Лабораторная работа №3

ПРЕДСТАВЛЕНИЕ ПРОСТРАНСТВ СОСТОЯНИЙ ПОСРЕДСТВОМ НЕДЕТЕРМИНИРОВАННЫХ ПРОГРАММ

Цель работы: Изучить процесс порождения пространства состояний. Ознакомиться с примерами задач представления в пространстве состояний.

Теоретические сведения

Процесс порождения пространства состояний может быть представлен блок-схемой, изображенной на рис. 1. Здесь символы *x* и *y* используются для обозначения произвольных совокупностей данных. Заметим, что оператор присваивания «положить *y* равным *некоторому* члену множества Г (*y*) элементов, непосредственно следующих за *y*», является *недетерминированным* в том смысле, что при его выполнении может быть выбран *любой* член множества Г(*y*). Множество (возможно, бесконечное) всех возможных способов выполнения программы, представленной этой блок-схемой, охватывает тогда полное пространство состояний. (При такой формулировке состояния описываются возможными величинами программной переменной *y*, которая может быть произвольно сложной совокупностью данных).

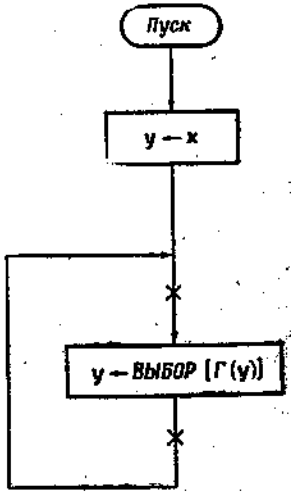
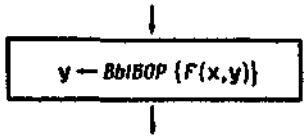


Рис. 1. Представление пространства состояний с помощью недетерминированной программы. Исходное положение: программная переменная y (пробегающая описания состояний) полагается равной входной структуре данных *x*, описывающей начальное состояние. Присваивание: новое значение программной переменной y полагается равным одному из - элементов множества Г (y) дочерних значений для прежней величины y.

Программы, в которых используются операторы присваивания (и другие), допускающие во время выполнения недетерминированные выборы, получили название *недетерминированных).* Часто бывает удобно представлять пространство состояний некоторой задачи неявно, с помощью некоторой недетерминированной блок-схемы. Такая блок-схема может быть столь же простой, как канонический пример рис.1., но может иметь и более сложный вид с несколькими недетерминированными и детерминированными элементами.

(Строго говоря, на нашей блок-схеме на рис.1. нужно было бы дать условия для окончания и привести другие подробности, такие, как тест для проверки непустоты множества Г (y). Проверка на окончание может появиться в любой из точек, помеченных крестиком на нашей блок-схеме.

В общем, недетерминированные программы связаны с расширением определений оператора обычного присваивания и оператора ветвления детерминированных программ. Мы уже видели, как можно пользоваться недетерминированными операторами присваивания. Этот тип операторов присваивания, называемый *операторами типа* **ВЫБОР**, на блок-схемах обозначается так:



Здесь совокупность данных *x* есть входная величина этой программы. Функция *F* является полной функцией, отображающей совместную область изменения *x* и *y* в некоторое непустое подмножество области изменения y. Для представления этого подмножества мы пользуемся обозначением *{F},* а ВЫБОР *[F]* означает выбор одного члена (любого члена) этого подмножества. Этот член затем присваивается как новое значение величине *y* . (Мы допускаем, что присваивание может зависеть как от значения входной переменной *x*, так и от имеющегося в данное время значения программной переменной *y*.)

Кроме того, мы расширяем понятие оператора ветвления. В операторе V-ветвления на *n* направлений используется *n* предикатов $p_1(x, y)$, ... , $p_n(x,y)$, принимающих либо значение *T* (истина), либо *F* (ложь) в совместной области изменения *x* и y, причем по крайней мере один из предикатов должен иметь значение *T.* Каждый

предикат соответствует некоторой ветви. Выбирается одна ветвь (любая), для которой соответствующий ей предикат имеет значение *T*. Этот тип недетерминированного ветвления называется V-ветвлением "и на блок-схемах обозначается следующим образом:

