**, 249—273 (1935). **Д1**.
498. Hull C. L., Hovland C. I., Ross R. T., Hall M., Perkins D. T., Fitch F. B., Mathematico-Deductive Theory of Rote Learning, Yale Univ. Press, New Haven, Conn., 1940.
499. Hull C. L., Principles of Behavior, Appleton-Century-Crofts, New York, 1943. **Д**.
500. Hull C. L., A Behavioral System, Yale Univ. Press, New Haven, Conn., 1952. **Д**.
501. Humphrey G., Thinking, Wiley, New York, 1951. **Л2**, **Ж4**; **22**, **36**.
502. Hunt E. B., An Experimental analysis and Computer Simulation of the Role of Memory in Concept Learning, диссертация, Yale Univ., 1960, **K4**.
503. Hunt E. B., Hovland C. I., Orders of consideration of different types of concepts, *J. Exper. Psychol.*, **59**, 220—225 (1960). **K4**.
504. Hunt E. B., Concept Formation: An Information Processing Problem, Wiley, New York, 1962.
505* Hunt E. B., Simulation and analytic models of memory, *J. verb. Learn. a. verb. Behav.*, **2**, 49—59 (1963). **22**.
506*. Hunt E. B., Concept Learning, an Information Processing Problem, Wiley, New York, 1965. **Д**; **22**.
507*. Hunt E. B., Marin J. K., Stone P., Experiments in induction, Acad. Press, New York, 1965. **K5**.
508*. Hunt J. McV., Intelligence and Experience, Ronald Press, New York, 1961. **22**.
509*. Irons E. T., "Structural Connections" in formal language, *Communic. Assoc. Computing Machinery*, **7**, 67—71 (1964). **И2**.
510. Jackson J. R., A computing procedure for a line balancing problem, *Management Sci.*, **2**, № 3, 261—271 (1956). **C8**.
511. Jackson R. F., "A Bridge-playing Program", presented at the Nat. Conf. of the Assoc. Computing Machinery, Pap. 89, 1960; abstract: *Communic. Assoc. Computing Machinery*, July, 398 (1960). **C5**.
512. Jackson W. (ed.), Proc. of a Symp. on Information Theory, Ministry of Supply, London, 1950 ([K], [I] и [X], по-видимому, служат продолжением этой конференции, хотя их названия несколько различаются). **И7**.

513*. Jacobson H., On models of reproduction, *Am. Scientist*, **46**, № 3, 255—284 (1958); русский перевод: Джекобсон Г., О моделях воспроизведения, Кибернетический сборник, вып. 7, ИЛ, 1963. **С4**.
514. Jacobson H., The informational content of mechanisms and circuits, *Information a. Control*, **2** (B), 285—296 (Sept. 1959). **И7**.
515. Jakowatz C. V., Shuey R. L., White G. M., Adaptive waveform recognition, см. [X]. **Р1, Д4, Г3, К1; 12, 13, 34**.
516*. James P., Stagle J., A Heuristic Program that Solves Symbol Integration Problems in Freshman Calculus, *J. Assoc. Computing Machinery*, **10**, № 4, 507—520 (1963).
517*. Jefferis R. P., Fegley K. A., Application of Dynamic Programming to Routing Problems, *IEEE Trans. on Systems Sci. a. Cybernetics*, **SSC-1**, № 1, 21—26 (1965). **В1; 21**.
518. Jefferson G., The mind of mechanical man, Lister Oration for 1949, *Brit. Medical J.*, **1**, 1105—1121 (1949).
519*. Jenkins D. P., List programming Introduct. Syst. Programm., Acad. Press, London—New York, 1964, pp. 283—293. **И3**.
520*. Joelur K. D., The art of simulation, London, 1964. **22**.
521. John E. R., Miller J. G., The acquisition and application of information in the problem-solving process, *Behavioral Sci.*, **2**, № 4, 291—300 (Oct. 1957). **Ж4**; 22, [434].
522. Johnson D. M., The Psychology of Thought and Judgement, Harper a. Row, New York, 1955. **Ж4, Л2**; 22.
523. Joseph R. D., Contributions to Perception Theory, Cornell Aeronaut. Lab. Rep. VG-1196-G-7, June 15, 1960. **С13, Д9**.
524. Julesz B., Binocular depth perception and pattern recognition, см. [X], pp. 212—224. **Г6; 12, 34**, 16.
525. Julesz B., Binocular depth perception of computer-generated patterns, *Bell Syst. Techn. J.*, **39**, 1125—1161 (1960). **Г6**.
526. Julesz B., Visual pattern discrimination, см. [AA], pp. 84—91. **Г6, Г11; 12, 16, 21, 22, 34**.
527. Julesz B., Miller J. E., Automatic stereoscopic presentation of functions of two variables, *Bell. Syst. Techn. J.*, **40**, № 2, 663—676 (1962).
528*. Jury E. I., Pavlidus T. A., A literature survey of biocontrol systems, *IEEE Trans.*, **AC-8**, № 3, 210—217 (1963); русский перевод: Джури И., Павлидус Т., Обзор литературы по биологическим управляемым системам, *Зарубежная радиоэлектроника*, № 6 (1964). **23, 31**.
529. Kac M., A note on learning signal detection, см. [AA], pp. **126—128**; русский перевод: Кэк М., Самообучающаяся схема обнаружения, *Зарубежная радиоэлектроника*, № 10, 35—40 (1962). **Д4, Д7**.
530. Kahrimanian H. G., Analytical Differentiation by a Digital Computer, диссертация, Temple Univ., Philadelphia, Pa., 1953. **И4; 12, 34**.
531. Kailath T., Optimum receivers for randomly varying channels, см. [X]. **В8, Д4; 21**.
532. Kalin T. A., Some Metric Considerations in Pattern Recognition, Res. Lab. Electronics, MIT, Cambridge, Mass., 1960. **Г1, Г12, Г8, М3; 21, 36**.
533. Kamentsky L. A., Pattern and character recognition systems — picture processing by nets of neuron-like elements, см. [P], pp. 304—309. **Б6; Г19; 12, 34**.
534*. Kamentsky L. A., Liu C. N., Computer — Automated Design of Multifont, Print Recognition Logic, *J. Res. a. Develpt.*, **7**, № 1, 2—13 (1963); русский перевод: Каментский Л., Лю К., Построение с помощью ЦВМ логики для распознавания различных печатных шрифтов, *Зарубежная радиоэлектроника*, № 9, 74—92 (1963). **Г10; 21**.

535. Kassler M., The decision of Arnold Schoenberg's twelve-note system and related systems, Library of Congress Catalog Card № 61-18855, 1961. **С6; 21, 11, 32**.
536. Katona G., Organizing and Memorizing, Columbia, New York, 1940 **Ж4, Ж5, П3; 16, 34, 36**.
537. Kattsoff L. O., Brains, thinking and machines, *Methods*, **6**, № 24, 279 (1954). **Н**.
538*. Kaufman H., Becker G. M., The empirical determination of game—theoretical strategies, *J. Exper. Psychol.*, **61**, № 6, 462—468 (1961). **С5, 11, 22, 33**.
539. Kazmierczak H., The potential field as an aid to character recognition, см. [R], pp. 244—247. **Г11, Г19; 13, 34**.
540. Kehl W. B., Horty J. F., Bacon C. R. T., Mitchell D. S., An information retrieval language for legal studies, *Communic. Assoc. Computing Machinery*, **4**, № 9, 380—389 (1961). **П1**.
541. Keller H. B., Finite automata, pattern recognition and perceptions, *J. Assoc. Computing Machinery*, **8**, № 1, 1—20 (1961). **А1, Б4, Б5, Г13, Д3** [930].
542. Kelly J. L., Selfridge O. G., Sophistication in computers: A disagreement, см. [AA], pp. 78—80.
543. Kemeny J. G., The use of simplicity in induction, *Philos. Rev.*, **62**, 391—408 (1953). **К6**.
544. Kemeny J. G., Man viewed as a machine, *Scientific American*, **192**, 58—67 (1955). **Н, Р, С4; 36**.
545. Kemeny J. G., Fair bets and inductive probabilities, *J. Symbolic Logic*, **20**, 263—273 (1955). **К1**.
546. Kemeny J. G., Two measures of complexity, *J. Philos.*, **52**, 722—733 (Nov. 1955). **К6**.
547*. Kemeny J. G., Snell J. L., Thompson G. L., Introduction to finite mathematics, Englewood Cliffs, N. Y., Prentice-Hall, New York, 1957; русский перевод: Кемени Дж., Снелл Дж., Томпсон Дж., Введение в конечную математику, ИЛ, 1963. **35**.
548*. Kessler A. R., Pool I. de Sola, *Crisiscom*, a Computer Simulation Processing During a Crisis, *IEEE Trans. on Systems Sci. a. Cybernetics*, **SSC-1**, № 1, 52—57 (1965).
549. Kilburn T., Grimsdale R. L., Summer F. H., Experiments in machine learning and thinking, см. [R]; русский перевод: Килберн Т., Гримсдейл Р., Саммер Ф., Эксперименты с обучающейся и мыслящей машиной, Кибернетический сборник, вып. 4, ИЛ, 1962. **Ж9, В9, В8, Д3; 24**.
550*. Kilner D. E., The characteristics of computers of the second decade (a review), *Computer Bull.*, pp. 88—112 (Dec. 1960); русский перевод: Килнер Д. Е., Характеристики вычислительных машин второго десятилетия. Обзор, Кибернетический сборник, вып. 5, ИЛ, 1962.
551. Kirsch R. A., Experiments with a Computer Learning Routine, Bureau of Standards Computer Seminar Notes, July 1954; NBS publication Computer Develpt. (SEAC and DYSEAC) at the NBS, 1955. **Д3, С5; 12, 34** [436].
552. Kirsch R. A., Cahn L., Ray L. C., Urban G. H., Experiments with processing pictorial information with a digital computer, Proc Eastern Joint Computer Conf., 1957, pp. 221—229. **Г1, Г8, Г11; 12, 13, 34, 36**.
553. Kister J., Stein P., Ulam S., Walden W., Wells M., Experiments in chess, *J. Assoc. Computing Machinery*, **4**, № 2, 174—177 (1957). **С5, Ж2, Л1; 12, 34** [X, 1043].

554 Klaus G, Relationship of causality and teleology from the cybernetics viewpoint, *German (East) Journ of Philos*, 8, № 10 (1960) Joint Publ Res Service Rep 8374, 1961, U S Dept of Commerce Washington **P2**
555 Kleene S C, General recursive functions of natural numbers *Am J Mathem*, **57**, 153—157, 219—244 (1935) **A7**
556 Kleene S C, Introduction to Metamathematics, Van Nostrand Princeton, N J, 1952, русский перевод Клини С, Введение в метаматематику, ИЛ, 1957 **A4, A7, И2 Ж7, 21, 32, 35**
557 Kleene S C, Representation of events in nerve nets and finite automata, см [G], pp 3—41 **A2, A1, Б6, И2 21 36** [676, 240]
558 Klein S, Simmons R F, A computational approach to grammatical coding of English words *J Assoc Computing Machinery*, **10** 334—347 (1963)
559* Knuth D E, On the Translation of Languages from Left to Right, *Information a Control*, 8, 607—639 (1965)
560* Кобринский Н Е, Трахтенброт Б А Введение в теорию конечных автоматов Физматгиз 1962 **А А1, 21, 35**
561 Kochen M, Levy M J, Jr, The logical nature of an action scheme, *Behavioral Sci*, **1**, № 4, 265—289 (1956) **Ж4, Е, Д, Е4 П4, 22**
562 Kochen M, Group behavior of robots, *Computers a Automation*, **6**, № 3, 16—21 48 (1957)
563 Kochen M, The acquisition and utilization of information in problem solving and thinking, *Information a Control*, **1**, № 3, 267—288 (1958) **Д2, К4, Л2 С2 Р3, Ж5, 11, 22, 34, 16, 36, 21**
564 Kochen M, Organized systems with discrete information transfer, *General Systems Yearbook*, 2, 30—47 (1958)
565 Kochen M, Extension of Moore Shannon model for relay circuits, *IBM J Res a Develpt*, 3, № 2, 169—186 (1959) **Б3**
566 Kochen M, Experimental study of hypothesis formation" by computer, см [X] **К4, Л2, Л1, В4, В6, Г4, Г11 21 32**
567 Kochen M, An experimental program for the selection of "disjunctive hypotheses", см [Z] pp 571—578
568 Koffka K, Principles of Gestalt Psychology, Harcourt, Brace a World, New York, 1935 **Г1 Г6 Д6 Е2 Ж4, И5, 22, 35**
569 Kohler W, Gestalt Psychology, Liveright, New York, 1929 **Г1, Г6, Д6, Е2, Ж4, И5, 22, 35 36**
570 Kohler W, Relational determination in perception см [C], pp 200—243 **Г6, М3**
571 Kolman E, Cybernetics, Joints Publ Res Service Rep 5002, U S Dept of Commerce, Washington, 1960 **P**
572 Kondo M, Murata H, On proof retrieval problem solving machines I *Proc Japan Acad*, **41**, № 4, 254—259 (1965) **Ж**
573 Koppel H, Digital computer plays NIM, *Electronics*, **25**, 155 (Nov 1952) **С5** (игра «Ним»), **12**
574* Ковалевский В А Корреляционный метод распознавания изображений, *Журн вычислит матем и матем физики*, 2, № 4 (1962) **Г, 21**
575* Ковалевский В А (ред), сб Читающие автоматы и распознавание образов, изд во «Наукова думка», Киев, 1965 **Г, Г10, 21**
576 Krieger F J, Future Science and Technology of the USSR RAND Corp Paper P 1647, Santa Monica, Calif, 1959
577 Kubie L S, Theoretical application to some neurological problems of properties of excitation waves which move in closed circuits, *Brain*, **53**, 166 (July 1930) **П3 М3, Р4, 22, 21, 36**

578 Kubie L S, Repetitive core of neurosis, *Psychoanal Quart*, **10**, 77 (Jan 1941) **М3, П3, Р4, 22, 21, 36**
579* Kuck D J, Krufee G K, A problem solver with formal discriptive inputs, *Computer a Inform Sci*, Spartan Books, Washington, 1964, pp 344—374 **Ж, 21.**
580 Kudielka V, Lucas P, Lernprogramme am Maliufteri', см [AE], pp 125—143 **Б6 К1 С** [1164, 1046, 1105, 1106]
581 Ladefoged P, The perception of speech, см [N], pp 397—415 **Г5, Г7, Г14, 11, 22, 36, 32**
582 Laing R, Book review, см Taube M, Computers and Common Sense, *Behavioral Sci*, 7, № 2, 238—240 (1962) **Н**
583 Landahl H D, McCulloch W S, Pitts W, A statistical consequence of the logical calculus of nervous nets, *Bull Mathem Biophys*, 5, 135 (1943) **Б6, Б3, А6**
584 Landahl H D, Mathematical theory of the central nervous system, см Rashevsky (ed), Mathematical Theories of Biological Phenomena, *Ann New York Acad Sci*, **96** (art 4), 1056—1070 (1962) **Б6, Д1, Д3, Д8, Д9, М1 М3, Г8**
585 Lashley K S, The problem of cerebral organization in behavior (Cattell), *Biol Symposia*, 7 302 (1942) **М3**
586 Lashley K S, The problem of serial order in behavior, см [C], pp 112—146 **М3 Д5, И5 23 36**
587 Laslett P (ed), The physical Basis of Mind, Macmillan New York, 1950 **Н4**
588 de Latil P Introduction a la Cybernetique, le Pensee Artificielle, Gallimard, Paris 1953 **P, 35**
589 de Latil P, Thinking by Machine, Houghton Mufflin, Boston, Mass, 1956 **P1, P3, P6, 13, 21, 31, 35** [47, 1129]
590* Laughery K R, Gregg L W, Simulation of human problem solving behavior, *Psychometrika*, **27**, № 3, 265—282 (1962) **Е6 Ж 22**
591* Lave L B, An experimental approach to economic conflicts, диссертация, Carnegie Inst Technol, Pittsburch, 1964 **P7, 22**
592 Lawrence D H, Coles G R, Accuracy of Recognition with alternatives before and after the stimulus, *J Exper Psychol*, **47**, 208—214 (1954) **Г6, П3**
593 Lawrence D H, LaBerge D L, The Relationship between recognition accuracy and order of reporting stimulus dimensions, *J Exper Psychol*, **51**, 12—18 (1956) **Г6, П3**
594 de Leeuw K, Moore E F, Shannon C E, Shapiro N, Computability by probabilistic machine, см [G], pp 183—212 **А6, А7, А4, 21**
595 Leimer J J, Design factors in the development of an optical character recognition machine, [AA] pp 167—170 **Г10**
596 Lenneberg E H, An Empirical Investigation into the Relation Between Language and Cognition, Harvard Univ, Cambridge, Mass, 1956 **И5**
597 Lettvin J Y Pitts W, A mathematical theory of the affective psychoses, *Bull Mathem Biophys*, **5**, 139 (1943) **Р4**
598 Lettvin J Y, Maturana H, McCulloch W S, Pitts W, What the Frog's Eye Tells the Frog's Brain, *Proc IRE*, **47**, 1940—1951 (1959), русский перевод Леттвин Дж и др, Что сообщает глаз лягушки мозгу лягушки, см [AJ_2], 1964 **М3, Г11, Г12 Г14, 23, 34, 36** [494]
599 Levien R E, Studies in the Theory of Computational Algorithms I Formalization Computability, Representation and Analysis Problems, RAND Corp Rep RM 3007, Applied Mathematics, Harvard Univ, Cambridge Mass, 1962 **И2**

