Лабораторная работа №1
ВВЕДЕНИЕ

1.1. РЕШЕНИЕ ЗАДАЧ И ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

Многие виды деятельности человека, такие, как решение головоломок, участие в играх, занятия математикой и даже вождение автомобиля, требуют, как это принято считать, участия «интеллекта». Если бы вычислительные машины могли справляться с деятельностью такого типа, то они (вместе с их программами), вероятно, обладали бы в какой-то степени искусственным интеллектом. Многие специалисты полагают, что в конечном итоге искусственный интеллект вычислительных машин превзойдет интеллект человека, хотя теперь все больше и больше осознается тот факт, что процессы, требуемые для выполнения даже самых обычных для человека задач, неизбежно будут чрезвычайно сложными. В настоящей книге мы подробно исследуем некоторые процессы, связанные с *решением задач,* в которых участвует интеллект.

Решение задач может показаться весьма неясным предметом, и тем не менее на нем концентрируется большая доля исследований по искусственному интеллекту. В самом широком смысле этих слов нахождение решений включает в себя всю вычислительную науку, поскольку всякая вычислительная задача может рассматриваться как задача, решение которой надо найти. Однако для наших целей нужно более узкое определение, которое исключает такие стандартные вычислительные методы, как методы, используемые, скажем, при обращении матрицы 50-го порядка или при решении системы линейных дифференциальных уравнений.

Если мы внимательно рассмотрим методы нахождения решений, изучаемые в исследованиях по искусственному интеллекту, то обнаружим, что в большинстве из них используется понятие поиска путем проб и ошибок. Это значит, что в этих методах задачи решаются посредством поиска решения в пространстве возможных решений. Наша цель состоит в разъяснении наиболее важных методов решения задач с использованием процедур поиска.

Имеются, конечно, и другие важные направления в изучении искусственного интеллекта. Типичные представители тех из них, которым было уделено особое внимание (кроме нахождения решений), следующие:

Обработка сенсорных данных (особенно зрительных образов и речи).

Сложные системы хранения и извлечения информации.

Обработка естественных языков.

К сожалению, никто еще не мог сказать ничего достаточно полезного относительно того, как названные элементы могли бы быть объединены вместе в одном общем «интеллекте» (каком бы то ни было). В действительности при внимательном анализе становится ясно, что любая из предполагаемых «фундаментальных» компонент интеллектуального поведения содержит по-видимому, в себе черты других фундаментальных компонент. Так, для сенсорного восприятий могут потребоваться весьма изощренные способы выбора решения, для которых в свою очередь возникает необходимость в достаточно эффективной системе извлечения информации, опирающейся, возможно, на дополнительный выбор решений и т.д. Наш опыт работы с этими сложными процессами все еще недостаточен для создания единой теории организации интеллекта. На самом деле в настоящее время нет никаких оснований полагать, что такая теория вообще могла бы существовать. Некоторые исследователи считают, что интеллектуальное поведение может быть получено на вычислительных машинах только посредством комбинирования специализированных программ, каждая, из которых содержит множество подходящих к данному случаю решений (или, как их часто называют, «программистских находок»), с возможностью обращения к магазину энциклопедических сведений, содержащему хорошо систематизированные факты. Однако сейчас нам не хотелось бы занимать определенную позицию по этому вопросу. Вместо этого мы опишем те приемы решения задач, которые, по-видимому, имеют достаточно широкую область применения.

1.2. ГОЛОВОЛОМКИ И ИГРЫ КАК ПРИМЕРЫ ЗАДАЧ

Мы еще не давали точного определения, что значит решить некоторую задачу с применением методов поиска. Точно так же мы не определили, что мы понимаем под задачей. По всей видимости, еще никем не было дано такого простого определения слова «задача», которое полностью бы соответствовало тому интуитивному значению, которое мы намереваемся здесь

использовать. Поэтому вместо того, чтобы пытаться дать формальное определение, мы начнем наше обсуждение с рассмотрения типичного примера задачи.

Головоломки и игры представляют собой неисчерпаемый источник примеров, полезных для иллюстрации и испытания методов решения задач. Для вычислительных машин были написаны программы решения многих видов головоломок, достаточно трудных для человека. Были написаны также другие программы, которые побеждали опытных игроков в настольные игры, такие, как шахматы и шашки. Как говорит Минский (1968, стр. 12): «Игры и математические задачи берутся не потому, что они просты и ясны, а потому, что они *при минимальных начальных структурах дают нам наибольшую сложность,* так что мы можем заняться некоторыми действительно трудными ситуациями, относительно мало отвлекаясь на вопросы программирования». В игровых задачах и решениях головоломок возникли и отшлифовались многие идеи, которые оказались по-настоящему полезными для менее легкомысленных задач.