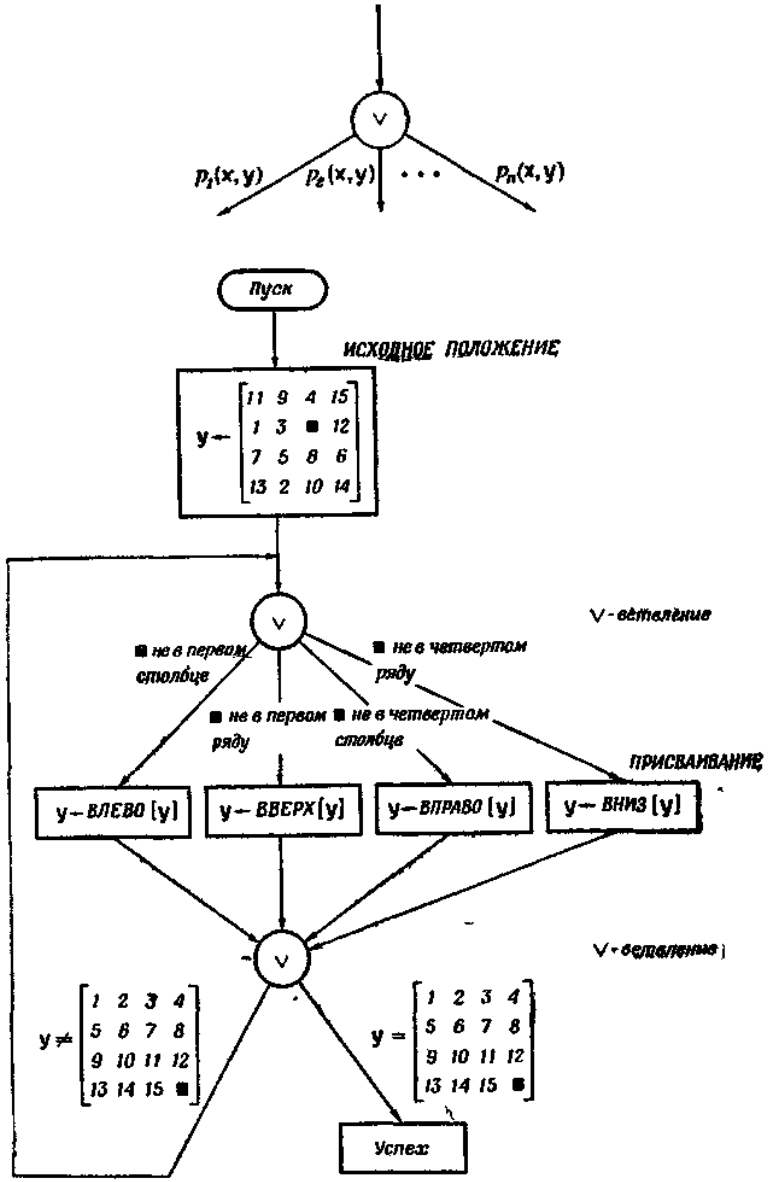


Рис. 2. Представление игры в пятнадцать в виде недетерминированной программы.

Отметим, что обычные детерминированные присваивания и ветвления - это простые частные случаи недетерминированных операторов.

При конкретном *выполнении* недетерминированной программы в операторах V-ветвление и ВЫБОР делаются конкретные выборы. Тогда множество всех возможных способов выполнения определяет пространство состояний. Если для любого входного массива существует, по крайней мере, одно конкретное выполнение программы, имеющее окончание, то говорят, что эта программа правильно определена.

В качестве того как некоторой задаче можно придать форму недетерминированной программы, на рис.2. дается одна возможная программа для игры в пятнадцать. Здесь состояние описывается значением программной переменной *y*, принадлежащем пространству массивов 4X4. Элементами этих массивов являются числа от 1 до 15 и символ • (представляющий пустую клетку). Функции ВЛЕВО, ВВЕРХ, ВПРАВО, ВНИЗ соответствуют операторам. Они изменяют массивы посредством перемещения символа • соответственно влево, вверх, вправо, вниз.

Оказывается, что можно определить также и другие элементы недетерминированных программ. Эти элементы полезны при обсуждении формулировок, основанных на сведении задачи к подзадачам.

НЕКОТОРЫЕ ПРИМЕРЫ ПРЕДСТАВЛЕНИЙ ЗАДАЧ

Для большого числа задач можно дать представления, связанные с пространством состояний. Для некоторых задач такое представление удается выбрать совершенно естественно, тогда как для других любое представление, связанное с введением пространства состояний, кажется весьма искусственным. Не следует предполагать, что каждая из формулировок, приведенных в разделе, наилучшая из всех возможных. Сейчас мы хотим лишь показать, что для некоторых различных типов задач действительно *возможно* представление в пространстве состояний.