600. Levinson J., Harmon L. D., Studies with artificial neurons, III. Mechanisms of flicker-fusion, *Kybernetik* (1962). **Б6**.
601. Lewine M. H., Retrieval of ordered hits from a content addressed memory, *RCA Rev.*, **23**, № 2, 215—219 (1962).
602*. Lewis B. N., Adaptive teaching machines, *Programmed Learning*, № 3, 4 (1963). **P1**.
603*. Lewis B. N., Automated group instruction, см. Goldsmith M. (ed.), Mechanization in the Classroom. Souvenier Press, London, 1963.
604*. Lewis B. N., Communication in problem-solving groups, см. Jones J. C., Thornley D. G. (eds.), Conf. on Design Methods, Pergamon, London, 1963.
605*. Lewis B. N., The adaptive control of small groups, см. Dodd B. (ed.), Programming' 63, Dept. of Education, Sheffield Univ., 1963. **P1**.
606*. Lewis B. N., The rationale of adaptive teaching machines, см. Goldsmith M. (ed.), Mechanization in the Classroom, Souvenier Press, London, 1963. **P1**; **15**, **21**.
607. Lewis P. M., II, The characteristic selection problem in recognition systems, см. [AA], 1962, pp. 171—179. **Г4**.
608. Licklider J. C. R., Man-computer symbiosis, *IRE Trans. on Human Factors in Electronics*, **1**, № 1, 4—11 (1960); русский перевод: Ликлайдер Дж., Симбиоз человека и машины, *Зарубежная радиоэлектроника*, № 9, 84—86 (1960). **С3**, **Ж4**; **36**.
609*. Lindgren N., Machine recognition of human language. Part I — Automatic speech recognition, *IEEE Spectrum*, **2**, № 3, 114—136 (1965); русский перевод: Линдгрен Н., Машинное распознавание человеческого языка, *Зарубежная радиоэлектроника*, № 3, 55—87 (1966). **Г7**; **21**.
610. Lindsay R. K., The Reading Machine Problem, CIP Working Paper 33, Graduate School of Industrial Administration, Carnegie Inst. Technol. Pittsburgh, Pa., 1960. **И1**, **И6**.
611. Lindsay R. K., Toward the Development of Machines Which Comprehend, Carnegie Inst. of Technology, Pittsburgh, Pa, 1961. **И1**, **И6**, **П2**.
612. Lindsay R. K., Information processing theory, см. Fields W. S., Abbot W. (eds.), Symp. on Information Storage and Neural Control, 10th Annual Meeting of the Houston Neurological Soc., 1962. **И1**, **И6**, **П2**.
613. Lindsay R. K., A program for sentences and making inferences about kinship relations, Hoggatt A. (ed.), Proc. of Western Management Science Conf. on Simulation, 1962. **И1**, **И6**, **П2** [417].
614*. Lindsay R. K., A heuristic parsing procedur for a language learning program. Information Processing Reps. 12 and 13, Univ. of Texas, Austin, Texas, 1964. **Д10**, **Л1**; **21**.
615. Linsky L. (ed.), Semantics and the Philosophy of Language, Univ. of Illinois Press, Urbana, Ill., 1952. **И5**, **Ж4**.
616. Lionnais F. L., L'imitation de la Pensée Créatrice par les Machines, Univ. de Paris, Paris, 1957. **Н2**.
617. Löfgren L., Automata of high capacity and methods of increasing their reliability by redundancy, *Information a. Control*, 1, 127 (1958); см. также [F], p. 493. **Б3**, **А4**, **С4**.
618. Löfgren L., Kinematic and Tesselation Models of Self-repair, Techn. Rep. 8, Contr. Nonr 1834(21), Electr. Eng. Res. Lab., Eng. Exper. Station, Univ. of Illinois, Urbana, Ill., 1961. **С4**.
619. Löfgren L., Self-repair as the limit for automatic error correction, см. [W], pp. 181—228. **Б3**, **С4**.
620*. Loehlin J. C., A computer program that simulates personality, см. Tomkins S. S., Messick S. (eds.), Computer simulation of personality, Wiley, New York, 1963. **Ж**, **Л**; **12**.

621*. Ломов Б. Ф. (ред.), Проблемы инженерной психологии, вып. 4, Ленинград, 1966.
622. Lovelace A. A., Translator's notes to an article on Babbage's Analytical Engine, *Sci. Memoirs*, 3, 691—731 (1842). См. также приложение к [150].
623. Loveland D. W., Heuristic Approximation, диссертация, MIT, Cambridge, Mass., 1958. **В8**, **Д3**; **12**, **34**.
624. Luce R. D., Raiffa H., Games and Decisions, Wiley, New York, 1957; русский перевод: Льюс Р. Д., Райфа Х., Игры и решения, ИЛ, 1961. **С1**, **С2**, **Е6**, **С5**, **К1**, **К5**; **35**, **32**, **21**.
625. Luchins A. S., Mechanization in problem-solving, *Psychol. Monogr.*, **54**, № 6, 248 (1942). **Ж4**, **Ж5**, **Ж6**; **22**, **34**.
626. Luhn H. P., A statistical approach to mechanized encoding and searching of literary information, *IBM J. Res. a. Develpt.*, **1**, № 4, 309—317 (1957). **П2**; **12**, **34**.
627. Luhn H. P., The automatic creation of literature abstracts, *IBM J. Res. a. Develpt.*, **2**, № 2, 159 (1958). **П1**; **12**, **34**.
628. Luhn H. P., Potentialities of Auto-encoding of Scientific Literature, IBM Research Center, Yorktown, N. Y., 1959. **П1**.
629. Lyapunov A. A., Sobolev S. L., Cybernetics and natural science, см. OTS-61-11565 (1958); англ. перевод работы: Ляпунов А., Соболев С., Кибернетика и естествознание, *Вопр. филос.*, № 5 (1958).
630. Ляпунов А. А. (ред.), Проблемы кибернетики, № 3, 1960. **Р**.
631*. MacCorquodale K., Meehl P. E., Edward Chase Tolman, см. Estes W. K. et al., Modern Learning Theory, Appleton-Century-Crofts, New York, 1954.
632. MacGowan R. A., Letter to the Editor, *Science*, July 22 (1960).
633. MacKay D. M., On the combination of digital and analogical techniques in the design of analytical engines, mimeogr., 1949; см. также дополнение к [N], pp. 53—65. **В9**, **А6**, **И6**, **М4**, **Н2**, **Е3**; **36**.
634. MacKay D. M., Quantal aspects of scientific information, *Philos. Mag.*, **41**, 289 (1950). **И7**, **Г5**.
635. MacKay D. M., Mind-like behaviour in artifacts, *Brit. J. Philos. Sci.*, 2, 105—121 (1951). **Н**, **М4**.
636. MacKay D. M., Mentality in machines, *Proc. Aristotelian Soc.*, **26**, suppl. pp. 61—86 (1952); включает также работы [1040, 1167]. **Н**, **М4**.
637. MacKay D. M., Generators of information, см. [K], pp. 475—485. **И7**.
638. MacKay D. M., Operational aspects of some fundamental concepts of human communication, *Syntheses* (Nederland), **9**, 182—194 (1954). **И5**.
639. MacKay D. M., Comparing the brain with machines, *Am. Scientist*, **42**, 261—268 (Apr. 1954). **М4**, **Н**.
640. MacKay D. M., Towards an information-flow model of human behavior, *Brit. J. Philos.*, **47**, № 1, 30—43 (Febr. 1956). **К4**, **Е1**, **Е2**.
641. MacKay D. M., The epistemological problem for automata, см. [C], pp. 235—251. **Е2**, **Г1**, **Г11**, **К1**, **И4**, **В4**, **Б6**, **К1**; **13**, **36**.
642. MacKay D. M., The place of "meaning" in the theory of information, см. [I], pp. 215—225. **И5**, **И7**, [644, 638, X].
643. MacKay D. M., Brain and will, *The Listener*, May 9, 16 (1957); *Faith and Thought*, **90**, 103—115 (1957).
644. MacKay D. M., Operational aspects of intellect, см. [N], pp. 39—54. **К1**, **И7**, **В6**, **Е3**.
645. MacKay D. M., On the logical indeterminacy of a free choice, *Mind*, **63**, 31—40 (1960). **Е2**, **Н3**, **Н4**.
646. MacKay D. M., Theoretical models of space perception, см. [AG]. **Г6**, **М4**, **Р2**, [640].

647. MacKay D. M., The informational analysis of questions and commands, см. [X]. **И5, И6, И7.**
648. MacKay D. M., Information and learning, см. [AE], pp. 40—49. **В3, Д, Д6, Е1, Е5, И7, К4,** [640, 641, 644].
649. MacKay D. M., The visual effects of non-redundant stimulation, *Nature*, **192**, 739—740 (1961). **Г6.**
650*. MacKay D. M. Theoretical models of space perception, см. Aspects Theory Artificial Intelligence, Plenum Press, New York, 1962, pp. 83—103, 239—245. **Г6; 21.**
651. MacKay D. M., The use of behavioural language to refer to mechanical processes, *Brit. J. Philos. Sci.*, (1962). **М4, Н2, Н3, Н4.**
652*. Maier N. R. F., Reasoning in humans. I: On direction., *J. Comp. Psychol.*, **10**, 115—143 (1930).
653*. Maier N. R. F., An aspect of human reasoning, *Brit. J. Psychol.*, **24**, № 2, 144—155 (1933).
654. Malin D., CONTRANS: Conceptual though random net simulation, Proc. Eastern Joint Computer Conf., 1961, pp. 124—234; см. также Computers: Key to Total Systems Control, AFIPS, vol. 20, MacMillan, New York, 1961. **Ж8, Б6, Л; 33.**
655*. Maltzman, Motivation and direction in thinking, *Psychol. Bull.*, **59**, № 6, 457—467 (1962).
656. Mandelbrot B, An informal theory of the structure of language based on theory of statistical matching of messages and coding, см. [K]. **И1, И6, И7.**
657. Marill T., Green D. M., Statistical recognition functions and the design of pattern recognizers, *IRE Trans. on Electronic Computers*, **EC-9**, № 4, 472—477 (Dec. 1960). **Г9, Г10, Г13, С2, Г1; 12, 21, 34.**
658. Marill T, Progress in artificial intelligence, вводная статья в [AF], p. 2. **36.**
659. Marill T, Automatic recognition of speech, см. [AF], pp. 34—38. **Г1, Г7, Г11; 31, 32, 35.**
660. Marimont R. B., A new method of checking the consistency of precedence matrices, *J. Assoc. Computing Machinery*, **6**, № 2, 164—171 (1959). **Г11.**
661. Marshall W. H., Talbot S. A., Recent evidence for neural mechanisms in vision, см. Kluver H. (ed.), Visual Mechanisms, Tempe, Ariz. 1942; Cattell. **Г6. М3; 23.**
662. Martens H. H., Two notes on machine "learning", *Information a. Control*, **2**, № 4, 364—379 (1959); русский перевод: Мартенс Г. Г., Две заметки об «обучении» машин, Кибернетический сборник, вып. 3. ИЛ, 1961. **В, Д10, К3, Л1, С5.**
663*. Martin R. M., Intension and Decision, Prentice Hall, N. J., 1963. **22.**
664*. Mathis S. J., Sass M. A., Wilcox R. H., Heuristic programs — fact, fad, or futility, Proc. 7th Nat. Convent. Military Electron., Washington, 1963, pp. 177—181. **Л, Л1.**
665. Marzocco F. N., Bartram P. R., Statistical learning models for behavior of an artificial organism, см. Biological Prototypes and synthetic Systems, vol. I, Plenum Press, New York, 1962.
666. Mathews M. V., An acoustic compiler for music and psychological stimuli, *Bell Syst. Techn. J.*, **40**, 677—694 (May 1961). **И3, С6; 14.**
667. Mattson R. L., A self-organizing logical system, см. [Q], pp. 212—217. **Г13, Г3, Г11, Г10, Д3, Д4; 12, 34.**
668*. Матвеев П. Н., Алехин Л. А., Моделирование самообучающихся гомеостатических систем на ЦВМ, Сб. трудов Ленингр. механ. инст., № 41, 27—34 (1964). **12; 21.**

669. Mays W., Mindlike behavior in artefacts and the concept of mind, *Brit. J. Philos. a. Sci.*, **3**, 191 (1952). **Н4.**
670. Mays W., Cybernetic models and thought processes, см. [F], p. 103. **Р.**
671. Mccallum D., Smith J. B., Machanized reasoning, *Electronic Eng.*, **23**, 126—133, 458 (1951). **П2, Ж8.**
672. McCarthy J., The inversion of functions defined by Turing machines, см. [G], pp. 177—181. **Л2, В1, В3, К6, А7; 11, 21, 36.**
673. McCarthy J., Programs with common sense, см. [N], pp. 75—84. **Л1, П2, Ж3, Ж9, И4, Ж8; 33.**
674. McCarthy J., Recursive functions of symbolic expressions, см. [T], pp. 184—195. **И3** (LISP), **А7, А8, И4; 15, 36.**
674a*. McCarthy J., Recursive functions of symbolic expressions and their computation by machine (The LISP Programming System), Quart. Progr. Rep. № 53, Res. Lab. Electronics, MIT, Cambridge, Mass., April 15, 1959.
675. McCarthy J., A basis for a mathematical theory of computation, см. [Z]. **А8, Ж7; 38.**
676. McCulloch W. S., Pitts W, A logical calculus of the ideas immanent in nervous activity, *Bull. Mathem. Biophys.*, **5**, 115—137 (1943). **А2, А1, Б6, Н1; 21.**
677. McCulloch W. S., A heterarchy of values determined by the topology of nervous nets, *Bull. Mathem. Biophys.*, **7**, 89—93 (1945). **Б6, Н; 21.**
678. McCulloch W. S., The brain as a computing machine, *Electrical Eng.*, **68**, № 6, 492 (1949). **М4; 35.**
679. McCulloch W. S., Pfeiffer J., Digital computers called brains, *Sci. Monthly*, **69**, 368—376 (1949). **М4, Н; 35.**
680. McCulloch W. S., Machines that know and want, см. Halstead (ed.), Brain and Behavior, Comparative Psychology Monographs, vol. 20 (№ 1), Univ. of Calif. Press, Berkeley, Calif., 1950.
681. McCulloch W. S., Dans l'Antre du Metaphysicien, *Thales* (Paris), 7, 37—49 (1951).
682. McCulloch W. S., Why the mind is in the head, см. [C], pp. 42—74. **М4, Б2, К6, Г12, Н; 36.**
683. McCulloch W S., Finality and Form, Thomas, Springfield, Ill., 1952. **И1.**
684. McCulloch W. S., The Past of a Delusion, Chicago Literary Club, Chicago, 1953 **Р4.**
685. McCulloch W. S., Through the den of the metaphysician, *Brit. J. Philos. Sci.*, **5**, 18—31 (1954). **Н1, Н3, П3, Н2; 36.**
686. McCulloch W. S., Mysterium iniquitatis of sinful men aspiring into the place of God, *Sci. Monthly*, **80**, 35—39 (1955). **Н, М4.**
687. McCulloch W. S., Symp.: The design of machines to simulate the behavior of the human brain, см. [J].
688. McCulloch W. S., Towards some circuitry of ethical robots, *Acta Biotheoretica*, **11**, 147 (1955). **С5, Н2, Л1, Ж3.**
689. McCulloch W. S., Biological computers, *IRE Trans. on Electronic Computers*, **EC-6**, № 3, 190—192 (1957). **М4, Н.**
690. McCulloch W. S., The stability of biological systems, см. Homeostatic Mechanisms, Brookhaven Symposia in Biology, vol. 10, 1957, pp. 207—215. **Б3, Б6, А6.**
691. McCulloch W. S., Agathe Tyche — of nervius nets — the lucky reckoners, см. [N], pp. 613—625. **Б6, Б3, А6; 21, 37.**
692. McCulloch W. S., Where Is Fancy Bred? Bicentennial Conf. on Exper. Psychiatry, Univ. Pittsburgh, 1959, p. 36.

693. McCulloch W. S., The reliability of biological systems, см. [U], pp. 264—281. **Б3, А6, М3; 21, 37.**
694. McLachlan D., Jr., Description mechanics, *Information a. Control*, **1**, № 3, 240—266 (1958). **Г4, Г11, Д4, Г3; 11.**
695. McNaughton R., The theory of automata, a survey, *Advances in Computers*, **2**, 379—421 (1961). **А1, А2, А3, А4, А5, А6, А7, А8.**
696. McPhee W. N., Note on a campaign simulator, *Public Opinion Quart.*, *Summer*, *25*, *184—193 (1961)*.
697*. McWhinney W. H., A study of self-organization in the communications network experiments, диссерт., Carnegie Inst. Technol., 1963.
698. Mehl L., Automation in the legal world, см. [N], vol. 2, p. 755. **О3.**
699*. Meier R. L., Blakelock E., Hinomoto, Simulation of ecological relationships, *Behavioral Sci.*, **9**, № 1, 67—76 (1964). **Р; 33.**
700. Merriman J., Wass D. W. G., Gill S., To what extent can administration be mechanized? см. [N], vol. 2, p. 809. **О3.**
701*. Mesarovic M. D., Foundations for a general systems theory, см. Views Gen. Syst. Theory, Wiley, New York — London — Sydney, 1964. **Р.**
702. Meszar J., Switching systems as mechanized brains, Bell Teleph. Labs Record, Febr. 1953. **Н3** (думающие машины).
703. Michie D., Trial and error, Penguin Sci. Library, № 2, 1961, pp. 129—145. **С5.**
704*. Michotte A., The Perception of Causality, Basic Books, New York, 1963. **Г; 22.**
705*. Микулич Л., Некоторые вопросы машинной эвристики, *Зарубежная радиоэлектроника*, № 10, 11 (1964). **В, Г, Е, Е6, Ж3, Ж6, Ж7, Ж8, И3, И4, Л, С5; 31, 32.**
706. Miller G. A., Language and Communication, McGraw-Hill, New York, 1951. **И5, И1, И2; 22, 21, 35.**
707. Miller G. A., Human memory and the storage of information, *IRE Trans. on Information Theory*, **IT-2**, № 3, 129—137 (1956).
708. Miller G. A., Information and memory, *Sci. American*, **195**, № 2, 42—46 (1956). **П3, И7; 36.**
709. Miller G. A., The magical number seven, *Psychol. Rev.*, **63**, № 2, 81—97 (1956); русский перевод: Миллер Дж. А., Магическое число семь, плюс или минус два, см. сб. Инженерная психология, изд-во «Прогресс», 1964. **Г6, П3, И7, Ж4; 34, 36.**
710. Miller G. A., A note on the remarkable memory of man, *IRE Trans. on Electronic Computers*, **EC-6**, 194—195 (Sept. 1957). **П3.**
711. Miller G. A., Speech and communication, *J. Acoust. Soc. America*, **30**, № 5, 397—398 (1958). **Г7, 36.**
712. Miller G. A., Decision units in the perception of speech, см. [AA], pp. 81—83 (1962). **Г5, Г14.**
713. Miller G. A., Chomsky A. N., Pattern Conception, Rep. AFCRC-TN-57-57, ASTIA Doc. AD-110076, Cambridge Res. Center, 1957. **К2, К4, Г, Ж4.**
714*. Miller G. A., Chomsky A. N., Finitary Models of Language Users, см. Mathematical Methods in Psychology, vol. 2, McGraw-Hill, New York, 1964. **И, К; 21.**
715. Miller G. A., Galanter E., Pribram K., Plans and the Structure of Behavior, Holt, New York, 1960; русский перевод: Миллер Дж., Галантер Е., Прибрам К., Планы и структура поведения, изд-во «Прогресс», 1965. **Е6, Л2, Л1, Г14, Ж3, Д2, П3, П1, Ж6, М4; 36, 31, 32.**
716. Miller G. A., Minker J., Reed W. G., Shindle W. E., A multilevel file structure for information processing, *Proc. Western Joint Computer Conf.*, **17**, 53—59 (1960). **И4.**

