2 Интеллектуальные агенты

В этой главе рассматриваются характеристики агентов, идеальных или неидеальных, разнообразие вариантов среды и вытекающая из этого классификация типов агентов.

Как было указано в главе 1, понятие **рационального агента** является центральным в применяемом авторами данной книги подходе к искусственному интеллекту. В этой главе указанное понятие раскрывается более подробно. В ней показано, что концепция рациональности может применяться к самым различным агентам, действующим в любой среде, которую только можно себе представить. План авторов состоит в том, чтобы использовать эту концепцию в данной книге для разработки небольшого набора принципов проектирования для создания успешно действующих агентов — систем, которые вполне можно было бы назвать **интеллектуальными**.

Начнем с изучения агентов, вариантов среды и связей между ними. Наблюдая за тем, что некоторые агенты действуют лучше, чем другие, можно вполне обоснованно выдвинуть идею рационального агента; таковым является агент, который действует настолько успешно, насколько это возможно. Успехи, которых может добиться агент, зависят от характера среды; некоторые варианты среды являются более сложными, чем другие. В этой главе дана грубая классификация вариантов среды и показано, как свойства среды влияют на проектирование агентов, наиболее подходящих для данной среды. Здесь описан ряд основных, "скелетных" проектов агентов, которые будут облечены в плоть в остальной части книги.

2.1. Агенты и варианты среды

Агентом является все, что может рассматриваться как воспринимающее свою **среду** с помощью **датчиков** и воздействующее на эту среду с помощью **исполнительных механизмов**. Эта простая идея иллюстрируется на рис. 2.1. Человек, рассматриваемый в роли агента, имеет глаза, уши и другие органы чувств, а исполнительными механизмами для него служат руки, ноги, рот и другие части тела. Робот, выполняющий функции агента, в качестве датчиков может иметь видеокамеры и инфракрасные дальномеры, а его исполнительными механизмами могут являться различные двигатели. Программное обеспечение, выступающее в роли аген-

та, в качестве входных сенсорных данных получает коды нажатия клавиш, содержимое файлов и сетевые пакеты, а его воздействие на среду выражается в том, что программное обеспечение выводит данные на экран, записывает файлы и передает сетевые пакеты. Мы принимаем общее допущение, что каждый агент может воспринимать свои собственные действия (но не всегда их результаты).

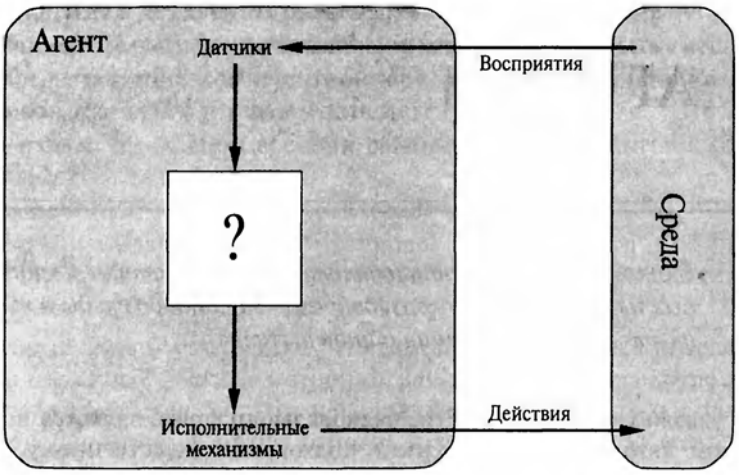

Рис. 2.1. Агент взаимодействует со средой с помощью датчиков и исполнительных механизмов

Мы используем термин **восприятие** для обозначения полученных агентом сенсорных данных в любой конкретный момент времени. **Последовательностью актов восприятия** агента называется полная история всего, что было когда-либо воспринято агентом. Вообще говоря, *выбор агентом действия в любой конкретный момент времени может зависеть от всей последовательности актов восприятия, наблюдавшихся до этого момента времени.* Если существует возможность определить, какое действие будет выбрано агентом в ответ на любую возможную последовательность актов восприятия, то может быть дано более или менее точное определение агента. С точки зрения математики это равносильно утверждению, что поведение некоторого агента может быть описано с помощью **функции агента**, которая отображает любую конкретную последовательность актов восприятия на некоторое действие.

Может рассматриваться задача табуляции функции агента, которая описывает любого конкретного агента; для большинства агентов это была бы очень большая таблица (фактически бесконечная), если не устанавливается предел длины последовательностей актов восприятия, которые должны учитываться в таблице. Проводя эксперименты с некоторым агентом, такую таблицу в принципе можно сконструировать, проверяя все возможные последовательности актов восприятия и регистрируя, какие действия в ответ выполняет агент[1]. Такая таблица, безусловно, является внешним описанием агента. Внутреннее описание состоит в определении того, ка-

[1] Если агент для выбора своих действий использует определенную рандомизацию, то может потребоваться проверить каждую последовательность многократно, чтобы определить вероятность каждого действия. На первый взгляд кажется, что выбор действий случайным образом является довольно неразумным, но ниже в этой главе будет показано, что такая организация функционирования может оказаться весьма интеллектуальной.

кая функция агента для данного искусственного агента реализуется с помощью ✍ **программы агента**. Важно различать два последних понятия. *Функция агента* представляет собой абстрактное математическое описание, а *программа агента* — это конкретная реализация, действующая в рамках архитектуры агента.

Для иллюстрации изложенных идей воспользуемся очень простым примером: рассмотрим показанный на рис. 2.2 мир, в котором работает пылесос. Этот мир настолько прост, что существует возможность описать все, что в нем происходит; кроме того, это — мир, созданный человеком, поэтому можно изобрести множество вариантов его организации. В данном конкретном мире имеются только два местонахождения: квадраты `A` и `B`. Пылесос, выполняющий роль агента, воспринимает, в каком квадрате он находится и есть ли мусор в этом квадрате. Агент может выбрать такие действия, как переход влево, вправо, всасывание мусора или бездействие. Одна из очень простых функций агента состоит в следующем: если в текущем квадрате имеется мусор, то всосать его, иначе перейти в другой квадрат. Частичная табуляция данной функции агента показана в табл. 2.1. Простая программа агента для этой функции агента приведена ниже в этой главе, в листинге 2.2.

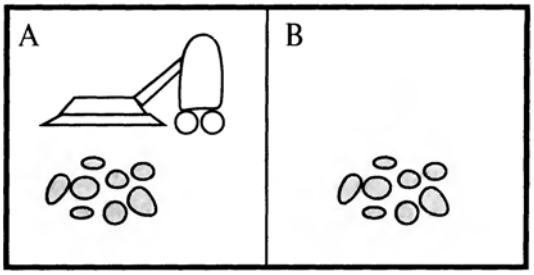

Рис. 2.2. Мир пылесоса, в котором имеются только два местонахождения

Таблица 2.1. Частичная табуляция функции простого агента для мира пылесоса, показанного на рис. 2.2

Последовательность актов восприятия	Действие
`[A,Clean]`	`Right`
`[A,Dirty]`	`Suck`
`[B,Clean]`	`Left`
`[B,Dirty]`	`Suck`
`[A,Clean],[A,Clean]`	`Right`
`[A,Clean],[A,Dirty]`	`Suck`
...	...
`[A,Clean],[A,Clean],[A,Clean]`	`Right`
`[A,Clean],[A,Clean],[A,Dirty]`	`Suck`
...	...

На основании табл. 2.1 можно сделать вывод, что для мира пылесоса можно определять различных агентов, заполняя разными способами правый столбец этой таблицы. Поэтому очевидный вопрос состоит в следующем: ☞ *"Какой способ заполнения этой таблицы является правильным?"* Иными словами, благодаря чему агент

становится хорошим или плохим, интеллектуальным или не соответствующим критериям интеллектуальности? Ответ на этот вопрос приведен в следующем разделе.

Прежде чем завершить этот раздел, необходимо отметить, что понятие агента рассматривается как инструмент для анализа систем, а не как абсолютная классификация, согласно которой мир делится на агентов и неагентов. Например, в качестве агента можно было бы рассматривать карманный калькулятор, который выбирает действие по отображению "4" после получения последовательности актов восприятия "2+2=", но подобный анализ вряд ли поможет понять работу калькулятора.

2.2. Качественное поведение: концепция рациональности

Рациональным агентом является такой агент, который выполняет правильные действия; выражаясь более формально, таковым является агент, в котором каждая запись в таблице для функции агента заполнена правильно. Очевидно, что выполнение правильных действий лучше, чем осуществление неправильных действий, но что подразумевается под выражением "выполнение правильных действий"? В первом приближении можно сказать, что правильным действием является такое действие, которое обеспечивает наиболее успешное функционирование агента. Поэтому требуется определенный способ измерения успеха. Критерии успеха, наряду с описанием среды, а также датчиков и исполнительных механизмов агента, предоставляют полную спецификацию задачи, с которой сталкивается агент. Имея эти компоненты, мы можем определить более точно, что подразумевается под словом "рациональный".

Показатели производительности

Показатели производительности воплощают в себе критерии оценки успешного поведения агента. После погружения в среду агент вырабатывает последовательность действий, соответствующих полученным им восприятиям. Эта последовательность действий вынуждает среду пройти через последовательность состояний. Если такая последовательность соответствует желаемому, то агент функционирует хорошо. Безусловно, что не может быть одного постоянного показателя, подходящего для всех агентов. Можно было бы узнать у агента его субъективное мнение о том, насколько он удовлетворен своей собственной производительностью, но некоторые агенты не будут способны ответить, а другие склонны заниматься самообманом[2]. Поэтому необходимо упорно добиваться применения объективных показателей производительности, и, как правило, проектировщик, конструирующий агента, предусматривает такие показатели.

Рассмотрим агент-пылесос, описанный в предыдущем разделе. Можно было бы предложить измерять показатели производительности по объему мусора, убранного за одну восьмичасовую смену. Но, безусловно, имея дело с рациональным агентом,

[2] Особенно известны тем, что недостигнутый успех для них — "зелен виноград", такие агенты, как люди. Не получив кое-что для себя весьма ценное, они искренне считают, что и не стремились к этому: "Подумаешь! Мне и даром не нужна эта дурацкая Нобелевская премия!"

вы получаете то, что просите. Рациональный агент может максимизировать такой показатель производительности, убирая мусор, затем вываливая весь его на пол, затем снова убирая, и т.д. Поэтому более приемлемые критерии производительности должны вознаграждать агента за то, что пол остается чистым. Например, одно очко могло бы присуждаться за каждый чистый квадрат в каждом интервале времени (возможно, в сочетании со штрафом за потребляемую электроэнергию и создаваемый шум). ☞ *В качестве общего правила следует указать, что лучше всего разрабатывать показатели производительности в соответствии с тем, чего действительно необходимо добиться в данной среде, а не в соответствии с тем, как, по мнению проектировщика, должен вести себя агент.*

Задача выбора показателей производительности не всегда является простой. Например, понятие "чистого пола", которое рассматривалось выше, основано на определении усредненной чистоты пола во времени. Но необходимо также учитывать, что одна и та же усредненная чистота может быть достигнута двумя различными агентами, один из которых постоянно, но неторопливо выполняет свою работу, а другой время от времени энергично занимается очисткой, но делает длинные перерывы. Может показаться, что определение того способа действий, который является в данном случае наиболее предпочтительным, относится к тонкостям домоводства, но фактически это — глубокий философский вопрос с далеко идущими последствиями. Что лучше — бесшабашная жизнь со взлетами и падениями или безопасное, но однообразное существование? Что лучше — экономика, в которой каждый живет в умеренной бедности, или такая экономика, в которой одни ни в чем не нуждаются, а другие еле сводят концы с концами? Оставляем задачу поиска ответов на эти вопросы в качестве упражнения для любознательного читателя.

Рациональность

В любой конкретный момент времени оценка рациональности действий агента зависит от четырех перечисленных ниже факторов.

- Показатели производительности, которые определяют критерии успеха.
- Знания агента о среде, приобретенные ранее.
- Действия, которые могут быть выполнены агентом.
- Последовательность актов восприятия агента, которые произошли до настоящего времени.

С учетом этих факторов можно сформулировать следующее ✍ **определение рационального агента**.

> ☞ *Для каждой возможной последовательности актов восприятия рациональный агент должен выбрать действие, которое, как ожидается, максимизирует его показатели производительности, с учетом фактов, предоставленных данной последовательностью актов восприятия и всех встроенных знаний, которыми обладает агент.*

Рассмотрим пример простого агента-пылесоса, который очищает квадрат, если в нем имеется мусор, и переходит в другой квадрат, если мусора в нем нет; результаты частичной табуляции такой функции агента приведены в табл. 2.1. Является ли этот агент рациональным? Ответ на этот вопрос не так уж прост! Вначале необходимо определить, в чем состоят показатели производительности, что известно о среде и ка-

кие датчики и исполнительные механизмы имеет агент. Примем перечисленные ниже предположения.

- Применяемые показатели производительности предусматривают вознаграждение в одно очко за каждый чистый квадрат в каждом интервале времени в течение "срока существования" агента, состоящего из 1000 интервалов времени.
- "География" среды известна заранее (рис. 2.2), но распределение мусора и первоначальное местонахождение агента не определены. Чистые квадраты остаются чистыми, а всасывание мусора приводит к очистке текущего квадрата. Действия `Left` и `Right` приводят к перемещению агента соответственно влево и вправо, за исключением тех случаев, когда они могли бы вывести агента за пределы среды, и в этих случаях агент остается там, где он находится.
- Единственными доступными действиями являются `Left`, `Right`, `Suck` (всосать мусор) и `NoOp` (ничего не делать).
- Агент правильно определяет свое местонахождение и воспринимает показания датчика, позволяющие узнать, имеется ли мусор в этом местонахождении.

Авторы утверждают, что в этих обстоятельствах агент действительно является рациональным; его ожидаемая производительность, по меньшей мере, не ниже, чем у любых других агентов. В упр. 2.4 предложено доказать это утверждение.

Можно легко обнаружить, что в других обстоятельствах тот же самый агент может стать нерациональным. Например, после того как весь мусор будет очищен, агент станет совершать ненужные периодические перемещения вперед и назад; если показатели производительности предусматривают штраф в одно очко за каждое передвижение в том или ином направлении, то агент не сможет хорошо зарабатывать. В таком случае лучший агент должен был бы ничего не делать до тех пор, пока он уверен в том, что все квадраты остаются чистыми. Если же чистые квадраты могут снова стать грязными, то агент обязан время от времени проводить проверку и снова очищать их по мере необходимости. А если география среды неизвестна, то агенту может потребоваться исследовать ее, а не ограничиваться квадратами *A* и *B*. В упр. 2.4 предложено спроектировать агентов для подобных случаев.

Всезнание, обучение и автономность

Необходимо тщательно проводить различие между рациональностью и **всезнанием**. Всезнающий агент знает фактический результат своих действий и может действовать соответствующим образом; но всезнание в действительности невозможно. Рассмотрим следующий пример: некий господин однажды гуляет в Париже по Елисейским Полям и видит на другой стороне улицы старого приятеля. Вблизи нет никаких машин, а наш господин никуда не спешит, поэтому, будучи рациональным агентом, он начинает переходить через дорогу. Между тем на высоте 10 000 метров у пролетающего самолета отваливается дверь грузового отсека[3], и прежде чем несчастный успевает достичь другой стороны улицы, расплющивает его в лепешку. Было ли нерациональным именно то, что этот господин решил перейти на другую сторону улицы? Весьма маловероятно, что в его некрологе написали бы: "Жертва идиотской попытки перейти улицу".

[3] См. заметку Н. Гендерсона "New door latches urged for Boeing 747 jumbo jets" (На дверях аэробусов Boeing 747 необходимо срочно установить новые замки). — Washington Post, 24 августа 1989 года.

Этот пример показывает, что рациональность нельзя рассматривать как равнозначную совершенству. Рациональность — это максимизация ожидаемой производительности, а совершенство — максимизация фактической производительности. Отказываясь от стремления к совершенству, мы не только применяем к агентам справедливые критерии, но и учитываем реальность. Дело в том, что если от агента требуют, чтобы он выполнял действия, которые оказываются наилучшими после их совершения, то задача проектирования агента, отвечающего этой спецификации, становится невыполнимой (по крайней мере, до тех пор, пока мы не сможем повысить эффективность машин времени или хрустальных шаров, применяемых гадалками).

Поэтому наше определение рациональности не требует всезнания, ведь рациональный выбор зависит только от последовательности актов восприятия, сформированной к данному моменту. Необходимо также следить за тем, чтобы мы непреднамеренно не позволили бы агенту участвовать в действиях, которые, безусловно, не являются интеллектуальными. Например, если агент не оглядывается влево и вправо, прежде чем пересечь дорогу с интенсивным движением, то полученная им до сих пор последовательность актов восприятия не сможет подсказать, что к нему на большой скорости приближается огромный грузовик. Указывает ли наше определение рациональности, что теперь агент может перейти через дорогу? Отнюдь нет! Во-первых, агент не был бы рациональным, если бы попытался перейти на другую сторону, получив такую неинформативную последовательность актов восприятия: риск несчастного случая при подобной попытке перейти автомагистраль, не оглянувшись, слишком велик. Во-вторых, рациональный агент должен выбрать действие "оглянуться", прежде чем ступить на дорогу, поскольку такой осмотр позволяет максимизировать ожидаемую производительность. Выполнение в целях модификации будущих восприятий определенных действий (иногда называемых **сбором информации**) составляет важную часть рациональности и подробно рассматривается в главе 16. Второй пример сбора информации выражается в том **исследовании ситуации**, которое должно быть предпринято агентом-пылесосом в среде, которая первоначально была для него неизвестной.