1. Задача о коммивояжере

Задача о коммивояжере - классическая комбинаторная проблема. Коммивояжер должен построить свой маршрут так, чтобы побывать в каждом из *n* городов в точности по разу и возвратиться в исходный город. Желательно, чтобы этот маршрут имел минимально возможную протяженность. Было разработано несколько эффективных методов

решения задачи, которые реализуемы лишь в том случае, когда число городов не превышает примерно 50. Приближенные методы дают хорошие решения (хотя не обязательно минимизирующие протяженность маршрута) уже для 200 городов. Задача о коммивояжере полезна для иллюстрации представлений, основанных на введении пространства состояний, как это видно из следующего простого частного примера

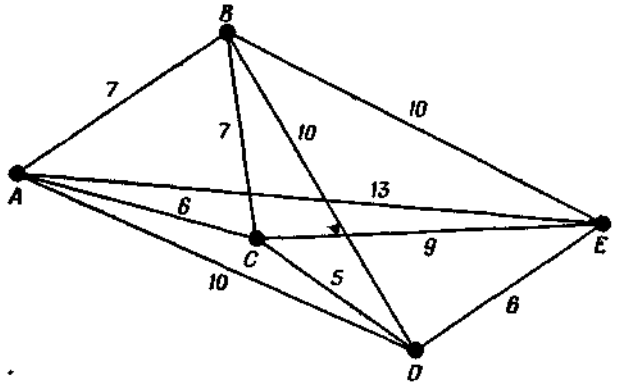


Рис. 3. Карта для задачи о коммивояжере.

Коммивояжер должен посетить каждый из пяти городов, изображенных на карте на рис.3. Между каждой парой городов имеется путь, длина которого указана на этой карте. Нужно, отправляясь из города *A,* найти самый короткий путь, по которому коммивояжер по одному разу проходит через каждый из городов и затем возвращается в город *A*.

Чтобы дать представление в пространстве состояний, мы должны определить следующее:

Описания состояний. Будем задавать состояния списком городов, пройденных к настоящему моменту. Так, начальным состоянием будет список *(A)*. Мы не будем допускать, чтобы в этом списке какой-то город упоминался более одного раза, с тем лишь исключением, что после того, как в нем будут упомянуты все остальные города, может быть снова упомянут *A*.

Операторы. Операторы суть вычисления, соответствующие поступкам: (1) направиться теперь в город *A,* (2) направиться теперь в город *B, ...,* (5) направиться теперь в город *E.* Оператор неприменим к некоторому описанию состояния, если он не преобразует его в некоторое допустимое описание. Так, оператор номер (1) (соответствующий «направиться теперь в город *A»)* неприменим ни к какому описанию, не содержащему названия всех городов.

Критерий достижения цели. Любое описание, начинающееся и оканчивающееся городом *A* и перечисляющее все другие города, есть описание состояния, удовлетворяющего поставленной цели.