717. Miller G. A., Selfridge J. A., Verbal context and the recall of meaningful material, *Am. J. Psychol.*, **63**, 176 (1956). **П3, И5.**
718. Milligan M., Machines are smarter than I am? *Data Processing Digest*, Oct. (1959). **М4.**
719. Milner P. M., The cell-assembly: Mark II, *Psychol. Rev.*, **64**, 242 (1957). **М2, М3, Б4, Б5, Б6, Д5, Д3; 23, 22** [458].
720. Milner P. M., Learning in neural systems, см. [U]). **М2, М3, Б4, Б6, Д5, Е2; 11, 23, 22, 34.**
721. Milner P. M., A neural mechanism for the immediate recall of sequences, *Kybernetik*, **1**, 76—81 (1961). **Б4, Б6, М2, Р3.**
722. Milner P. M., The application of physiology to learning theory, см. Current Trends in Psychological Theory (Symp.), Univ. of Pittsburgh Press, Pittsburgh, 1961. **М2, М3.**
723*. Milsum J. H., The nature of living control systems, *Eng. J.*, **47**, № 5, 34—38 (1964). **Р1; 31.**
724. Minot O. N., Automatic Devices for Recognition of Visible Two-dimensional Patterns: A Survey of the Field, Rep. TM-364, June 25, U. S. Navy Electronics Lab., San Diego, Calif., 1959. **Г11, Г1, Г10, Г8, Г4; 31, 32, 36, 13, 15.**
725. Minot O. N., Counting and outlining of "Two-dimensional" Patterns by Digital Computer, Techn. Memo. TM-414, Aug. 4, U. S. Navy Electronics Lab., San Diego, Calif., 1960. **Г12.**
726. Minsky M. L., Neural Nets and the Brain Model Problem, диссертация, Princeton Univ., 1954; см. Univ. Microfilms, Ann Arbor, Mich., 1954. **А2, Б6, Д3, Б4** (SNARC), **Б5, А6, Е2, Е1, Ж3, М1, М2, М3; (13, 34), 33, 36.**
727. Minsky M. L., Discrete Selection Processes, Rep. 1954-494-03-21, Navy Contr. Nonr-494(03), Tufts College, Medford, Mass., 1954. **В, В6, В7, В8, С7.**
728. Minsky M. L., Some universal elements for finite automata, см. [G]. **А2, Б6.**
729. Minsky M. L., Heuristic Aspects of the Artificial Intelligence Problem, Group Rep. 34-55, ASTIA Doc. AD 236885, 1956 (MIT Hayden Library № H-58), Lincoln Labs, MIT, Lexington, Mass. **Ж1, Ж10, Ж7, Ж3, Е1, Е2, Е3, Е4, Е5, Л1, Г4, Г11, К1, К2, Б5, Б6, В8, Д1, Д3.**
730. Minsky M. L., Notes on the Geometry Problem, I and II, Artificial Intelligence Project, Dartmouth College, Hanover, Vt., Aug. 1956 (mimeogr.). **Ж1, Ж7, Л1, Е3.**
731. Minsky M. L., Learning systems and artificial intelligence, см. Applications of Logic to Advanced Digital Computer Programming, Univ. of Michigan Press, Ann Arbor, Mich., 1957.
732. Minsky M. L., Some methods of heuristic programming and artificial intelligence, см. [N], pp. 3—36. **Н2, Ж1, Г1, Г5, Ж10, Ж5, Е3, Е4, Е5, Ж7, Г11, К6, К2, Б4; 37.**
733. Minsky M. L., Physicial Machines and Their Abstract Counterparts, Group Rep. 54-4, March, Lincoln Labs, MIT, Lexington, Mass., 1959. **А1, А2, Б6; 35.**
734. Minsky M. L., Steps toward artificial intelligence, см. [Y], pp. 8—30. **В, Г1, Д, Е, Ж, К2, Л1, Н; 36, 24, 32, 31.**
735. Minsky M. L., Descriptive languages and problem solving, см. [Z], pp. 215—218. **Д6, Е, Е3, Е5, Ж8, И5, Л1, П1, П2.**
736. Minsky M. L., A selected descriptor-indexed bibliography to the literature on artificial intelligence, см. [AF].
737. Minsky M. L., Recursive unsolvability of Post's problem of "Tag", *Annals of Mathem.* (1961). **А4, А7,** [876, 889, 1132].
738. Minsky M. L., Problems of formulation in the artificial intelligence

area, Proc Sump on Mathem Problems in Biology, Am Math Soc, 1962 **А6, Б6, В1, Г14, Д9, Ж1, Ж7, К1, К2, К5, К6, Л1**

739* Minsky M L, Artificial intelligence, *Discovery*, 23, № 10, 33—37 (1962) **В, Г, Д, Е, Ж, И3, И5, К, Л1, Н**

740 Minsky M L, Selfridge O G, Learning in random nets, см. [X], ASTIA Doc AD 238220, 1960 **Г13, Г11, В3, В6, Д3, Д4, М2, Г9, В7 В8, Б4**

741 Mitchell J, A Computational Procedure for Balancing Zoned Assembly Lines, Res Rep 6 94801 1 R3, Westinghouse Res Labs, Pittsburgh, Pa, 1957 **С8**

742 Moiseyev K, Man and the 'Thinking" Machine, Joint Publ Res Service Rep 2200 N, U S Dept Commerce, Washington, 1960 **Н2, Р.**

743* Moles A, Theorie de l'information et perception esthetique, Flammarion, Paris, 1958, русский перевод Моль А, Теория информации и эстетическое восприятие, изд во «Мир», 1966 **Г, И7, 32**

744* Moles A, Heuristische Prozess und Information theorie, *Electron Rechenanlage*, 6, Beih 7, 40—62 (1964) **Л, И7**

745* Moodie C L, Young H H, A heuristic method of assembly line balancing for assumptions of constant or variable work element times, *J Ind Engng*, **16**, № 1, 23—29 (1965) **Е4, Л1, Л2, Ж1, И4, В4, В5, В6, 14, 34, 36**

746 Mooers C N, Ciphering chemical formulas — the Zatopleg system, Zator Techn Bull, 59, Zator Co, Cambridge, Mass, 1951, **Г5** [903]

747 Mooers C N, ZATOCODING applied to mechanical organization of knowledge, *Am Documentation*, **2**, № 1, 20 (1951) **П1**

748 Mooers C N, ZATOCODING and developments in information retrieval, *Aslib Proc.*, 8, № 1, 3—22 (1956) **П1, И5, И6**

749 Mooers C N, Information retrieval in structured content, см [I], pp 212—234 **П1, П2, И5, И6**

750 Mooers C N, Some mathematical fundamentals of the use of symbols in information retrieval, см [R] **П2, П1, И5**

751 Moore E F, Gedanken experiments on sequential machines, см [G], pp 129—156 **А1**

752 Moore E F, Artificial living plants, *Scientific American*, **195**, 118—122 (Oct 1956) **С4**

753 Moore E F, On the shortest path through a maze, Proc Int Symp on Switching Theory, см Annals of Computation Lab, vols 29, 30, Cambridge, Mass, Harvard, 1959 **С7**

754 Moore O K, Anderson S B, Modern logic and tasks for experiments on problem solving behavior, *J Psychol*, **38**, 151—160 (1954) **Ж4, Ж8**

755 Moore O K, Anderson S B, Search behavior in individual and group problem solving, *Am Sociol Rev*, **19**, № 6, 702—714 (1954) **Ж4**

756 Moore O K, Shannon C E, Reliable circuits using less reliable relays, *J Franklin Inst*, **262**, 191—208, 281—297 (1956), русский перевод Мур Э Ф, Шеннон К Э, Надежные схемы из ненадежных реле, Кибернетический сборник, вып 1, ИЛ, 1960 **Б3, А3, А6, 21**

757* Morimoto Hiroshi, On an attempt at reappearance of human thinking by an automatic calculator, *Math Japan*, **8**, № 1—2, 59—79 (1963)

758 Mowrer O H, The psychologist looks at language, *Am Psychologist*, **9**, № 10, 660—694 (1954) **И5**

759* Mullery A P, A procedur oriented machine language, *IEEE Trans Electronic Computers*, **13**, № 4, 449—455 (1964) **И2**

760 Mullin A A, Some mathematical aspects of the analysis and synthesis of biological computers, см [O], pp 1—18 **А1, А2, А3**

761 Murray A E, A review of the perceptron program, *Proc Nat Electronics Conf*, **15** (1959) **Г13, Д9**

762 Murray A E, Perception applications in photo interpretation, *Photogrammetric Eng*, 627—637, Sept (1961) **Г13, Д9**

763 Murray F J, Mechanisms and robots, *J Assoc Computing Machinery*, **2**, 61—82 (1955), русский перевод Меррей Ф Дж, Механизмы и автоматы, Кибернетический сборник, вып 1, ИЛ, 1960 **А1**

764* Myers J, Atkinson R, Choice behavior and reward structure, *J Mathem Psychol*, **1**, № 1, 170—203 (1964) **Д, 33**

765* Myhill J, The abstract theory of selfreproduction, Viens Gen Syst. Theory, Wiley, New York—London—Sydney, 1964, pp 106—118 **С4, 21, 31, 36**

766* Напалков А В, Некоторые принципы работы головного мозга, сб *Проблемы кибернетики*, № 4, Физматгиз, 1960, стр 183 **М3**

767* Напалков А В, Изучение принципов переработки информации головным мозгом, сб Кибернетику на службу коммунизму, Госэнергоиздат, 1961, стр 153—173 **М, 23**

768* Напалков А В, Переработка информации головным мозгом, см Биологические аспекты кибернетики, Изд во АН СССР, 1962, стр 134—144 **М3, 23**

769* Napalkov A V, Information processes of the brain, см [AN], 1963, pp 59—70 **Ж1, М3, 23**

770* Напалков А В, Кибернетика и пути изучения мозга, см [AQ], 1964, стр 182—190 **Ж1, М3, Л2, 31**

771* Напалков А В, Бобнева М И, Анализ информационных процессов мозга человека, *Вопр психол*, № 6, 40—54 (1962) **М, 33**

772* Напалков А В, Орфеев Ю В, Актуальные задачи эвристического программирования, *Вопр филос*, № 6 (1965) **Ж1, И3, Л1, Л2**

773*. Напалков А В, Родман Н Н, Дашевский Б А, Эвристические программы как основа работы самоорганизующихся и самообучающихся систем, сб Проблемы нейрокибернетики, изд Ростовского университета, 1966 **Д, Ж, И3, Л, М, П1, П; 36**

774* Напалков А В, Воронин Л Г, Systematic in the working of the brain and some problems of cybernetics, 3rd Int Congr on Cybernetics, Sept 11—15, 1961, Namur, Belgium Assoc de Cybernétique, 1965

775* Напалков А В Чичварина Н А, Мозг и кибернетика, изд во «Знание», 1965 **В, Г1, Д1, Д2, Д, Е, Ж, Ж3, И, И4, Л1, Л2, М; 31, 36**

776 Nash J F, Parallel Control, Monograph RM 1361, ASTIA Doc AD 86392, RAND Corp, Santa Monica, Calif, 1954 **Б2**

777 Neisser U, A Preliminary Study of Human Pattern-Recognition, Group Rep 34-75, Lincoln Labs, MIT, Lexington, Mass, 1959 **Г14, Г6**

778 Neisser U, Hierarchies in Pattern Recognition, Rep 54 G 0009, Lincoln Labs, MIT, Lexington, Mass, 1959 **Г6, Г14, 16, 34**

779 Neisser U, A theory of Cognitive Processes, Group Rep 54 19, Lincoln Labs, MIT, Lexington, Mass, 1960 **Г14, Г6, К4, 16, 34, 36**

780* Neisser U, The imitation of man by machine, *Science*, **139**, 193—197 (1963)

781* Nelson E E, Wolf H F, A sampled data analog of neuron properties, *IEEE Trans Electron Comput*, **13**, № 6, 744—745 (1964)

782 von Neumann J, The general and logical theory of automata, см [C], pp 1—31, а также [H], p 2070, русский перевод приложение к книге Тьюринг А, Может ли машина мыслить?, Физматгиз, 1960. **А1, А4, А5, А6, Б3, С4, М4**

783 von Neumann J, Probabilistic logics and the synthesis of reliable

organisms from unreliable components, см. [G], pp. 43—98. **Б3, Б6, А1, Д8; 21, 36.**

784. von Neumann J., The Computer and the Brain, Yale Univ. Press, New Haven, 1958; русский перевод: Нейман Дж., Вычислительная машина и мозг, Кибернетический сборник, вып. 1, ИЛ, 1960. **М4, Б3, С4, И6.**

785 von Neumann J., Morgenstern O., Theory of Games and Economic Behavior, Princeton, N. J., 1947. **С1, Е6.**

786. Newell A., The chess machine, см. [E], pp. 101—108. **Л1, Л5, С5** (шахматы), **Ж2, Ж3, И5, И6, Е, Д4, Д6; 33.**

787. Newell A., Simon H. A., The logic theory machine, *IRE Trans. on Information Theory*, **IT-2**, № 3, 61—79 (1956). **Л1, Л2, В2, Ж7, Ж8, К2, И4, И2; 14** [IPL], **34, 36.**

788. Newell A., Simon H A., Problem-solving in Humans and Computers, RAND Corp. Paper P-987, Santa Monica, Calif., 1956.

789. Newell A, Shaw J. C., Simon H. A., Empirical explorations of the logic theory machine, Proc. Western Joint Computer Conf. 1957, pp. 218—239. **Ж1, Ж4, Н2, Ж8, Ж7, В2, Л1, Л2, П1, В3; 12, 14, 34, 36.**

790. Newell A., Shaw J. C., Programming the logic theory machine, Proc. Western Joint Computer Conf. 1957, pp. 230—240. **И2, И3, И4; 12, 14, 15.**

791. Newell A, Shaw J. C., Simon H. A., Elements of a theory of human problem-solving, *Psychol. Rev.*, **65**, 151—166 (1958). **Ж6. Ж4, И4, П2, И3, Ж5, Л1, Л2, Ж3; 23, 36.**

792. Newell A., Shaw J. C., Simon H. A. Chess-playing programs and the problem of complexity, *IBM J. Res. a. Develpt.*, **2**, № 4, 320—335 (1958). **Ж2, Ж4, С5, В1, В2, И4, П2, Ж5, Л1, Л2, Ж3; 23, 36** (**С5; 31**), [1086, 553, 120, 952].

793. Newell A., Shaw J. C., Simon H A., The Processes of Creative Thinking, RAND Corp. Paper P-1320, Santa Monica, Calif, 1958. **Ж6, Ж4, Ж5, В1, В2, И4, П, Л1, Л2, Ж3.**

794. Newell A, Simon H. A, Heuristic problem-solving: The next advance in operations research, *J. Operations Research Soc. America*, **6**, № 1 (1958)

795. Newell A, Shaw J. C., Simon H. A., Report on a general problem-solving program, см. [R], pp. 256—264. **Ж1, Ж10, Ж4, Ж6, И4, И2, И3, Ж7, Ж8, Л1, Л2; 33, 36.**

796. Newell A., Shaw J. C., Simon H A., A general problem-solving program for a computer, *Computers a. Automation*, **8**, № 7, 10—16 (1959).

797. Newell A., Shaw J. C., Simon H. A., Report on the Play of Chess Player 1—5 of a Book Game of Morphy vs. Duke Karl of Brunswick and Count Isouard, CIP Working Paper 21, Graduate School of Ind. Administration, Carnegie Inst. Technol., 1959. **С5** (шахматы).

798. Newell A., Shaw J. C., Simon H. A., A variety of intelligent learning in a general problem solver, см. [U], pp. 153—189. **Г11, Ж1, Ж4, Ж6, И4, И2, И3, И6, Ж7, Ж8, Ж9, Е4, Е5, Ж10, Д6, К4, Д1, Д2; 34, 33.**

799. Newell A., On programming a highly parallel machine to be an intelligent technician, см. [V], pp. 267—282. **Л1, Л2, Б2, Ж, И3, В** ([481]).

800. Newell A., Tonge F. M., An introduction to information processing language IPL-V, см. [T], pp. 205—211. **И3, И4; 15, 35.**

800a*. Newell A., Tonge F. M., Feigenbaum E. A., Mealy G. H., Saber N., Green B. F., Jr., Wolf A. K., Information Processing Language V Manual, Sect. I: The Elements of IPL Programming, RAND Corp., Paper P-1897, 1960.

8006*. Newell A., Tonge F. M., Feigenbaum E. A., Mealy G. H., Saber N., Green B. F., Jr., Wolf A. K., Information Processing Language V Manual, Sect. II: Programmers Reference Manual, RAND Corp., Paper P-1918, 1960.

801. Newell A., Simon H. A, GPS — a program that simulates human problem-solving, см. [AE].

802. Newell A., Simon H. A., The simulation of human thought, см. Current Trends in Psychological Theory, Univ. of Pittsburgh Press, Pittsburgh, Pa., 1961. pp. 152—179. **Ж, Ж3, Ж4, Ж6, Ж7, Л2, М4, Н.**

803. Newell A., Simon H. A., GPS, a program that simulates human thought, см. Billing H. (ed.) Lernende Automaten, Oldenbourg, München, 1961, S. 109—124. **Ж, Ж3, Ж4, Ж6, Ж7, Л2, М4, Н.**

804. Newell A., Simon H. A., Computer simulation of human thinking, *Science*, **134**, № 3495, 2011—2017 (1961). **Ж, Ж3, Ж4, Ж6, Ж7, Л2, М4, Н.**

805. Newell A. (ed.), Information Processing Language V Manual, Prentice-Hall, Englewood Cliffs, N. J., 1961. **И4.**

806*. Newell A., Shaw J. C., Simon H. A., The processes of creative thinking, см. Gruber H. E., Terrell G., Wertheimer M. (eds.), Contemporary Approaches to Creative Thinking, New York, 1962. **Л.**

807. Newell A, Simon H. A., Computer simulation of human thinking and problem-solving, см. [AD], pp. 94—133; см. также *Computers a. Automation*, **10**, 18—19 и след. (Apr. 1961), и *Datamation*, № 6, 7 (1961) **Д, Д2, Ж, Ж4, И1, Л2.**

808. Newell A., Some problems of basic organization in problem-solving programs, см. Self-organizing Systems—1962, Spartan books, New York, 1962.