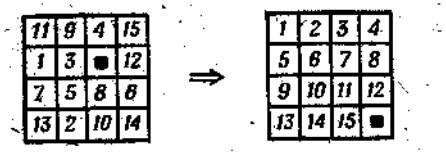


Рис. 1.1. Игра в пятнадцать (слева - начальная конфигурация, справа - целевая).

Для иллюстрации понятий, возникающих при решении задач, мы часто будем пользоваться головоломкой; известной игра в пятнадцать. В ней используется пятнадцать пронумерованных подвижных фишек, расположенных на площадке размером 4X4 клетки. Одна клетка этой площадки остается всегда пустой, так что всегда одну из соседних с ней фишек; передвинуть на место этой пустой клетки, «передвинув», таким образом, и эту пустую клетку. Игра в пятнадцать иллюстрируется на рис. 1.1, на котором изображены две конфигурации фишек. Рассмотрим задачу перевода начальной конфигурации заданную целевую конфигурацию. Решением этой служит подходящая последовательность ходов, такая, мер, как «передвинуть фишку 12 влево, фишку 15 вниз и т.д.

Игра в пятнадцать - замечательный пример одного класса задач, для которого лучше всего приспособлены методы излагаемые далее. В этой задаче имеется точно определенная начальная ситуация и точно определенная цель. Имеется также некоторое множество операций, или ходов, переводящих одну ситуацию в другую. Мы начнем с введения, некоторых фундаментальных понятий, связанных с нахождением которые могут быть использованы для нахождения решения игры в пятнадцать.

1.3. СОСТОЯНИЯ И ОПЕРАТОРЫ

По-видимому, самый прямолинейный подход при поиске решения для игры в пятнадцать состоит в попытке перепробовать различные ходы, пока не удастся получить целевую конфигурацию. Такого рода попытка по существу связана с поиском при помощи проб и ошибок. (Мы, разумеется, предполагаем, что такой поиск может быть выполнен в принципе, скажем, на некоторой вычислительной машине, а не с привлечением реальной игры в пятнадцать). Отправляясь от начальной конфигурации, мы могли бы построить все конфигурации, возникающие в результате выполнения каждого из возможных ходов, затем построить следующее множество конфигураций после применения следующего хода и т. д., пока не будет достигнута целевая конфигурация.

Для обсуждения такого сорта методов поиска решения оказывается полезным введение понятий *состояний* и *операторов* для данной задачи. Для игры в пятнадцать состояние задачи - это просто некоторое конкретное расположение фишек. Начальная и целевая конфигурации представляют собой соответственно начальное и целевое состояния. *Пространство* состояний, достижимых из начального состояния, состоит из всех тех конфигураций фишек, которые могут быть образованы в результате допустимых правилами перемещении фишек. Многие из задач, с которыми мы будем сталкиваться, имеют чрезвычайно большие (если не бесконечные) пространства состояний (*в игре в пятнадцать имеется 16 различных конфигураций из фишек и пустой клетки, половина из них (или примерно 10,5 10^{12}) достижима из данной начальной конфигурации*).

Оператор преобразует одно состояние в другое Игру в пятнадцать естественнее всего интерпретировать как игру, имеющую четыре оператора, соответствующие следующим ходам передвинуть пустую клетку (пробел) влево, вверх, вправо и вниз. В некоторых случаях оператор может оказаться неприложимым к какому-то

состоянию: так, оператор «передвинуть пробел вправо» не может быть применен к целевому состоянию на рис. 1.1. На нашем языке состояний и операторов решение некоторой проблемы есть последовательность операторов, которая преобразует начальное состояние в целевое.

Пространство состояний, достижимых из данного начального состояния, полезно представлять себе в виде графа, вершины которого соответствуют этим состояниям. Вершины такого графа связаны между собой дугами, отвечающими операторам. На рис. 1.2 показана небольшая часть графа для игры в пятнадцать. На этом графе в каждой вершине помещена та конфигурация фишек, которую она представляет.