Наше определение требует, чтобы рациональный агент не только собирал информацию, но также **обучался** в максимально возможной степени на тех данных, которые он воспринимает. Начальная конфигурация агента может отражать некоторые предварительные знания о среде, но по мере приобретения агентом опыта эти знания могут модифицироваться и пополняться. Существуют крайние случаи, в которых среда полностью известна заранее. В подобных случаях агенту не требуется воспринимать информацию или обучаться; он просто сразу действует правильно. Безусловно, такие агенты являются весьма уязвимыми. Рассмотрим скромного навозного жука. Выкопав гнездо и отложив яйца, он скатывает шарик навоза, набрав его из ближайшей навозной кучи, чтобы заткнуть вход в гнездо. Если шарик навоза будет удален непосредственно перед тем, как жук его схватит, жук продолжает манипулировать им и изображает такую пантомиму, как будто он затыкает гнездо несуществующим шариком навоза, даже не замечая, что этот шарик отсутствует. В результате эволюции поведение этого жука было сформировано на основании определенного предположения, а если это предположение нарушается, то за этим следует безуспешное поведение. Немного более интеллектуальными являются осы-сфексы. Самка сфекса выкапывает норку, выходит из нее, жалит гусеницу и затаскивает ее в норку, затем снова выходит из норки, чтобы проверить, все ли в порядке, вытаски-

вает гусеницу наружу и откладывает в нее яйца. Гусеница служит в качестве источника питания во время развития яиц. До сих пор все идет хорошо, но если энтомолог переместит гусеницу на несколько дюймов в сторону, пока сфекс выполняет свою проверку, это насекомое снова возвращается к этапу "перетаскивания" своего плана и продолжает выполнять план без изменений, даже после десятков вмешательств в процедуру перемещения гусеницы. Оса-сфекс не способна обучиться действовать в такой ситуации, когда ее врожденный план нарушается, и поэтому не может его изменить.

В успешно действующих агентах задача вычисления функции агента разбивается на три отдельных периода: при проектирования агента некоторые вычисления осуществляются его проектировщиками; дополнительные вычисления агент производит, выбирая одно из своих очередных действий; а по мере того как агент учится на основании опыта, он осуществляет другие вспомогательные вычисления для принятия решения о том, как модифицировать свое поведение.

Если степень, в которой агент полагается на априорные знания своего проектировщика, а не на свои восприятия, слишком высока, то такой агент рассматривается как обладающий недостаточной **автономностью**. Рациональный агент должен быть автономным — он должен обучаться всему, что может освоить, для компенсации неполных или неправильных априорных знаний. Например, агент-пылесос, который обучается прогнозированию того, где и когда появится дополнительный мусор, безусловно, будет работать лучше, чем тот агент, который на это не способен. С точки зрения практики агенту редко предъявляется требование, чтобы он был полностью автономным с самого начала: если агент имеет мало опыта или вообще не имеет опыта, то вынужден действовать случайным образом, если проектировщик не оказал ему определенную помощь. Поэтому, как и эволюция предоставила животным достаточное количество врожденных рефлексов, позволяющих им прожить после рождения настолько долго, чтобы успеть обучиться самостоятельно, так и искусственному интеллектуальному агенту было бы разумно предоставить некоторые начальные знания, а не только наделить его способностью обучаться. После достаточного опыта существования в своей среде поведение рационального агента может по сути стать независимым от его априорных знаний. Поэтому включение в проект способности к обучению позволяет проектировать простых рациональных агентов, которые могут действовать успешно в исключительно разнообразных вариантах среды.

2.3. Определение характера среды

Теперь, после разработки определения рациональности, мы почти готовы приступить к созданию рациональных агентов. Но вначале необходимо определить, чем является **проблемная среда**, по сути представляющая собой "проблему", для которой рациональный агент служит "решением". Начнем с демонстрации того, как определить проблемную среду, и проиллюстрируем этот процесс на ряде примеров. Затем в этом разделе будет показано, что проблемная среда может иметь целый ряд разновидностей. Выбор проекта, наиболее подходящего для программы конкретного агента, непосредственно зависит от рассматриваемой разновидности проблемной среды.

Определение проблемной среды

В приведенном выше исследовании рациональности простого агента-пылесоса нам пришлось определить показатели производительности, среду, а также исполнительные механизмы и датчики агента. Сгруппируем описание всех этих факторов под заголовком **проблемная среда**. Для тех, кто любит аббревиатуры, авторы сокращенно обозначили соответствующее описание как **PEAS** (Performance, Environment, Actuators, Sensors — производительность, среда, исполнительные механизмы, датчики). Первый этап проектирования любого агента всегда должен состоять в определении проблемной среды с наибольшей возможной полнотой.

Пример, в котором рассматривался мир пылесоса, был несложным; теперь рассмотрим более сложную проблему — создание автоматизированного водителя такси. Этот пример будет использоваться во всей оставшейся части данной главы. Прежде чем читатель почувствует тревогу за безопасность будущих пассажиров, хотим сразу же отметить, что задача создания полностью автоматизированного водителя такси в настоящее время все еще выходит за пределы возможностей существующей технологии. (См. с. 69, где приведено описание существующего робота-водителя; с состоянием дел в этой области можно также ознакомиться по трудам конференций, посвященных интеллектуальным транспортным системам, в названиях которых есть слова *Intelligent Transportation Systems*.) Полное решение проблемы вождения автомобиля является чрезвычайно трудоемким, поскольку нет предела появлению все новых и новых комбинаций обстоятельств, которые могут возникать в процессе вождения; это еще одна из причин, по которой мы выбрали данную проблему для обсуждения. В табл. 2.2 приведено итоговое описание PEAS для проблемной среды вождения такси. Каждый из элементов этого описания рассматривается более подробно в настоящей главе.

Таблица 2.2. Описание PEAS проблемной среды для автоматизированного водителя такси

Тип агента	Показатели производительности	Среда	Исполнительные механизмы	Датчики
Водитель такси	Безопасная, быстрая, комфортная езда в рамках правил дорожного движения, максимизация прибыли	Дороги, другие транспортные средства, пешеходы, клиенты	Рулевое управление, акселератор, тормоз, световые сигналы, клаксон, дисплей	Видеокамеры, ультразвуковой дальномер, спидометр, глобальная система навигации и определения положения, одометр, акселерометр, датчики двигателя, клавиатура

Прежде всего необходимо определить **показатели производительности**, которыми мы могли бы стимулировать деятельность нашего автоматизированного водителя. К желаемым качествам относится успешное достижение нужного места назначения; минимизация потребления топлива, износа и старения; минимизация продолжительности и/или стоимости поездки; минимизация количества нарушений правил дорожного движения и помех другим водителям; максимизация безопасности и комфорта пассажиров; максимизация прибыли. Безусловно, что некоторые из этих целей конфликтуют, поэтому должны рассматриваться возможные компромиссы.

Затем рассмотрим, в чем состоит **среда вождения**, в которой действует такси. Любому водителю такси приходится иметь дело с самыми различными дорогами, начи-

ная с проселков и узких городских переулков и заканчивая автострадами с двенадцатью полосами движения. На дороге встречаются другие транспортные средства, беспризорные животные, пешеходы, рабочие, производящие дорожные работы, полицейские автомобили, лужи и выбоины. Водителю такси приходится также иметь дело с потенциальными и действительными пассажирами. Кроме того, имеется еще несколько важных дополнительных факторов. Таксисту может выпасть участь работать в Южной Калифорнии, где редко возникает такая проблема, как снег, или на Аляске, где снега на дорогах не бывает очень редко. Может оказаться, что водителю всю жизнь придется ездить по правой стороне или от него может потребоваться, чтобы он сумел достаточно успешно приспособиться к езде по левой стороне во время пребывания в Британии или в Японии. Безусловно, чем более ограниченной является среда, тем проще задача проектирования.

Исполнительные механизмы, имеющиеся в автоматизированном такси, должны быть в большей или меньшей степени такими же, как и те, которые находятся в распоряжении водителя-человека: средства управления двигателем с помощью акселератора и средства управления вождением с помощью руля и тормозов. Кроме того, для него могут потребоваться средства вывода на экран дисплея или синтеза речи для передачи ответных сообщений пассажирам и, возможно, определенные способы общения с водителями других транспортных средств, иногда вежливого, а иногда и не совсем.

Для достижения своих целей в данной среде вождения таксисту необходимо будет знать, где он находится, кто еще едет по этой дороге и с какой скоростью движется он сам. Поэтому в число его основных **датчиков** должны входить одна или несколько управляемых телевизионных камер, спидометр и одометр. Для правильного управления автомобилем, особенно на поворотах, в нем должен быть предусмотрен акселерометр; водителю потребуется также знать механическое состояние автомобиля, поэтому для него будет нужен обычный набор датчиков для двигателя и электрической системы. Автоматизированный водитель может также иметь приборы, недоступные для среднего водителя-человека: спутниковую глобальную систему навигации и определения положения (Global Positioning System — GPS) для получения точной информации о местонахождении по отношению к электронной карте, а также инфракрасные или ультразвуковые датчики для измерения расстояний до других автомобилей и препятствий. Наконец, ему потребуется клавиатура или микрофон для пассажиров, чтобы они могли указать место своего назначения.

В табл. 2.3 кратко перечислены основные элементы PEAS для целого ряда других типов агентов. Дополнительные примеры приведены в упр. 2.5. Некоторым читателям может показаться удивительным то, что авторы включили в этот список типов агентов некоторые программы, которые функционируют в полностью искусственной среде, ограничиваемой вводом с клавиатуры и выводом символов на экран. Кое-кто мог бы сказать: "Разумеется, это же не реальная среда, не правда ли?" В действительности суть состоит не в различиях между "реальными" и "искусственными" вариантами среды, а в том, какова сложность связей между поведением агента, последовательностью актов восприятия, вырабатываемой этой средой, и показателями производительности. Некоторые "реальные" варианты среды фактически являются чрезвычайно простыми. Например, для робота, предназначенного для контроля деталей, проходящих мимо него на ленточном конвейере, может использоваться целый ряд упрощающих допущений, например, что освещение всегда включено, что единственными предме-

тами на ленте конвейера являются детали того типа, который ему известен, и что существуют только два действия (принять изделие или забраковать его).

Таблица 2.3. Примеры типов агентов и их описаний PEAS

Тип агента	Показатели производительности	Среда	Исполнительные механизмы	Датчики
Медицинская диагностическая система	Успешное излечение пациента, минимизация затрат, отсутствие поводов для судебных тяжб	Пациент, больница, персонал	Вывод вопросов, тестов, диагнозов, рекомендаций, направлений	Ввод с клавиатуры симптомов, результатов лабораторных исследований, ответов пациента
Система анализа изображений, полученных со спутника	Правильная классификация изображения	Канал передачи данных от орбитального спутника	Вывод на дисплей результатов классификации определенного фрагмента изображения	Массивы пикселов с данными о цвете
Робот-сортировщик деталей	Процентные показатели безошибочной сортировки по лоткам	Ленточный конвейер с движущимися на нем деталями; лотки	Шарнирный манипулятор и захват	Видеокамера, датчики углов поворота шарниров
Контроллер очистительной установки	Максимизация степени очистки, продуктивности, безопасности	Очистительная установка, операторы	Клапаны, насосы, нагреватели, дисплеи	Температура, давление, датчики химического состава
Интерактивная программа обучения английскому языку	Максимизация оценок студентов на экзаменах	Множество студентов, экзаменационное агентство	Вывод на дисплей упражнений, рекомендаций, исправлений	Ввод с клавиатуры

В отличие от этого некоторые **программные агенты** (называемые также **программными роботами** или **софтботами**) существуют в сложных, неограниченных проблемных областях. Представьте себе программный робот, предназначенный для управления тренажером, имитирующим крупный пассажирский самолет. Этот тренажер представляет собой очень детально промоделированную, сложную среду, в которой имитируются движения других самолетов и работа наземных служб, а программный агент должен выбирать в реальном времени наиболее целесообразные действия из широкого диапазона действий. Еще одним примером может служить программный робот, предназначенный для просмотра источников новостей в Internet и показа клиентам интересующих их сообщений. Для успешной работы ему требуются определенные способности к обработке текста на естественном языке, он должен в процессе обучения определять, что интересует каждого заказчика, а также должен уметь изменять свои планы динамически, допустим, когда соединение с каким-либо из источников новостей закрывается или в оперативный режим переходит новый источник новостей. Internet представляет собой среду, которая по своей сложности соперничает с физическим миром, а в число обитателей этой сети входит много искусственных агентов.

Свойства проблемной среды

Несомненно, что разнообразие вариантов проблемной среды, которые могут возникать в искусственном интеллекте, весьма велико. Тем не менее существует возможность определить относительно небольшое количество измерений, по которым могут быть классифицированы варианты проблемной среды. Эти измерения в значительной степени определяют наиболее приемлемый проект агента и применимость каждого из основных семейств методов для реализации агента. Вначале в этом разделе будет приведен список измерений, а затем проанализировано несколько вариантов проблемной среды для иллюстрации этих идей. Приведенные здесь определения являются неформальными; более точные утверждения и примеры вариантов среды каждого типа описаны в следующих главах.

- **Полностью наблюдаемая** или **частично наблюдаемая**

  Если датчики агента предоставляют ему доступ к полной информации о состоянии среды в каждый момент времени, то такая проблемная среда называется полностью наблюдаемой[4]. По сути, проблемная среда является полностью наблюдаемой, если датчики выявляют все данные, которые являются релевантными для выбора агентом действия; релевантность, в свою очередь, зависит от показателей производительности. Полностью наблюдаемые варианты среды являются удобными, поскольку агенту не требуется поддерживать какое-либо внутреннее состояние для того, чтобы быть в курсе всего происходящего в этом мире. Среда может оказаться частично наблюдаемой из-за создающих шум и неточных датчиков или из-за того, что отдельные характеристики ее состояния просто отсутствуют в информации, полученной от датчиков; например, агент-пылесос, в котором имеется только локальный датчик мусора, не может определить, имеется ли мусор в других квадратах, а автоматизированный водитель такси не имеет сведений о том, какие маневры намереваются выполнить другие водители.

- **Детерминированная** или **стохастическая**

  Если следующее состояние среды полностью определяется текущим состоянием и действием, выполненным агентом, то такая среда называется детерминированной; в противном случае она является стохастической. В принципе в полностью наблюдаемой детерминированной среде агенту не приходится действовать в условиях неопределенности. Но если среда — частично наблюдаемая, то может создаться впечатление, что она является стохастической. Это отчасти справедливо, если среда — сложная и агенту нелегко следить за всеми ее ненаблюдаемыми аспектами. В связи с этим часто бывает более удобно классифицировать среду как детерминированную или стохастическую с точки зрения агента. Очевидно, что при такой трактовке среда вождения такси является стохастической, поскольку никто не может точно предсказать поведение всех других транспортных средств; более того, в любом автомобиле совершенно неожиданно может произойти прокол шины или остановка двигателя.

[4] В первом издании этой книги использовались термины **доступная** среда и **недоступная** среда вместо терминов **полностью наблюдаемая** среда и **частично наблюдаемая** среда; **недетерминированная** вместо **стохастической** и **неэпизодическая** вместо **последовательной**. Применяемая в этом издании новая терминология более совместима со сложившейся в этой области терминологией.

Описанный здесь мир пылесоса является детерминированным, но другие варианты этой среды могут включать стохастические элементы, такие как случайное появление мусора и ненадежная работа механизма всасывания (см. упр. 2.12). Если среда является детерминированной во всех отношениях, кроме действий других агентов, то авторы данной книги называют эту среду **стратегической**.

- **Эпизодическая** или **последовательная**[5]

  В эпизодической проблемной среде опыт агента состоит из неразрывных эпизодов. Каждый эпизод включает в себя восприятие среды агентом, а затем выполнение одного действия. При этом крайне важно то, что следующий эпизод не зависит от действий, предпринятых в предыдущих эпизодах. В эпизодических вариантах среды выбор действия в каждом эпизоде зависит только от самого эпизода. Эпизодическими являются многие задачи классификации. Например, агент, который должен распознавать дефектные детали на сборочной линии, формирует каждое решение применительно к текущей детали, независимо от предыдущих решений; более того, от текущего решения не зависит то, будет ли определена как дефектная следующая деталь. С другой стороны, в последовательных вариантах среды текущее решение может повлиять на все будущие решения. Последовательными являются такие задачи, как игра в шахматы и вождение такси: в обоих случаях кратковременные действия могут иметь долговременные последствия. Эпизодические варианты среды гораздо проще по сравнению с последовательными, поскольку в них агенту не нужно думать наперед.

- **Статическая** или **динамическая**

  Если среда может измениться в ходе того, как агент выбирает очередное действие, то такая среда называется динамической для данного агента; в противном случае она является статической. Действовать в условиях статической среды проще, поскольку агенту не требуется наблюдать за миром в процессе выработки решения о выполнении очередного действия, к тому же агенту не приходится беспокоиться о том, что он затрачивает на размышления слишком много времени. Динамические варианты среды, с другой стороны, как бы непрерывно спрашивают агента, что он собирается делать, а если он еще ничего не решил, то это рассматривается как решение ничего не делать. Если с течением времени сама среда не изменяется, а изменяются показатели производительности агента, то такая среда называется **полудинамической**. Очевидно, что среда вождения такси является динамической, поскольку другие автомобили и само такси продолжают движение и в ходе того, как алгоритм вождения определяет, что делать дальше. Игра в шахматы с контролем времени является полудинамической, а задача решения кроссворда — статической.

- **Дискретная** или **непрерывная**

  Различие между дискретными и непрерывными вариантами среды может относиться к состоянию среды, способу учета времени, а также восприятиям и действиям агента. Например, такая среда с дискретными состояниями, как

[5] Термин "последовательный" используется также в компьютерных науках как антоним термина "параллельный". Соответствующие два толкования фактически не связаны друг с другом.

игра в шахматы, имеет конечное количество различимых состояний. Кроме того, игра в шахматы связана с дискретным множеством восприятий и действий. Вождение такси — это проблема с непрерывно меняющимся состоянием и непрерывно текущим временем, поскольку скорость и местонахождение самого такси и других транспортных средств изменяются в определенном диапазоне непрерывных значений, причем эти изменения происходят во времени плавно. Действия по вождению такси также являются непрерывными (непрерывная регулировка угла поворота руля и т.д.). Строго говоря, входные данные от цифровых камер поступают дискретно, но обычно рассматриваются как представляющие непрерывно изменяющиеся скорости и местонахождения.