809*. Newell A. et al., Information Processing Language V Manual, 2nd ed., Prentice-Hall, Englewood Cliffs, N. J., 1964. **И4.**

810*. Newell A., Simon H. A., Simulation of human processing of information, *Am. Mathem. Monthly*, **72**, № 2, pt. 2, 111—118 (1965). **Л2, Ж, Ж3, Ж4, Н.**

810a*. Newell A., Simon H. A., Programs as theories of higher mental processes, см. [AU].

810б*. Newell A., On the analysis of human problem solving protocols, Intern. Symp. on Mathem. a. Comput. Methods in the Social Sciences, Rome, July, 1966.

811. Newman E. A., Machines that try to think, *Control*, **1**, 294—295 (Dec. 1958). **Р; 31.**

812. Newman E. A., An analysis of non-mathematical data processing, см. [N], pp. 863—876.

813*. Norman M. F., A two-phase model and an application to verbal discrimination learning, Stud. Mathem. Psychol., Univ. Press Stanford, Calif., 1964, pp. 173—187. **Д2.**

814*. Novotny G. V., Machines that think like people, *Electronics*, **37**, № 20 (1964). **Н.**

815. Oettinger A. G, Programming a digital computer to learn, *Philos. Mag.*, **43**, 1243—1262 (1952); см. также *Methods*, **11** (1959); сокращенный вариант см. *Proc. Assoc. Computing Machinery*, Sept. (1952). **Д3, Д9, Б1; 12, 34, 36, 32.**

816. Ogden C. K., Basic English: An Introduction with Rules and Grammar, 4th ed, Kegan Paul, Trench, Trubner a. Co., London, 1933. **И1, И2.**

817. Orchard-Hays W., The evolution of programming systems, см. [Y], pp. 283—295. **И3, И4; 31, 32, 35.**

818. Osgood C. E., Suci G., Tannenbaum P., The Measurement of Meaning, Univ. of Illinois Press, Urbana, Ill., 1957. **И5.**

819*. Overton R. K., Intelligent machines and nazy questions, *Computers a. Automation*, **14**, № 7, 26—30 (1965); русский перевод: Овертон Р., Разумные машины и туманные вопросы, *Зарубежная радиоэлектроника*, № 3, 68 (1966). **Н**.
820*. Padelty M. J., An approach to machine intelligence. Cleaver-Hume Press, London, 1963. **В, Г1, Д, Е, Ж, К2, Л1, Н; 36, 32, 31**.
821. Pahl P. M., Johnson D. L., Pattern recognition in an electronic reader, *Trend*, **11**, 16—21 (July 1959). **Г10**.
822. Papert S., Some mathematical models of learning, см. [X]. **Г4, Г3, Г9, Д3, Д6, Г11, В3; 21, 33**.
823*. Парин В. В., Баевский Р. М., Кибернетика в медицине и физиологии, Медгиз, 1963.
824. Pask G., Organic control and the cybernetic method, *Cybernetica*, **1**, № 3, 155—173 (1958). **Р; 11, 24, 36**.
825. Pask G., Physical analogues to the growth of a concept, см. [N], vol. 2, pp. 877—928. **Н1, Б4, Д, Р5; 13, 24, 34**.
826. Pask G., Von Foerster H., A proposed evolutionary model, см. [N]. **Д9, Е6, С1, С5, Р5, Р6; 16, 31, 35**.
827. Pask G., Von Foerster H., A predictive model for self-organizing systems, *Cybernetica*, **3**, № 4, 258—300 (1960); **4**, № 1, 20—55 (1961). **Д2, Д7, Ж, Н, Р, С1, С3; 21, 24, 33, 36**.
828*. Pask G., Teaching machines, Proc. 2nd Congr. Int. Assoc. Cybernetics, Namur, 1958; Gauthier Villars, 1960. **Р; 31, 36**.
829. Pask G., The natural history of networks, см. [U], pp 232—263. **Н, Б4, Д, И5, Р5; 13, 24, 34**.
830*. Pask G., The teaching machines as a control mechanism, *Trans. Soc. Instr. Technol.*, June (1960). **Р; 33, 36**.
831*. Pask G., Interaction between a group of subjects and an adaptive automation to produce a self-organizing system for decision making, см. [AO]. **Д, Р5; 21, 24, 33, 36**.
832. Pask G., An Approach to Cybernetics, Hutchinson, London, 1961; Harper a. Row, New York, 1962. **Д, Е, Ж, М, Н, Р**.
833. Pask G., The simulation of learning and decision-making behavior, см. [AG]. **Р5**.
834*. Pask G., A discussion of the cybernetics of learning behavior, см. [AO], pp. 177—215.
835*. Pask G., Learning machines, Survey paper at 2nd IFAC Conf., Basel, 1963. **Д; 21, 24, 31, 36**.
836*. Pask G., Adaptive Teaching Machines, см. Austwick K. (ed.), Teaching Machines, Pergamon, London, 1964. **Д, Д7, Р, С3; 21, 24, 34**.
837*. Pask G., A discussion of artificial intelligence and self-organization, Acad. Press, 1964. **Н, Р, Р5; 21, 24, 33, 36**.
838*. Pask G., Tests for some features on a Cybernetic Model of learning, Symp. on Cybernetic Problems in Psychology, Humboldt Univ., DDR, Berlin, 1964. **Д; 21, 24, 33**.
839*. Pask G., The developments of communication skills under adaptively controlled conditions, *Programmed Learning*, **2**, 69—88 (1964). **Д7**.
840*. Pask G., Comments on an indeterminancy that characterises a self-organizing system, Cybernetics of Neural Processes, Consiglio Nazionale Delle Richerche, Rome, 1965. **Р5, 21, 24, 33, 36**.
841*. Pask G., Some comments on the organization of men, machines and concepts, см. Heilprin, Markuson, Goodman (eds.), Education for Information Science, Spartan a. MacMillan, 1965. **Д, Н, Р; 21, 24, 36**.
842*. Pask G., Teaching as a control-engineering process, *Control*, 1—4 (1965).
843*. Pask G., Man as a system that needs to learn, см. Beer S., George F. H., Goodman R. (eds.), Advances in Cybernetics, Acad. Press, New York, 1966. **Д, Н, Р, Ж, С3; 21, 24, 33, 36**.

844. Paycha F., Medical diagnosis and cybernetics, см. [N], pp. 635—659. **П1, П2**.
845. Peirce C. S., Logical machines, *Am. J. Psychol.*, **1**, 165—170 (1887). **Ж8**.
846. Pendergraft E., Machine Language Translation Project, 9th Quart. Progr. Rep., 1 May—31 July 1961, U. S Army Signal Corps Contr. DA 36-039 SC 78911, File № 19678-PM 59-91-91 (6909). **И1**.
847. Penfield W., Rasmussen T., The Cerebral Cortex of Man, Macmillan, New York, 1950. **М3; 23, 34**.
848*. Penfield W., Roberts L., Speech and brain mechanisms, Princeton Univ. Press, Princeton, N. J., 1959. **И, М3**.
849. Penrose L. S., Automatic mechanical self-reproduction, см. New Biology, № 28, Penguin, Baltimore, 1959, pp. 92—117. **С4**.
850. Penrose L. S., Self-reproducing machines, *Scientific American*, **200**, № 6, 105—112, 114, 202 (1959). **С4**.
851. Pervin I. A., On algorithms and programming for playing at dominoes, англ. перевод работы Первин И. А., *ДАН СССР*, **124**, № 1, 31—33 (1959), см. также *Automation Express*, **1**, № 8, 26—28 (1959); **В1, С5; 12, 34**.
852*. Петренко А. И., Свечников С. В., Основные направления в развитии читающих автоматов, *Изв. вузов, Радиотехн.*, **4**, № 3 (1961). **Г10**.
853. Petrick S. R., Talking to a computer, *New Scientist*, **10**, № 235, 370—372 (1961). **Г7, Г2**.
854. Petrick S. R., Use of a list processing language in programming simplification procedures, Proc. 2nd Annual AIEE Symp. on Switching Circuit Theory a. Logical Design, AIEE S-134, 1961, pp. 18—24, **И4, И3**.
855. Pfeiffer J., This mouse is smarter than you are, *Popular Sci.*, **160**, 99—100 (March 1952). **С7; 13, 34**, [984].
856. Pfeiffer J., The Human Brain, Harper a. Row, New York, 1955. **М3, М4**.
857. Piaget J., The Language and Thought of the Child, Harcourt, Brace a. World, New York, 1926. Meridian Books, New York, 1955. **П10, Ж4, Ж5; 22**.
858. Piaget J., The Psychology of Intelligence, Harcourt, Brace a. World, New York, 1950. **Ж4; 22**.
859. Piaget J., The Construction of Reality in the Child, Basic Books, New York, 1954. **Ж4; 22**.
860. Pierce J. R., см. [243].
861. Pierce J. R., David E. E., Man's World of Sound, Garden City, N. Y., Doubleday, 1958. **Г7, И7, М3; 12, 13, 15, 34, 35, 31**.
862. Pierce J. R., Symbols, Signals and Noise, Harper a. Row, New York, 1961; русский перевод: Пирс Дж., Сигналы и шумы, **И5, И7, К3, Г**.
863*. Пийль Е. И., Способ размещения внутренних состояний асинхронного конечного автомата, см. Принципы построения сетей и систем управления, изд-во «Наука», 1964. **А1; 21**.
864. Pitts W., A general theory of learning and conditioning, *Psychometrika*, **8**, № 1, 131 (1943). **Д8**.
865. Pitts W., McCulloch W. S., How we know universals, *Bull. Mathem. Biophys.*, **9**, 127—147 (1947). **Г12, Г6, М3, Р1; 23, 21, 36**.
866. Pitts W., замечания, см. [E], pp. 108—111, и [J]. **Н, Г1, М4**.
867. Poincaré H., Science and hypothesis, выдержки см. [H]); русский перевод: Пуанкаре А., Наука и гипотеза, СПБ, 1906. **Ж7, Ж4; 12, 22, 21**.
868. Polanyi M., Problem-solving, *Brit. J. Philos. of Sci.*, **8**, 89 (1957);

см также Polanyi M, Personal Knowledge, London, 1958, pp 120—135 **В, Ж, Ж6, Л2, 24**
869* Полетаев И А, Сигнал О некоторых понятиях кибернетики, изд во «Советское радио», 1958 **Р, 35**
870 Polya G, How to Solve It, Princeton, N J, 1954, русский перевод Пойа Д, Как решать задачи Учпедгиз, 1961 **Ж4, Ж5, Л2 К1, К5, Е6, 35, 36**
871 Polya G, Induction and analogy in mathematics, Mathematics and Plausible Reasoning, vol 1, Princeton N J, Dover, New York, 1954, русский перевод Пойа Д, Математика и правдоподобные рассуждения, ИЛ, 1957 **Ж7, Ж4, Л2, Ж5 К1, К5, Е6**
872* Polya G, Patterns of plausible inference, Princeton, N J, 1954
873 Popper K R, Indeterminism in quantum physics and in classical physics, *Brit J Philos of Sci*, **1**, 117—133 (1950)
874 Popper K R, The logic of scientific discovery, review by Good, *Mathem Rev*, **21**, 1171—1173 (1960) **К5, К4, К6, К1, Н 24, 36 21**
875 Porter A, McLean D, Human performance in the solution of logical problems, Actes 2 congr int, rech operat, Aixen, Provence, 1960 Paris, 1961, pp 128—138 **Ж4, 21, 34**
876 Post E, Formal reductions of the general combinatorial decision problem, *Am J Mathem*, **65**, № 2, 197—215 (1943) **И2, А7**
877 Prawitz D, Prawitz H, Vogera N, A mechanical proof procedure and its realization in an electronic computer, *J Assoc Computing Machinery*, **7**, № 2, 102—128 (1960) **В1, Ж7, 12, 21, 34, 32**
878 Precker J A, Toward a theoretical brain model, *J Personality*, 22 310—325 (March 1954) **М3**
879 Pribram K, On the neurology of thinking, *Behavioral Sci*, **4**, 265—287 (Oct 1959) **М3**
880 Pringle J W S, On the parallel between learning and evolution, *Behaviour*, **3**, 174 (1951) **В10**
881 Prinz D G, Robot chess, *Research*, **5**, 261 (1952) **С5**
882 Prywes N, Data processing aspects of some psychological experiments, *Perceptual and Motor Skills*, **12**, 155—160 (1961)
883 Prywes N, Gray H J, The organization of a multilist type associative memory, Cigacycle Systems, AIEE Publ 5 136, Jan 29—Febr 2, 1962, pp 87—101 **Б1, В6, В3, Д5, И2, П1**
884' Putter P S, Zur Theorie der Denkvorgange, *Elektron Datenverarb*, № 1 27—40, № 2, 88—101 № 3 123—132, № 5 (1963)
885* Пушкин В Н, Оперативное мышление в больших системах изд во «Энергия», 1965 **Ж1, Ж4, С5, С8, Л; 13, 15, 16, 22, 32, 34, 36.**
886 Quastler H, The complexity of biological computers, *IRE Trans on Electronic Computers*, **EC-6**, № 3, 192—194 (1957) **М4**
887 Quillian R, Word concepts a theory and simulation of some basic semantic capabilities CIP working paper 79 Carnegie Inst Technol, Pittsburgh, 1965
888 Quine W V O, A proof procedure for quantification theory, *J Symbolic Logic*, **20**, 141—149 (1955) **В1, Ж7**
889 Rabin M O, Scott D, Finite automata and their decision problems, *IBM J Res a Develpt*, **3**, 114—125 (Apr 1959), русский перевод Рабин М О, Скотт Д, Конечные автоматы и задачи их разрешения, Кибернетический сборник, вып 4 ИЛ, 1962 **А1, 21**
890* Rabin M O, Probabilistic automata, *Information a Control*, **6**, № 3, 230—244 (1963), русский перевод Рабин М О, Вероятностные автоматы Кибернетический сборник, вып 9, изд во «Мир», 1964 **А6, 21**
891 Rapoport A, Cycle distribution in random nets, *Bull Mathem Biophys*, **10**, (1948) **Б6, Б4, 21**

892 Rapoport A, Shimbel A, Mathematical biophysics, cybernetics and general semantics, *Rev General Semantics*, **6**, 145 (1949) **Р, Б6**
893 Rapoport A, On the application of the information concept to learning theory, *Bull Mathem Biophys*, **18**, 317 (1956) **Д9, И7**
894 Rapoport A, Fights, games and debates, Univ of Michigan Press, Ann Arbor, Ill, 1960 **С5, 36**
895 Rapoport A, Hays W, Birch J, The formation of hypotheses and styles in the process of solving a logical task, Decisions, Values and Grow, Vol 1, Pergamon, Oxford—London—New York—Paris, 1960, pp 50—69 **Д6, Ж4, 22**
896 Rappaport D (ed), The Organization and Pathology of Thought, New York, 1951 **Ж4, 22**
897 Rashevsky N, Advances and Applications of Mathematical Biology, Univ of Chicago Press, Chicago, 1940 **Б6, Б5, Г1, Д8, М3, Р, 21, 31, 32**
898 Rashevsky N, The mathematical biophysics of some mental phenomena, *Bull. Mathem Biophys*, **7**, 115 (1945) **М3, 21**
899 Rashevsky N, Mathematical biophysics of abstraction and logical thinking, *Bull Mathem Biophys*, **7**, 133—148 (1945) **Ж6, Г, Ж8**
900 Rashevsky N, The neural mechanism of logical thinking, *Bull Mathem Biophys* **8**, 29—40 (1946) **М3, Ж6, Ж8**
901 Rashevsky N, Mathematical Biophysics, Univ of Chicago Press, Chicago 1938, Dover, New York, 1960 **Б6, Б5, Г1 Д8, М3, Р, 21, 31, 32, 36**
902* Rashevsky N, Mathematical foundations of general biology, *Ann N Y Acad Sci*, **96**, № 4, 1105—1116 (1962) **Г, Б6, Д8, М3, 21**
903 Ray L C, Kirsch R A, Finding chemical records by digital computers, *Science*, **126**, № 3278, 814 (1957) **П1, 12, 34, 36**
904 Razran G, Soviet psychology and psychophysiology, *Science*, **128**, 1187—1194 (1958)
905 Reich D L Ernst H A, A Mechanical Hand, Quart Progr Rep 56, Res Lab Electronics, MIT, Cambridge, Mass, 1960, pp 156—157 **Р1, 13, 12, 33 34**
906* Reid J W An experimental study of analysis of goal" in problem solving, *J Gen Psychol*, **44**, № 1 51—69 (1951), русский перевод **[АО]**
907 Reitman W R Heuristic programs, computer simulation and higher mental processes, *Behavioral Sci*, **4**, 330—335 (1959) **Ж1, М3, Л2, Л1, 31**
908 Reitman W R Programming intelligent problem solvers, см [AE], pp 26—33 **Ж1, И3, Л2, Л1, С6; 31.**
909 Reitman W R, Book review, *Science*, March 2 (1962) **Н5, Н2**
910 Reitman W R, Personality as a problem solving coalition, см Tomkins S S, Messick S (eds), Computer simulation of personality, Wiley, New York, 1963 **Ж, Л2**
911 Reitman W R, Grove R B, Shoup R G, Argus an information processing model of thinking, *Behavioral Sci*, **9**, 270—281 (1964) **Ж1, Ж4, Ж6, И4, И3, Ж8, Л1, Л2, 33, 36**
912 Reitman W R, Heuristic decision procedures, open constraints and the structure of ill defined problems, см Shelly M W, Bryan G L (eds), Human Judgements and Optimality, Wiley, New York, 1964 **Ж1, Ж4, Ж6, Ж8, Л**
913' Reitman W R, Cognition and Thought, An Information Processing Approach, Wiley, New York—London—Sydney, 1965 **Е4, Е5, Ж1, Ж4, Ж6, Ж8, Ж9, Л1, Л2, М3, Р7, Г, И2 И3, И4, Е6, 33, 34, 32**