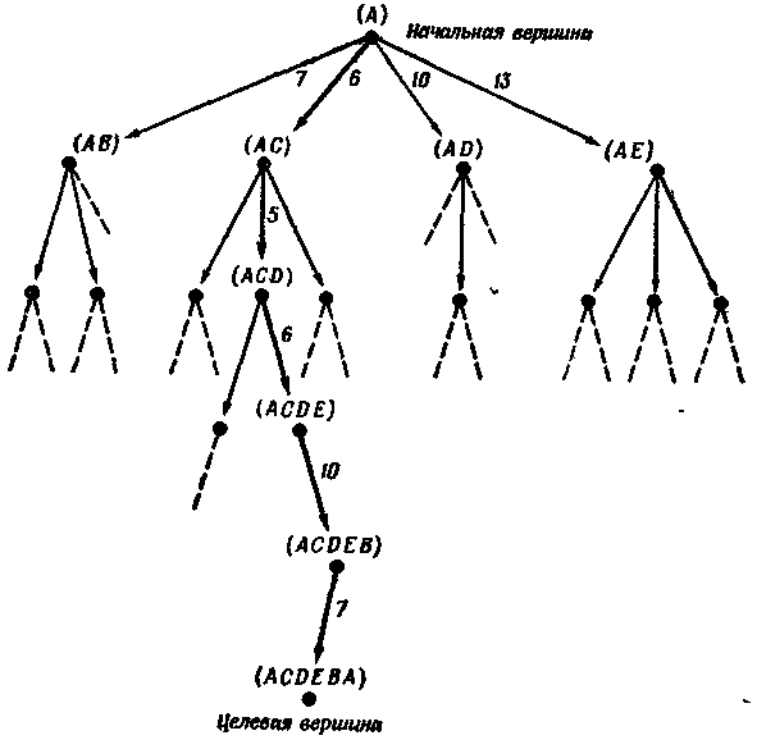


Рис.4. Часть графа для задачи о коммивояжере.

На рис.4. показано представление этого пространства состояний в виде графа. (Явно указаны лишь некоторые из его вершин.) Числа, написанные около дуг графа, указывают стоимости этих дуг. Мы полагаем эти стоимости равными расстояниям между соответствующими городами (см. рис.3). В вершинах графа стоят описания тех состояний, которые они представляют. Достоинства представления в виде графа состоят в том, что приписывание дугам стоимостей дает нам удобный способ вычисления полной длины маршрута, а следовательно, и способ поиска кратчайшего из них. Кратчайший (34 мили) для нашего случая показан на графе жирными стрелками.

Задача о коммивояжере представляет собой пример задачи, в которой информация, содержащаяся в ее формулировке, представима в

графической форме (карте расстояний). Следует быть внимательным и не смешивать какие-либо графы, используемые при формулировке задачи, с графом пространства состояний, который строится при решении задачи.

2. Задачи синтаксического анализа

При работе с языками часто сталкиваются с задачей *синтаксического анализа.* В таких задачах сначала дается некоторое формальное определение через задание *грамматики,* которая выделяет определенный класс строк символов. А затем возникает вопрос о том, принадлежит ли к этому классу произвольная строка. Следующий пример иллюстрирует этот тип задач.

Грамматика. Предположим, что мы определяем *предложение как* строку одного из следующих видов:

за символом *a* следует символ *b*

за символом *a* следует некоторое предложение

за некоторым предложением следует символ *b*

за некоторым предложением следует другое предложение.

Примеры предложений: *aab, abaabab, aaaaab.* Некоторые строки, не являющиеся предложениями *aaa, aba, abaa.*

Предположим, что мы захотели определить, является ли строка *abaabab* предложением. Тогда формулировка этой задачи в пространстве состояний выглядит так:

Описания состояний. Один из возможных путей формулировки этой задачи состоит в том, чтобы выбрать в качестве начального состояния рассматриваемую строку *abaabab.* Тогда множеством допустимых состояний будет множество строк, получающихся из нее путем применения тех правил переписывания (они даются ниже), которыми определяются операторы.

Операторы. Мы определяем операторы через следующие правила переписывания:

$\$_1 ab \$_2 \rightarrow \$_1 S \$_2$ (подстрока *ab* может быть заменена символом *S,* обозначающим предложение)

$\$_1 aS \$_2 \rightarrow \$_1 S \$_2$

$\$_1 Sb \$_2 \rightarrow \$_1 S \$_2$

$\$_1 SS \$_2 \rightarrow \$_1 S \$_2$

Мы видим, что эти правила просто выражают грамматику, определяющую понятие предложения.

Критерий цели. Целевое состояние описывается строкой, которая состоит из одиночного символа S.

Последовательность состояний, представляющая собой решение этой задачи, имеет следующий вид:

abaabab
Saabab
SaSab
SSab
SSS
SS
S

Граф, изображающий пространство состояний для этой задачи, показан на рис.5. В этой задаче оказалось так, что в силу заданной грамматики любая строка, начинающаяся с *a* и оканчивающаяся на *b,* является предложением. Знание такого факта, очевидно, сильно бы упростило решение вопроса о том, будет ли некоторая произвольная строка предложением. Иногда оказывается, что заданная грамматика может быть представлена в эквивалентном, но более простом виде. Обнаружение таких упрощений позволяет строить меньшие пространства для перебора.