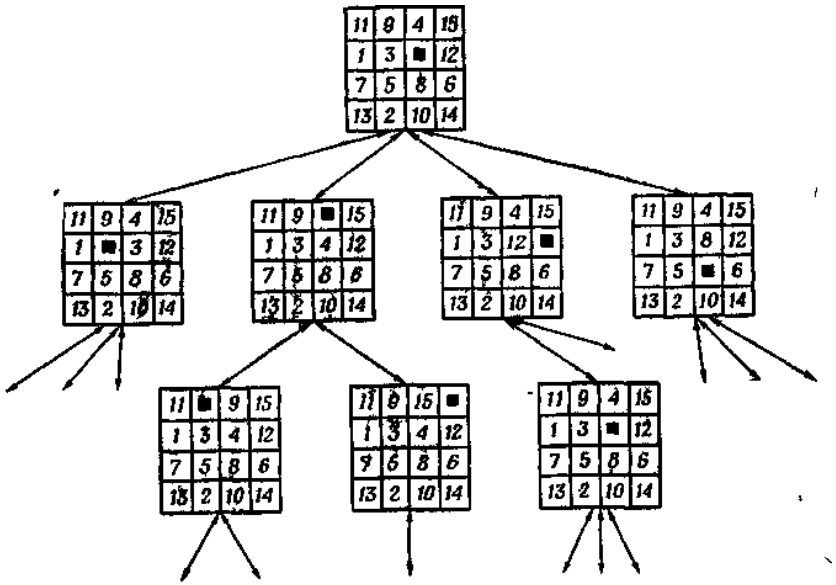


Рис. 1.2. Часть графа для игры в пятнадцать.

Решение игры в пятнадцать можно было бы получить, используя процесс поиска (перебора) '), при котором прежде всего применяют операторы к начальному состоянию, с тем чтобы получить новые состояния, к которым также применяют операторы, и т. д. до тех пор, пока не будет построено состояние, отвечающее цели Методы организации такого поиска целевого состояния удобнее всего объяснять, пользуясь представлением в виде графа рис. 1.2.

Про метод решения задач, основанный на понятиях состояний и операторов, можно было бы сказать, что это подход к задаче с точки зрения *пространства состояний.* В общем случае мы будем связывать последний термин с методами, в которых опробываемые последовательности операторов строят постепенно, отправляясь от некоторого начального оператора и добавляя затем каждый раз по одному оператору до тех пор, пока не будет достигнуто целевое состояние.

1.4. СВЕДЕНИЕ ЗАДАЧ ПОДЗАДАЧАМ

В некотором смысле более тонкий подход к решению задачи связан с понятием *подзадач.* При таком подходе производится Исследований исходной задачи с целью выделения такого множества подзадач, чтобы решение некоторого определенного подмножества этих подзадач содержало в себе решение исходной задачи. Рассмотрим, например, задачу о проезде на автомобиле из Пала-Альто (шт. Калифорния) в Кембридж (шт. Массачусетс). Эта задача может быть сведена, скажем, к следующим подзадачам:

Подзадача 1. Проехать из Пало-Альто в Сан-Франциско.

Подзадача 2. Проехать из Сан-Франциско в Чикаго, шт. Иллинойс.

Подзадача 3. Проехать из Чикаго в Олбани, шт. Нью-Йорк.

Подзадача 4. Проехать из Олбани в Кембридж.

Здесь решение всех четырех подзадач обеспечило бы некоторое решение первоначальной задачи.

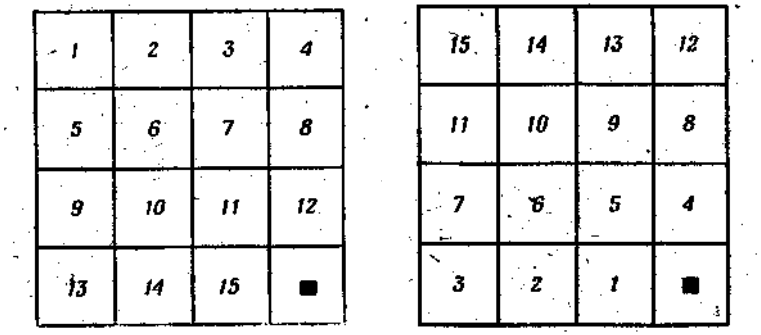
Каждая из подзадач может быть решена с применением какого-либо метода. К ним могут быть применены методы, использующие пространство состояний, или же их можно проанализировать с целью выделения для каждой своих подзадач и т.д. Если продолжить процесс разбиения возникающих подзадач на ещё более мелкие, то, в конце концов, мы придем к некоторым *элементарным* задачам, решение которых может считаться тривиальным. Про всякий метод решения задачи путем выработки и последующего решения подзадач мы будем говорить, что в нем используется подход, основанный на *редукции задачи.* Заметим, что строго говоря, подход с использованием пространства состояний можно рассматривать как вырожденный случай подхода, основанного на редукции, задачи, ибо каждое применение оператора сводит задачу к несколько более простой подзадаче как правило, мы будем иметь дело со случаями, когда подзадачи, возникающие при редукции, получаются не столь тривиальным

образом. Важно отметить, что поиск методом проб и ошибок по-прежнему играет важную роль в подходе, основанном на редукции.