- **Одноагентная** или **мультиагентная**

  Различие между одноагентными и мультиагентными вариантами среды на первый взгляд может показаться достаточно простым. Например, очевидно, что агент, самостоятельно решающий кроссворд, находится в одноагентной среде, а агент, играющий в шахматы, действует в двухагентной среде. Тем не менее при анализе этого классификационного признака возникают некоторые нюансы. Прежде всего, выше было описано, на каком основании некоторая сущность *может* рассматриваться как агент, но не было указано, какие сущности *должны* рассматриваться как агенты. Должен ли агент *A* (например, водитель такси) считать агентом объект *B* (другой автомобиль), или может относиться к нему просто как к стохастически действующему объекту, который можно сравнить с волнами, набегающими на берег, или с листьями, трепещущими на ветру? Ключевое различие состоит в том, следует ли или не следует описывать поведение объекта *B* как максимизирующее личные показатели производительности, значения которых зависят от поведения агента *A*. Например, в шахматах соперничающая сущность *B* пытается максимизировать свои показатели производительности, а это по правилам шахмат приводит к минимизации показателей производительности агента *A*. Таким образом, шахматы — это **конкурентная** мультиагентная среда. А в среде вождения такси, с другой стороны, предотвращение столкновений максимизирует показатели производительности всех агентов, поэтому она может служить примером частично **кооперативной** мультиагентной среды. Она является также частично конкурентной, поскольку, например, парковочную площадку может занять только один автомобиль. Проблемы проектирования агентов, возникающие в мультиагентной среде, часто полностью отличаются от тех, с которыми приходится сталкиваться в одноагентных вариантах среды; например, одним из признаков рационального поведения в мультиагентной среде часто бывает **поддержка связи**, а в некоторых вариантах частично наблюдаемой конкурентной среды рациональным становится **стохастическое поведение**, поскольку оно позволяет избежать *ловушек предсказуемости*.

Как и следует ожидать, наиболее сложными вариантами среды являются частично наблюдаемые, стохастические, последовательные, динамические, непрерывные и мультиагентные. Кроме того, часто обнаруживается, что многие реальные ситуации являются настолько сложными, что неясно даже, действительно ли их можно считать детерминированными. С точки зрения практики их следует рассматривать как стохастические. Проблема вождения такси является сложной во всех указанных отношениях.

В табл. 2.4 перечислены свойства многих известных вариантов среды. Следует отметить, что в отдельных случаях приведенные в ней описания являются слишком краткими и сухими. Например, в ней указано, что шахматы — это полностью наблюдаемая среда, но строго говоря, это утверждение является ложным, поскольку некоторые правила, касающиеся рокировки, взятия пешки на проходе и объявления ничьи при повторении ходов, требуют запоминания определенных фактов об истории игры, которые нельзя выявить из анализа позиции на доске. Но эти исключения из определения наблюдаемости, безусловно, являются незначительными по сравнению с теми необычными ситуациями, с которыми сталкивается автоматизированный водитель такси, интерактивная система преподавания английского языка или медицинская диагностическая система.

Таблица 2.4. Примеры вариантов проблемной среды и их характеристик

Проблемная среда	Наблюдаемая полностью или частично	Детерминированная, стратегическая или стохастическая	Эпизодическая или последовательная	Статическая, динамическая или полудинамическая	Дискретная или непрерывная	Одноагентная или мультиагентная
Решение кроссворда	Полностью наблюдаемая	Детерминированная	Последовательная	Статическая	Дискретная	Одноагентная
Игра в шахматы с контролем времени	Полностью наблюдаемая	Стохастическая	Последовательная	Полудинамическая	Дискретная	Мультиагентная
Игра в покер	Частично наблюдаемая	Стохастическая	Последовательная	Статическая	Дискретная	Мультиагентная
Игра в нарды	Полностью наблюдаемая	Стохастическая	Последовательная	Статическая	Дискретная	Мультиагентная
Вождение такси	Частично наблюдаемая	Стохастическая	Последовательная	Динамическая	Непрерывная	Мультиагентная
Медицинская диагностика	Частично наблюдаемая	Стохастическая	Последовательная	Динамическая	Непрерывная	Одноагентная
Анализ изображений	Полностью наблюдаемая	Детерминированная	Эпизодическая	Полудинамическая	Непрерывная	Одноагентная
Робот-сортировщик деталей	Частично наблюдаемая	Стохастическая	Эпизодическая	Динамическая	Непрерывная	Одноагентная
Контроллер очистительной установки	Частично наблюдаемая	Стохастическая	Последовательная	Динамическая	Непрерывная	Одноагентная
Интерактивная программа, обучающая английскому языку	Частично наблюдаемая	Стохастическая	Последовательная	Динамическая	Дискретная	Мультиагентная

Некоторые другие ответы в этой таблице зависят от того, как определена проблемная среда. Например, в ней задача медицинского диагноза определена как одноагентная, поскольку сам процесс развития заболевания у пациента нецелесообразно моделировать в качестве агента, но системе медицинской диагностики иногда приходится сталкиваться с пациентами, не желающими принимать ее рекомендации, и со скептически настроенным персоналом, поэтому ее среда может иметь мультиагентный аспект. Кроме того, медицинская диагностика является эпизодической, если она рассматривается как задача выбора диагноза на основе анализа перечня симптомов, но эта проблема становится последовательной, если решаемая при этом задача может включать выработку рекомендаций по выполнению ряда лабораторных исследований, оценку прогресса в ходе лечения и т.д. К тому же многие варианты среды являются эпизодическими на более высоких уровнях по сравнению с отдельными действиями агента. Например, шахматный турнир состоит из ряда игр; каждая игра является эпизодом, поскольку (вообще говоря) от ходов, сделанных в предыдущей игре, не зависит то, как повлияют на общую производительность агента ходы, сделанные им в текущей игре. С другой стороны, принятие решений в одной и той же игре, безусловно, происходит последовательно.

Репозитарий кода, который относится к данной книге (`aima.cs.berkeley.edu`), включает реализации многих вариантов среды, наряду с имитатором среды общего назначения, который помещает одного или нескольких агентов в моделируемую среду, наблюдает за их поведением в течение определенного времени и оценивает их действия в соответствии с заданными показателями производительности. Такие эксперименты часто выполняются применительно не к одному варианту среды, а ко многим вариантам, сформированным на основе некоторого **класса вариантов среды**. Например, чтобы оценить действия водителя такси в моделируемой ситуации дорожного движения, может потребоваться провести много сеансов моделирования с различными условиями трафика, освещения и погоды. Если бы мы спроектировали этого агента для одного сценария, то могли бы лучше воспользоваться специфическими свойствами данного конкретного случая, но не имели бы возможности определить приемлемый проект решения задачи автоматизированного вождения в целом. По этой причине репозитарий кода включает также **генератор вариантов среды** для каждого класса вариантов среды; этот генератор выбирает определенные варианты среды (с некоторой вероятностью), в которых выполняется проверка агента. Например, генератор вариантов среды пылесоса инициализирует случайным образом такие исходные данные, как распределение мусора и местонахождение агента. Дело в том, что наибольший интерес представляет то, какую среднюю производительность будет иметь данный конкретный агент в некотором классе вариантов среды. Рациональный агент для определенного класса вариантов среды максимизирует свою среднюю производительность. Процесс разработки класса вариантов среды и оценки в них различных агентов иллюстрируется в упр. 2.7–2.12.

2.4. Структура агентов

До сих пор в этой книге свойства агентов рассматривались на основании анализа их поведения — действий, выполняемых агентом после получения любой заданной последовательности актов восприятия. Теперь нам поневоле придется сменить тему

и перейти к описанию того, как организовано их внутреннее функционирование. Задача искусственного интеллекта состоит в разработке **программы агента**, которая реализует функцию агента, отображая восприятия на действия. Предполагается, что эта программа должна работать в своего рода вычислительном устройстве с физическими датчиками и исполнительными механизмами; в целом эти компоненты именуются в данной книге **архитектурой**, а структура агента условно обозначается следующей формулой:

```
Агент = Архитектура + Программа
```

Очевидно, что выбранная программа должна быть подходящей для этой архитектуры. Например, если в программе осуществляется выработка рекомендаций по выполнению таких действий, как `Walk` (ходьба), то в архитектуре целесообразно предусмотреть использование опорно-двигательного аппарата. Архитектура может представлять собой обычный персональный компьютер или может быть воплощена в виде роботизированного автомобиля с несколькими бортовыми компьютерами, видеокамерами и другими датчиками. Вообще говоря, архитектура обеспечивает передачу в программу результатов восприятия, полученных от датчиков, выполнение программы и передачу исполнительным механизмам вариантов действий, выбранных программой, по мере их выработки. Основная часть данной книги посвящена проектированию программ агентов, а главы 24 и 25 касаются непосредственно датчиков и исполнительных механизмов.

Программы агентов

Все программы агентов, которые будут разработаны в этой книге, имеют одну и ту же структуру: они принимают от датчиков в качестве входных данных результаты текущего восприятия и возвращают исполнительным механизмам выбранный вариант действия[6]. Необходимо указать на различие между программой агента, которая принимает в качестве входных данных результаты текущего восприятия, и функцией агента, которая принимает на входе всю историю актов восприятия. Программа агента получает в качестве входных данных только результаты текущего восприятия, поскольку больше ничего не может узнать из своей среды; если действия агента зависят от всей последовательности актов восприятия, то агент должен сам запоминать результаты этих актов восприятия.

Для описания программ агентов будет применяться простой язык псевдокода, который определен в приложении Б. (Оперативный репозитарий кода содержит реализации на реальных языках программирования.) Например, в листинге 2.1 показана довольно несложная программа агента, которая регистрирует последовательность актов восприятия, а затем использует полученную последовательность для доступа по индексу к таблице действий и определения того, что нужно сделать. Таблица явно отображает функцию агента, воплощаемую данной программой агента. Чтобы создать рационального агента таким образом, проектировщики должны сформировать

[6] Существуют и другие варианты структуры программы агента, например, можно было бы использовать в виде программ агентов **сопроцедуры**, которые действуют асинхронно со средой. Каждая такая сопроцедура имеет входной и выходной порты, а ее работа организована в виде цикла, в котором из входного порта считываются результаты восприятий, а в выходной порт записываются варианты действий.

таблицу, которая содержит подходящее действие для любой возможной последовательности актов восприятия.

Листинг 2.1. Программа `Table-Driven-Agent`, которая вызывается после каждого восприятия новых данных и каждый раз возвращает вариант действия; программа регистрирует последовательность актов восприятия с использованием своей собственной закрытой структуры данных

```
function Table-Driven-Agent(percept) returns действие action
    static: percepts, последовательность актов восприятия,
                      первоначально пустая
            table, таблица действий, индексированная по
                   последовательностям актов восприятия и
                   полностью заданная с самого начала

    добавить результаты восприятия percept к концу
        последовательности percepts
    action ← Lookup(percepts, table)
    return action
```

Анализ того, почему такой подход к созданию агента, основанный на использовании таблицы, обречен на неудачу, является весьма поучительным. Допустим, что $\mathcal{P}$ — множество возможных актов восприятия, а T — срок существования агента (общее количество актов восприятия, которое может быть им получено). Поисковая таблица будет содержать

$$\sum_{t=1}^{T} |\mathcal{P}|^t \text{ записей.}$$

Рассмотрим автоматизированное такси: визуальные входные данные от одной телекамеры поступают со скоростью примерно 27 мегабайтов в секунду (30 кадров в секунду, 640×480 пикселов с 24 битами информации о цвете). Согласно этим данным поисковая таблица, рассчитанная на 1 час вождения, должна содержать количество записей, превышающее $10^{250\,000\,000\,000}$. И даже поисковая таблица для шахмат (крошечного, хорошо изученного фрагмента реального мира) имела бы, по меньшей мере, 10^{150} записей. Ошеломляющий размер этих таблиц (притом что количество атомов в наблюдаемой вселенной не превышает 10^{80}) означает, что, во-первых, ни один физический агент в нашей вселенной не имеет пространства для хранения такой таблицы, во-вторых, проектировщик не сможет найти достаточно времени для создания этой таблицы, в-третьих, ни один агент никогда не сможет обучиться тому, что содержится во всех правильных записях этой таблицы, на основании собственного опыта, и, в-четвертых, даже если среда достаточно проста для того, чтобы можно было создать таблицу приемлемых размеров, все равно у проектировщика нет руководящих сведений о том, как следует заполнять записи подобной таблицы.

Несмотря на все сказанное, программа `Table-Driven-Agent` выполняет именно то, что от нее требуется: она реализует желаемую функцию агента. Основная сложность, стоящая перед искусственным интеллектом как научным направлением, состоит в том, чтобы узнать, как создавать программы, которые в рамках возможного вырабатывают рациональное поведение с использованием небольшого объема кода, а не большого количества записей таблицы. Существует множество примеров, показывающих, что такая задача может быть выполнена успешно в других областях; например, огромные таблицы квадратных корней, использовавшиеся инженерами и

школьниками до 1970-х годов, теперь заменены работающей в электронных калькуляторах программой из пяти строк, в которой применяется метод Ньютона. Вопрос заключается в том, может ли искусственный интеллект сделать для интеллектуального поведения в целом то, что Ньютон сделал для упрощения вычисления квадратных корней? Авторы данной книги полагают, что ответ на этот вопрос является положительным.

В остальной части этого раздела рассматриваются четыре основных вида программ агентов, которые воплощают принципы, лежащие в основе почти всех интеллектуальных систем:

- простые рефлексные агенты;
- рефлексные агенты, основанные на модели;
- агенты, действующие на основе цели;
- агенты, действующие на основе полезности.

Затем приведено описание в общих терминах того, как преобразовать агентов всех этих типов в обучающихся агентов.

Простые рефлексные агенты

Простейшим видом агента является **простой рефлексный агент**. Подобные агенты выбирают действия на основе текущего акта восприятия, игнорируя всю остальную историю актов восприятия. Например, агент-пылесос, для которого результаты табуляции функции агента приведены в табл. 2.1, представляет собой простой рефлексный агент, поскольку его решения основаны только на информации о текущем местонахождении и о том, содержит ли оно мусор. Программа для данного агента приведена в листинге 2.2.

Листинг 2.2. Программа простого рефлексного агента в среде пылесоса с двумя состояниями. Эта программа реализует функцию агента, которая табулирована в табл. 2.1

```
function Reflex-Vacuum-Agent([location,status]) returns действие
action

    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

Обратите внимание на то, что эта программа агента-пылесоса действительно очень мала по сравнению с соответствующей таблицей. Наиболее очевидное сокращение обусловлено тем, что в ней игнорируется история актов восприятия, в результате чего количество возможных вариантов сокращается от 4^T просто до 4. Дополнительное небольшое сокращение обусловлено тем фактом, что если в текущем квадрате имеется мусор, то выполняемое при этом действие не зависит от местонахождения пылесоса.

Представьте себя на месте водителя автоматизированного такси. Если движущийся впереди автомобиль тормозит и загораются его тормозные огни, то вы должны заметить это и тоже начать торможение. Иными словами, над визуальными входными данными выполняется определенная обработка для выявления условия,

которое обозначается как "*car-in-front-is-braking*" (движущийся впереди автомобиль тормозит). Затем это условие активизирует некоторую связь с действием "*initiate-braking*" (начать торможение), установленную в программе агента. Такая связь называется **правилом условие–действие**[7] и записывается следующим образом:

if *car-in-front-is-braking* **then** *initiate-braking*

Люди также используют большое количество таких связей, причем некоторые из них представляют собой сложные отклики, освоенные в результате обучения (как при вождении автомобиля), а другие являются врожденными рефлексами (такими как моргание, которое происходит при приближении к глазу постороннего предмета). В разных главах данной книги будет показано несколько различных способов, с помощью которых можно организовать обучение агента и реализацию таких связей.

Программа, приведенная в листинге 2.2, специализирована для одной конкретной среды пылесоса. Более общий и гибкий подход состоит в том, чтобы вначале создать интерпретатор общего назначения для правил условие–действие, а затем определить наборы правил для конкретной проблемной среды. На рис. 2.3 приведена структура такой общей программы в схематической форме и показано, каким образом правила условие–действие позволяют агенту создать связь от восприятия к действию. (Не следует беспокоиться, если такой способ покажется тривиальным; вскоре он обнаружит намного более интересные возможности.) В подобных схемах для обозначения текущего внутреннего состояния процесса принятия решения агентом используются прямоугольники, а для представления фоновой информации, применяемой в этом процессе, служат овалы. Программа этого агента, которая также является очень простой, приведена в листинге 2.3. Функция `Interpret-Input` вырабатывает абстрагированное описание текущего состояния по результатам восприятия, а функция `Rule-Match` возвращает первое правило во множестве правил, которое соответствует заданному описанию состояния. Следует отметить, что приведенное здесь изложение в терминах "правил" и "соответствия" является чисто концептуальным; фактические реализации могут быть настолько простыми, как совокупность логических элементов, реализующих логическую схему.

Листинг 2.3. Программа простого рефлексного агента, который действует согласно правилу, условие которого соответствует текущему состоянию, определяемому результатом восприятия