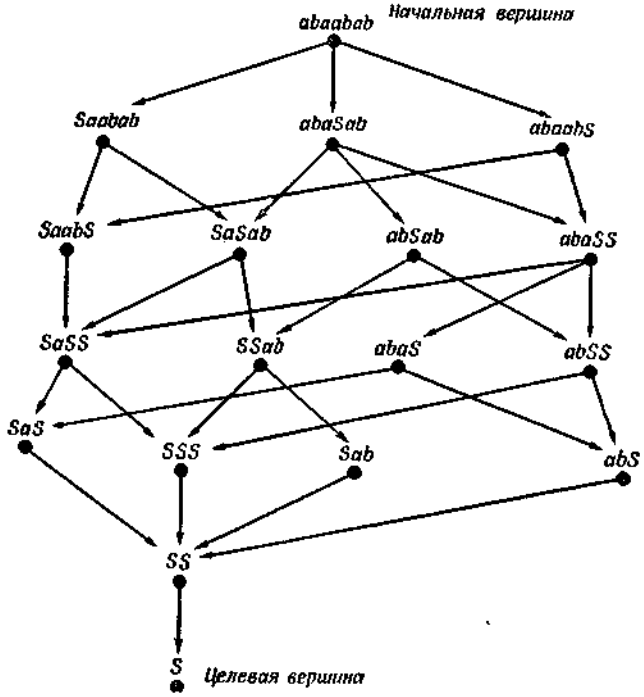


Рис. 5. Граф для задачи синтаксического анализа.

3. Задачи распределения

Следующая простая задача типична для класса задач, называемых иногда *задачами распределения.* Имеются два источника жидкости: *A* дает 100 галлонов в минуту, а *B* — 50. Источники должны снабжать два бассейна *C* и *D,* потребность каждого из которых 75 галлонов в минуту. Жидкость может подаваться от источника к бассейну с помощью труб с максимальной пропускной способностью 75 галлонов в минуту. Пусть источники и бассейны расположены так, как это показано на рис.6., и соединения труб допускаются только в местах расположения источников и бассейнов. Спрашивается, как следует подсоединять трубы, чтобы при этом полная длина труб была наименьшей.

Представление этой задачи в пространстве состояний выглядит следующим образом:

Описания состояний. Состояния описываются списком величин избыточного расхода жидкости, который имеется в точках *A*, *B*, *C* и *D,* Так, начальное состояние описывается списком *(A* = 100, *B* = 50, *C* = *0*, *D* = *0*).

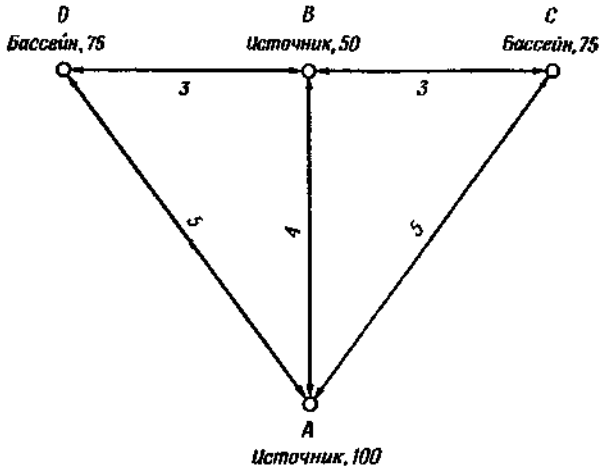


Рис. 6. Расположение источников жидкости и бассейнов (расстояния измеряются в милях).

Операторы. Операторы соответствуют передаче избытка «жидкости в минуту» из одной точки в другую. В задачах, подобных этой, в качестве подходящего избытка выступает наибольший общий делитель пропускных способностей и потребностей в жидкости в различных точках. Таким образом, у нас есть операторы:

1. Передать 25 галлон/мин из *A* в *B.*

2. Передать 25 галлон/мин из *A* в С.

……..

12. Передать 25 галлон/мин из В в *A*.

Разумеется, операторы применимы лишь тогда, когда имеется достаточный избыток жидкости в той точке, от которой жидкость отбирается для передачи в другую точку. И, конечно, для осуществления каждой такой передачи нужно иметь соответствующую трубу.

Критерий цели. Целевое состояние описывается списком *(A* = 0, *B* = *0*, *C* = 75, *D* = 75).