На каждом из этапов может возникнуть несколько альтернативных множеств подзадач, к которым может быть сведена данная задача. Так как некоторые из этих множеств в конечном итоге, возможно, и не приведут к окончательному решению задачи, то, как правило, для решения первоначальной задачи необходим поиск в пространстве множеств подзадач.

1.5. ИСПОЛЬЗОВАНИЕ ФОРМАЛЬНОЙ ЛОГИКИ ПРИ РЕШЕНИИ ЗАДАЧ

Часто для решения задачи либо требуется проведение логического анализа в определенном объеме, либо поиск решения существенно облегчается после такого анализа. Иногда такой анализ показывает, что определенные проблемы неразрешимы.



В игре в пятнадцать, например, можно доказать, что целевая: конфигурация на рисунке слева не может быть получена из начальной конфигурации на рисунке справа (*так как в этой игре множество всевозможных конфигураций может быть разбито на два непересекающихся подмножества A и B, причем никакой элемент подмножества A не может быть преобразован в элемент подмножества B и обратно*).

Необходимость делать логические заключения возникает как в подходах, основанных на использовании пространства состояний, так и в подходах, связанных с редукцией. В подходах первого типа логических выводов может потребовать тот тест, с помощью которого определяется, будет ли которое состояние состоянием, отвечающим поставленной Кроме того, логические умозаключения могут понадобиться определении, какой из операторов применим к данному состоянию.

Как мы уже видели, иногда можно доказать, что некоторая задача неразрешима. В подходах, основанных на редукции задачи, доказательство такого рода позволило бы избежать тщетных попыток разрешить неразрешимые подзадачи. В дополнение к таким приложениям мы хотим также иметь возможность решать задачи, которые представляют собой задачи на доказательство. Например, возможно, мы захотим найти доказательство некоторой математической теоремы, записанной в определенной формальной системе, такой, как исчисление предикатов первого порядка.

Таким образом, полное исследование приемов решения задач должно включать рассмотрение машинных методов поиска доказательства. Некоторые из этих методов опираются на стратегии поиска, подобные тем, которые мы будем обсуждать в связи с подходами, основанными на пространстве состояний и редукции задач. Хотя известно много способов выбора конкретного логического формализма, мы будем рассматривать разработанную в последнее время методику доказательства теорем в исчислении предикатов первого порядка, основанную на принципе резольвенций, и применения такой методики к решению задач.

При обсуждении автоматического доказательства теорем мы покажем, что даже нематематические задачи могут быть сформулированы как теоремы, подлежащие доказательству. Многие из головоломок, которые мы рассмотрим, так же как и многие возникающие в реальной действительности задачи, требующие для их анализа здравого смысла, могут быть в принципе сформулированы в рамках определенного логического формализма и после этого решены методом доказательства теорем. Использование формальной логики и методов доказательства теорем позволяют нам думать о действительно «универсальном» решателе задач. Новая информация в такой решатель задач могла бы вводиться просто в форме внесения в его память новых дополнительных аксиом, а не посредством переделывания его программы Он мог бы решать задачи из достаточно широких областей, поскольку существуют логические формализмы, достаточно универсальные для того, чтобы выразить любую информацию и записать любую задачу.

1.6. ДВА СОСТАВНЫХ ЭЛЕМЕНТА ПРОЦЕССА РЕШЕНИЯ ЗАДАЧ: ПРЕДСТАВЛЕНИЕ И ПОИСК (ПЕРЕБОР)

В каждом из подходов к решению задач, о которых мы говорили,

для построения решения необходим поиск какого-либо типа. Эта книга написана главным образом о том, как проводить такой поиск настолько эффективно, насколько это возможно. Но прежде чем такой процесс поиска может быть начат, сама задача должна быть поставлена либо в рамках подхода, основанного на пространстве состояний или на редукции к подзадачам, либо же как теорема, подлежащая доказательству. Обычно при решении человеком той или иной задачи мы восхищаемся не быстрым и упорядоченным поиском в пространстве всевозможных решений, а умением найти такую ясную точку зрения на рассматриваемую задачу, которая делает решение элегантно простым