```
function Simple-Reflex-Agent(percept) returns действие action
    static: rules, множество правил условие-действие

    state ← Interpret-Input(percept)
    rule ← Rule-Match(state, rules)
    action ← Rule-Action[rule]
    return action
```

Простые рефлексные агенты характеризуются той замечательной особенностью, что они чрезвычайно просты, но зато обладают весьма ограниченным интеллектом. Агент, программа которого приведена в листинге 2.3, работает, ☞ *только если правильное решение может быть принято на основе исключительно текущего восприятия,*

[7] Эти связи называются также **правилами ситуация–действие**, **продукциями** или **правилами if-then**.

иначе говоря, только если среда является полностью наблюдаемой. Внесение даже небольшой доли ненаблюдаемости может вызвать серьезное нарушение его работы. Например, в приведенном выше правиле торможения принято предположение, что условие `car-in-front-is-braking` может быть определено из текущего восприятия (текущего видеоизображения), если движущийся автомобиль имеет тормозной сигнал, расположенный на центральном месте среди других сигналов. К сожалению, некоторые более старые модели имеют другие конфигурации задних фар, тормозных сигналов, габаритных огней, сигналов торможения и сигналов поворота, поэтому не всегда возможно определить из единственного изображения, тормозит ли этот автомобиль или нет. Простой рефлексный агент, ведущий свой автомобиль вслед за таким автомобилем, либо будет постоянно тормозить без всякой необходимости, либо, что еще хуже, вообще не станет тормозить.

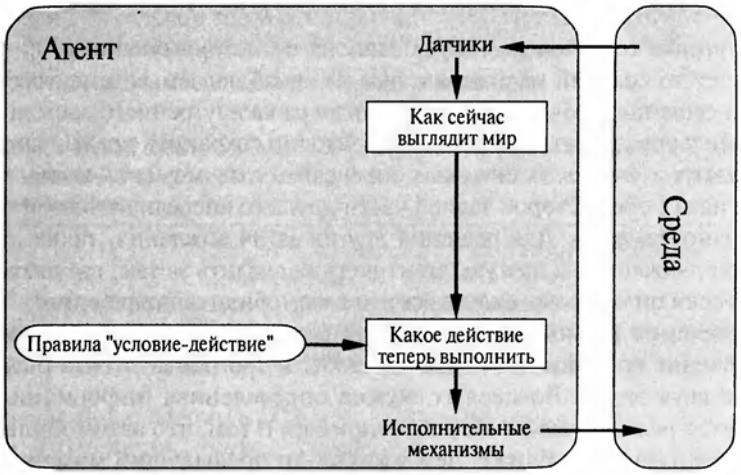

Рис. 2.3. Схематическое изображение структуры простого рефлексного агента

Возникновение аналогичной проблемы можно обнаружить и в мире пылесоса. Предположим, что в простом рефлексном агенте-пылесосе испортился датчик местонахождения и работает только датчик мусора. Такой агент получает только два возможных восприятия: `[Dirty]` и `[Clean]`. Он может выполнить действие `Suck` в ответ на восприятие `[Dirty]`, а что он должен делать в ответ на восприятие `[Clean]`? Выполнение движения `Left` завершится отказом (на неопределенно долгое время), если окажется, что он начинает это движение с квадрата `A`, а если он начинает движение с квадрата `B`, то завершится отказом на неопределенно долгое время движение `Right`. Для простых рефлексных агентов, действующих в частично наблюдаемых вариантах среды, часто бывают неизбежными бесконечные циклы.

Выход из бесконечных циклов становится возможным, если агент обладает способностью **рандомизировать** свои действия (вводить в них элемент случайности). Например, если агент-пылесос получает результат восприятия `[Clean]`, то может подбросить монету, чтобы выбрать между движениями `Left` и `Right`. Легко показать, что агент достигнет другого квадрата в среднем за два этапа. Затем, если в этом квадрате имеется мусор, то пылесос его уберет и задача очистки будет выполнена.

Поэтому рандомизированный простой рефлексный агент может превзойти по своей производительности детерминированного простого рефлексного агента.

В разделе 2.3 уже упоминалось, что в некоторых мультиагентных вариантах среды может оказаться рациональным рандомизированное поведение правильного типа. А в одноагентных вариантах среды рандомизация обычно не является рациональной. Это лишь полезный трюк, который помогает простому рефлексному агенту в некоторых ситуациях, но в большинстве случаев можно добиться гораздо большего с помощью более сложных детерминированных агентов.

Рефлексные агенты, основанные на модели

Наиболее эффективный способ организации работы в условиях частичной наблюдаемости состоит в том, чтобы агент отслеживал ту часть мира, которая воспринимается им в текущий момент. Это означает, что агент должен поддерживать своего рода **внутреннее состояние**, которое зависит от истории актов восприятия и поэтому отражает по крайней мере некоторые из ненаблюдаемых аспектов текущего состояния. Для решения задачи торможения поддержка внутреннего состояния не требует слишком больших затрат — для этого достаточно сохранить предыдущий кадр, снятый видеокамерой, чтобы агент мог определить тот момент, когда два красных световых сигнала с обеих сторон задней части идущего впереди автомобиля загораются или гаснут одновременно. Для решения других задач вождения, таких как переход с одной полосы движения на другую, агент должен следить за тем, где находятся другие автомобили, если он не может видеть все эти автомобили одновременно.

Для обеспечения возможности обновления этой внутренней информации о состоянии в течение времени необходимо, чтобы в программе агента были закодированы знания двух видов. Во-первых, нужна определенная информация о том, как мир изменяется независимо от агента, например, о том, что автомобиль, идущий на обгон, обычно становится ближе, чем в какой-то предыдущий момент. Во-вторых, требуется определенная информация о том, как влияют на мир собственные действия агента, например, что при повороте агентом рулевого колеса по часовой стрелке автомобиль поворачивает вправо или что после проезда по автомагистрали в течение пяти минут на север автомобиль находится на пять миль севернее от того места, где он был пять минут назад. Эти знания о том, "как работает мир" (которые могут быть воплощены в простых логических схемах или в сложных научных теориях) называются **моделью мира**. Агент, в котором используется такая модель, называется **агентом, основанным на модели**.

На рис. 2.4 приведена структура рефлексного агента, действующего с учетом внутреннего состояния, и показано, как текущее восприятие комбинируется с прежним внутренним состоянием для выработки обновленного описания текущего состояния. Программа такого агента приведена в листинге 2.4. В этом листинге интерес представляет функция `Update-State`, которая отвечает за создание нового описания внутреннего состояния. Эта функция не только интерпретирует результаты нового восприятия в свете существующих знаний о состоянии, но и использует информацию о том, как изменяется мир, для слежения за невидимыми частями мира, поэтому должна учитывать информацию о том, как действия агента влияют на состояние мира. Более подробные примеры приведены в главах 10 и 17.

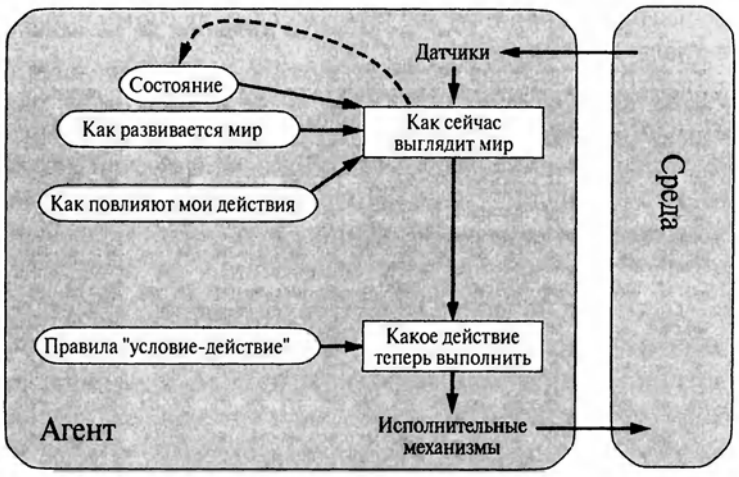

Рис. 2.4. Рефлексный агент, основанный на модели

Листинг 2.4. Рефлексный агент, основанный на модели, который следит за текущим состоянием мира с использованием внутренней модели, затем выбирает действие таким же образом, как и простой рефлексный агент

```
function Reflex-Agent-With-State(percept) returns действие action
    static: state, описание текущего состояния мира
            rules, множество правил условие-действие
            action, последнее по времени действие;
                    первоначально не определено