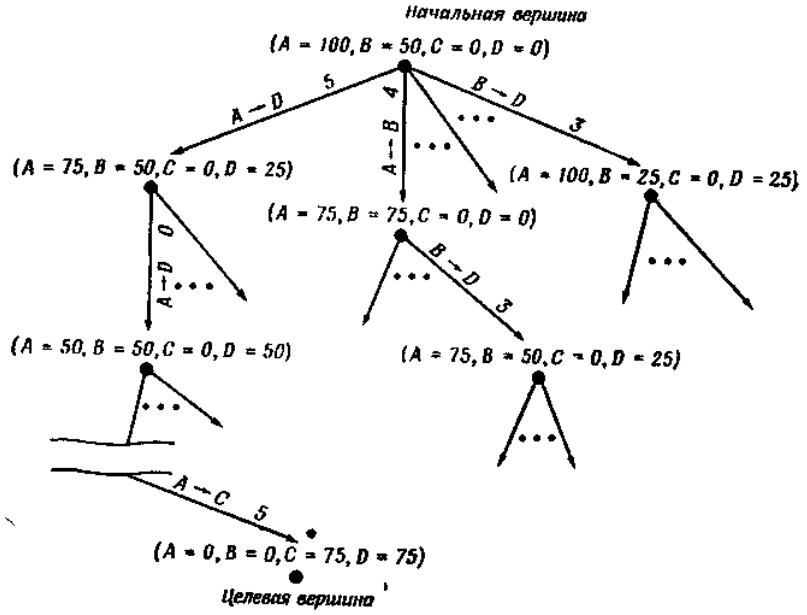


Рис.7. Часть графа для задачи распределения.

Часть графа, получающегося таким образом пространства состояния, показана на рис.7. Обозначение типа $A \rightarrow D$ около дуг графа показывает, что соответствующий оператор передает избыток в 25 галлон/мин от *A к D.* Стоимости, написанные рядом с каждой дугой, показывают, сколько миль труб нужно добавить для подачи этого избытка. Число нуль при этом означает, что нет необходимости добавлять еще трубу, поскольку уже имеющаяся труба обладает достаточной дополнительной пропускной способностью. Граф на рис.7. изображен не полностью - многие из его вершин не показаны. Исследовав полный граф, можно установить, что путь, ведущий от начальной вершины к целевой и обладающий наименьшей стоимостью, требует 12 миль труб.

4. Задачи управления

В типичной задаче *управления* имеется процесс, представленный системой «устанавливаемых» переменных, которые должны управляться с помощью соответствующего управления, обеспечиваемого некоторым множеством управляющих переменных.

Интересным примером служит задача о перевернутом маят*нике* на тележке (рис.8.). В этой задаче масса *M* прикреплена к концу стержня длины *l*, другой конец которого шарнирно закреплен на тележке, так что стержень может свободно вращаться в вертикальной плоскости, совпадающей с направлением движения тележки, снабженной колесами. Устанавливаемые переменные — угол наклона стержня θ, координата *x* тележки и производная по времени θ'. Требуется, чтобы значения каждой из этих переменных поддерживались в определенных, заранее указанных границах. Управляющей переменной служит скорость тележки *x,* которая может принимать одно из двух значений + *v* и - *v*. (Мы предполагаем для простоты, что эти значения могут сменять друг друга мгновенно.) Главная задача здесь состоит в принятии в данный момент решения о том, следует ли перемещать тележку со скоростью *v* вправо или со скоростью *v* влево.

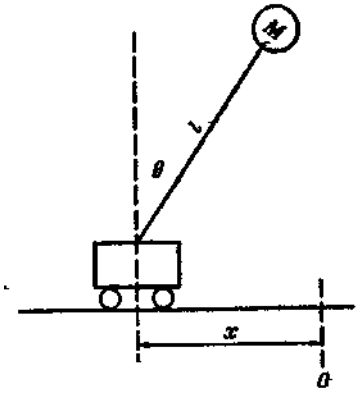


Рис.8. Перевернутый маятник на тележке

Описание состояний. Предположив, что переменные θ', θ и x принимают дискретные значения с достаточно мелким шагом, можно считать состоянием вектор, составленный из этих трех переменных (пространством состояний при этом служит решетка в трехмерном пространстве θ', θ и x*)*.

Операторы. Имеются ровно два оператора:

1. Применить управление + *v*.
2. Применить управление —*v*.

Состояние, возникающее в результате применения одного из этих операторов, — это просто то состояние, которое описывается вектором (θ, θ', x) по истечении Δt секунд. (Во многих типичных задачах управления действия операторов могут быть компактно представлены с помощью дифференциальных уравнений.)