Мы еще обсудим вопрос о постановке и представлении задачи в такой форме, чтобы ее можно было решать методом, основанным на рассмотрении пространства состояний. Мы увидим, что существует несколько вариантов представлений для одной и той же задачи, причем некоторые представления дают намного более узкие пространства состояний, чем другие. Так как даже самые эффективные методы поиска будут непригодны, если пространство, в котором ведется поиск, слишком велико, то важно уметь представлять задачу самым экономным из возможных способов. Вопрос о выборе представления - общий для любого способа решения задач, но, к сожалению, в исследованиях по искусственному интеллекту еще не выработано универсального автоматического метода для нахождения искусных формулировок задач. Поэтому, несмотря на то, что имеется два аспекта в автоматическом решении задач, а именно представление и поиск, в настоящей книге мы вынуждены ограничиться рассмотрением главным образом вопросов поиска.

Лабораторная работа № 2

ПРЕДСТАВЛЕНИЕ ЗАДАЧ В ПРОСТРАНСТВЕ СОСТОЯНИИ

Цель работы: научится методике представления алгебраического выражения в виде графа и префиксных операторов выражения. Использование правил переписывания для упрощения выражений.

2.1. ОПИСАНИЯ СОСТОЯНИЙ

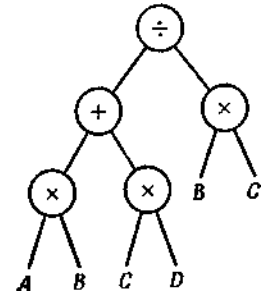
В предыдущей главе мы ввели понятия состояний и операторов. Здесь мы займемся детальной разработкой этих идей и дадим несколько примеров формулировки задач в терминах пространства состояний.

Чтобы построить описание задачи с использованием пространства состояний, мы должны иметь определенное представление о том, что представляют собой состояния в этой задаче. В игре в пятнадцать выбор в качестве состояний различных конфигураций из фишек был достаточно очевидным. Но процесс решения задачи, в котором решение ищется без реального перемещения настоящих фишек, может работать лишь *с описанием* конфигураций, а не с самими конфигурациями. Таким образом, важным этапом построения какого-либо описания задачи с использованием пространства состояний является выбор некоторой конкретной *формы* описания состояний этой задачи.

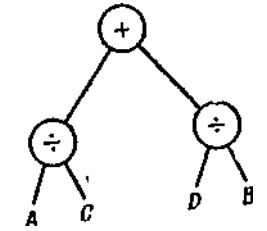
В сущности любая структура величин может быть использована для описания состояний. Это могут быть строки символов, векторы, двумерные массивы, деревья и списки. Часто выбираемая форма описания имеет сходство с некоторым физическим свойством решаемой задачи. Так, в игре в пятнадцать естественной формой описания состояний может быть массив 4X4. Выбирая форму описания состояний, нужно позаботиться и о том, чтобы применение оператора, преобразующего одно описание состояния в другое, оказалось бы достаточно легким.

Проиллюстрируем выбор формы описания состояний на простом примере. Рассмотрим задачу преобразования алгебраического выражения *(AB + CD)/BC* в более простое выражение *A/C + D/B*. Очевидно, что в качестве состояний задачи здесь должны выступать алгебраические выражения, но необходимо еще принять решение относительно формы описания состояний.

В широко используемом описании употребляются двоичные деревья. Не концевые вершины в таком дереве описания представляют арифметические знаки (+ , - , × , ÷), а концевые вершины представляют переменные или постоянные символы *(A, B, C, D),* появляющиеся в этом выражении. Таким образом, деревом представления для выражения *(AB + CD)/BC* должно быть



Здесь ветвь, отходящая от вершины ÷ влево, представляет числитель дроби, а ветвь, отходящая вправо, - ее знаменатель. Применение законов алгебраических преобразований (операторов в пространстве состояний) привело бы к преобразованию этого описания в другие описания. Наша задача состоит в преобразовании его в состояние, описываемое деревом



Другой известной формой описания служит линейная строка. Возможное описание, для выражения *(AB* + CD) */BC* в виде строки такое ÷ + × *AB* × *CD* × *BC*. Здесь арифметические операторы (÷, + и ×) называются *префиксными операторами,* поскольку они предшествуют в этой строке своим операндам. Так как нам известно, что каждый из этих операторов относится ровно к двум операндам, то в этой строке нет необходимости в пунктуации. Операндами для символа + в этой строке, например, должны быть две идущие непосредственно друг за другом подстроки, которые представляют собой алгебраические выражения × *AB* и × *CD*. Используя описание в форме строки, можно сформулировать стоящую перед нами проблему как задачу преобразования строки ÷ + × *AB* × × *CD* × *BC* в строку + ÷ *AC* ÷ *DB*.