    state ← Update-State(state, action, percept)
    rule ← Rule-Match(state, rules)
    action ← Rule-Action[rule]
    return action
```

Агенты, основанные на цели

Знаний о текущем состоянии среды не всегда достаточно для принятия решения о том, что делать. Например, на перекрестке дорог такси может повернуть налево, повернуть направо или ехать прямо. Правильное решение зависит от того, куда должно попасть это такси. Иными словами, агенту требуется не только описание текущего состояния, но и своего рода информация о **цели**, которая описывает желаемые ситуации, такие как доставка пассажира в место назначения. Программа агента может комбинировать эту информацию с информацией о результатах возможных действий (с такой же информацией, как и та, что использовалась при обновлении внутреннего состояния рефлексного агента) для выбора действий, позволяющих достичь этой цели. Структура агента, действующего на основе цели, показана на рис. 2.5.

Иногда задача выбора действия на основе цели решается просто, когда достижение цели немедленно становится результатом единственного действия, а иногда эта задача становится более сложной, и агенту требуется рассмотреть длинные последовательности движений и поворотов, чтобы найти способ достижения цели. Подобластями искусственного интеллекта, посвященными выработке последовательно-

стей действий, позволяющих агенту достичь его целей, являются **поиск** (главы 3–6) и **планирование** (главы 11 и 12).

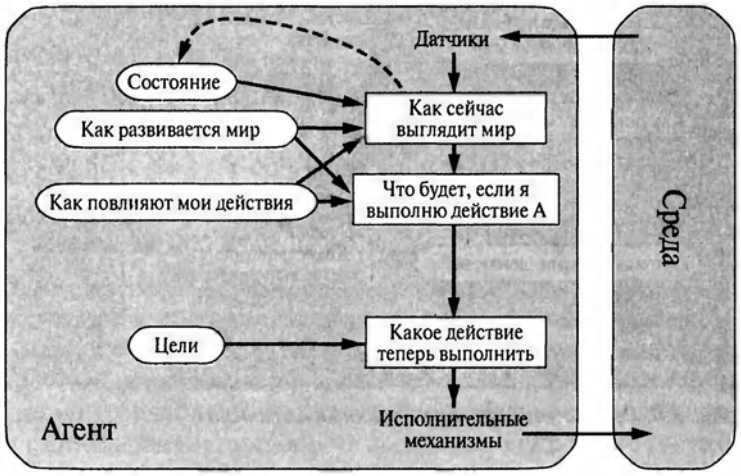

Рис. 2.5. Агент, основанный на модели и на цели. Он следит за состоянием мира, а также за множеством целей, которых он пытается достичь, и выбирает действие, позволяющее (в конечном итоге) добиться достижения этих целей

Следует учитывать, что процедура принятия решений такого рода имеет фундаментальные отличия от описанной выше процедуры применения правил условие–действие, поскольку в ней приходится размышлять о будущем, отвечая на два вопроса: "Что произойдет, если я сделаю то-то и то-то?" и "Позволит ли это мне достичь удовлетворения?" В проектах рефлексных агентов такая информация не представлена явно, поскольку встроенные правила устанавливают непосредственное соответствие между восприятиями и действиями. Рефлексный агент тормозит, увидев сигналы торможения движущего впереди автомобиля, а агент, основанный на цели, может рассудить, что если на движущемся впереди автомобиле загорелись тормозные огни, то он замедляет свое движение. Учитывая принцип, по которому обычно изменяется этот мир, для него единственным действием, позволяющим достичь такой цели, как предотвращение столкновения с другими автомобилями, является торможение.

Хотя на первый взгляд кажется, что агент, основанный на цели, менее эффективен, он является более гибким, поскольку знания, на которые опираются его решения, представлены явно и могут быть модифицированы. Если начинается дождь, агент может обновить свои знания о том, насколько эффективно теперь будут работать его тормоза; это автоматически вызывает изменение всех соответствующих правил поведения с учетом новых условий. Для рефлексного агента, с другой стороны, в таком случае пришлось бы переписать целый ряд правил условие–действие. Поведение агента, основанного на цели, можно легко изменить, чтобы направить его в другое место, а правила рефлексного агента, которые указывают, где поворачивать и где ехать прямо, окажутся применимыми только для единственного места назначения; для того чтобы этого агента можно было направить в другое место, все эти правила должны быть заменены.

Агенты, основанные на полезности

В действительности в большинстве вариантов среды для выработки высококачественного поведения одного лишь учета целей недостаточно. Например, обычно существует много последовательностей действий, позволяющих такси добраться до места назначения (и тем самым достичь поставленной цели), но некоторые из этих последовательностей обеспечивают более быструю, безопасную, надежную или недорогую поездку, чем другие. Цели позволяют провести лишь жесткое бинарное различие между состояниями "удовлетворенности" и "неудовлетворенности", тогда как более общие показатели производительности должны обеспечивать сравнение различных состояний мира в точном соответствии с тем, насколько удовлетворенным станет агент, если их удастся достичь. Поскольку понятие "удовлетворенности" представляется не совсем научным, чаще применяется терминология, согласно которой состояние мира, более предпочтительное по сравнению с другим, рассматривается как имеющее более высокую **полезность** для агента[8].

Функция полезности отображает состояние (или последовательность состояний) на вещественное число, которое обозначает соответствующую степень удовлетворенности агента. Полная спецификация функции полезности обеспечивает возможность принимать рациональные решения в описанных ниже двух случаях, когда этого не позволяют сделать цели. Во-первых, если имеются конфликтующие цели, такие, что могут быть достигнуты только некоторые из них (например, или скорость, или безопасность), то функция полезности позволяет найти приемлемый компромисс. Во-вторых, если имеется несколько целей, к которым может стремиться агент, но ни одна из них не может быть достигнута со всей определенностью, то функция полезности предоставляет удобный способ взвешенной оценки вероятности успеха с учетом важности целей.

В главе 16 будет показано, что любой рациональный агент должен вести себя так, как если бы он обладал функцией полезности, ожидаемое значение которой он пытается максимизировать. Поэтому агент, обладающий явно заданной функцией полезности, имеет возможность принимать рациональные решения и способен делать это с помощью алгоритма общего назначения, не зависящего от конкретной максимизируемой функции полезности. Благодаря этому "глобальное" определение рациональности (согласно которому рациональными считаются функции агента, имеющие наивысшую производительность) преобразуется в "локальное" ограничение на проекты рациональных агентов, которое может быть выражено в виде простой программы.

Структура агента, действующего с учетом полезности, показана на рис. 2.6. Программы агентов, действующих с учетом полезности, приведены в части V, которая посвящена проектированию агентов, принимающих решения, способных учитывать неопределенность, свойственную частично наблюдаемым вариантам среды.

---

[8] Термин "полезность" имеет англоязычный эквивалент "utility", который в данном контексте обозначает "свойство быть полезным", а не электростанцию или предприятие, предоставляющее коммунальные услуги.

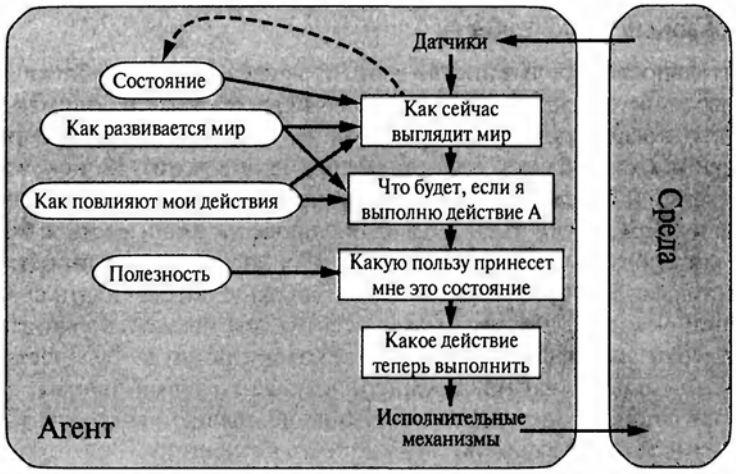

Рис. 2.6. Агент, основанный на модели и на полезности. В нем модель мира используется наряду с функцией полезности, которая измеряет предпочтения агента применительно к состояниям мира. Затем агент выбирает действие, которое ведет к наилучшей ожидаемой полезности. Для вычисления ожидаемой полезности выполняется усреднение по всем возможным результирующим состояниям с учетом коэффициента, определяющего вероятность каждого результата

Обучающиеся агенты

Выше были описаны программы агентов, в которых применяются различные методы выбора действий. Но до сих пор еще не были приведены сведения о том, как создаются программы агентов. В своей знаменитой ранней статье Тьюринг [1520] проанализировал идею о том, как фактически должно осуществляться программирование предложенных им интеллектуальных машин вручную. Он оценил объем работы, который для этого потребуется, и пришел к такому выводу: "Желательно было бы иметь какой-то более продуктивный метод". Предложенный им метод заключался в том, что необходимо создавать обучающиеся машины, а затем проводить их обучение. Теперь этот метод стал доминирующим методом создания наиболее современных систем во многих областях искусственного интеллекта. Как отмечалось выше, обучение имеет еще одно преимущество: оно позволяет агенту функционировать в первоначально неизвестных ему вариантах среды и становиться более компетентным по сравнению с тем, что могли бы позволить только его начальные знания. В данном разделе кратко представлены основные сведения об обучающихся агентах. Существующие возможности и методы обучения агентов конкретных типов рассматриваются почти в каждой главе данной книги, а в части VI более подробно описываются сами алгоритмы обучения.

Как показано на рис. 2.7, структура обучающегося агента может подразделяться на четыре концептуальных компонента. Наиболее важное различие наблюдается между **обучающим компонентом**, который отвечает за внесение усовершенствований, и **производительным компонентом**, который обеспечивает выбор внешних действий. Производительным компонентом является то, что до сих пор в данной книге рассматривалось в качестве всего агента: он получает воспринимаемую ин-

формацию и принимает решение о выполнении действий. Обучающий компонент использует информацию обратной связи от **критика** с оценкой того, как действует агент, и определяет, каким образом должен быть модифицирован производительный компонент для того, чтобы он успешнее действовал в будущем.

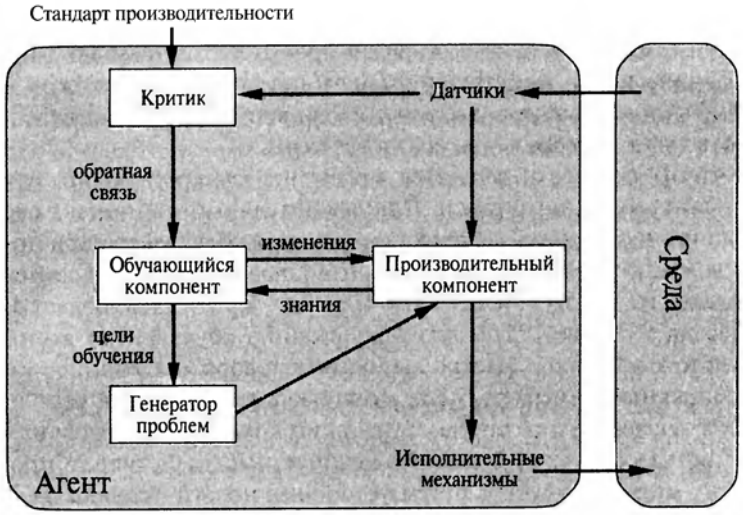

Рис. 2.7. Общая модель обучающихся агентов

Проект обучающего компонента во многом зависит от проекта производительного компонента. Осуществляя попытку спроектировать агента, который обучается определенным способностям, необходимо прежде всего стремиться найти ответ на вопрос: "Какого рода производительный компонент потребуется моему агенту после того, как он будет обучен тому, как выполнять свои функции?", а не на вопрос: "Как приступить к решению задачи обучения его выполнению этих функций?" После того как спроектирован сам агент, можно приступать к конструированию обучающих механизмов, позволяющих усовершенствовать любую часть этого агента.

Критик сообщает обучающему компоненту, насколько хорошо действует агент с учетом постоянного стандарта производительности. Критик необходим, поскольку сами результаты восприятия не дают никаких указаний на то, успешно ли действует агент. Например, шахматная программа может получить результаты восприятия, указывающие на то, что она поставила мат своему противнику, но ей требуется стандарт производительности, который позволил бы определить, что это — хороший результат; сами данные восприятия ничего об этом не говорят. Важно, чтобы стандарт производительности был постоянным. В принципе этот стандарт следует рассматривать как полностью внешний по отношению к агенту, поскольку агент не должен иметь возможности его модифицировать так, чтобы он в большей степени соответствовал его собственному поведению.

Последним компонентом обучающегося агента является **генератор проблем**. Его задача состоит в том, чтобы предлагать действия, которые должны привести к получению нового и информативного опыта. Дело в том, что если производительный компонент предоставлен самому себе, то продолжает выполнять действия, которые являются наилучшими с точки зрения того, что он знает. Но если агент готов

к тому, чтобы немного поэкспериментировать и в кратковременной перспективе выполнять действия, которые, возможно, окажутся не совсем оптимальными, то он может обнаружить гораздо более лучшие действия с точки зрения долговременной перспективы. Генератор проблем предназначен именно для того, чтобы предлагать такие исследовательские действия. Именно этим занимаются ученые, проводя эксперименты. Галилей не считал, что сбрасывание камней с вершины Пизанской башни является самоцелью. Он не старался просто вдрызг разбить эти булыжники или оказать физическое воздействие на головы неудачливых прохожих. Его замысел состоял в том, чтобы изменить взгляды, сложившиеся в его собственной голове, сформулировав лучшую теорию движения объектов.

Для того чтобы перевести весь этот проект на конкретную почву, вернемся к примеру автоматизированного такси. Производительный компонент состоит из той коллекции знаний и процедур, которая применяется водителем такси при выборе им действий по вождению. Водитель такси с помощью этого производительного компонента выезжает на дорогу и ведет свою машину. Критик наблюдает за миром и в ходе этого передает соответствующую информацию обучающему компоненту. Например, после того как такси быстро выполняет поворот налево, пересекая три полосы движения, критик замечает, какие шокирующие выражения используют другие водители. На основании этого опыта обучающий компонент способен сформулировать правило, которое гласит, что это — недопустимое действие, а производительный компонент модифицируется путем установки нового правила. Генератор проблем может определить некоторые области поведения, требующие усовершенствования, и предложить эксперименты, такие как проверка тормозов на разных дорожных покрытиях и при различных условиях.

Обучающий компонент может вносить изменения в любой из компонентов "знаний", показанных на схемах агентов (см. рис. 2.3–2.6). В простейших случаях обучение будет осуществляться непосредственно на основании последовательности актов восприятия. Наблюдение за парами последовательных состояний среды позволяет агенту освоить информацию о том, "как изменяется мир", а наблюдение за результатами своих действий может дать агенту возможность узнать, "какое влияние оказывают мои действия". Например, после того как водитель такси приложит определенное тормозное давление во время езды по мокрой дороге, он вскоре узнает, какое снижение скорости фактически было достигнуто. Очевидно, что эти две задачи обучения становятся более сложными, если среда наблюдаема лишь частично.

Те формы обучения, которые были описаны в предыдущем абзаце, не требуют доступа к внешнему стандарту производительности, вернее, в них применяется универсальный стандарт, согласно которому сделанные прогнозы должны быть согласованы с экспериментом. Ситуация становится немного сложнее, когда речь идет об агенте, основанном на полезности, который стремится освоить в процессе обучения информацию о полезности. Например, предположим, что агент, занимающийся вождением такси, перестает получать чаевые от пассажиров, которые в ходе утомительной поездки почувствовали себя полностью разбитыми. Внешний стандарт производительности должен информировать агента, что отсутствие чаевых — это отрицательный вклад в его общую производительность; в таком случае агент получает возможность освоить в результате обучения, что грубые маневры, утомляющие пассажиров, не позволяют повысить оценку его собственной функции полезности. В этом смысле стандарт производительности позволяет выделить определенную

часть входных результатов восприятия как **вознаграждение** (или **штраф**), непосредственно предоставлемое данными обратной связи, влияющими на качество поведения агента. Именно с этой точки зрения могут рассматриваться жестко закрепленные стандарты производительности, такие как боль или голод, которыми характеризуется жизнь животных. Эта тема рассматривается более подробно в главе 21.

Подводя итог, отметим, что агенты имеют самые различные компоненты, а сами эти компоненты могут быть представлены в программе агента многими способами, поэтому создается впечатление, что разнообразие методов обучения чрезвычайно велико. Тем не менее все эти методы имеют единый объединяющий их аспект. Процесс обучения, осуществляемый в интеллектуальных агентах, можно в целом охарактеризовать как процесс модификации каждого компонента агента для обеспечения более точного соответствия этих компонентов доступной информации обратной связи и тем самым улучшения общей производительности агента.

2.5. Резюме

Эту главу можно сравнить с головокружительным полетом над обширным ландшафтом искусственного интеллекта, который мы рассматриваем как науку проектирования агентов. Ниже кратко приведены основные идеи, которые рассматривались в данной главе.

- **Агентом** является нечто воспринимающее и действующее в определенной среде. **Функция агента** определяет действие, предпринимаемое агентом в ответ на любую последовательность актов восприятия.
- **Показатели производительности** оценивают поведение агента в среде. **Рациональный агент** действует так, чтобы максимизировать ожидаемые значения показателей производительности, с учетом последовательности актов восприятия, полученной агентом к данному моменту.
- Спецификация **проблемной среды** включает определения показателей производительности, внешней среды, исполнительных механизмов и датчиков. Первым этапом проектирования агента всегда должно быть определение проблемной среды с наибольшей возможной полнотой.
- Варианты проблемной среды классифицируются по нескольким важным размерностям. Они могут быть полностью или частично наблюдаемыми, детерминированными или стохастическими, эпизодическими или последовательными, статическими или динамическими, дискретными или непрерывными, а также одноагентными или мультиагентными.
- **Программа агента** реализует функцию агента. Существует целый ряд основных проектов программ агента, соответствующих характеру явно воспринимаемой информации, которая используется в процессе принятия решения. Разные проекты характеризуются различной эффективностью, компактностью и гибкостью. Выбор наиболее подходящего проекта программы агента зависит от характера среды.
- **Простые рефлексные агенты** отвечают непосредственно на акты восприятия, тогда как **рефлексные агенты, основанные на модели**, поддерживают внутреннее

состояние, прослеживая те аспекты среды, которые не наблюдаются в текущем акте восприятия. **Агенты, действующие на основе цели**, организуют свои действия так, чтобы достигнуть своих целей, а **агенты, действующие с учетом полезности**, пытаются максимизировать свою собственную ожидаемую "удовлетворенность".

- Все агенты способны улучшать свою работу благодаря **обучению**.

Библиографические и исторические заметки

Истоки представлений о центральной роли действий в интеллекте (сформулированных в виде понятия *практических рассуждений*) прослеживаются до *Никомаховой этики* Аристотеля. Практические рассуждения были также темой влиятельной статьи Маккарти *Programs with Common Sense* (Программы со здравым смыслом) [1009]. Такие области науки, как робототехника и теория управления, по самому своему характеру преимущественно касаются проблем конструирования физических агентов. Понятие **контроллера** в теории управления идентично понятию агента в искусственном интеллекте. И хотя на первый взгляд это может показаться удивительным, но на протяжении большей части истории развития искусственного интеллекта основные усилия в этой области были сосредоточены на исследовании отдельных компонентов агентов (в качестве примеров можно привести системы поиска ответов на вопросы, программы автоматического доказательства теорем, системы технического зрения и т.д.), а не самих агентов, рассматриваемых как единое целое. Важным исключением из этого правила, оказавшим значительное влияние на дальнейшее развитие данной области, стало обсуждение проблематики агентов в работе Генезерета и Нильссона [537]. В настоящее время в данной научной области широко применяется подход, основанный на изучении всего агента, и результаты, достигнутые в рамках этого подхода, стали центральной темой новейших работ [1146], [1227].

В главе 1 было показано, что корни понятия *рациональности* прослеживаются в философии и экономике. В искусственном интеллекте это понятие не привлекало значительного интереса до тех пор, пока в середине 1980-х не началось широкое обсуждение проблемы создания подходящих теоретических основ данной области. В статье Джона Дойла [410] было предсказано, что со временем проектирование рациональных агентов станет рассматриваться в качестве основного назначения искусственного интеллекта, притом что другие популярные ныне темы исследований послужат основой формирования новых дисциплин.

Стремление к тщательному изучению свойств среды и их влияния на выбор самого подходящего проекта рационального агента наиболее ярко проявляется в традиционных областях теории управления; например, в исследованиях по классическим системам управления [405] рассматриваются полностью наблюдаемые, детерминированные варианты среды; темой работ по стохастическому оптимальному управлению [865] являются частично наблюдаемые, стохастические варианты среды; а работы по гибридному управлению [649] касаются таких вариантов среды, которые содержат и дискретные, и непрерывные элементы. Описание различий между полностью и частично наблюдаемыми вариантами среды является также центральной темой работ по **динамическому программированию**, проводимых в области исследования операций [1244], которая будет обсуждаться в главе 17.

Рефлексные агенты были основной моделью для психологических бихевиористов, таких как Скиннер [1423], которые пытались свести все знания в области психологии организмов исключительно к отображениям "ввод–вывод" или "стимул–отклик". В результате прогресса в области психологии, связанного с переходом от бихевиоризма к функционализму, который был по крайней мере отчасти обусловлен распространением на агентов трактовки понятия компьютера как новой метафоры в мировоззрении человека [923], [1246], возникло понимание того, что нужно также учитывать внутреннее состояние агента. В большинстве работ по искусственному интеллекту идея чисто рефлексных агентов с внутренним состоянием рассматривалась как слишком простая для того, чтобы на ее основе можно было добиться значительных успехов, но исследования Розеншайна [1308] и Брукса [189] позволили поставить под сомнение это предположение (см. главу 25). В последние годы значительная часть работ была посвящена поиску эффективных алгоритмов слежения за сложными вариантами среды [610]. Наиболее впечатляющим примером достигнутых при этом результатов является программа Remote Agent, которая управляла космическим аппаратом Deep Space One (описанная на с. 69) [744], [1108].

Понимание необходимости создания агентов на основе цели просматривается во всем, что стало источником идей искусственного интеллекта, начиная с введенного Аристотелем определения практического рассуждения и заканчивая ранними статьями Маккарти по логическому искусственному интеллекту. Робот Shakey [466], [1143] был первым воплощением логического агента на основе цели в виде автоматизированного устройства. Полный логический анализ агентов на основе цели приведен в книге Генезерета и Нильссона [537], а методология программирования на основе цели, получившая название *агентно-ориентированного программирования*, была разработана Шохемом [1404].

Кроме того, начиная с книги *Human Problem Solving* [1130], оказавшей чрезвычайно большое влияние на ход дальнейших исследований, в той области когнитивной психологии, которая посвящена изучению процедур решения задач человеком, ведущее место занял подход, основанный на понятии цели; на этом подходе базируется также вся последняя работа Ньюэлла [1125]. Цели, дополнительно подразделяемые на желания (общие замыслы) и намерения (осуществляемые в настоящее время), являются основой теории агентов, разработанной Братманом [176]. Эта теория оказала влияние и на работы в области понимания естественного языка, и на исследования мультиагентных систем.

Горвиц, наряду с другими исследователями [686], особо подчеркивал важность использования в качестве основы для искусственного интеллекта понятия *рациональности*, рассматриваемой как средство максимизации ожидаемой полезности. Книга Перла [1191] была первой работой в области искусственного интеллекта, в которой дан глубокий анализ проблем применения теории вероятности и теории полезности; описанные им практические методы проведения рассуждений и принятия решений в условиях неопределенности, по-видимому, послужили наиболее важной причиной быстрой смены направления исследований в 1990-х годах и перехода к изучению агентов, действующих с учетом полезности (подробнее об этом рассказывается в части V).

Общий проект для обучающихся агентов, приведенный на рис. 2.7, является классическим образцом такого проекта в литературе по машинному обучению [203], [1064]. Одним из первых примеров воплощения этого проекта в программах являет-

17 Принятие сложных решений

В данной главе рассматриваются методы принятия решений о том, что следует делать в настоящее время, при условии, что в дальнейшем может быть принято другое решение.

В этой главе описано, какие расчеты связаны с принятием решений. В главе 16 речь шла о задачах принятия единоразовых или эпизодических решений, в которых полезность результата каждого действия была вполне известна, а в настоящей главе рассматриваются ✍ **задачи последовательного принятия решений**, в которых полезность действий агента зависит от последовательности решений. Задачи последовательного принятия решений, в которых рассматриваются полезности, степени неопределенности и результаты восприятия, являются обобщением задач поиска и планирования, описанных в частях II и IV. В разделе 17.1 описано, как должны быть определены задачи последовательного принятия решений, а в разделах 17.2 и 17.3 показано, как их следует решать, чтобы выработать оптимальные правила поведения, в которых уравновешиваются риски и вознаграждения, связанные с осуществлением действий в неопределенной среде. В разделе 17.4 эти идеи распространяются на случай частично наблюдаемых вариантов среды, а в разделе 17.5 разрабатывается полный проект для агентов, действующих на основе теории принятия решений в частично наблюдаемых вариантах среды; в этом проекте объединяются динамические байесовские сети, описанные в главе 15, и сети принятия решений, описанные в главе 16.

Во второй части данной главы рассматриваются варианты среды с многочисленными агентами. В таких вариантах среды понятие оптимального поведения становится гораздо более сложным из-за взаимодействия агентов. В разделе 17.6 представлены основные идеи **теории игр**, включая ту идею, что рациональным агентам может потребоваться вести себя случайным образом. В разделе 17.7 показано, как следует проектировать мультиагентные системы для того, чтобы несколько агентов могли достичь общей цели.

17.1. Задачи последовательного принятия решений

Пример

Предположим, что агент находится в среде с размерами 4×3, показанной на рис. 17.1, *а*. Начиная с начального состояния, он должен выбирать какое-то действие в каждом временном интервале. Взаимодействие со средой оканчивается после того, как агент достигает одного из целевых состояний, обозначенных +1 и -1. В каждом местонахождении в распоряжении агента имеются действия *Up* (Вверх), *Down* (Вниз), *Left* (Влево) и *Right* (Вправо). На данный момент предполагается, что эта среда является **полностью наблюдаемой**, поэтому агент всегда знает, где он находится.

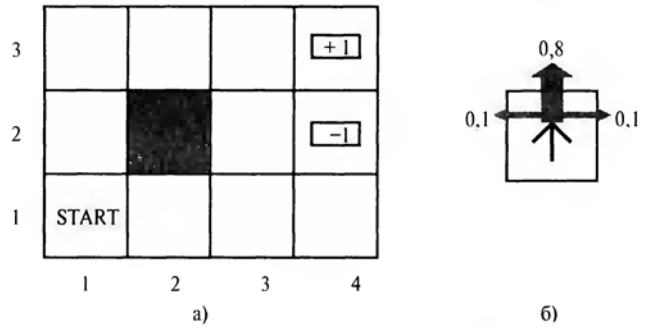

Рис. 17.1. Определение задачи: простая среда с размерами 4×3, в которой перед агентом поставлена задача последовательного принятия решений (а); модель перехода для этой среды: "намеченный" результат достигается с вероятностью 0.8, а с вероятностью 0.2 агент движется под прямыми углами влево или вправо от намеченного направления (б). Столкновение со стеной приводит к тому, что дальнейшее движение не происходит. С двумя конечными состояниями связаны вознаграждения +1 и -1 соответственно, а со всеми другими состояниями связано вознаграждение -0.04

Если бы эта среда была полностью детерминированной, то достижение требуемого решения было бы несложным: [*Up*, *Up*, *Right*, *Right*, *Right*]. К сожалению, среда не всегда реагирует правильно на осуществление этого решения, поскольку действия выполняются ненадежно. Конкретная принятая нами модель стохастического движения показана на рис. 17.1, *б*. Каждое действие достигает намеченной цели с вероятностью 0.8, но в течение всего остального времени в результате выполнения действия агент движется под прямыми углами к выбранному направлению. Более того, если агент ударяется в стену, то остается в том же квадрате. Например, выполняемое из начального квадрата (1,1) действие *Up* перемещает агента в квадрат (1,2) с вероятностью 0.8, но с вероятностью 0.1 агент движется вправо, в квадрат (2,1), а с вероятностью 0.1 он движется влево, ударяется в стену и остается в квадрате (1,1). В такой среде последовательность действий [*Up*, *Up*, *Right*, *Right*, *Right*] позволяет обойти барьер и достичь целевого состояния, квадрата (4,3), с вероятностью

$0.8^5=0.32768$. Существует также небольшой шанс случайно достичь цели, обойдя барьер с другой стороны с вероятностью $0.1^4\times0.8$, поэтому суммарная вероятность достижения цели равна 0.32776 (см. также упр. 17.1).

Спецификацию вероятностей результатов каждого действия в каждом возможном состоянии принято называть **моделью перехода** (или просто "моделью", если не может возникнуть путаница). Для обозначения вероятности достижения состояния s', если в состоянии s было выполнено действие a, будет применяться запись $T(s,a,s')$. Предполагается, что эти переходы являются **марковскими** в том смысле, какой указан в главе 15, т.е. что вероятность достижения состояния s' из s зависит только от s, а не от истории пребывания в предыдущих состояниях. На данный момент запись $T(s,a,s')$ может рассматриваться как большая трехмерная таблица, содержащая вероятности. В дальнейшем, в разделе 17.5, будет показано, что модель перехода может быть представлена как **динамическая байесовская сеть**, точно так же, как и в главе 15.

В завершение этого определения среды задачи необходимо сформулировать функцию полезности для агента. Поскольку эта задача принятия решений является последовательной, функция полезности должна зависеть от последовательности состояний (от **истории пребывания в среде**), а не от отдельного состояния. Ниже в этом разделе будет приведено описание того, как такие функции полезности могут быть определены в целом, а на данный момент просто примем предположение, что в каждом состоянии s агент получает **вознаграждение** $R(s)$, которое может быть положительным или отрицательным, но должно быть ограниченным. В данном конкретном примере вознаграждение равно -0.04 во всех состояниях, кроме конечных (с которыми связаны вознаграждения +1 и -1). Полезность, связанная с историей пребывания в среде (на данный момент), рассматривается как сумма полученных вознаграждений. Например, если агент достиг состояния +1 после 10 шагов, суммарная полезность его действий будет равна 0.6. Отрицательное вознаграждение -0.04 побуждает агента быстрее достичь квадрата (4,3), поэтому данная среда представляет собой стохастическое обобщение вариантов среды, которые рассматривались в задачах поиска в главе 3. Еще один способ описать эту игровую ситуацию состоит в том, что агенту "не нравится" находиться в этой среде, поэтому он стремится выйти из игры как можно быстрее.

Такая спецификация задачи последовательного принятия решений для полностью наблюдаемой среды с марковской моделью перехода и дополнительными вознаграждениями называется спецификацией **марковского процесса принятия решений**, или сокращенно **MDP** (Markov Decision Process). Любая задача MDP определяется тремя перечисленными ниже компонентами.

- Начальное состояние — S_0.
- Модель перехода — $T(s,a,s')$.
- Функция вознаграждения[1] — $R(s)$.

[1] В некоторых определениях задач MDP допускается, чтобы вознаграждение зависело также от действия и результата, поэтому функция вознаграждения принимает вид R(s,a,s'). Такой подход позволяет упростить описание некоторых вариантов среды, но не приводит к какому-либо фундаментальному изменению самой задачи.

Следующий вопрос состоит в том, как должно выглядеть решение этой задачи. Выше в данной главе было показано, что какая-либо фиксированная последовательность действий не может служить решением этой задачи, поскольку в конечном итоге после ее выполнения агент может оказаться в состоянии, отличном от целевого. Поэтому в решении должно быть указано, что следует делать агенту в любом состоянии, которого он может достичь. Решение такого рода — это так называемая **стратегия**. Для обозначения стратегии обычно принято использовать π; а $\pi(s)$ — это действие, рекомендованное в соответствии со стратегией π для состояния s. Если агент имеет полное описание стратегии, то всегда знает, что делать дальше, независимо от результата любого действия.

Каждый раз, когда осуществляется данная конкретная стратегия, начиная с начального состояния, стохастический характер среды приводит к формированию другой истории пребывания в среде. Поэтому качество определения стратегии измеряется по ожидаемой полезности возможных историй пребывания в среде, создаваемых с помощью этой стратегии. **Оптимальной стратегией** называется такая стратегия, которая позволяет достичь максимальной ожидаемой полезности. Для обозначения оптимальной стратегии принято использовать запись π^*. Если агенту указана стратегия π^*, он принимает решение, что делать, проверяя свои текущие результаты восприятия, которые сообщают ему, что он находится в текущем состоянии s, а затем выполняя действие $\pi^*(s)$. В любой стратегии функция агента представлена явно, поэтому стратегия является описанием простого рефлексного агента, сформированным с учетом информации, которая используется агентом, действующим на основе полезности.

Оптимальная стратегия для мира, приведенного на рис. 17.1, показана на рис. 17.2, *а*. Обратите внимание на то, что стоимость выполнения одного шага довольно мала по сравнению со штрафом, который связан со случайным попаданием в квадрат (4,2), поэтому оптимальная стратегия для состояния (3,1) является предельно осторожной. Этот стратегия рекомендует, что нужно совершить дальний обход препятствия, а не пытаться пройти по короткому пути и тем самым подвергнуться риску попасть в квадрат (4,2).

Равновесие между риском и вознаграждением изменяется в зависимости от значения функции $R(s)$ для нетерминальных состояний. На рис. 17.2, *б* показаны оптимальные стратегии для четырех различных диапазонов изменения значения $R(s)$. Если $R(s) \leq -1.6284$, жизнь настолько мучительна, что агент направляется прямо к ближайшему выходу, даже если стоимость этого выхода равна -1. Если $-0.4278 \leq R(s) \leq -0.0850$, жизнь довольно дискомфортна; агент выбирает кратчайший маршрут к состоянию +1 и стремится избежать риска случайного попадания в состояние -1. В частности, агент выбирает короткий путь из квадрата (3,1). А если жизнь не так уж неприятна ($-0.0221 < R(s) < 0$), оптимальная стратегия состоит в том, чтобы избегать вообще какого-либо риска. В квадратах (4,1) и (3,2) агент направляется буквально прочь от состояния -1, чтобы случайно не попасть туда ни при каких обстоятельствах, даже несмотря на то, что из-за этого ему приходится несколько раз удариться головой о стену. Наконец, если $R(s) > 0$, то жизнь агента становится весьма приятной и он избегает обоих выходов. При условии, что используются действия, показанные в квадратах (4,1), (3,2) и (3,3), любая стратегия является оптимальной и агент получает бесконечно боль-

шое суммарное вознаграждение, поскольку он никогда не попадает в терминальное состояние. Как это ни удивительно, но оказывается, что существуют шесть других оптимальных стратегий для различных диапазонов значений $R(s)$; в упр. 17.7 предлагается найти эти стратегии.

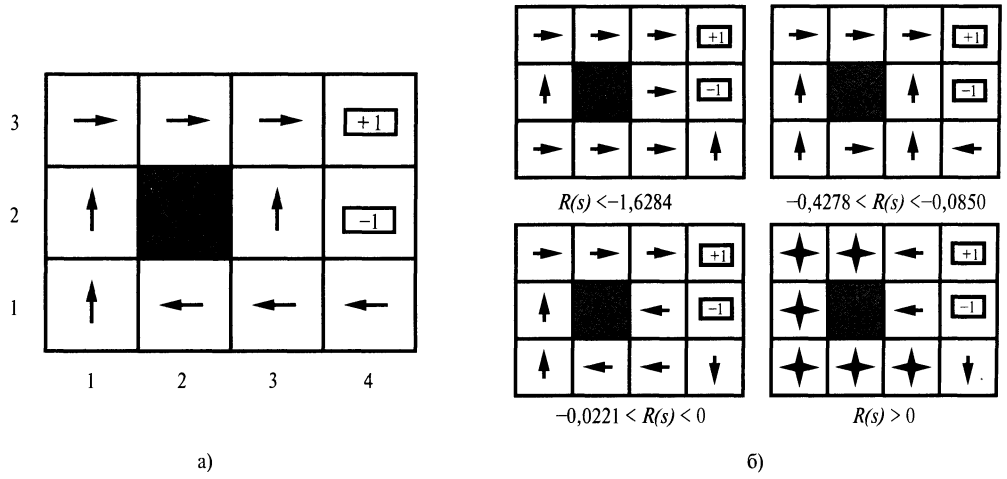

Рис. 17.2. Примеры оптимальных стратегий: оптимальная стратегия для стохастической среды со значениями `R(s) = -0.04` *в нетерминальных состояниях (а); оптимальные стратегии для четырех различных диапазонов значений* `R(s)` *(б)*

Тщательное уравновешивание риска и вознаграждения является характерной особенностью задач MDP, которая не возникает в детерминированных задачах поиска; более того, такое уравновешивание характерно для многих реальных задач принятия решений. По этой причине задачи MDP изучаются в нескольких научных областях, включая искусственный интеллект, исследование операций, экономику и теорию управления. Для вычисления оптимальных стратегий были предложены десятки алгоритмов. В разделах 17.2 и 17.3 описываются два наиболее важных семейства алгоритмов. Но вначале мы должны завершить начатое исследование полезностей и стратегий для задач последовательного принятия решений.

Оптимальность в задачах последовательного принятия решений

В примере задачи MDP (см. рис. 17.1) производительность агента определялась по сумме вознаграждений, связанных с посещенными состояниями. Такой выбор показателя производительности нельзя назвать произвольным, но он не является также единственно допустимым. В данном разделе рассматриваются возможные варианты показателей производительности, т.е. варианты способов определения функции полезности по историям пребывания в среде, которые могут записываться как $U_h([s_0, s_1, \ldots, s_n])$. Этот раздел основан на идеях, изложенных в главе 16, и является довольно формальным; основные его пункты подытожены в конце.

Первый вопрос, на который нужно найти ответ, состоит в том, существует ли **конечный горизонт** или **бесконечный горизонт** для принятия решений. Наличие конечного горизонта означает, что есть такое фиксированное время N, после кото-

рого все теряет смысл, — так сказать, игра все равно окончена. Таким образом, $U_h([s_0, s_1, \ldots, s_{N+k}]) = U_h([s_0, s_1, \ldots, s_N])$ для всех $k>0$. Например, предположим, что агент начинает свое движение с квадрата (3,1) в мире с размерами 4×3, показанном на рис. 17.1, а также допустим, что $N=3$. В таком случае, чтобы получить хоть малейший шанс достичь состояния +1, агент должен направиться непосредственно к нему, и оптимальное действие состоит в том, чтобы двигаться в направлении *Up*. С другой стороны, если $N=100$, то запас времени настолько велик, что можно выбрать безопасный маршрут в направлении *Left*. ☞ *Поэтому при наличии конечного горизонта оптимальное действие в каждом конкретном состоянии со временем может измениться.* Принято считать, что оптимальная стратегия при наличии конечного горизонта является **нестационарной**. С другой стороны, если нет заданного предела времени, то нет смысла вести себя поразному в одном и том же состоянии в разное время. Поэтому оптимальное действие зависит только от текущего состояния и оптимальная стратегия является **стационарной**. Таким образом, стратегии для случая с бесконечным горизонтом проще по сравнению с теми, которые применяются в случае с конечным горизонтом, и в данной главе будет в основном рассматриваться случай с бесконечным горизонтом[2]. Обратите внимание на то, что понятие "бесконечного горизонта" не обязательно означает, что все последовательности состояний являются бесконечными; оно просто говорит о том, что для их выполнения не устанавливаются фиксированные сроки. В частности, в любой задаче MDP с бесконечным горизонтом могут существовать конечные последовательности состояний, содержащие терминальное состояние.

Следующий вопрос, на который необходимо найти ответ, состоит в том, как рассчитать полезность последовательностей состояний. Мы будем рассматривать задачу поиска ответа на этот вопрос как задачу **многоатрибутной теории полезности** (см. раздел 16.4), где каждое состояние s_i рассматривается как атрибут последовательности состояний $[s_0, s_1, s_2, \ldots]$. Чтобы получить простое выражение в терминах атрибутов, необходимо принять своего рода предположение о независимости предпочтений. Наиболее естественное предположение состоит в том, что отношение предпочтения агента между последовательностями состояний является **стационарным**. Стационарность предпочтений означает следующее: если две последовательности состояний, $[s_0, s_1, s_2, \ldots]$ и $[s_0', s_1', s_2', \ldots]$, начинаются с одного и того же состояния (т.е. $s_0=s_0'$), то эти две последовательности должны быть упорядочены по предпочтениям таким же образом, как и последовательности $[s_1, s_2, \ldots]$ и $[s_1', s_2', \ldots]$. На естественном языке эту мысль можно выразить так, что если вы предпочитаете одно будущее развитие событий, начинающееся завтра, другому развитию событий, то вы должны также предпочесть это будущее развитие событий, если оно начнется сегодня. На первый взгляд, предположение о стационарности выглядит довольно безобидно, но влечет за собой весьма важные последствия: как оказалось, в условиях стационарности существуют только два способа присваивания значений полезности последовательностям, которые описаны ниже.

1. **Аддитивные вознаграждения**. Полезность последовательности состояний определяется следующим образом:

$$U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \ldots$$

[2] Это утверждение касается полностью наблюдаемых вариантов среды. Как будет показано ниже, для частично наблюдаемых вариантов среды случай с бесконечным горизонтом не так уж прост.

В мире 4×3, показанном на рис. 17.1, используются аддитивные вознаграждения. Обратите внимание на то, что свойство аддитивности уже было определено неявно в используемых нами функциях стоимости пути для алгоритмов эвристического поиска (см. главу 4).

2. **Обесцениваемые вознаграждения**, Полезность последовательности состояний определяется с помощью следующего соотношения:

   $U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$

   где γ — это **коэффициент обесценивания**, который представляет собой число от 0 до 1. Коэффициент обесценивания описывает предпочтение агентом текущих вознаграждений перед будущими вознаграждениями. Если коэффициент γ близок к 0, вознаграждения, которые должны быть получены в отдаленном будущем, рассматриваются как малозначащие, а если коэффициент γ равен 1, то обесцениваемые вознаграждения полностью эквивалентны аддитивным вознаграждениям, поэтому аддитивные вознаграждения представляют собой частный случай обесцениваемых вознаграждений. По-видимому, обесценивание представляет собой хорошую модель изменения во времени предпочтений и животных, и человека. Коэффициент обесценивания γ эквивалентен процентной ставке $(1/\gamma) - 1$.

По причинам, которые вскоре станут очевидными, до конца этой главы предполагается, что используются обесцениваемые вознаграждения, хотя иногда допускается применение значения $\gamma=1$.

Сделанный нами выбор бесконечных горизонтов становится причиной возникновения определенной проблемы: если среда не содержит терминальное состояние или если агент никогда его не достигает, то все истории пребывания в среде будут иметь бесконечную длину, а полезности, связанные с аддитивными вознаграждениями, в общем случае будут бесконечными. Дело в том, что, безусловно, $+\infty$ лучше, чем $-\infty$, но гораздо сложнее сравнивать две последовательности состояний, притом что обе имеют полезность $+\infty$. Для решения этой проблемы можно применить три описанных ниже подхода, два из которых уже упоминались в этой главе.

1. При наличии обесцениваемых вознаграждений полезность любой бесконечной последовательности является конечной. В действительности, если вознаграждения ограничены значением R_{max} и $\gamma<1$, то можно получить следующее соотношение с использованием стандартной формулы суммы бесконечных геометрических рядов:

$$U_h([s_0, s_1, s_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = R_{max}/(1-\gamma) \qquad (17.1)$$

2. Если среда содержит терминальные состояния и гарантируется достижение агентом в конечном итоге одного из этих состояний, то нам никогда не придется сравнивать бесконечные последовательности действий. Стратегия, который гарантирует достижение терминального состояния, называется **правильной стратегией**. При наличии правильных стратегий можно использовать $\gamma=1$ (т.е. аддитивные вознаграждения). Первые три стратегии, пока-

занные на рис. 17.2, *б*, являются правильными, а четвертая — неправильной. В ней достигается бесконечное суммарное вознаграждение за счет предотвращения попадания в терминальные состояния, притом что вознаграждение за пребывание в нетерминальных состояниях является положительным. Само существование неправильных стратегий в сочетании с использованием аддитивных вознаграждений может стать причиной неудачного завершения стандартных алгоритмов решения задач MDP, поэтому является весомым доводом в пользу применения обесцениваемых вознаграждений.

3. Еще один подход состоит в том, чтобы сравнивать бесконечные последовательности по **среднему вознаграждению**, получаемому в расчете на каждый временной интервал. Предположим, что с квадратом `(1,1)` в мире 4×3 связано вознаграждение `0.1`, тогда как для других нетерминальных состояний предусмотрено вознаграждение `0.01`. В таком случае стратегия, в которой агент предпочтет оставаться в квадрате `(1,1)`, позволит получать более высокое среднее вознаграждение по сравнению с той стратегией, в которой агент находится в каком-то другом квадрате. Среднее вознаграждение представляет собой полезный критерий для некоторых задач, но анализ алгоритмов со средним вознаграждением выходит за рамки данной книги.

Подводя итог, можно сказать, что использование обесцениваемых вознаграждений связано с наименьшими трудностями при оценке последовательностей состояний. Заключительный этап состоит в том, чтобы показать, как осуществляется выбор между стратегиями с учетом того, что каждая конкретная стратегия π вырабатывает не только одну последовательность состояний, но целый ряд возможных последовательностей состояний, притом что каждая из этих последовательностей имеет конкретную вероятность, определяемую моделью перехода для данной среды. Таким образом, стоимость любой стратегии представляет собой ожидаемую сумму полученных обесцениваемых вознаграждений, где это ожидаемое значение вычисляется по всем возможным последовательностям состояний, которые могут возникнуть при осуществлении данной стратегии. Любая оптимальная стратегия π^* удовлетворяет следующему соотношению:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right] \qquad (17.2)$$

В следующих двух разделах описаны алгоритмы поиска оптимальных стратегий.

17.2. Итерация по значениям

В этом разделе представлен алгоритм вычисления оптимальной стратегии, называемый **итерацией по значениям**. Основная его идея состоит в том, что нужно рассчитать полезность каждого состояния, а затем использовать полезности состояний для выбора оптимального действия в каждом состоянии.

Полезности состояний

Полезность состояний определяется в терминах полезности последовательностей состояний. Грубо говоря, полезность любого состояния представляет собой ожидаемую полезность последовательностей состояний, которые могут привести к этому состоянию. Очевидно, что перечень таких последовательностей состояний зависит от осуществляемой стратегии, поэтому начнем с определения полезности $U^{p}(s)$ по отношению к конкретной стратегии π. Если мы предположим, что s_t — это состояние, в котором находится агент после осуществления стратегии π в течение t шагов (обратите внимание на то, что s_t — случайная переменная), то получим следующее:

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^{t} R(s_t) \mid \pi, s_0 = s\right] \qquad (17.3)$$

На основании этого определения можно утверждать, что истинная полезность любого состояния, которую обозначим как $U(s)$, представляет собой $U^{\pi^*}(s)$, т.е. ожидаемую сумму обесцениваемых вознаграждений, при условии, что агент осуществляет оптимальную стратегию. Обратите внимание на то, что $U(s)$ и $R(s)$ — совершенно разные величины; $R(s)$ — это "кратковременное" вознаграждение за пребывание в состоянии s; $U(s)$ — "долговременное" суммарное вознаграждение, которое начинается с состояния s и продолжается дальше. На рис. 17.3 показаны рассматриваемые значения полезности для мира 4×3. Заслуживает внимание то, что значения полезности по мере приближения состояний к выходу +1 становятся выше, поскольку уменьшается количество шагов, требуемых для достижения этого выхода.

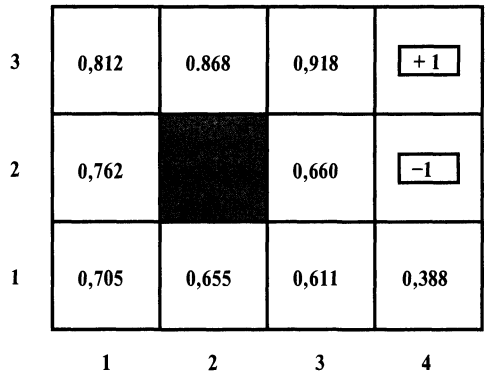

Рис. 17.3. Полезности состояний в мире 4×3, рассчитанные при $\gamma=1$ и $R(s)=-0.04$ для нетерминальных состояний

Эта функция полезности $U(s)$ позволяет агенту выбирать действия с использованием принципа максимальной ожидаемой полезности, приведенного в главе 16, т.е. выбирать действие, которое максимизирует ожидаемую полезность в следующем состоянии:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s,a,s')\ U(s') \qquad (17.4)$$

Итак, если полезность некоторого состояния представляет собой ожидаемую сумму обесцениваемых вознаграждений, начиная с данного момента и дальше, то существует прямая связь между полезностью состояния и полезностью его соседних состояний: ☞ *полезность некоторого состояния равна сумме непосредственного вознаграждения за пребывание в этом состоянии и ожидаемой обесцениваемой полезности следующего состояния, при условии, что агент выбирает оптимальное действие.* Это означает, что полезность любого состояния можно определить с помощью следующего соотношения:

$$U(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s,a,s')\ U(s') \qquad (17.5)$$

Уравнение 17.5 называется **уравнением Беллмана** в честь Ричарда Беллмана [97]. Полезности состояний (определяемые с помощью уравнения 17.3 как ожидаемые полезности дальнейших последовательностей состояний) являются решениями множества уравнений Беллмана. В действительности, как будет показано в следующих двух разделах, они являются уникальными решениями.

Рассмотрим одно из уравнений Беллмана для мира 4×3. Уравнение для состояния (1,1) приведено ниже.