914*. Restle F., The selection of strategies in cue learning, *Psychol. Rev.*, **69**, № 4, 329—343 (1962). **Д2**; **34**, **36**.
915. Richards P. I., Machines which can learn, *Am. Scientist*, **39**, 711 (1951). **С5**, **Д**.
916. Richards P. I., On game-learning machines, *Sci. Monthly*, **74**, № 4, 201—205 (1952). **С5**, **Д**.
917*. Risak V., Lernprozesse in Automaten, *Elektrotechnik u. Maschinenbau*, **82**, № 9, 417—421 (1965). **А**, **Д**; **21**.
918. Roberts L. G., Pattern recognition with an adaptive network, *IRE Int. Convent. Record*, pt. 2, 66—70 (1960); русский перевод: Роберте Л. Дж., Распознавание изображений при помощи приспосабливающихся систем, Кибернетический сборник, вып. 4, ИЛ, 1962. **Г13**, **Г12**, **Д3**, **Г10**.
919. Roberts L. G., Picture coding using pseudo-random noise, см. [AA], pp. 145—154. **Г6**.
920. Robinson A., Proving a theorem (as done by man, logician or machine), Proc. Cornell Summer Inst. of Logic, Transcription, Cornell, Ithaca, N. Y., 1957. **Ж6**, **Ж7**, **Ж4**; **36**.
921. Robinson A., On the mechanization of the theory of equations, *Bull. Res. Council Israel*, 9F, № 2, 47—70 (1960).
922. Robinson G. U., Binary choice in a dynamic environment, *Human Factors*, **6**, № 3, 287—291 (1964). **Д6**, **Ж4**; **34**.
923. Rochester N., Symbolic programming, *IRE Trans. on Electronic Computers*, **EC-2**, № 1, 10—15 (1953). **И3**.
924. Rochester N., Holland J. H., Haibt L. H., Duda W. L., Test on a cell assembly theory of the action of the brain, using a large digital computer, *IRE Trans. on Information Theory*, **IT-2**, № 3, 80—93 (1956). **М2**, **Б4**, **Б6**; **12**, **34**.
925. Rochester N., Goldberg S. H., Edwards D. J., Machine Manipulation of Algebraic Expressions, Quart. Progr. Rep. 55, Res. Lab. Electronics, MIT, Cambridge, Mass., 1959, pp. 132—134. **И4**, **И3**; **14** (LISP), **34**.
926. Rogers H., Jr., The present theory of Turing machine computability, *J. Soc. Ind. a. Appl. Mathem.* (SIAM), **7**, № 1, 114—130 (1959). **А4**, **А7**; **35**.
927. Rogers H., Jr., Review of Gödel's proof by Newman and Nagel, *Am. Mathem. Monthly*, **67**, 98 (Jan. 1960). **А7**, **Н2**; **31**.
928. Rome B. K., Rome S. C., Leviathan: A Simulation of Behavioral Systems, to Operate Dynamically on a Digital Computer, Int. Conf. for Standards on a Common Language for Machine Searching and Translation, Cleveland, 1959. **Р7**.
929*. Rome B., Rome S., Programming the bureaucratic computer, *IEEE Spectrum*, **1**, № 12, 72—78, 83—92 (1964), **Р7**, **С**; **12**, **31**.
930. Rosenblatt F., The Perceptron, a Theory of Statistical Separability in Cognitive Systems, Rep. VG-1196-G-1, Cornell Aeronaut. Lab., Buffalo, N. Y., 1958. **Б4**, **Б5**, **Г13**, **Г10**, **Д3**, **М1**, **Г9**; **13**, **33**, **39**.
931. Rosenblatt F., The perceptron: A probabilistic model for information storage and organization in the brain, *Psychol. Rev.*, **65**, 386—407 (Nov. 1958). **Б4**, **Б5**, **Г13**, **Д3**, **М1**, **П**, **Г9**; **13**, **33**.
932. Rosenblatt F., On the Convergence of Reinforcement Procedures in Simple Perceptrons, Rep. VG-1196-G-3, Cornell Aeronaut. Lab., Cornell Univ., Buffalo, N. Y., 1959. **Г13**, **Г9**, **Д3**, **Д4**; **21**.
933. Rosenblatt F., Two theorems of statistical separability in the perceptron, см. [N], vol. 1, pp. 421—456. **Б4**, **Б5**, **Г9**, **Г13**; **21**.
934. Rosenblatt F., Perceptual generalization over transformation groups, см. [U]. **Б4**, **Б5**, **Г13**, **Г9**, **Г12**, **Е**, **К3**; **11**, **33**, **36**.

935. Rosenblatt F., Perceptron experiments, *Proc. IRE*, **48**, 301—309 (1960).
936. Rosenblatt F., Principles of Neurodynamics, Cornell Aeronaut. Lab. Rep. 1196-G-8; Spartan, Washington, 1962; русский перевод: Розенблатт Ф., Принципы нейродинамики, изд-во «Мир», 1965. **Б5**, **Г13**, **Г9**, **Г14**, **Д**, **М1**, **М3**; **12**, **13**, **21**, **23**, **34**, **36**.
937*. Rosenblatt F., Analytic techniques for the study of neural nets, *IEEE Trans. on Appl. and Ind.*, 83, № 74, 285—292 (Sept. 1964); русский перевод: Розенблатт Ф., Аналитические методы изучения нейронных сетей, *Зарубежная радиоэлектроника*, № 8, 43—57 (1965).
938. Rosenblith W. A., Some quantifiable aspects of the electrical activity of the nervous system, *Rev. Modern Phys.*, **31**, № 2, 532—545 (1959). **12**, **23**, **34**, **32**, **31**.
939. Rosenbloom P. C., The Elements of Mathematical Logic, Dover, New York, 1950. **И2**, **Н2**; **31**, **35**.
940. Rosenblueth A., Wiener N., Bigelow J., Behavior, purpose and teleology, *Philos. of Sci.*, **10**, 18—24 (1943). **Р2**, **Н**.
941. Rosenblueth A., Wiener N., Purposeful and non-purposeful behavior, *Philos. of Sci.*, **17**, 318 (1950). **Р2**, **Н**.
942. Ross T., Machines that think, *Scientific American*, **148**, 206—208 (Apr. 1933). **Н2**.
943. Rothstein J., Information, organization and systems, *IRE Transactions on Information Theory*, **IT-4**, 64—66 (1954). **И7**, **И5**, **Р**.
944*. Рубинштейн С. Л., О мышлении и путях его исследования, Изд-во АН СССР, 1958. **М**; **22**.
945. Russell B., History of Western Philosophy, Simon a Schuster, New York, 1940. **Н3**.
946. Russell D. H., Children's Thinking, Ginn, Boston, 1956. **Ж5**; **22**.
947. Russell G., Learning Machines and Adaptive Control Mechanisms, Radar Research Establishment Memo. 1369, Brit. Ministry of Supply, Malvern, England, 1957. **Д**, **Р**, **Р1**, **Б5**, **В**, **Г14**, **К**; **13**, **34**.
948. Ryle G., The Concept of Mind, Hutchinson, Lnd., 1949. **Н**, **Н4**, **И5**, **К5**; **24**.
949. Salveson M. E., The assembly-line balancing problem, *Trans. ASME*, 77, № 16 (1955). [1075], **Ж11**.
950. Samuel A. L., Some studies in machine learning using the game of checkers, *IBM J. Res. a. Develpt.* 3, № 3, 210—229 (1959). **С5** (шашки). **Ж2**, **П1**, **Д4**, **Д3**, **Г4**, **В3**, **Д10**, **П1**; **12**, **34**, **36**.
951. Samuel A. L., Machine learning, *Technol. Rev.*, **62**, 42—45 (Nov. 1959).
952. Samuel A. L., Programming a computer to play games, см. Alt F. (ed.), Advances in Computers, Acad. Press, New York, 1960. **С5**; **31**, **32**, **35**.
953. Samuel A. L., Letter to the editor, *Science*, **132**, № 3429 (1960). **Н2**, **Н3**.
954. Samuel A. L., Artificial intelligence — a frontier of automation, *Ann. Am. Acad. Polit. a. Soc. Sci.*, **340**, 10—20 (1962).
955*. Samuel A. L., Artificial intelligence: progress and problems, *Computers a. automation*, **12**, № 3, 28—35 (1963); русский перевод: Самуэль А., Искусственный разум: прогресс и проблемы, *Зарубежная радиоэлектроника*, № 9, 63—73 (1963); см. также [00].
956. Samuelson P. A., Foundations of Economic Analysis, Cambridge, Mass., Harvard, 1947.
957. Sapir E., Language, Harcourt, Brace a. World, New York, 1939. **И1**, **И5**.
958*. Сапожков М. А., Речевой сигнал в кибернетике и связи, М. [illegible]

959* Sayre K M, Crosson F J (eds), The Modeling of Mind (Computers and Intelligence), Univ Notre Dame Press, Notre Dame, Indiana, 1963 **М**
960* Шалютин С М, Алгоритмы и возможности кибернетики, *Вопр филос*, № 8, 78—87 (1962) **Р**
961* Шанин Н А, Давыдов Г В и др, Алгоритм машинного поиска естественного логического вывода в исчислении высказываний, изд-во «Наука», 1965 **Ж, Ж1, 21**
962* Schay G, Spruth W, Analysis of a file addressing method, *Communic Assoc Computing Machinery*, **5**, № 8, 459—462 (1962) **И3, 21, 31**
963* Schelling T C, The strategy of conflict, Cambridge, Mass, Harvard, 1960 **Р7, С**
964 Schultz H, The Theory and Measurement of Demand, Univ of Chicago Press, Chicago, 1938
965 Schutzenberger M P, A tentative classification of goal seeking behavior, *J Mental Sci*, **100**, № 1, 97—102 (1954) **Р2**
966* Seidmore A K, Weinberg M, Storage and search properties of a treeorganized memory system, *Communic Assoc Computing Machinery*, **6**, № 1, 28—31 (1963) **А**
967 Scott P, Williams K G, A note on temporal coding as a mechanism in sensory perception, *Information a Control*, **2**, № 4, 380—385 (1959) **Г6, И7**
968 Scriven M, The mechanical concept of mind, *Mind*, **62**, 230—240 (1953). **Н**
969* Sears R W, Adaptive Representations for Pattern Recognition, *IEEE Trans on Syst Sci a Cybernetics*, **SSC-1**, № 1, 59—67 (1965) **Д7, Г, 21**
970 Sebestyen G S, Classification Decisions in Pattern Recognition, Rep 381, Res Lab of Electronics, MIT, Cambridge, Mass, 1960 **Г9, Г11**
971 Sebestyen G S, Recognition of membership in classes, *Proc IRE on Information Theory*, **IT-7**, № 1, 40 (1961) **Г9, Г11**
972 Selfridge O G, Some notes on the theory of flutter, *Archives Inst Cardiology, Mexico*, **18**, № 2, 177—187 (1948) **Б6, 22**
973 Selfridge O G, Pattern recognition and modern computers, см [E], pp 91—93 **Г1, Г4, Г11, Г3, Г8, Д9, К2, К1, 12, 34, 33, 36** [X, 284]
974 Selfridge O G, Pattern recognition and learning, см [I], p 345, а также *Methods*, 8, 31 (1956) **Г11, Г1, Г8, Г4, Г3, Д9, К2, К1, Л1, В6, В8, С5, И5, Ж3, К3**
975 Selfridge O G, Minsky M L, Digital Computers and Pattern Recognition, Annual Meeting of the AAAS, 1956
976 Selfridge O G, Pandemonium a paradigm for learning, см [N], pp 511—529 **Г14, Г13, В8, Д3, Б2, Г11, М2, 12, 34, 33, 36**
977 Selfridge O G, Neisser U, Pattern recognition by machine, *Scientific American*, **203**, № 3, 60—68 (1960) **Г1, Г2, Г3, Г6, Г10, Г12, Г14, 31, 35** [188, 393, 285, X]
978 Shaginyan M, In the World of Cybernetics, Joint Publ Res Service Rep 718-D, U S Dept of Commerce, Washington, 1959 **Р**
979 Shannon C E, Weaver W, The Mathematical Theory of Communication, Univ of Illinois Press, Urbana, Ill, 1949, русский перевод этой и последующих работ Шеннона см Шеннон К, Собрание сочинений ИЛ, 1964 **И7, И1, К3**
980 Shannon C E, Synthesis of two terminal switching networks, *Bell Syst Techn J*, 28, № 1, 59—98 (1949) **А2, А3, К6**
981 Shannon C E, Communication theory of secrecy systems, *Bell Syst Techn J*, **28**, № 4, 656—715 (1949) **И7**

982 Shannon C E, Automatic chess player, *Scientific American*, **182**, 48 (1950). **С5, Ж2, Ж3, Е6, 33**
983 Shannon C E, Programming a digital computer for playing chess, *Philos Mag*, **41**, 356—375 (1950) **С5, Ж2, Ж3, Е6, 33**, см также [H]
984 Shannon C E, Presentation of a maze solving machine, см [B], 8th conf, pp 173—180 **Д10, С7, 13, 34**
985 Shannon C E, Computers and automata, *Proc IRE*, **41**, № 10, 1234—1241 (1954), см также *Methods*, **6**, 115 (1954) **А1, Б1**
986 Shannon C E, Game playing machines, *J Franklin Inst*, **260**, № 6, 447—453 (1955) **С5, Е4, К1, Д10, 13, 34** [436, X]
987 Shannon C E, A universal Turing machine with two internal states, см [G] **А4**
988 Shannon C E, Von Neumann's contributions to automata theory, *Bull Am. Mathem Soc*, **64**, № 3, pt 2, 123—129 (1959) **31**
989* Shaw C J, U and the machine, *Datamation*, **10**, № 10, 38—42 (1964). **14**
990 Shaw J C, Newell A, Simon H A, Ellis T O, A command structure for complex information processing, Proc Western Joint Computer Conf, May 6—8, 1958 **И3, И4, 15, 35, 38** [790]
991 Shepard R N, Chang J J, Stimulus Generalization in the Learning of Classifications, mimeogr rep, Bell Telephone Lab, 1961 **Д2, Г**
992 Shepard R N, Hovland C I Jenkins H M, Learning and memorization of classifications, *Psychol Monogr*, **517**, № 75 (1961) **Д2, Г**
993 Sherman H, A quasi topological method for machine recognition of line patterns, см [R] **Г10, Г5 Г8, 21, 34**
994 Shimbel A, Contributions to the mathematical biophysics of the central nervous system, with special reference to learning, *Bull Mathem Biophys*, **12**, 24 (1950) **Д, М3**
995 Sholl D A, Uttley A M, Pattern discrimination and the visual cortex, *Nature*, **171**, 387—388 (1953) **Б6, Г, М3**
996 Sholl D A, The Organizaiton of the Cerebral Cortex, Methuen, London, 1956, **М3**
997 Shoulders K R, On microelectronic components, interconnections and system fabrication, см [V] **Б1, Б2**
998 Shubik M, Games, decisions and industrial organizations, *Management Sci*, **6**, 455—474 (1960) **С8, С1, С2**
999 Shubik M, Bibliography on simulation, gaming, artificial intelligence, and allied topics, *J Am Statist Assoc*, **55**, № 292, 736—751 (1960) **С8, С1, С2, Р3, 32**
1000* Shubik M (ed), Game theory and related approaches to social behavior, Selections, Wiley, New York—London—Lidney, 1964 **С1, С2, Р7**
1001 Shultz G L, Use of IBM 704 in simulation of speech-recognition systems, Proc of the Eastern Joint Computer Conf, 1957, pp 214—218 **Г7, 12, 34**
1002 Siegel A I, Wolf I I, A technique for evaluating man — machine system designs, *Human Factors*, **3**, № 1, 18—28 (March 1961), русский перевод Сиджел А, Вольф И, Метод оценки вариантов построения системы человек — машина, *Зарубежная радиоэлектроника*, № 8, 36—54 (1962) **С3**
1003 Simmons R. F., Anticipated Developments in Machine Literature Processing in the Next Decade, SP 129, System Develpt Corp, Santa Monica, Calif, 1960 **П1**
1004* Simmons R F, Answering English question by computer a survey, *Commun of the ACM*, **8**, № 1 (1965), русский перевод Симонс Р,