Задание 1: Представьте в видах графа и префиксных операторов выражения: (задание у преподавателя).

2.2. ОПЕРАТОРЫ

Операторы переводят одно состояние в другое. Таким образом, их можно рассматривать как функции, определенные на множестве состояний и принимающие значения из этого множества. Так как наши процессы решения задач основаны на работе с описаниями состояний, то мы будем предполагать, что операторы суть функции этих описаний, а их значения суть новые описания. Мы могли бы, конечно, определять наши операторные функции с помощью таблицы, связывающей с каждым «входным» описанием состояния некоторое «выходное» описание. Для больших задач такая таблица была бы практически непригодной, поэтому в общем случае мы будем предполагать, что операторы - это *вычисления,* преобразующие одни описания состояний в другие.

Для описаний состояний в форме строки имеется очень удобный способ представления операторных вычислений. Он основан на идее *правил переписывания* (называемых иногда продукциями (productions)).

Множество правил переписывания определяет возможные способы преобразования одной строки в другую. Все правила переписывания имеют форму $S_i \rightarrow S_j$, означающую, что строка S_i может быть преобразована в строку S_j. Пример правила переписывания таков:

$$A\$ \rightarrow B\$.$$

Оно означает, что если символ *A* появляется в качестве первого символа некоторой строки, то он может быть заменен на символ B. Знак $ - произвольная подстрока (включая пустую строку). В приведенном правиле переписывания знак $ указывает, что часть строки (какова бы она ни была), идущая непосредственно за *A,* не изменяется, когда *A* заменяется на *B*. Для указания нескольких различных подстрок в правилах переписывания может быть использовано несколько знаков $. Так мы получаем следующие примеры возможных правил переписывания:

1. $A\$A \rightarrow A$ (строка, начинающаяся и кончающаяся символом *A,* может быть заменена одиночным символом *A*).

2. $\$_1BAB\$_2 \rightarrow \$_1BB\$_2$ (одиночный символ *A,* стоящий между двумя символами B, можно исключить).

3. $\$_1\$_2\$_3 \rightarrow \$_1\$_2\$_2\$_3$ (каждая подстрока может быть повторена).

4. $\$_1\$_2\$_2\$_3 \rightarrow \$_1\$_2\$_3$ (одна из двух стоящих рядом одинаковых подстрок может быть исключена).

Используя, например, два последних правила, строку *ABCBABC* можно преобразовать в строку ABC следующим образом:

ABCBABC (3) → ABABCBABC(4) → ABABC(4) →ABC

(в скобках приписано правило, на основании которого происходит замена).

Правило переписывания часто может применяться к некоторой строке несколькими различными способами. Так, в приведенном примере правило *4* к строке *ABCBABC* вообще не может быть применено, тогда как правило 3 может применяться к ней несколькими способами. Конечно, определенному оператору отвечает некоторое *конкретное* применение данного правила переписывания. Поскольку одно правило переписывания может представлять много различных операторов, то правила переписывания находят широкое применение при решении задач.

Представление операторов с помощью правил переписывания не должно быть обязательно ограничено ситуациями, в которых состояния описываются строками. Аналогичные идеи могут быть использованы, например, для игры в пятнадцать, в которой естественным описанием состояний служит массив 4X4. Проиллюстрируем это обобщенное понятие правил переписывания на примере игры в восемь — упрощенном варианте игры в пятнадцать. В этой игре восемь пронумерованных фишек расположены на площадке размером 3X3.

Один из способов представления допустимых ходов в этой игре состоит в задании множества правил переписывания, определенных над массивами. Эти правила определяют пути, по которым массивы 3X3 могут быть преобразованы в другие массивы того же размера. На рис. 2.1 изображено множество правил переписывания для игры в восемь. Каждое правило представляет собой в действительности два правила (как это показано двунаправленными стрелками), объединенных для экономии места в одно; в каждом случае левая запись может быть заменена правой и обратно. В каждом из правил переписывания допустимый ход определяется путем подстановки

чисел 1,2, ..., 8 на место переменных $X_1, X_2, \ldots, X_8$, по каждую сторону от стрелки (при условии $X_i \neq X_j$).