```
U(1,1) = -0.04 + γ max {0.8 U(1,2) + 0.1 U(2,1) + 0.1 U(1,1),   (Up)
                        0.9 U(1,1) + 0.1 U(1,2),                (Left)
                        0.9 U(1,1) + 0.1 U(2,1),                (Down)
                        0.8 U(2,1) + 0.1 U(1,2) + 0.1 U(1,1)}   (Right)
```

После подстановки в это уравнение чисел, приведенных на рис. 17.3, можно обнаружить, что наилучшим действием является *Up*.

Алгоритм итерации по значениям

Уравнение Беллмана является основой алгоритма итерации по значениям, применяемого для решения задач MDP. Если существует n возможных состояний, то количество уравнений Беллмана также равно n, по одному для каждого состояния. Эти n уравнений содержат n неизвестных — полезностей состояний. Поэтому можно было бы заняться поиском решений системы этих уравнений, чтобы определить полезности. Тем не менее возникает одна проблема, связанная с тем, что эти уравнения являются нелинейными, поскольку оператор "max" — это нелинейный оператор. Системы линейных уравнений могут быть решены очень быстро с использованием методов линейной алгебры, а для решения систем нелинейных уравнений необходимо преодолеть некоторые проблемы. Один из возможных подходов состоит в использовании итерационных методов. Для этого нужно начать с произвольных исходных значений полезностей, вычислить правую часть уравнения и подставить ее в левую, тем самым обновляя значение полезности каждого состояния с учетом полезностей его соседних состояний. Такая операция повторяется до тех пор, пока не достигается равновесие. Допустим, что $U_i(s)$ — это значение полезности для со-

стояния s в i-й итерации. Шаг итерации, называемый **обновлением Беллмана**, выглядит следующим образом:

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s,a,s')\, U_i(s') \qquad (17.6)$$

Если обновление Беллмана используется неопределенно большое количество раз, то гарантируется достижение равновесия (см. следующий подраздел), и в этом случае конечные значения полезности должны представлять собой решения уравнений Беллмана. В действительности они также представляют собой уникальные решения, и соответствующая стратегия (полученная с помощью уравнения 17.4) является оптимальной. Применяемый при этом алгоритм, называемый `Value-Iteration`, показан в листинге 17.1.

Листинг 17.1. Алгоритм итерации по значениям для вычисления полезностей состояний. Условие завершения работы взято из уравнения 17.8