Критерий достижения цели. Предположим, что целевое состояние описывается вектором (θ = 0, θ' = 0, x = 0). Тогда нам нужно найти последовательность операторов, которая будет преобразовывать любое данное состояние в целое. Конечно, такая последовательность не должна приводить ни к какому состоянию, описываемому переменными θ', θ и x, для которого неизбежно в конце концов полное нарушение работы системы. (Оно происходит при θ = ±90°.)

В некоторых типичных задачах управления часто можно получить (аналитическими методами) уравнения разделяющих поверхностей, которые разбивают векторное пространство состояний на непересекающиеся области, такие, что для всех векторов из данной области в данный момент должно быть применено одно и то же управление (один и тот же оператор). В этих случаях несложное вычисление может дать ответ, который иначе получался бы с помощью поиска. Читатель должен понимать, что мы не собираемся предлагать использовать поисковые процессы в случаях, когда известны прямые методы решения. Мы хотим лишь подчеркнуть, что часто можно воспользоваться эффективными методами перебора для решения тех задач, для которых прямые методы еще не найдены.

ВЫБОР «ХОРОШИХ» ПРЕДСТАВЛЕНИИ

Выбор для данной задачи определенного представления в пространстве состояний существенно определяет те поисковые усилия, которые придется приложить для ее решения. Очевидно, что предпочтительны представления с малыми пространствами состояний (т. е. такие, графы которых имеют небольшое число вершин). Существует множество примеров задач, кажущихся трудными, но таких, что при правильной их трактовке соответствующие пространства состояний оказываются очень узкими. Подчас данное пространство состояний «сжимается в точку» после того, как обнаруживается, что некоторые операторы могут быть отброшены за ненадобностью, а другие — объединены в более мощные операторы. И даже если такие простые преобразования неосуществимы, может оказаться, что полная переформулировка задачи, изменяющая само понятие состояния, приведет к меньшему пространству.

По-видимому, желаемый прогресс в представлении задачи зависит от опыта, накопленного в попытках решить задачу в рамках данного представления. Этот опыт позволяет выделить упрощающие понятия, такие, как свойства симметрии или полезные последовательности операторов, которые следует объединить в *макрооператоры*.

Чрезвычайно важная идея в области представлений задач связана с использованием переменных в описаниях состояний. Тогда выражения, содержащие переменные, могут быть использованы для описания целого множества состояний, а не только одного. Подстановка каких-то частных значений (констант) вместо этих переменных в указанные выражения дает конкретные описания состояний. Выражение, содержащее переменные, которое используется для описания множества состояний в указанном смысле, называется *схемой* для описания состояний. Мы проиллюстрируем на примере использование схем для описания состояний.

К задаче об обезьяне и бананах часто обращаются в литературе по искусственному интеллекту при демонстрации работы автоматических решателей задач, создаваемых для осуществления умозаключений, опирающихся на здравый смысл. Задачу можно сформулировать так. В комнате, где находится обезьяна, имеются ящик и связка бананов, причем бананы подвешены к потолку настолько высоко, что обезьяна не может до них дотянуться с пола. Какая последовательность действий позволит обезьяне достать эти бананы? (Предполагается, что обезьяна должна подойти к ящику, подтащить его к бананам, забраться на него и достать бананы.)

Как следует строить для этой задачи представления в пространстве состояний? В описании состояния должны непременно появиться следующие элементы: координаты обезьяны в комнате (по горизонтали и по вертикали), координаты ящика в комнате и наличие у обезьяны бананов. Эти элементы удобно представить в виде четырехэлементного списка (w, x, y, z), где

w — координаты обезьяны в горизонтальной плоскости (двумерный вектор),

x — это 1 или 0 в зависимости от того, где обезьяна находится, на ящике или нет,

y — координаты ящика в горизонтальной плоскости (двумерный вектор),

z— это 1 или 0 в зависимости от того, достала обезьяна бананы или нет.

Если бы каждое отдельное значение списка (w, x, y, z) описывало ровно одно состояние, то мы имели бы бесконечное число состояний, поскольку число различных расположений предметов в комнате бесконечно. Мы могли бы сделать пространство состояний конечным, допустив, что предметы могут находиться лишь в конечном числе точек (скажем, точек решетки), но все же число точек, достаточное для поставленной задачи, привело бы к чрезвычайно большому пространству состояний. Другой путь состоит в использовании схем. Для входящих в схему переменных имеются частные значения (например, константы), которые должны подставляться на их место при применении оператора или при проверке того, что цель достигнута.