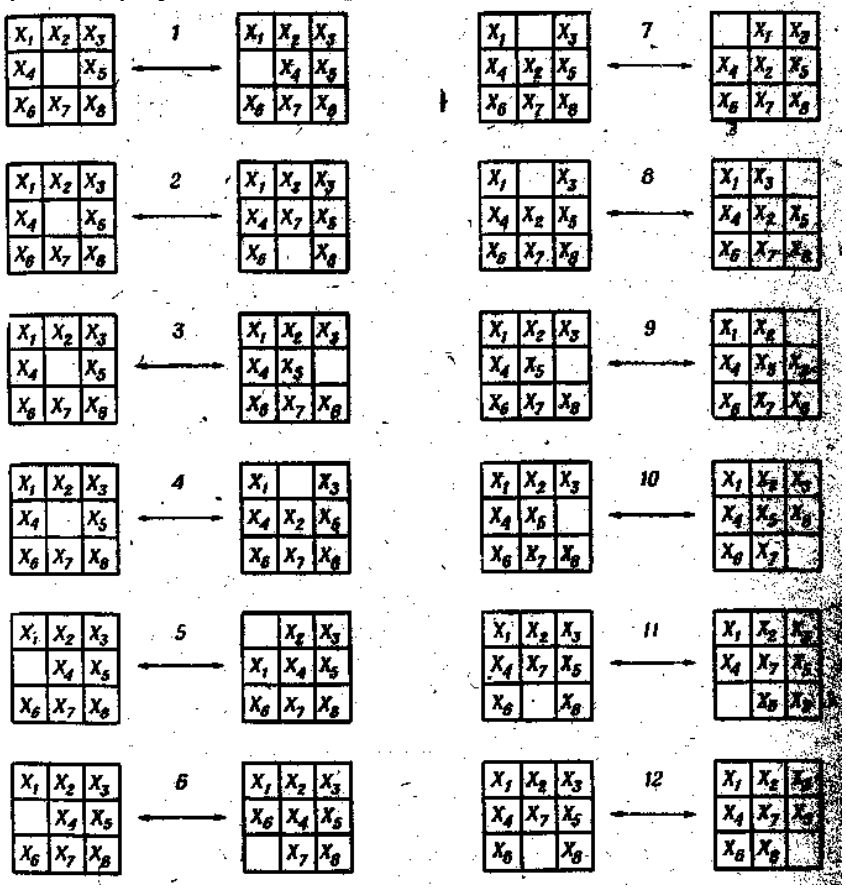
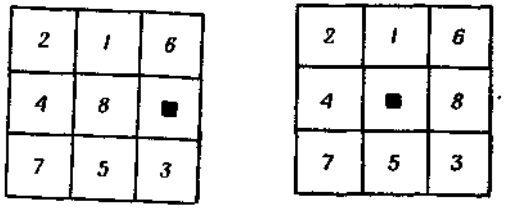


Рис. 2.1. Правила переписывания для игры в восемь.

Так, рисунок слева может быть преобразован в рисунок справа посредством применения правила 3.



Задание2: Упростите выражение … с помощью правила 1, а затем - 2. (задание у преподавателя).

2.3. *ЦЕЛЕВЫЕ СОСТОЯНИЯ*

Во все наши процедуры исследований входит построение новых описаний пространства состояний входит построение новых описаний состояний исходя из старых с последующей проверкой новых описаний состояний, с тем чтобы убедиться, не описывают ли они состояние, отвечающее поставленной цели. Часто это просто проверка того, соответствует ли некоторое описание состояния данному целевому описанию состояния, но иногда должна быть произведена более сложная проверка. Например, для игры в пятнадцать целью может быть создание конфигурации из фишек, в которой в верхних двух рядах не будет фишек с номерами, превосходящими 12. Во всяком случае, то *свойство,* которому должно удовлетворять описание состояния, для того чтобы это состояние было целевым, должно быть охарактеризовано исчерпывающим образом.

В некоторых задачах оптимизации недостаточно найти любой путь, ведущий к цели, а необходимо найти путь, оптимизирующий некоторый критерий (например, минимизирующий число применений операторов). С такими задачами проще всего работать, сделав так, чтобы поиск не оканчивался до тех пор, пока не будет найдено некоторое оптимальное решение. Методы поиска в пространстве состояний, рассматриваемые в следующей главе, позволяют получить оптимальное решение.

Таким образом, мы видим, что для полного представления задачи в пространстве состояний необходимо задать: а) форму описания состояний и, в частности, описание начального состояния, б) множество операторов и их воздействий на описания состояний, в) свойства описания целевого состояния.