```
function Value-Iteration(mdp, ε) returns функция полезности
    inputs: mdp, задача MDP с состояниями S, моделью перехода T,
                 функцией вознаграждения R, коэффициентом
                 обесценивания γ
            ε, максимально допустимая ошибка определения полезности
               любого состояния
    local variables: U, U', векторы полезностей для состояний из S,
                           первоначально равные нулю
                     δ, максимальное изменение полезности любого
                        состояния во время итерации

    repeat
          U ← U'; δ ← 0
          for each состояние s in S do
                U'[s] ← R[s] + γ max_a Σ_s' T(s,a,s') U[s']
                if |U'[s]-U[s]|>δ then δ ← |U'[s]-U[s]|
    until δ < ε(1-γ)/γ
    return U
```

Мы можем применить алгоритм итерации по значениям к миру 4×3 (см. рис. 17.1, *а*). Начиная с исходных значений, равных нулю, полезности изменяются, как показано на рис. 17.4, *а*. Обратите внимание на то, как состояния, находящиеся на различных расстояниях от квадрата `(4,3)`, накапливают отрицательное вознаграждение до тех пор, пока в какой-то момент не обнаруживается путь к состоянию `(4,3)`, после чего значения полезности начинают возрастать. Алгоритм итерации по значениям может рассматриваться как способ распространения информации через пространство состояний с помощью локальных обновлений.

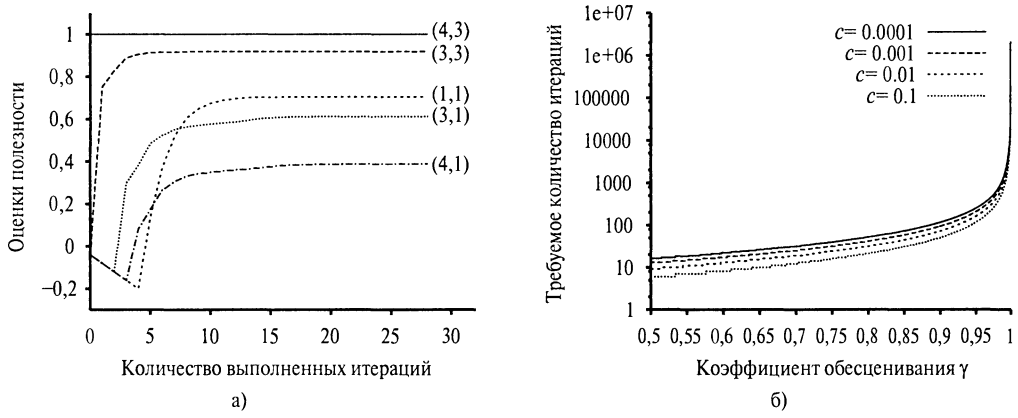

Рис. 17.4. Пример применения алгоритма итерации по значениям: график, показывающий изменение полезностей выбранных состояний в процессе итерации по значениям (а); количество итераций по значениям k, *необходимое для того, чтобы можно было гарантировать, что ошибка не превышает* $\varepsilon = c \cdot R_{max}$ *для различных значений* c, *как функция от коэффициента обесценивания* γ *(б)*

Сходимость итерации по значениям

Выше было указано, что процедура итерации по значениям в конечном итоге сходится к уникальному множеству решений уравнений Беллмана. В этом разделе показано, почему это происходит. В ходе этого будут представлены некоторые полезные математические идеи и получены определенные методы оценки ошибки в значении функции полезности, возвращаемом при преждевременном завершении работы алгоритма; это важно, поскольку означает, что количество применяемых итераций алгоритма не обязательно должно стремиться к бесконечности. Изложение в этом разделе является весьма формальным.

Основным понятием, используемым при доказательстве того, что процедура итерации по значениям сходится, является **сжатие**. Грубо говоря, функция сжатия — это функция от одного параметра, которая после ее последовательного применения к двум различным входным значениям вырабатывает два выходных значения, которые "ближе друг к другу" по меньшей мере на некоторую постоянную величину, чем первоначальные фактические параметры. Например, функция "деления на два" представляет собой функцию сжатия, поскольку после деления двух чисел на два разница между ними уменьшается наполовину. Обратите внимание на то, что функция "деления на два" имеет фиксированную точку, а именно нуль, которая остается неизменной в результате применения этой функции. На основании данного примера можно установить два важных свойства функций сжатия, описанных ниже.

- Функция сжатия имеет только одну фиксированную точку; если бы были две фиксированные точки, они бы не приближались друг к другу после применения функции, поэтому такая функция не соответствовала бы определению функции сжатия.
- После применения функции к любому параметру полученное значение должно стать ближе к фиксированной точке (поскольку фиксированная точка не

движется), поэтому в пределе повторное применение функции сжатия всегда приводит к достижению фиксированной точки.

Теперь предположим, что обновление Беллмана (уравнение 17.6) рассматривается как оператор B, который используется для одновременного обновления значений полезности каждого состояния. Обозначим через U_i вектор полезностей для всех состояний в i-й итерации. В таком случае уравнение обновления Беллмана может быть записано следующим образом:

$$U_{i+1} \leftarrow BU_i$$

Затем нужно найти способ измерения расстояний между векторами полезностей. Мы будем использовать **нормализованный максимум**, который измеряет длину вектора по длине его максимального компонента, следующим образом:

$$||U|| = \max_s |U(s)|$$

При использовании этого определения "расстояние" между двумя векторами, $||U-U'||$, представляет собой максимальную разность между двумя соответствующими элементами. Основным математическим результатом данного раздела является такое утверждение: *допустим, что U_i и U_i' — два вектора полезностей. В таком случае получим следующее:*

$$||BU_i - BU_i'|| \leq \gamma\, ||U_i - U_i'|| \qquad (17.7)$$

Это означает, что обновление Беллмана представляет собой функцию сжатия на коэффициент γ, применяемую к пространству векторов полезностей. Таким образом, процедура итерации по значениям всегда сходится к уникальному решению уравнений Беллмана.

В частности, можно заменить значение U_i' в уравнении 17.7 истинными полезностями U, для которых $BU=U$. В таком случае будет получено следующее неравенство:

$$||BU_i - U|| \leq \gamma\, ||U_i - U||$$

Поэтому, если $||U_i - U||$ рассматривается как ошибка в оценке U_i, то можно видеть, что эта ошибка уменьшается при каждой итерации на коэффициент, по меньшей мере равный γ. Это означает, что процедура итерации по значениям сходится экспоненциально быстро. Можно легко рассчитать количество итераций, требуемых для достижения заданной предельной ошибки ε, как описано ниже. Вначале напомним, что, как показывает уравнение 17.1, полезности всех состояний ограничены значением $\pm R_{max}/(1-\gamma)$. Из этого следует, что максимальная начальная ошибка определяется соотношением $||U_0 - U|| \leq 2R_{max}/(1-\gamma)$. Предположим, что для достижения ошибки, не превышающей ε, выполнено N итераций. В таком случае потребуется $\gamma^N \cdot 2R_{max}/(1-\gamma) \leq \varepsilon$ итераций, поскольку ошибка уменьшается каждый раз по меньшей мере на величину γ. Взяв логарифмы от этого выражения, можно определить, что достаточно применить следующее количество итераций:

$$N = \lceil \log(2R_{max}/\varepsilon(1-\gamma))/\log(1/\gamma) \rceil$$

На рис. 17.4, *б* показано, как количество итераций N изменяется в зависимости от γ при различных значениях отношения ε/R_{max}. Положительной особенностью этого соотношения является то, что из-за экспоненциально быстрой сходимости значение

N не очень зависит от отношения ε / R_{max}, а отрицательной особенностью — то, что N быстро возрастает по мере приближения значения γ к 1. Уменьшение значения γ позволяет добиться ускорения сходимости, но это фактически приводит к сужению горизонта агента и может не позволить агенту обнаруживать долговременные последствия своих действий.

Анализ предельной ошибки, приведенный выше, позволяет получить определенное представление о том, какие факторы влияют на продолжительность прогона данного алгоритма, но сам подход, основанный на определении предельной ошибки, иногда становится слишком консервативным способом принятия решения о прекращении итераций. Для последней цели можно использовать предел, связывающий ошибку с размерами обновления Беллмана в каждой конкретной итерации. На основании свойства сжатия (уравнение 17.7) можно показать, что если обновление невелико (т.е. не происходит значительного изменения полезности ни одного состояния), то ошибка также является небольшой по сравнению с истинным значением функции полезности. Точнее, выполняется следующее условие:

$$\textbf{if } ||U_{i+1}-U_i|| < \varepsilon(1-\gamma)/\gamma \textbf{ then } ||U_{i+1}-U|| < \varepsilon \qquad (17.8)$$

В этом и состоит условие завершения, используемое в алгоритме `Value-Iteration`, который приведен в листинге 17.1.

До сих пор мы анализировали ошибку в значении функции полезности, возвращаемом алгоритмом итерации по значениям. ☞ *Но для агента фактически гораздо важнее то, насколько успешно он будет действовать, принимая свои решения на основе данной функции полезности.* Предположим, что после i итераций в процедуре итерации по значениям агент получает оценку U_i истинной полезности U и определяет максимальную ожидаемую полезность стратегии π_i на основе прогнозирования на один шаг вперед с использованием значения U_i (как в уравнении 17.4). Будет ли выбранное в итоге поведение почти столь же хорошим, как и оптимальное поведение? Это — крайне важный вопрос для любого реального агента, и было показано, что ответ на него является положительным. Значение $U^{\pi_i}(s)$ — это полезность, достигаемая, если, начиная с состояния s, осуществляется стратегия π_i, а **убыточность стратегии** $||U^{\pi_i}-U||$ — это самая бо́льшая часть полезности, которую агент может потерять, осуществляя стратегию π_i вместо оптимальной стратегии π^*. Убыточность стратегии π_i связана с ошибкой в значении полезности U_i следующим неравенством:

$$\textbf{if } ||U_i-U|| < \varepsilon \textbf{ then } ||U^{\pi_i}-U|| < 2\varepsilon\gamma/(1-\gamma) \qquad (17.9)$$

На практике часто происходит так, что стратегия π_i становится оптимальной задолго до того, как сходится значение U_i. На рис. 17.5 показано, как максимальная ошибка в значении U_i и убыточность стратегии приближаются к нулю по мере осуществления процедуры итерации по значениям для среды 4×3 со значением $\gamma = 0.9$. Стратегия π_i становится оптимальной при $i=4$, даже несмотря на то, что максимальная ошибка в значении U_i все еще остается равной 0.46.

Теперь подготовлено все необходимое для использования процедуры итерации по значениям на практике. Известно, что процедура итерации по значениям в пределе сходится к правильным значениям полезности; ошибка в оценках полезностей может быть ограничена, даже если процедура итерации по значениям останавливается после конечного количества итераций; кроме того, может быть ограничена убы-

точность стратегии, которая связана с осуществлением соответствующей стратегии с максимальной ожидаемой полезностью. В качестве заключительного замечания отметим, что все результаты, приведенные в данном разделе, соответствуют такому случаю, когда применяется обесценивание полезностей, а $\gamma<1$. Если $\gamma=1$ и среда содержит терминальные состояния, то можно вывести аналогичное множество результатов оценки сходимости и определения предельных значений ошибок, если выполняются некоторые формальные условия.

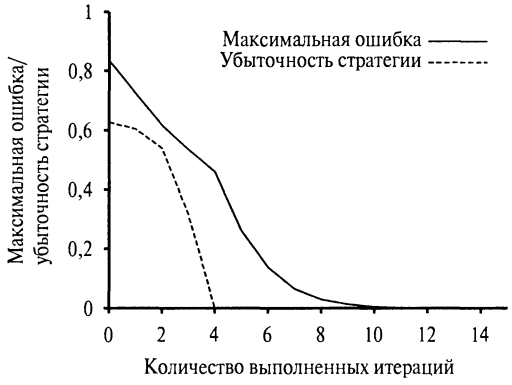

Рис. 17.5. Зависимости максимальной ошибки $||U_i-U||$ в оценках полезности и убыточности стратегии $||U^{\pi_i}-U||$ по сравнению с оптимальной стратегией от количества итераций в процедуре итерации по значениям

17.3. Итерация по стратегиям

В предыдущем разделе было показано, что возможно выработать оптимальную стратегию, даже если оценка функции полезности является неточной. Если очевидно, что одно действие лучше по сравнению со всеми остальными, то нет необходимости точно определять истинные значения величины полезности всех рассматриваемых состояний. Эта идея подсказывает альтернативный метод поиска оптимальных стратегий. В алгоритме **итерации по стратегиям** чередуются описанные ниже два этапа, начиная с некоторой исходной стратегии π_0.

- **Оценка стратегии**. На основе стратегии π_i вычислить $U_i=U^{\pi_i}$, полезность каждого состояния, если будет осуществлена стратегия π_i.
- **Усовершенствование стратегии**. Вычислить новую стратегию π_{i+1} с максимальной ожидаемой полезностью, используя прогноз на один шаг и исходя из значения U_i (как в уравнении 17.4).

Алгоритм завершает свою работу после того, как этап усовершенствования стратегии не приводит к изменению значений полезности. Как известно, в этот момент функция полезности U_i представляет собой фиксированную точку обновления Беллмана, поэтому является решением уравнений Беллмана, а π_i должна быть оп-

тимальной стратегией. Поскольку для каждого конечного пространства состояний количество возможных стратегий является конечным и можно показать, что каждая итерация приводит к определению лучшей стратегии, то алгоритм итерации по стратегиям должен всегда завершать свою работу. Этот алгоритм показан в листинге 17.2.

Листинг 17.2. Алгоритм итерации по стратегиям для вычисления оптимальной стратегии