ЭВМ, отвечающие на вопросы, заданные по-английски, *Зарубежная радиоэлектроника*, № 7 (1965). **И6, И3, С3.**
1005. Simon H. A., A behavioral model of rational choice, *Quart. J. Economics*, **69**, 99 (1955). **С2.**
1006. Simon H. A., Newell A., Models: Their uses and limitations, см. White (ed.), The State of the Social Sciences, Univ. of Chicago Press, Chicago, 1956. **Ж4.**
1007. Simon H. A., Rational choice and the difficulty of the environment, *Psychol. Rev.*, **63**, 129 (1956). **С2, Ж5.**
1008. Simon H. A., Models of Man, Wiley, New York, 1957. **Е6, Ж4; 22, 21.**
1009. Simon H. A., Newell A., Heuristic problem-solving, *Operations Research*, **6**, № 1, 1—10 (1958). **Л1, Л2, Ж; 35.**
1010*. Simon H. A., Theories of decision-making in economics and behavioral science, *Am. Econ. Review*, **49**, 253—283 (1959).
1011. Simon H. A., The New Science of Management Decision, Harper a. Row, New York, 1960.
1012. Simon H. A., Modelling human mental processes, см. [Z], pp. 111—120. **Ж, Ж4, Л1, Л2.**
1013. Simon H. A., The control of the mind by reality: Human cognition and problem solving, см. Farber, Wilson (eds.), Control of the Mind, McGraw-Hill, New York, 1961, pp. 219—232. **Д2, Ж3, Ж4, Л2, Н2.**
1014. Simon H. A., Experiments with a heuristic compiler, RAND Corp. Paper P-2349, Santa Monica, Calif., 1961.
1015*. Simon H. A., The architecture of complexity, *Proc. Am. Philos. Soc.*, **106**, 467—482 (1962).
1016*. Simon H. A., Simon P. A., Trial and error search in solving difficult problems: evidence from the game and chess, *Behavioral Sci.*, **7**, 425—429 (1962). **С5.**
1017*. Simon H. A., Experiments with a heuristic compiler, *J. Assoc. Computing Machinery*, **10**, № 4, 493—506 (1963). **Л1; 14.**
1018*. Simon H. A., Kotovsky K., Human acquisition of concepts for sequential patterns, *Psychol. Rev.*, **70**, 534—546 (1963). **Г.**
1019*. Simon H. A., Newell A., Information processing in computer and man, *Am. Scientist*, **52**, № 3, 281—300 (1964). **М4.**
1020. Singer J. R., Electronic analog of the human recognition system, *J. Opt. Soc. America*, **51**, 61—69 (1961). **Г, Д, Б.**
1021. Skinner B. F., Science and Human Behavior, Macmillan, New York, 1953. **Ж5, Д2; 22, 35.**
1022. Skinner B. F., Verbal Behavior, Appleton-Century-Crofts, New York, 1957. **И5, И1, Д2, Д3.**
1023. Skinner B. F., Cumulative Record: Enlarged Edition, Appleton-Century-Crofts, New York, 1961. **Д1, Д2, Д3, И1, Н1, Н3, Н4, Ж5.**
1024. Slagle J., A Computer Program for Solving Problems in Freshman Calculus (SAINT), диссертация, MIT, Cambridge, Mass., 1961. **Ж1, Ж10, Ж7, Л1, Л2, Г4, И4; 12, 14, 21, 34.**
1025. Sluckin W., Minds and Machines, Pelican, London; Penguin, Baltimore, 1954. **Д, Р; 35.**
1026. Sluckin W., Information theory, probability models and psychology, *Penguin Sci. News*, **47**, 65 (1958). **И7, К1.**
1027*. Смолян Г. Л., Техника и мозг. *Вопр. филос.*, № 5, 83—94 (1965). **М4.**
1028*. Соболев С. Л., Ляпунов А. А., Кибернетика и естествознание, *Вопр. филос.*, № 5, 125—138 (1958). **Р.**
1029*. Соболев С. Л., Ляпунов А. А., Математические проблемы современной кибернетики, *Изв. Сибирск. отд. АН СССР*, № 5, 3—13 (1959). **Р.**
1030*. Солодовников В. В., Некоторые вопросы теории и принципы построения аналитических самонастраивающихся систем, сб. Аналитические самонастраивающиеся системы автоматического управления, М., 1965. **Д7.**
1031. Solomonoff R. J., An inductive inference machine, *IRE Nat. Conv. Rec.*, pt. 2, pp. 56—62, 1957. **К4, К6, Л1, Ж5, Д6, Г11; 33, 36.**
1032. Solomonoff R. J., The mechanization of linguistic learning, см. [AB]; см. также Zator Techn. Bull. ZTB-125, ASTIA Doc. AD 212226, 1958. **К2, К3, К4, И2.**
1033. Solomonoff R. J., A new method for discovering the grammars of phrase structure languages, см. [R], ASTIA Doc. AD 210390, 1959. **К2, К4, К3, К6, И2, Д6.**
1034. Solomonoff R. J., On machines to learn to translate languages and retrieve information, Zator Techn. Bull., AFOSR-TN-59646, 1959. **К2, И2, П1.**
1035. Solomonoff R. J., A preliminary report on a general theory of inductive inference, Zator Techn. Bull. ZTB-138, Contr. AF 49 (638)-376, Zator Co., Cambridge, Mass., 1960. **К2, К1, К5, К6, А4, А6; 21, 24.**
1035а*. Solomonoff R. J., A formal theory of inductive inference, Pt. I, *Inform. a. Control*, 1—2 (March 1964).
1035б*. Solomonoff R. J., A formal theory of inductive inference, Pt. II, *Inform. a. Control*, 224—254 (June 1964).
1035в*. Solomonoff R. J., Some recent work in artificial intelligence, *Proc. IEEE*, **54**, № 12 (1966); русский перевод: *ТИИЭР*, **54**, № 12 (1966).
1036. Somenzi V., Can induction be mechanized?, см. [I], p. 226. **Н2, К5.**
1037. Sperry R. W., Mechanisms of neural maturation, см. Stevens S. S. (ed.), Handbook of Experimental Psychology, Wiley, New York, 1951, pp. 236—280. **М3.**
1038. Sperry R. W., Neurology and the mind-brain problem, *Am. Scientist*, **40**, 291 (1952). **Г1, Н4, М4.**
1039. Spiegelthal E. S., Redundancy exploitation in the computer solution of double-crostics. Proc. Eastern Joint Computer Conf., 1960, pp. 39—56. **Ж3, В5.**
1040. Spitsbury R. J., Mentality in machines, *Proc. of the Aristotelian Society*, **26**, suppl., 61—86 (1952). **М4, П4.**
1041. Sprick W., Ganzhorn K., An analogous method for pattern recognition by following the boundary, см. [R], pp. 238—244. **Г11, Г10, Г3; 13.**
1042*. Stafford R. A., Multy-layer learning networks, *Techn. Papers*, **4**, № 3, 25—28 (1963); русский перевод: Стаффорд Р., Многослойные обучающиеся схемы, *Зарубежная радиоэлектроника*, № 8 (1965).
1043. Stein P., Ulam S., Experiments in chess on electronic computing machines, *Comput. a. Automation*, **6**, 14 (1957). **С5, Ж2, Л1.** [553, 120].
1044. Steinbuch K., Automatic symbol recognition, *Nachrichtentechn. Zs.*, **11**, № 4, 210; № 5, 237 (1958); Electrical Eng. Abstract, № 3722, 4872, 1958. **Г10, Г3, Г4; 13, 31, 32, 36.**
1045. Steinbuch K., Automatic speech recognition, *Nachrichtentechnische Zeitschrift*, **11**, 446 (Sept. 1958). **Г7.**
1046. Steinbuch K. Die Lernmatrix, *Kybernetik*, **1**, № 1, 36—45 (1961). [580], **Д4** [580].
1047*. Steinbuch K., Frank H., Nichtdigitale Lernmatrizen als Perzeptron *Kybernetik*, **1**, 117—124 (1961). **Г, Д, Б.**
1048. Stevens M. E., A Survey of Automatic Reading Techniques, Rep 5643, NBS, Washington, 1957. **Г10, Г3, Г1; 31, 32** [1050].
1049. Stevens M. E., A machine model of recall, см. [R], pp. 309—31. **П1, П2, Д5; 12, 34.**

1050. Stevens M. E., Automatic Character Recognition — a State-of-the-art Rep., NBS Techn. Note 112, PB № 161613, Washington, 1961. **Г10, Г1, Г2, Г4, Г5, Г8, Г9, Г12, Д7; 31, 36, 35, 32** (549 ссылок).
1051. Stevens M. E., Abstract shape recognition by machine, *Proc. Eastern Joint Computer Conf.*, **21**, 332—351 (1961). **Г8, Г11, Г12**, [279].
1052. Stevens S. S. (ed.), Handbook of Experimental Psychology, Wiley, New York, 1951; русский перевод: Анохин П. К., Артемов В. А. (ред.), Экспериментальная психология, ИЛ, том I, 1960; том II, 1963.
1053. Strachey C. S., Logical or non-mathematical programmes, Proc. of the Assoc. Computing Machinery, Ontario, Canada, Sept. 1952. **С5, И4**.
1054. Sutherland N. S., Stimulus analysing mechanisms, см. [N], pp. 575—609. **Г1, Г6, Д1, М3; 22, 23, 31, 32, 36**.
1055. Sutherland W. R., Malcolm J., Mugglin M. G., Sutherland I., An electromechanical model of simple animals, *Computers a. Automation*, **7**, 6—8, 23—25, 32 (1958); русский перевод: Сузерленд У. Р., Малкольм Дж., Магглин М., Сузерленд И., Электромеханическая модель простых животных, Кибернетический сборник, вып. 1, ИЛ, 1960. **Р; 13**.
1056. Sutro L., Emergency simulation of the duties of the president of the United States, см. [P], pp. 314—323. **Р4; 12, 33, 36**.
1057. Swallow R., Weston P., On the design of artificial nerve nets, см [O], pp. 34—46. **Б6**.
1058. Székely L., Knowledge and thinking, *Acta psychol.*, **7**, № 1, 1—24 (1950); русский перевод: см. [AL]. **Ж; 22**.
1059. Székely L., Productive processes in learning and thinking, *Acta psychol.*, **7**, № 2—4, 388—407 (1950); русский перевод см [AL]. **Ж, Д; 22**.
1060. Székely L., Some comments on problem-solving availability and test-magic, *Acta psychol.*, **14**, № 2, 152—157 (1958); русский перевод: см. [AL] **Ж; 22**.
1061*. Тарасенко Ф. П., К определению понятия «информация» в кибернетике, *Вопр. филос.*, № 4, 76—84 (1963) **И7**.
1062. Tarjan R., Neuronal automata, *Cybernetica*, **1**, № 3, 189 (1958). **А2, Б6**.
1063. Taube M., Man-machine relationship, *Datamation*, Jan.-Febr. (1959). **С3**.
1064. Taube M., Letter to the editor, *Science* (Aug. 26, 1960). **Н2**.
1065. Taube M., Computers and Common Sense, The Myth of Thinking Machines, Columbia, New York, 1961; русский перевод: Таубе М., Вычислительные машины и здравый смысл. Миф о думающих машинах, изд-во «Прогресс», 1964. **М4, Н1**.
1066. Taylor W. K., Electrical simulation of some nervous system functional activities, см. [I]. **М3**.
1067. Taylor W. K., Pattern recognition by means of automatic analog apparatus, *Proc. IEE*, pt. B, № 106, 198—209 (1959). **Г; 13**.
1068. Taylor W. K., Automatic control by visual signals, см. [N], pp. 843—861. **Г3, Г10, Г13; 13**.
1069. Thomac G., Biocybernetics: Mathematical analysis of organic systems, *Arch. Environment. Health*, **9**, № 3, 298—302 (1964).
1070. Thomson R., Sluckin W., Cybernetics and mental functioning, *Brit. J. Philos. of Sci.*, **4**, 130 (1954); Psychol. Abstract 3532. **Р4**.
1071. Thomson R., Sluckin W., Machines, robots and minds, *Durham Univ. J.*, **15** (1954). **Н4**.
1072. Thorpe W. H., Learning and Instinct in Animals, Cambridge, Mass., Harvard, 1956. **Д1; 35** [1073].

1073. Tinbergen N., The Study of Instinct, Oxford, New York, 1951. **Д1, Д10, Г14**.
1074*. Tomkins S. S., Messick S. (eds.), Computer simulation of personality, Wiley, New York, 1963. **Ж, Ж4, Л1, Л2**.
1075. Tonge F. M., An assembly line balancing procedure, *Management Science*, **7**, № 1, 21—42 (1960). **Е4, Л1, Л2, Ж1, И4, В4, В5, В6; 14, 34, 36, 31**.
1076. Tonge F. M., A Heuristic Program for Assembly Line-balancing, Prentice-Hall, Englewood Cliffs, N. J., 1961.
1077. Tonge F. M., The use of heuristic programming in management science, *Management Science*, **7**, № 3, Apr. (1961).
1078*. Torgerson W. S., обзор книги Luce R. D., Bush R. R., Galanter E. (eds.), Handbook of mathem. psychol., vol. 1., *Science*, **143**, 1319—1320 (1964). **Ж, Г, Д**.
1079. Трахтенброт Б. А., Об операторах, реализуемых в логических сетях, *ДАН СССР*, **112**, № 6 (1957). **А2; 21**.
1080*. Трахтенброт Б. А., Алгоритмы и машинное решение задач, Физматгиз, 1960. **Ж**.
1081*. Трахтенгерц Э. А., Языки программирования, *Автоматика и телемеханика*, **26**, № 6, 1115—1130 (1965). **И2, И3, И4**.
1082. Troll J. H., The thinking of men and machines, *Atlantic Monthly*, July, 62—65 (1954). **Н2**.
1083. Tsien H. S., Engineering Cybernetics, McGraw-Hill, New York, 1954; русский перевод с первого издания: Цзян Сюэ-сэнь, Техническая кибернетика, ИЛ, 1952.
1084. Turing A. M., On computable numbers with an application to the Entscheidungsproblem, *Proc. London Mathem. Soc.* (ser. 2), **42**, 230—265 (1936); **43**, 544 (1937). **А4, А7, Н1**.
1085. Turing A. M., Computing machinery and intelligence, *Mind*, **59** (n. s. 236), 433—460 (1950); см. также [H], pp. 2099 и след., и *Methodos*, **6**, 195 (1954); русский перевод: Тьюринг А., Может ли машина мыслить? Физматгиз, 1960. **Н, Н2, Н3, Н4, Д; 24, 36** [556, гл. 30].
1086. Turing A. M., соавтор гл. 25 (стр. 288—295) книги Bowden B. V. (ed.), Faster than Thought, Pitman, London, 1953. **С5, Ж2** (шахматы), **Г13, Л1; 12** (моделирование вручную), **34** [792].
1087. Uhr L., Machine perception of printed and hand-written forms by means of procedures for assessing and recognizing gestalts, см. 14th Nat. Meeting of the Assoc. Computing Machinery, Boston, Preprint 34, Mental Health Res. Inst., Ann Arbor, Mich., 1959. **Г11, Г5, Г10**.
1088. Uhr L., Latest methods for the conception and education of intelligent machines, *Behavioral Sci.*, **4**, 248—251 (1959) (обзор работы [P]).
1089. Uhr L., Intelligence in computing machines: The psychology of perception in people and in machines, *Behavioral Sci.*, **5**, 177—182 (1960). **Г6, Г11**.
1090. Uhr L., Vossler C., Suggestions for self-adapting computer models of brain functions, *Behavioral Sci.*, **5**, 91—97 (1961). **Г, М, Д7**.
1091. Uhr L., A possibly misleading conclusion as to the inferiority of one method for pattern recognition to a second method to which it is guaranteed to be superior, *IRE Trans. on Electronic Computers*, **EC-10**, 96—97 (1961). **Г9, Г**.
1092. Uhr L., Vossler C., A pattern recognition program that generates, evaluates and adjusts its own operators, см. [A], pp. 555—569 (1961). **Г11, К2, К4**.
1093. Uhr L., Vossler C., Recognition of speech by a computer program that was written to simulate a model for human visual pattern recognition, *J. Acoust. Soc.*, **33**, 1426 (1961).

1094. Uhr L., Vossler C., Uleman J., Pattern recognition over distortions, by human subjects and by a computer simulation of a model for human perception, *J. Exper. Psychol.*, **63**, 227—234 (1962).
1095. Uhr L., "Pattern recognition" computer as models for form perception, *Psychol. Bull.*, **60**, 40—73 (1963).
1096. Uhr L., The development of perception and language simulated models, см. Tomkins S. S., Messick S. (eds.), Computer simulation of personality, Wiley, New York, 1963. **Г, И**.
1097. Unger S. H., A computer oriented toward spatial problems, *Proc. IRE*, **46**, № 10, 1744—1750 (1958). **Б2, И3, А1, Г9, Г8**.
1098. Unger S. H., Pattern detection and recognition, *Proc. IRE*, **47**, № 10, 1737—1752 (1959).
1099*. Unger S. H., GIT — a heuristic program for testing pairs of directed line graphs for isomorphism, *Communic. Assoc. Computing Machinery*, **7**, № 1, 26—34 (1964). **Л1; 21, 35**.
1100. Uttley A. M., The classification of signals in the nervous system, *EEG Clinical Neurophysiol.*, **6**, 479 (1954). **Г14, Г1, М3, Г9, Г13, Б4, Б5, Б6**.
1101. Uttley A. M., The probability of neural connections, *Proc. Roy. Soc.*, **144B**, 299 (1955). **Б4, М3; 23**.
1102. Uttley A. M., Conditional probability machines, см. [G], pp. 253—275. **К1, Д5, Д9, Г9, Б2, Б6**.
1103. Uttley A. M., Temporal and spatial patterns in a conditional probability machine, см. [G], pp. 277—285. **К1, Д5, Д9, Г9, Г1, Г14**
1104. Uttley A. M., Conditional probability as a principle of learning, см. [F], pp. 830—856. **Г14, Г13, Г9, Д5, К1**.
1105. Uttley A. M., The design of conditional probability computers, *Information a Control*, **2**, 1—24 (Apr. 1959); русский перевод: Аттли А., Построение машин условной вероятности, Кибернетический сборник, вып. 8, изд-во «Мир», 1964. **К1, Д5, Д9, Г9; 13, 34** [580].
1106. Uttley A. M., Conditional probability computing in a nervous system, см. [N], pp. 121—146. **К1, Б2, Б4, Б6, Б5, М3, Д5, Г13, Г14, Г9**.
1107. Uttley A. M., The engineering approach to the problem of neural organization, *Progr. in Biophys.*, **2**, 25—52 (1961).
1108. Uttley A. M., Properties of plastic networks, *Biophys. J., Suppl.*, Rockefeller Inst. Press, 1966.
1109*. Van Heerden P. I., Theory of optical information storage in solids, *Appl. Opt.*, **2**, № 4, 393—400 (Apr. 1963); русский перевод: Ван Хирден, Теория оптического запоминающего устройства на твердом теле. *Зарубежная радиоэлектроника*, № 2, 32—45 (1964). **Г**.
1110*. Veenker G., Ein Programm zur Lösung von Schachaufgaben, *Elektron-Rechenanlag.*, **7**, № 1, 25—29 (1965). **Ж3, И5, И6, Е, Д4, Д6; 33**.
1111*. Вентцель Е. С., Элементы динамического программирования, изд-во «Наука», 1964. **В1, В4, В5, В6, Ж2, Ж3; 21**.
1112. Волгин Л. Н., К вопросу о создании обучаемых и самообучающихся систем автоматического управления, *Изв. АН СССР*, Техн. кибернетика, № 6, 81—84 (1964). **Д; 21, 36**.
1113. Verbeek L. A. M., On error-minimizing neuronal nets, см. [W]. **Б3, А6, Р6, Б2**.
1114. Verbeek L. A. M., Reliable computation with unreliable circuitry, см. [S]. **А6, Б2, Б3**.
1115. Vinacke W. E., The Psychology of Thinking, McGraw-Hill, New York, 1962. **Л2, Ж4; 22**.
1116. Von Foerster H., Quantum mechanical theory of memory, см. [B], 6th conf. **П3, М3**.