Мы уже отмечали, что пространство состояний полезно представлять себе в виде направленного графа. Такое представление особенно полезно для исследования различных методов поиска в пространстве состояний. В следующем разделе мы приведем некоторые необходимые сведения из теории графов.

2.4. ЗАПИСЬ В ВИДЕ ГРАФА

В гл. 1 мы использовали граф с целью иллюстрации пространства состояний для игры в пятнадцать. До сих пор наше рассмотрение графов носило интуитивный характер, а в настоящем разделе будут введены некоторые полезные формальные понятия, относящиеся к графам.

Граф состоит из множества (не обязательно конечного) *вершин.* Некоторые пары вершин соединены с помощью *дуг,* и эти дуги направлены от одного члена этой пары к другому. Такие графы носят название *направленных графов.* Если некоторая дуга направлена от вершины n_i к вершине n_j, то говорят, что вершина n_i является *дочерней вершиной* для вершины n_j, а вершина n_i является *родительской вершиной* для n_j. Может оказаться, что некие две вершины будут дочерними друг для друга; в этом случае пара направленных дуг называется иногда *ребром* графа. В случае когда граф используется для представления пространства состояний, с *его* вершинами связывают описания состояний, а с его дугами - операторы.

Последовательность вершин n_{i1}, n_{i2}, ... , n_{ik}, в которой каждая вершина n_{ij} дочерняя для $n_{i,\,j-1}$, $j = 2, ..., k$, называется *путем* длины k от вершины n_{i1} к вершине n_{ik}. Если существует путь, ведущий от вершины n_i к вершине n_j,то вершину n_j называют *достижимой* из вершины n_i или *потомком* вершины n_{i1}.

В этом случае вершина n_i - называется также *предком* для вершины n_j. Видно, что проблема нахождения последовательности операторов преобразующих одно состояние в другое, эквивалентна задаче поиска пути на графе.

Часто бывает удобным приписывать, дугам графа стоимости, отражающие стоимость применения соответствующего оператора. Мы будем использовать запись $c(n_i , n_j)$ для обозначения стоимости дуги, направленной из вершины n_i в n_j. Стоимость пути между двумя вершинами определяется так, как сумма стоимостей всех дуг, соединяющих вершины этого пути. В задачах оптимизации возникнет необходимость найти путь между двумя вершинами, имеющий минимальную стоимость. В задачах простейшего типа нам необходимо найти путь (возможно, имеющий минимальную стоимость) между заданной вершиной s (представляющей начальное состояние) и другой заданной вершиной t (представляющей целевое состояние).

Два, очевидных усложнения этой простейшей задачи следующие:

Найти путь между вершиной s и *любым* элементом множества вершин $\{t_i\}$.

Найти путь между любым элементом множества $\{s_i\}$ и любым элементом множества $\{t_i\}$.

Множество $\{t_i\}$, называемое *целевым множеством,* не должно

быть обязательно задано явным образом. Оно может определяться неявно через свойства, которыми обладают описания соответствующих состояний, отвечающих цели.

Граф может быть задан как явным образом, так и неявным. При явном задании его вершины и дуги (с соответствующими стоимостями) должны быть перечислены явным образом, скажем, в виде некоторой таблицы. Эта таблица может содержать перечень всех вершин графа, их дочерних вершин и стоимостей всех связанных с ними дуг. Очевидно, что явное задание оказывается практически неприемлемым для больших графов, а для графов, имеющих бесконечное число вершин, оно невозможно.

При неявном способе задания определяется некоторое конечное множество $\{s_i\}$ вершин, являющихся *начальными* вершинами. Кроме того, определяется оператор Г, который, будучи примененным к любой вершине, дает *все* ее дочерние вершины и стоимости соответствующих дуг. (В нашей терминологии пространства состояний этот оператор «наследования» определяется как множество операторов, применимых к данному описанию состояния.) Последовательное применение оператора Г к элементам множества $\{s_i\}$, к их дочерним элементам и так до бесконечности дает, таким образом, представление для графа, определенного неявно через Г и $\{s_i\}$.

При этом процесс поиска в пространстве состояний той последовательности операторов, которая решает задачу, соответствует преобразованию в явную форму достаточно большой части неявно заданного графа, такой, чтобы в нее входила вершина, отвечающая цели. Таким образом, центральным пунктом решения задачи *с* использованием пространства состояний является поиск на графе указанного типа. Рассмотрение приемов поиска на графе мы отложим до следующей главы.