Операторы в этой задаче соответствуют четырем возможным действиям, которые могут выполняться обезьяной:

1) подойти (**u**) - обезьяна идет к точке **u** в плоскости пола комнаты (**u** — переменная),

2) передвинуть (**v**) - обезьяна передвигает ящик в точку **v** пола комнаты (**v** - переменная),

3) взобраться - обезьяна забирается на ящик,

4) схватить - обезьяна хватает бананы.

В связи с наличием переменных в «подойти», «передвинуть» эти операторы на самом деле являются схемами. Условия применимости и действия этих операторов даются следующими правилами переписывания:

$$(w, 0, y, z) \xrightarrow{\text{подойти (u)}} (u, 0, y, z),$$

$$(w, 0, w, z) \xrightarrow{\text{передвинуть (v)}} (v, 0, v, z),$$

$$(w, 0, w, z) \xrightarrow{\text{взобраться}} (w, 1, w, z),$$

$$(c, 1, c, 0) \xrightarrow{\text{схватить}} (c, 1, c, 1),$$

где **c** — координаты точки пола, расположенной непосредственно под бананами (двумерный вектор).

Применение некоторых операторов, например «передвинуть», связано с ограничением значений переменных, представляющих координаты обезьяны и ящика, от которых теперь требуется, чтобы они были одинаковыми. Мы будем считать идентичными две схемы для описания состояний, если они отличаются лишь наименованием переменных.

В такой формулировке элементы множества целевых состояний описываются любыми списками, последний элемент которых есть

единица.

Предположим, что вначале обезьяна находится в точке **a** пола, а ящик — в **b**. Тогда описанием начального состояния будет (**a**, 0, **b**, 0). Единственный оператор, который применим в этом состоянии, — это «подойти», приводящий к схеме (**u**, 0, **b**, 0). Теперь применимы три оператора. Если **u = b**, то обезьяна может либо забраться на ящик, либо передвинуть его. Независимо от величины **u** обезьяна могла бы переместиться куда-нибудь еще. Влезание на ящик приводит к состоянию с описанием (**b**, 1, **b**, 0); перемещение ящика в **v** приводит к схеме (**v**, 0, **v**, 0), а переход в другое место, описываемое новой переменной, не изменяет описания.

Продолжая процесс применения всех операторов, мы построим пространство состояний, иллюстрируемое графом на рис.9. Мы видим, что этот граф весьма невелик и на нем легко можно найти путь, решающий нашу задачу (жирные стрелки). Подставив вместо переменных их соответствующие частные значения, как это показано на рис.9., мы получаем последовательность операторов: подойти (**b**), передвинуть (**c**), взобраться, схватить.

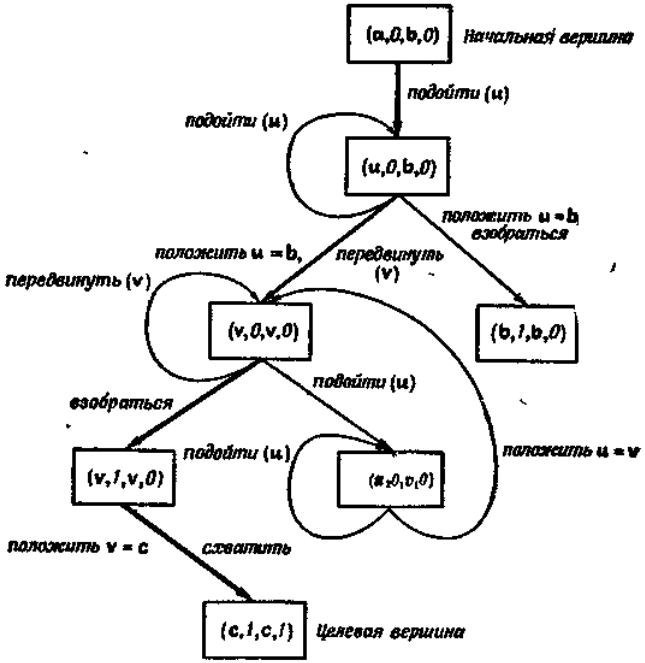


Рис. 9. Граф для задачи об обезьяне и бананах.