1117. Von Foerster H., Basic concepts of homeostasis, см. Homeostatic Mechanisms, Brookhaven Symp. in Biology, № 10, Upton, N. Y., 1957, pp. 216—242. **Р6, Р5; 21**.
1118. Von Foerster H., Some aspects in the design of biological computers, см. [O], pp. 47—64. **А2, Б2, Б4, Б6, Г1, Р5; 13**.
1119. Von Foerster H., On self-organizing systems and their environments, см. [U], pp. 31—50 **Р5**.
1120*. Von Foerster H., Biologics, см. Bernard E. E., Kare M. R. (eds.), Biological Prototypes and Synthetic Systems, Plenum Press, New York, 1962. **М**.
1121*. Von Foerster H., Communication amongst automata, *Am. J. Psychiat.* (1962). **А; 21**.
1122. Von Foerster H., Zopf G. W., Jr. (eds.), Principles of self-organization, Pergamon Press, New York, 1962; русский перевод: Принципы самоорганизации, изд-во «Мир», 1966. **Б6, В, Г11, Д**.
1123*. Vygotsky L. S., Thought and Language, Wiley, New York, 1962.
1124. Wada H. et al., An electronic reading machine, см. [R], pp. 227—232. **Г3, Г10, Г13**.
1125. Wald A., Statistical Decision Functions, Wiley, New York, 1950. **С2**.
1126. Wall P. D., Melzak R., Neural mechanisms which discriminate events on the skin, см. [AA], pp. 120—125. **Г7, М3, Р6**.
1127. Wallace R. A., The maze-solving computer. Proc. Assoc. Computing Machinery, Rimbach, Toronto, Pittsburgh, Pa., 1952, pp. 119—125. **С7, Д10**.
1128. Walter W. G., An imitation of life, *Scientific American*, **182**, № 5, 42 (1950). **Д8, М4; 13**.
1129. Walter W. G., A machine that learns, *Scientific American*, **185**, № 2, 60 (1951). **Д8, М4; 13, 34**.
1130. Walter W. G., The Living Brain, Norton, New York, 1953. **М3; 23, 31, 35, 36**.
1131. Walter W. G., Networks of neurons, Brooklyn Information Networks Symp., Brooklyn Polytechn. Inst., Brooklyn, N. Y., 1954, p. 185. **Б6**.
1132. Wang H., A variant to Turing's theory of computing machines, *J. Assoc. Computing Machinery*, **2**, 63 (1957). **А4, А6, Н2; 21** [737].
1133. Wang H., Toward mechanical mathematics, *IBM J. Res. a. Develpt.*, **4**, № 1, 2—22 (1960); русский перевод: Ван Хао, На пути к машинной математике, Кибернетический сборник, вып. 5, ИЛ, 1963. **Ж7, И2, Л1, Л2, В1, Н2, В2, В5; 21, 34, 36**.
1134. Wang H., Proving theorems by pattern recognition, I, см. [T], p 220. **Ж7**.
1135. Wang H., Proving theorems by pattern recognition, II, Bell Telephone Labs., Murray Hill, N. J., 1961. **Ж7**.
1136. Ware (ed.), Soviet computer technology — 1959, *IRE Trans. on Electronic Computers*, 72—120 (1960). **Б1**.
1137. Watanabe S., Information-theoretic aspects of inductive and deductive inference, *IBM J. Res. a. Develpt.*, **2**, № 4, 208—231 (1960). **К5, К1, К4, П2; 21, 12, 34**.
1138. Watson A. J., Some questions concerning the explanation of learning in animals, см. [N], pp. 693—728. **Д1, Г11, Д4; 22, 36**.
1139*. Weaver H. E., Madden E. H., "Direction" in problem solving, *J. Gen. Psychol.*, **27**, № 2, 331—345 (1949).
1140. Weber C. O., Homeostasis and servo-mechanisms for what? *Psychol. Rev.*, **56**, 234—236 (1949). **Р6, Р**.
1141. Wechsler D., The nature of intelligence, см. The Measurement and Appraisal of Adult Intelligence, Williams a. Wilkins, Baltimore, 1958. **Н2**.

1142. Wechsler D., Intelligence, quantum resonance and machine thinking, *Trans. New York Acad. Sci.*, **22**, № 4, 259—266 (1960). **Н2, М3, П**.
1143*. Wehr K. C., Conditioned-reflex circuits, *Electronics*, **37**, № 23, 66—70 (1964). **Д8; 21**.
1144. Weinberg M., Mechanism in neurosis, *Am. Scientist*, **39**, 74—98 (Jan. 1951). **Р4**.
1145. Weir J. M., A Learning Process Suitable for Mechanization, presented at the 11th nat. meeting Assoc. Computing Machinery, Los Angeles, Calif., Aug. 1956; A Physical Model of an Abstract Learning Process, presented at the 13th nat. meeting Assoc. Computing Machinery, June 1958. **Д, К4**.
1146. Wertheimer M., Productive Thinking, Harper a. Row, New York, 1945. **Ж4, Л2**.
1147. White G. M., Penny matching machines, *Information a. Control*, **2**, 349—363 (Dec. 1959); русский перевод: Уайт Дж. М., Машины, играющие в сравнение монет, Кибернетический сборник, вып. 3, ИЛ, 1961. **Д9, К1, С1, С5; 21**.
1148. Wickelgren W., A simulation program for conservative focusing, *Behavioral Sci.*, **7**, 245—247 (Apr. 1962).
1149. Wickelgren W., Cohen D. H., An Artificial Language and Memory Approach to Concept Attainment, *Psychological Rep.*, **10**, 815—827 (1962).
1150. Widrow B., Adaptive sampled-data systems — a statistical theory of adaptation, IRE Wescon Conv., San Francisco, 1959. **Р1, Д9, В8**.
1151. Wiener N., Rosenblueth A., Conduction of impulses in cardiac muscle, *Archives Inst. of Cardiol., Mexico*, **16**, 205—265 (1946); русский перевод: Винер Н., Розенблют А., Поведение импульсов в сердечной мышце, Кибернетический сборник, вып. 3, ИЛ, 1961. **Б6**.
1152. Wiener N., Cybernetics, Wiley, New York, 1948; русский перевод: Винер Н., Кибернетика, изд-во «Советское радио», 1958; см. также: Винер Н., Новые главы кибернетики, изд-во «Советское радио», 1963. **И7, Р1, Р6, Р2, Р3, Р4, Г12, Д3, М, Н, Ж2, С5**.
1153. Wiener N., Extrapolation, Interpolation and Smoothing of Stationary Time Series, MIT, Technology Press, Cambridge, Mass., 1949 **К1, Г9**.
1154. Wiener N., Sensory prostheses, см. [В]. **Г**.
1155. Wiener N., The Human Use of Human Beings, Houghton Mifflin, Boston, Mass., 1950; русский перевод: Винер Н., Кибернетика и общество, ИЛ, 1958. **Р3**.
1156. Wiener N., Speech, language and learning, *J. Acoust. Soc. America*, Wiener, Noted Scientist, U. S. News and World Report, № 24, 84—86.
1157. Wiener N., My connection with cybernetics: Its origin and its future, *Cybernetics*, **1**, № 1, 1—14 (1958). **Р**.
1158. Wiener N., Some moral and technical consequences of automation, *Science*, May 6 (1960). **Р3**.
1159. Wiener N., Machines smarter than men? Interview with Dr. Norbert Wiener, Noted Scientist U. S. News and World Report, № 24, 84—86. (Febr. 1964); русский перевод: Машины — изобретение людей? (Интервью Н. Винера для журнала «Юнайтед стейтс ньюс энд уорлд рипорт»), *Зарубежная радиоэлектроника*, № 7, 47—50 (1964). **Н**.
1160. Wilkes M. V., Can machines think? *Discovery*, **14**, 151 (1953); см. также *Proc. IRE*, **41**, 1230 (Oct. 1953). **Н2**.
1161. Wilkes M. V., Appendix on Can machines think?, см. Automatic Digital Computers, Methuen, London; Wiley, New York, 1956. **Н2**.
1162. Williams J. D., Toward Intelligent Machines, RAND Corp. Paper P-2170, Santa Monica, Calif., 1960. **Н2**.

1163. Williams T., Translating from Ordinary Discourse into Formal Logic, ASTIA doc. AD 98813, Res. Dept. Avion Div. ACF Industr., 1956. **И1, И2, И4**.
1164. Willis D. G., Plastic neurons as memory elements, см. [R]. **П1, Б6 (М1, Г13), Д3; 12, 34**. [580].
1165. Willis D. G., The functional domain of complex systems, см. [W]. **В3, В5**.
1166. Wisdom J. O., The hypothesis of cybernetics, *Gen. Systems Yearbook*, **1**, 111 (1951); см. также *Brit. J. Philos. of Sci.*, **2**, 1, 248, 312 (1951). **Р; 24**.
1167. Wisdom J. O., Mentality in machines, *Proc. Aristotelian Soc.*, **26**, suppl., 1—26 (1952) **Н**.
1168*. Wolin B. R., Decision processes research, Data Processing Yearbook, 1963—1964, Am. Data Process, Inc., Detroit, 1963, pp. 139—144. **Ж**.
1169. Woodward P. M., Jenkins D. P., Atoms and lists, *Computer J.*, **4**, № 1 (Apr. 1961).
1170*. Wooldridge D. E., The Machinery of the Brain, McGraw-Hill, New York, 1963; русский перевод: Вулдридж Д., Механизмы мозга, изд-во «Мир», 1965.
1171. Wozencraft J., Horstein M., Coding for two-way channels, см. [X]. **И7, Л1**.
1172*. Wozencraft J. M., Reiffen B., Sequential decoding, MIT Press, Cambridge, Mass., 1961; русский перевод: Возенкрафт Дж. М., Рейффен Б., Последовательное декодирование, ИЛ, 1963.
1173. Wright M. A., Can machines be intelligent? *Process Control a. Automation*, **6**, 2—6 (Jan. 1959). **Н2**.
1174. Wulfeck J. W., Taylor J. H. (eds.), Form Discrimination as Related to Military Problems, Publ. 561, Nat. Acad. Sci. Nat. Res. Counc., Washington, 1957. **Г1**.
1175. Wyckoff L. B., A mathematical model and an electronic model for learning, *Psychol. Rev.*, **61**, 89—97 (1954). **Д**.
1176. Xenakis I., Freie stochastische Musik durch den Elektronenrechner, *Gravesaner Bl.*, **7**, № 26, 54—78, 79—92 (1965).
1177. Yngve V. H., A programming language for mechanical translation, *Mechanical Translation*, **5**, № 1 (July 1956); русский перевод: Ингве В. Х., Язык для программирования задач машинного перевода, Кибернетический сборник, вып. 6, ИЛ, 1963. **И4, И1**.
1178. Yngve V. H., A model and an hypothesis for language structure, *Proc. Am. Philos. Soc.*, **104**, 444—466 (1961). **И2, И1**.
1179. Yngve V. H., Computer programs for translation, *Scientific American*, **206**, № 6, 68—76 (1962).
1180. Yntema D. B., Torgerson W. S., Man-computer cooperation in decisions requiring common sense, см. [AF], pp. 20—26; русский перевод: Интема Д., Торгерсон В., Совместная работа человека и вычислительной машины при принятии решений, требующих здравого смысла. *Зарубежная радиоэлектроника*, № 11, 68—80 (1961). **С3**.
1181*. Yntema D. B., Klem L., Telling a computer how to evaluate multidimensional situations. *IEEE Trans. Human Factors El.*, **6**, № 1, 3—13 (1965). **Ж**.
1182. Young J. Z., The organization within nerve cells, *Endeavor*, **15**, 5—19 (Jan. 1956). **М3**.
1183. Young J. Z., Doubt and Certainty in Science, Oxford, New York, 1957. **М, Н, П; 23, 34**.
1184*. Young J. Z., Models for a Brain. Oxford Univ. Press, 1964. **М3**.
1185. Zemanek H., Kretz H., Angyan A. J., A model for neurophysiological functions, см. [X], pp. 270—284. **Д1, Д8, К1, М3**.

Некоторые симпозиумы, собрания трудов и другие специальные сборники, посвященные проблемам искусственного разума

[A] Teleological mechanisms, *Ann. New York Acad. Sci.*, **50**, 189. (Frank, Hutchinson, Livingston, McCulloch, Wiener.)

[B] Von Foerster H. (ed.), Cybernetics: Circular Causal and Feedback Mechanisms in Biological and Social Systems, Transactions of the 6th (1949), 7th (1950), 8th (1951), 9th (1953) and 10th (1955) conferences, sponsored and published by the Josiah Macy, Jr., Foundation, New York.

[C] Jeffress L. A. (ed.), Cerebral Mechanisms in Behavior: The Hixon symposium, Wiley, New York; Chapman a. Hall, London, 1951. (von Neumann, McCulloch, Lashley, Kohler, Halstead and discussants.)

[D] Couffignal L. (ed.), Les Machines à Penser, Editions de Minuit, Paris, 1952.

[E] Proc. of the 1955 Western Joint Computer Conf., Session on Learning Machines (Ware W. H., Chairman), pp. 85—111, March 1—3, 1955. (Clark and Farley, Selfridge, Dinneen, Newell; discussion: Pitts, Miller.)

[F] Proc. of the Ist Intern. Congr. on Cybernetics, Namur, Belgique, June 26—29, 1956, Gauthier-Villars, Paris; 2nd conf., см. [AB].

[G] Shannon C. E., McCarthy J. (eds.), Automata studies, Annals of Mathematics Studies, vol. 34, Princeton, N. J.; русский перевод: Шеннон К. Э., Маккарти Дж. (ред.), сб. Автоматы, ИЛ, 1956.

[H] Newman (ed.), The World of Mathematics, vol. 4, Simon and Schuster, New York. (von Neumann, 1951; Poincaré, 1954; Shannon, 1950; Turing, 1950.)

[I] Cherry C. (ed.), Information theory, Proc. of the 3rd London Symposium on Information Theory, Butterworth, London; Acad. Press, New York, 1956.

[J] Tompkins H. E. et al., Symposium on the design of machines to simulate the behavior of the human brain, *IRE Trans. on Electronic Computers*, **EC-5**, 240—255 (1956). (Tompkins, moderator; McCulloch, Oettinger, Schmitt, Rochester, Pitts, Rothstein, others); русский перевод: Томпкинс Г. Э. (председатель), Симпозиум: Проектирование машин, имитирующих поведение человеческого мозга, Кибернетический сборник, вып. I, ИЛ, 1960.

[K] Jackson W. (ed.), Communication theory, Proc. of the 2nd London Symposium on Applications of Communications Theory, Butterworth, London, 1953.

[L] Carr J. W., III, Computer Programming and Artificial Intelligence: An Intensive Course for Practicing Scientists and Engineers, Univ. of Michigan, Ann Arbor: Summer Conf., College of Engineering, 1958.

[M] Liapunov A. A. (ed.), Problems of Cybernetics, JPRS 876-D, Joint Publ. Research Service, Washington; английский перевод сборников «Проблемы кибернетики» под ред. Ляпунова А. А.

[N] Blake D. V., Uttley A. M. (eds.), Proc. of the Symposium on Mechanization of Thought Processes, National Physical Lab., Teddington, England, H. M. Stationary Office, London, 1959, 2 vols.

[O] Von Foerster H. (ed.), The Realization of Biological Computers Rep. 5, Electrical Engineering Research Lab., Univ. of Illinois, Urbana, Ill., 1959.

[P] Proc. Western Joint Computer Conf., March 3—5, 1959.

[Q] Proc. Eastern Joint Computer Conf., Dec. 9—13, 1959.

[R] Proc. of the Intern. Conf. on Information Processing (ICIP), 1959, Paris: UNESCO House.

[S] Steele E. J. (ed.) (USAF Air Research and Development Command), Symposium on BIONICS, 1960 (Armer, Savely, Blum, Verbeek, Cowan, Stark, McCulloch, Schreiver, Kelly, Novikoff, Tanner, Dersch, Loebner, de Ross, Vallese, Maitra, van Bergeijk, Harmon, Reitman, Metzelaar, Lee, Ryle, Scott, Steinman, Noble, Taube, Schmitt, Steele.)

[T] Communications of the Association for Computing Machinery, vol. 3, 1960; preprints of the Conference on Symbol Manipulation Programs.

[U] Yovitts M., Cameron S. (eds.), Self-organizing Systems, Pergamon Press, New York, 1960 (Farley, von Foerster, Estes, Rosenblatt, Newell et al., Milner, Campbell, Pask, McCulloch, Burks a. o.); русский перевод см.: Самоорганизующиеся системы, изд-во «Мир», 1964.

[V] Proc. of the Western Joint Computer Conf., May 3—5, 1960.

[W] Von Foerster H. (ed.), Illinois Symposium on Principles of Self-Organization, Univ. of Illinois, Urbana, Ill., 1961 (Pask, Ashby, Rapaport, Beer, Verbeek, Blum, Cowan, Lofgren, Sperry, Beurle, Zopf, Hayek, Hawkins, Platt, Novikoff, Rosen, Willis, Rosenblatt, Greene); русский перевод: Принципы самоорганизации, изд-во «Мир», 1966.

[X] Cherry C. (ed.), Proc. of the 4th London Symposium on Information Theory, Butterworth, London; Acad. Press, New York, 1961.

[Y] Proc. IRE, Special computer issue; review articles about artificial intelligence, automatic programming, adaptive servomechanisms, adaptive networks, etc., 1961; русский перевод: *ТИРИ*, **49**, № 1 (1961).

[Z] Proc. of the Western Joint Computer Conf., May 9—11, 1961.

[AA] David E. E. (ed.), Special issue on sensory information processing, *IRE Trans. on Information Theory*, **IT-8**, № 2, 74—191 (1962).

[AB] 2nd Intern. Congr. on Cybernetics, Sept. 3—10, 1958, Association de Cybernétique, Namur, Belgique, 1962.

[AC_1] Delafresnaye J. F. (ed.), Brain Mechanisms and Consciousness, Council for International Organizations of Medical Sciences, Blackwell Sci. Publ., Oxford, 1954.

[AC_2] Delafresnaye J. F. (ed.), Brain Mechanisms and Learning, Blackwell Sci. Publ., Oxford, 1961.

[AD] Greenberger M. (ed.), Management and the Computer of the Future, Wiley, New York, 1962.

[AE] Lernende Automaten, (Symposium), Technische Hochschule Karlsruhe, Apr. 11—14, 1961, Oldenbourg, München, 1961.

[AF] Marill T. (ed.), Special issue on artificial intelligence, *IRE Trans. on Human Factors in Electronics*, **HFE-2**, № 1 (1961).

[AG] Symposium on Biosimulation, Barth Foundation, Locarno, Switzerland, June—July, 1960.

[AH] Liapunov A. A. (ed.), Problems of Cybernetics, № 1 (1958), № 2 (1959), № 3 (1960), № 4 (1960), № 5 (1961); английский перевод сборников «Проблемы кибернетики» (под ред. Ляпунова А. А.), № 1 (1958), № 2 (1959), № 3 (1960), № 4 (1960), № 5 (1961). Семинар по кибернетике в МГУ, 1959—1960; Всесоюзная конф. по вычислительной математике и технике, 1960. Содержание излагается в работе. Friedman M. D., 8th Reference Bibliography (Artificial Intelligence — Soviet Bloc), MIT Lincoln Lab. Library, Dec. 27, 1961, ASTIA MIT Hayden Library; американская оценка статей дана в работе Фейгенбаума [321].

[AI]* Шеннон К., Работы по теории информации и кибернетике, ИЛ, 1963.

[AJ_1]* Анохин П. К. (ред.), Механизмы целого мозга, сб. статей, ИЛ, 1963

[AJ_2]* Анохин П. К. (ред.), Электроника и кибернетика в биологии и медицине, ИЛ, 1963.

[AK]* Ершов А. П. (ред.), Автоматизация программирования, сб. переводов, Физматгиз, 1961.
[AL]* Матюшкин А. М. (ред.), Психология мышления, изд-во «Прогресс», 1965.
[AM]* Панов, Зинченко (ред.), Инженерная психология, изд-во «Прогресс», 1964.
[AN]* Wiener N., Schadé J. P. (eds)., Nerve, Brain and Memory Models, Progress in brain research, vol. 2, Elsevier Publ. Comp, Amsterdam—London—New York, 1963.
[AO]* Models and Analogues in Biology, Symposia of the Society for Experimental Biology, № XIV, Univ. Press, Cambridge, 1960; русский перевод: Бернштейн Н. А. (ред.), Моделирование в биологии, ИЛ, 1963.
[AP]* Yovits M. C., Jacobi G. T., Goldstein G. D. (eds.), Self-Organizing Systems, 1962, Spartan Press, Washington, 1962.
[AQ]* Петров Б. Н. (ред.), Самонастраивающиеся автоматические системы, Труды Международного симпозиума (ИФАК), изд-во «Наука», 1964. (Truxal, Поспелов, Steinbuch, Pask и др.).
[AR]* Greenberger M. (ed.), Computers and the World of the Future, MIT Press, Cambridge, Mass., 1964.
[AS]* William S., Fields M. D., Albott W. (eds.), Information Storage and Neural Control, Thomas Publ., Springfield, Ill., 1963; русский перевод: Гурфенкель В. С. (ред.), Концепция информации и биологические системы, изд-во «Мир», 1966.
[AT]* Bellman R. (ed.), Mathematical Problems in the Biological Sciences, Proc. of Symp. in Appl. Math., Am. Math. Soc., vol. 14, 1962; русский перевод: Беллман Р. (ред.), Математические проблемы в биологии, изд-во «Мир», 1966.
[AU]* Stacy R. W., Waxman B. (eds.), Computers in Biomedical Research, Acad. Press, New York, 1965, vol. II.
[AV]* Melton A. W. (ed.), Categories on Human Learning, Acad. Press, New York, 1964.

АЛФАВИТНЫЙ УКАЗАТЕЛЬ

Алгебра различий 451
Алгоритм 29, 118, 145, 442
— Британского музея 120
Амарел С. 333, 335
Анализ соотношения средств и цели 282, 450; см. также Универсальный решатель задач
Аналогия с «черным ящиком» 319
Армер П. 31, 401
Аткинсон Р. С. 303
Аттли А. М. 423
Аш С. 385

Бартлетт Ф. К. 333
Бар-Хиллел И. 235
Беллман Р. 433
Бернстейн А 45, 56, 62, 63, 433
Бихевиоризм 284
Блау П. М. 386
Бледсоу В. В. 422
Бомба Дж. С. 426
Брайтон Б. 181, 200
Браунинг И. 422
Бэйлс Р. Ф. 385

Ван Хао Г. 159, 160, 171, 441, 442
Винер Н. 410, 416
Возенкрафт Дж. 446
Вторичное подкрепление 431
Вулф А. 221
Выбор ответа 333; см. также Модель формирования понятий
Вычислительные машины, определение 23
Вюрцбургская школа 284

Галантер Е. 431, 452
Галлахорн Дж. И. 279, 385
Галлахорн Дж. Т. 279, 385
Гёделя теорема 454
Гелернтер Г. 112, 145, 165, 237, 443, 447, 448, 452
Генераторы ходов 50, 59
Гештальт-психология 284, 426
Гибсон Е. Дж. 303
Гилмор П. К. 159, 171
Гильберт Д. 150
Голомб 446
Горн С. 457
GPS, см. Универсальный решатель задач
Грамматическая индукция 455
Гримсдейл Р. Л. 426, 438
Грин Б. Ф. 220
Гудмен Н. 71
Гуднау Дж. Дж. 307

Дайнин Г. П. 307, 417
Дарлингтон К. Д. 438
Двоичные связующие члены в оценочной функции 108
Девис М. 442
Дерево вариантов 27
Джексон Дж. Р. 181, 194, 196, 197
Дойл В. 422
Доказательство теорем, см. Программа для доказательства геометрических теорем, Программа «Логик-теоретик», Программа SAINT

Забывание 316
Задача (problem) 27, 117

Игра в имитацию 171
— «Ним» 444
Инвариантные признаки 415
Ингве В. Г. 245
Индуктивный метод 454—456

Интеллект 453
Интерференция 316
Инференциальный анализ 160, 172
IPL. см. Язык для переработки информации
Искажения 411
Искусственный разум, определение 25
Использование предшествующих результатов 143, 438
Испытание на невозможность доказательства 441
— на подобие 132, 441
— на простоту 441
Исчисление высказываний 439
— предикатов 442

Каменский Л. А. 422
Кво Т. Т. 181
Келлер Г. 334
Кель В. Б. 234
Килберн Л. 438
Кирш Р. А. 422, 426
Кларк В. А. 72, 457
Кларксон Дж. П. Е. 33, 279, 337, 338, 358
Колер В. 426
Комбинирование признаков 418
COMIT 199, 458
Компаратор 405
Кохен 255, 335, 336
Крейк К. Дж. 454

Лабиринтная модель задачи 27
Линдсей Р. К 220, 233, 307
LISP 199, 204, 319, 445, 452
Ловленд Д. В. 165
Ложный (локальный) максимум 408
Лоферн К. 221
Люн Г. П. 234
Лямбда-символика 425

Магазинные списки 464
Майнот О. Н. 426
Маккаллок В. С. 72, 416
Маккарти Дж. 204, 319, 336, 445, 446, 452
Маккей Д. М. 426, 454
Макфи В. 385
Маримонт Б. Б. 200
Марковские процессы 385
Маурер О. Г. 236
Машина, доказывающая геометрические теоремы 145—175
Машина «Логик-теоретик» 113—144, 177, 183, 205, 211, 217, 333, 439—443
— чертежник 151, 152
Машинная математика 442
Машины, отвечающие на вопросы 219—254
Методика «рассуждения вслух» 286, 339, 361
Методы классификации 409
Миллер Г. А. 232, 431, 452, 455
Миллер П. М. 457
Минимаксная процедура 38, 55, 77, 433
Минский М. 166, 302, 307, 401, 402
Митчелл Дж. 182, 196, 198
Модель 411
— обучения понятиям 318—337
— процесса инвестиции 358—384
— формирования понятий 317—337
— элементарного социального поведения 385—400
Модели восприятия: см. Распознавание образов
Морфи 61
Мостеллер Ф. 41
Муэрс К. Н. 423, 443

Направленные графы 444
Нейронные сети 72
Нейссер У. 422
Нормализация (объектов) 412
Ньюэлл А. 33, 113, 166, 199, 205, 211, 279, 282, 283, 302, 307, 319, 405, 418, 424, 433, 434, 439, 442, 446, 450, 453, 458

Обобщение прошлого опыта 86
— стимула 315; см. также Программа ЕРАМ
Образ 411
Обучающая последовательность 430, 438, 443
Обучающиеся системы 426—438
Обучение 427; см. также Программа для игры в шашки; Модель обучения понятиям; Программа ЕРАМ; Распознавание образов
— с помощью запоминания позиций 80—86
Огден К. К. 246
Орчард-Хэйсс В. 452
Осгуд К. И. 236
Остин Г. А. 307
Оценка, оценочная функция 51, 74—77, 106—109, 433

Память, способная к выводам 233
Паперт С. 422
Парно-ассоциированное предъявление 304
Паск Г. 454
Патнэм Г. 442
Перенос, определение 332
Питтс В. 416
Планирование 30, 282, 432, 438
Подкрепление 427—432
Подъем по градиенту 404, 406—409
Поиск 404
— информации 234; см. также Машины, отвечающие на вопросы
Пойя Г. 146, 335, 457
Понятие дизъюнктивное 318
— конъюнктивное 318
— определение 317
Поток ассоциаций 283
Правила вывода в программе «Логик-теоретик» 115
Предсказание и ожидание 432
Предъявление с предсказанием последовательности 304
Прибрам К. 431, 452
Проблема «ум—мозг» 454
Проблемы «гештальт»-типа 426
Программа 24
— балансирования сброчного конвейера 178—204
— Бейсбол 8, 221—233
— BSAINT 227
— Гомункулус 387—399
— грамматического разбора предложения: см. Программа SAD SAM
— для игры в шахматы 33—71
— — — — — Бернстейна 45—48
— — — — — Лос-Аламосская 41, 44—45
— — — — — Ньюэлла, Шоу и Саймона 48—71
— — — — — Тьюринга 40—41
— — — — — Шеннона 39—46
— — — — шашки 71—111, 433—435
— ЕРАМ 302—317
— моделирующая поведение в эксперименте с двоичным выбором 338—358
— SAD SAM 220, 239—254
— SAINT (Символический Автоматический Интегратор) 177, 204—219
— семантического анализа, см. Программа SAD SAM
— элементарного восприятия и запоминания—см. Программа ЕРАМ
Процедура сравнения 128

Распознавание образов 216, 218, 254—272, 409—426
— речи 271
Рассел Б. 112, 116, 117, 159, 439, 441
Расчлененное описание объектов 425
Рашевский Н. 457
Ретроактивное торможение 305, 315
Решение задач (problem solving) 281—301, 438
Робертс Л. Г. 422, 436
Розенблатт Ф. 72, 331, 335, 421, 422, 436
Розенблюм 453
Рочестер Н. 72, 443, 447, 457

Саймон Г. А. 33, 113, 166, 205, 211, 279, 282, 283, 302, 307, 313, 319, 418, 424, 433, 434, 439, 442, 446, 450, 453
Салвсон М. И. 181
Самнер Ф. Г. 438
Самонаблюдение 284
Селфридж О. 254, 255, 307, 331, 335, 416—418, 419, 426, 457
Силвер Р. 445
Синтаксическая машина 151
Синтаксический анализ 225—227
Система ACSI-MATIC 232
Системы пассивной классификации 426
Скиннер Б. Ф. 428
Скрытое обучение 332
Сложные системы обработки информации 113
Случайные сети 421; см. также Нейронные сети
Слэйджл Дж. 21, 177, 204, 446
Соломонов Р. Дж. 418, 439, 455
Социальное поведение, см. Модель элементарного социального поведения
Списковая структура 239, 464
Список адресов текущих команд 466
— свободной памяти 465
— спецификационный 222
Статистическая оценка 39, 57, 59
Статистические схемы обучения 430
Стивенс М. И. 426
Стивенс С. С. 305
Стрейчи К. С. 74, 76
СТРЕТЧ 219

Схемы условной вероятности 418, 447
— — — обучающиеся 421
Сэмюэль А. Л. 33, 71, 302, 405, 430, 433, 454

Тарский А. 150
Творческая деятельность 453
Телеологические требования к классификации 410
Тинберген Н 426
Тонг Ф. 176, 178, 445, 458
Топологическая эквивалентность 413
Тьюринг А. Н. 40, 49, 53, 79, 146, 288, 454

Уайтхед А. Н. 112, 116, 117, 159, 439, 441
Унгер С. Г. 426
Универсальный решатель задач (GPS) 30, 283—301, 307, 443
Условное предсказание 346
Условные рефлексы 431
Установка 332
Уэллс Г. Г. 329

Фейгенбаум Е. А. 279, 301, 336, 400
Фельдман Дж. 279, 305, 306, 337, 338, 400
Фордер 150
Формирование признаков 416
ФОРТРАН 112, 452
Фосслер К. 255
Фреймер М. 433
Фридберг Р. М. 436—437
Фэрли Б. Г. 72, 436, 457

Хайлимен В. К. 422
Халл К. Л. 303, 305
Ханзен Дж. Р. 165
Хант И. 279, 302, 317
Харрис З. С. 226
Хебб Д. О. 426
Хейс Д. Г. 351
Хелджсон В. Б. 181
Ховленд К. И. 279, 302, 305, 317
Ходс Л. 425
Холленд Дж. Г. 426
Холлер Н. 426
Хоманс Дж. 386—397
Хомский А. Н. 239, 455
Хоукинс В К. 422
Хэйк Ф. А. 454, 457

Цель, определение 450; см. также Универсальный решатель задач; Машина для доказательства геометрических теорем; Программа SAINT

Чёрч А. 317

Шеннон К. 37, 49, 62, 74, 405, 433, 444
Шерман Г. 426
Шоу Дж. К. 33, 113, 166, 205, 211, 282, 302, 307, 319, 418, 424, 433, 434, 439, 442, 446, 450, 453
Шубик М. 430, 457

Эвристика, определение 29, 113, 119, 146, 148, 182—184, 205, 404, 443
Эвристическая машина 151—156
— связь 406
Эвристические методы для выбора подзадач 443—446
Эвристически эффективное определение 410
Эквивалентные преобразования 413
Эксперимент с двоичным выбором 339
Эшби В. Р. 405

Юр Л. 255

Язык для переработки информации (IPL) 459
— — — — IPL-IV 63, 199, 459
— — — — IPL-V 199, 200, 221, 314, 388, 399, 458
— — работы с символами, см. ФОРТРАН; LISP
— ФОТРАН для обработки списков. 112, 167, 452
Языки 455

СОДЕРЖАНИЕ

Вступительная статья 5
Предисловие к английскому изданию *(перевод А. Л. Чернявского)* . . 19

Часть I

ИСКУССТВЕННЫЙ РАЗУМ

Введение. Э. Фейгенбаум и Дж. Фельдман *(перевод Н. В. Завалишина)* 23

Раздел 1. **Играющие машины** 32
1. Программа для игры в шахматы и проблема сложности. А. Ньюэлл, Дж Шоу и Г. Саймон *(перевод А. Л. Чернявского)* 33
2. Некоторые исследования возможности обучения машин на примере игры в шашки. А Сэмюэль *(перевод А. Л. Чернявского)* 71

Раздел 2. **Машины для доказательства математических теорем** . . 111
3. Эмпирические исследования машины «Логик-теоретик»; пример изучения эвристики. А. Ньюэлл, Дж. Шоу и Г. Саймон *(перевод А. А. Дорофеюка)* 113
4. Реализация машины, доказывающей геометрические теоремы. Г. Гелернтер *(перевод А. А. Дорофеюка)* 145
5. Экспериментальное исследование машины для доказательства геометрических теорем Г. Гелернтер, Дж Ханзен и Д. Ловленд *(перевод А. А. Дорофеюка)* 165

Раздел 3. **Две важные прикладные задачи** 175
6. Эвристические методы балансирования работы сборочного конвейера. Ф Тонг *(перевод Б. А. Дашевского)* 178
7 Эвристическая программа, решающая задачи символического интегрирования в объеме первого курса университета. Д. Слэйджл *(перевод А. А. Дорофеюка)* 204

Раздел 4. **Машины, отвечающие на вопросы** 219
8. «Бейсбол» — программа, автоматически отвечающая на вопросы. Б. Грин мл., А. Вулф, К. Чомский и К. Лофери *(перевод Н. В. Завалишина)* 221
9. Память, способная к выводам, как основа машин, понимающих человеческий язык. Р. Линдсей *(перевод М. Е. Киссина и В Е. Божкова)* 233

Раздел 5. **Распознавание образов** 254
10. Программа распознавания образов, которая вырабатывает, оценивает и улучшает свои операторы. Л. Юр и Ч. Фосслер *(перевод А. А. Дорофеюка)* 255

Часть II

МОДЕЛИРОВАНИЕ ПРОЦЕССОВ ПОЗНАНИЯ

Введение. Э. Фейгенбаум и Дж. Фельдман *(перевод М. Е. Киссина и В. Е. Божкова)* 273

Раздел 1. **Решение задач** 281
11. GPS — программа, моделирующая процесс человеческого мышления. А. Ньюэлл и Г. Саймон *(перевод Н. А. Чичвариной)* 283

Раздел 2. **Вербальное обучение и обучение понятиям** 301
12. Моделирование вербального обучения. Э. Фейгенбаум (*перевод Г. М. Хованова*) 302
13. Машинная модель формирования человеческих понятий. Э. Хант и К. Ховленд (*перевод М. Е. Киссина* и *В. Е. Божкова*) 317

Раздел 3. **Принятие решения в условиях неопределенности** 337
14. Моделирование поведения в эксперименте с двоичным выбором. Дж. Фельдман (*перевод М. Е. Киссина* и *В. Е. Божкова*) 338
15. Модель процесса доверительной инвестиции. Дж. Кларксон (*перевод А. Л. Чернявского*) 358

Раздел 4. **Социальное поведение** 384
16. Машинная модель элементарного социального поведения. Дж. Галлахорн и Дж. Галлахорн (*перевод М. Е. Киссина* и *В. Е. Божкова*) 385

Часть III

ОБЗОР ПОДХОДОВ И МЕТОДОВ

17. На пути к созданию искусственного разума. М. Минский (*перевод В. Н. Оксеня*) 402
Приложение: Краткое описание языка IPL-V. А. Ньюэлл и Ф. Тонг (*перевод А. Л. Чернявского*) 458

Часть IV

БИБЛИОГРАФИЯ РАБОТ ПО ИСКУССТВЕННОМУ РАЗУМУ

М. Минский (*перевод В. Н. Оксеня*)
Введение 475
Тематический указатель к библиографии 479
Библиография 491
Алфавитный указатель 547

ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ И МЫШЛЕНИЕ

Редактор В. Я. Фридман Художник С. А. Бычков

Художественный редактор Н. А. Фильчагина

Технический редактор В. П. Сизова Корректор М. А. Смирнов

Сдано в производство 28/XII 1966 г. Подписано к печати 18/V 1967 г. Бумага 60×90¹/₁₆ Бум. л. 17,25. Печ. л. 34,5. Уч.-изд. л. 36,01. Изд. № 20/3651 Цена 2 р. 77 к. Зак. № 9
ИЗДАТЕЛЬСТВО «МИР»
Москва, 1-й Рижский пер., 2

Ленинградская типография № 8 Главполиграфпрома
Комитета по печати при Совете Министров СССР.
Ленинград, Прачечный пер., 6