```
function Policy-Iteration(mdp) returns стратегия
    inputs: mdp, задача MDP с состояниями S, моделью перехода T
    local variables: U, U', векторы полезностей для состояний из S,
                             первоначально равные нулю
                     π, вектор стратегий, индексированный по состояниям,
                        первоначально сформированный случайным образом

    repeat
          U ← Policy-Evaluation(π, U, mdp)
          unchanged? ← истинное значение
          for each состояние s in S do
                if max_a Σ_s' T(s,a,s') U[s'] > Σ_s' T(s,π[s],s') U[s'] then
                      π[s] ← argmax_a Σ_s' T(s,a,s') U[s']
                      unchanged? ← ложное значение
    until unchanged?
    return π
```

Очевидно, что этап усовершенствования стратегии является несложным, а как реализовать процедуру Policy-Evaluation? Оказалось, что осуществление такого подхода намного проще по сравнению с решением стандартных уравнений Беллмана (а именно это происходит в алгоритме итерации по значениям), поскольку действие, применяемое в каждом состоянии, зафиксировано в соответствии с выбранной стратегией. Стратегия π_i определяет действие $\pi_i(s)$, выполняемое в состоянии s на i-й итерации. Это означает, что можно воспользоваться упрощенной версией уравнения Беллмана (17.5), которая связывает полезность состояния s (соответствующую стратегии π_i) с полезностями его соседних состояний, следующим образом:

$$U_i(s) = R(s) + \gamma \sum_{s'} T(s,\pi_i(s),s')\ U_i(s') \quad (17.10)$$

Например, предположим, что π_i — это стратегия, показанная на рис. 17.2, *а*. В таком случае имеет место $\pi_i(1,1) = Up$, $\pi_i(1,2) = Up$ и т.д., а упрощенные уравнения Беллмана принимают следующий вид:

$$U_i(1,1) = -0.04 + 0.8\ U_i(1,2) + 0.1\ U_i(1,1) + 0.1\ U_i(2,1)$$
$$U_i(1,2) = -0.04 + 0.8\ U_i(1,3) + 0.2\ U_i(1,2)$$
...

Важно отметить, что эти уравнения — линейные, поскольку оператор "max" был удален. Для n состояний имеется n линейных уравнений с n неизвестными, которые

могут быть решены точно за время $O(n^3)$ с помощью стандартных методов линейной алгебры.

Для небольших пространств состояний оценка стратегии с использованием точных методов решения часто является наиболее эффективным подходом, а для больших пространств состояний затраты времени $O(n^3)$ могут оказаться чрезмерно большими. К счастью, точная оценка стратегии не требуется. Вместо этого можно выполнить некоторое количество упрощенных этапов итерации по значениям (они являются упрощенными, поскольку стратегия зафиксирована) для получения достаточно хорошей аппроксимации полезности. Упрощенное обновление Беллмана для этого процесса определяется таким соотношением:

$$U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} T(s,\pi_i(s),s')\ U_i(s')$$

и определяемая в нем операция подстановки повторяется k раз для получения следующей оценки полезности. Результирующий алгоритм называется **модифицированной итерацией по стратегиям**. Он часто оказывается намного более эффективным, чем стандартная итерация по стратегиям или итерация по значениям.

Алгоритмы, описанные до сих пор в данной главе, требуют одновременного обновления полезности или стратегии для всех состояний. Как оказалось, применение такой организации работы не является строго необходимым. В действительности в каждой итерации можно выбирать любое подмножество состояний и применять к этому подмножеству либо тот, либо другой вид обновления (усовершенствование стратегии или упрощенную итерацию по значениям). Такой наиболее общий алгоритм называется **асинхронной итерацией по стратегиям**. При соблюдении определенных условий выбора исходной стратегии и функции полезности гарантируется сходимость асинхронной итерации по стратегиям к определенной оптимальной стратегии. А то, что мы вправе выбирать для работы с ними любые состояния, означает, что могут быть разработаны гораздо более эффективные эвристические алгоритмы, например, алгоритмы, которые сосредоточиваются на обновлении значений состояний, которые с наибольшей вероятностью будут достигнуты при осуществлении качественной стратегии. Такой подход имеет гораздо больше смысла в реальной жизни — если человек не намеревается попасть на прибрежную полосу, спрыгнув с высокой скалы, то для него нет смысла заниматься точной оценкой стоимости связанных с этим результирующих состояний.

17.4. Марковские процессы принятия решений в частично наблюдаемых вариантах среды

В описании марковских процессов принятия решений, приведенном в разделе 17.1, предполагалось, что среда является **полностью наблюдаемой**. При использовании этого предположения агент всегда знает, в каком состоянии он находится. Это предположение, в сочетании с предположением о марковости модели перехода, означает, что оптимальная стратегия зависит только от текущего состояния. А если среда является только **частично наблюдаемой**, то вполне очевидно, что ситуация становится гораздо менее ясной. Агент не всегда точно знает, в каком состоянии находится, поэтому не

Для более сложных задач POMDP с непустыми результатами наблюдений приближенный поиск оптимальных стратегий является очень сложным (фактически такие задачи являются PSPACE-трудными, т.е. действительно чрезвычайно трудными). Задачи с несколькими десятками состояний часто оказываются неразрешимыми. В следующем разделе описан другой, приближенный метод решения задач POMDP, основанный на опережающем поиске.

17.5. Агенты, действующие на основе теории решений

В этом разделе будет описан исчерпывающий подход к проектированию агентов для частично наблюдаемых, стохастических вариантов среды. Как показано ниже, основные элементы этого проекта должны быть уже знакомы читателю.

- Модели перехода и наблюдения представлены в виде **динамических байесовских сетей** (см. главу 15).
- Динамическая байесовская сеть дополняется узлами принятия решений и узлами полезности, по аналогии с теми, которые использовались в **сетях принятия решений** в главе 16. Результирующая модель называется **динамической сетью принятия решений** (Dynamic Decision Network — DDN).
- Для учета данных о каждом новом восприятии и действии и для обновления представления доверительного состояния используется алгоритм фильтрации.
- Решения принимаются путем проектирования в прямом направлении возможных последовательностей действий и выбора наилучших из этих последовательностей.

Основное преимущество использования динамической байесовской сети для представления модели перехода и модели восприятия состоит в том, что такая сеть позволяет применять декомпозицию описания состояния на множество случайных переменных во многом аналогично тому, как в алгоритмах планирования используются логические представления для декомпозиции пространства состояний, применяемого в алгоритмах поиска. Поэтому проект агента представляет собой практическую реализацию **агента, действующего с учетом полезности**, который был кратко описан в главе 2.

Поскольку в этом разделе будут использоваться динамические байесовские сети, вернемся к системе обозначений главы 15, где символом $\mathbf{X}_t$ обозначается множество переменных состояния во время t, а $\mathbf{E}_t$ — переменные свидетельства. Таким образом, там, где до сих пор в этой главе использовалось обозначение s_t (состояние во время t), теперь будет применяться обозначение $\mathbf{X}_t$. Для обозначения действия во время t будет использоваться запись A_t, поэтому модель перехода $T(s,a,s')$ представляет собой не что иное, как $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{X}_t,A_t)$, а модель наблюдения $O(s,o)$ — то же, что и $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$. Для обозначения вознаграждения, полученного во время t, будет применяться запись R_t, а для обозначения полезности состояния во время t — запись U_t. При использовании такой системы обозначений динамическая сеть принятия решений принимает вид, подобный показанному на рис. 17.7.

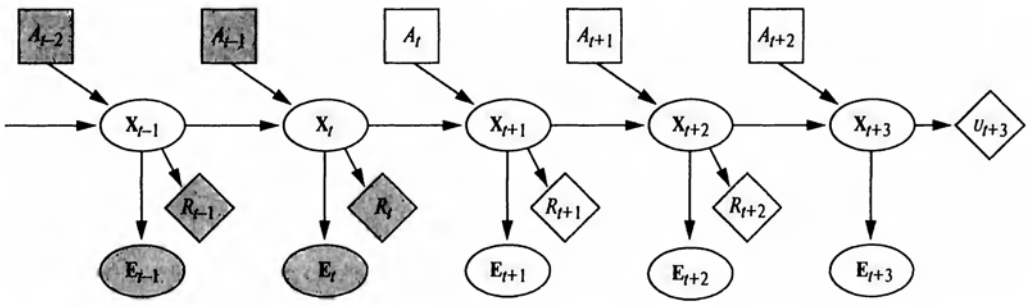

Рис. 17.7. Универсальная структура динамической сети принятия решений. Переменные с известными значениями выделены затенением. Текущим временем является `t`*, а агент должен решить, что делать дальше, т.е. выбрать значение для* A_t*. Сеть развернута в будущее на три этапа и представляет будущее вознаграждение, а также полезность состояния на горизонте прогноза*

Динамические сети принятия решений позволяют получить краткое представление крупных задач POMDP, поэтому могут использоваться в качестве входных данных для любого алгоритма POMDP, включая те из них, которые относятся к методам итерации по значениям и итерации по стратегиям. В этом разделе мы сосредоточимся на прогностических методах, которые проектируют последовательности действий в прямом направлении от текущего доверительного состояния во многом таким же образом, как и алгоритмы ведения игр, описанные в главе 6. Сеть, приведенная на рис. 17.7, спроектирована в будущее на три этапа; неизвестными являются все текущие и будущие решения, а также будущие наблюдения и вознаграждения. Обратите внимание на то, что сеть включает узлы вознаграждений для $\mathbf{X}_{t+1}$ и $\mathbf{X}_{t+2}$, но для $\mathbf{X}_{t+3}$ — только узел полезности. Это связано с тем, что агент должен максимизировать (обесцениваемую) сумму всех будущих вознаграждений, а полезность $U(\mathbf{X}_{t+3})$ представляет и вознаграждение, относящееся к $\mathbf{X}_{t+3}$, и все будущие вознаграждения. Как и в главе 6, предполагается, что значение полезности U доступно только в некоторой приближенной форме, поскольку, если бы были известны точные значения полезности, то не было бы необходимости заглядывать вперед на глубину больше `1`.

На рис. 17.8 показана часть дерева поиска, соответствующего приведенной на рис. 17.7 сети DDN, развернутой в будущее на три этапа. Каждый из узлов, обозначенных треугольником, представляет собой доверительное состояние, в котором агент принимает решение A_{t+i} для $i=0,1,2,\ldots$. Узлы, обозначенные кружками, соответствуют выборам, сделанным средой, а именно тому, какие получены результаты наблюдений $\mathbf{E}_{t+i}$. Обратите внимание на то, что в этой сети нет узлов жеребьевки, соответствующих результатам действий; это связано с тем, что обновление доверительного состояния для любого действия является детерминированным, независимо от фактического результата.

Доверительное состояние каждого обозначенного треугольником узла можно вычислить, применив алгоритм фильтрации к ведущей к нему последовательности наблюдений и действий. Таким образом, в алгоритме учитывается тот факт, что для принятия решения о выполнении действия A_{t+i} агент должен иметь доступные результаты восприятия $\mathbf{E}_{t+1},\ldots,\mathbf{E}_{t+i}$, даже несмотря на то, что во время t он не знает, какими будут эти результаты восприятия. Благодаря этому агент, действующий на основе теории принятия решений, автоматически учитывает стоимость информации и выполняет действия по сбору информации, когда это потребуется.

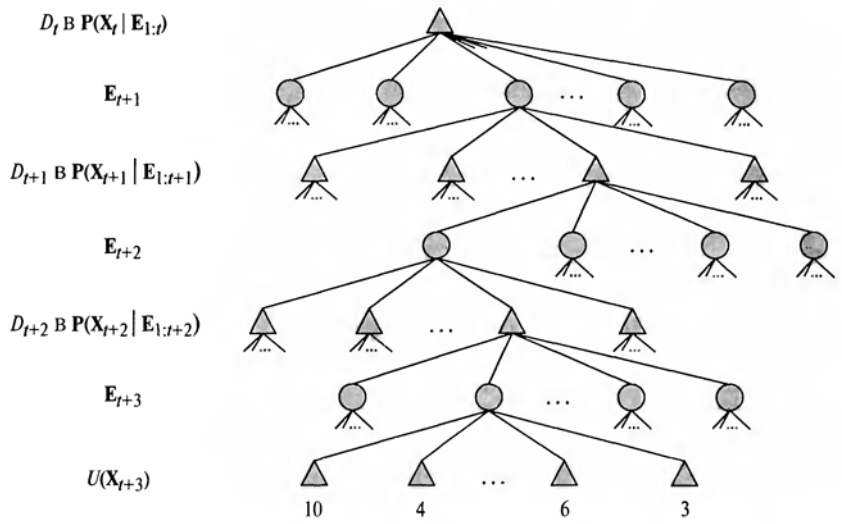

Рис. 17.8. Часть прогностического решения для сети DDN, приведенной на рис. 17.7

Из данного дерева поиска можно извлечь информацию о решении, резервируя значения полезности, взятые из листовых узлов, вычисляя среднее в узлах жеребьевки и определяя максимальные значения в узлах принятия решений. Такая организация работы аналогична применяемой в алгоритме `Expectiminimax` для деревьев игр с узлами жеребьевки, за исключением того, что, во-первых, вознаграждения могут быть предусмотрены и в состояниях, отличных от листовых, и, во-вторых, узлы принятия решений соответствуют доверительным состояниям, а не фактическим состояниям. Временная сложность исчерпывающего поиска до глубины d выражается как $O(|D|^d \cdot |\mathbf{E}|^d)$, где $|D|$ — количество доступных действий; $\mathbf{E}$ — количество возможных наблюдений. Для получения решений, близких к оптимальным, при решении таких задач, в которых коэффициент обесценивания γ не слишком близок к 1, часто бывает вполне приемлемым поверхностный поиск. Существует также возможность аппроксимировать этап усреднения в узлах жеребьевки, осуществляя выборку из множества возможных наблюдений вместо суммирования по всем возможным наблюдениям. Могут также применяться некоторые другие способы быстрого поиска качественных приближенных решений, но мы отложим их описание до главы 21.

Действующие по принципам теории решений агенты, основанные на динамических сетях принятия решений, имеют целый ряд преимуществ по сравнению с другими, более простыми агентами, проекты которых представлены в предыдущих главах. В частности, они способны функционировать в частично наблюдаемых неопределенных вариантах среды и могут легко пересматривать свои "планы" с учетом непредвиденных результатов наблюдений. При использовании подходящих моделей восприятия эти агенты могут справиться с ситуациями наподобие отказа датчиков и способны составлять планы по сбору информации. Под давлением жестких требований ко времени и в сложных вариантах среды эти агенты проявляют способность осуществлять "корректный переход к более примитивному поведению" (graceful degradation) с использованием различных методов вычисления приближенных решений. Так чего же в них недостает? Наиболее важным недостатком агентов, осно-

ванных на использовании алгоритмов динамической сети принятия решений, является то, что в них используется прямой поиск, полностью аналогичный тому, который предусмотрен в алгоритмах поиска в пространстве состояний, описанных в части II. В части IV было показано, что способность рассматривать частично упорядоченные, абстрактные планы с использованием целенаправленного поиска обеспечивает существенное расширение возможностей решения задач, особенно если при этом применяются библиотеки планов. Кроме того, были предприняты попытки распространить эти методы на вероятностную проблемную область, но до сих пор они оказывались неэффективными. Еще одной связанной с этим проблемой является слишком простой язык динамических сетей принятия решений, который по сути является пропозициональным. Было бы желательно распространить на задачи принятия решений некоторые идеи вероятностных языков первого порядка, описанные в разделе 14.6. Современные исследования показали, что такое расширение возможно и что оно позволяет получить существенные преимущества, как описано в заметках в конце данной главы.

17.6. Принятие решений при наличии нескольких агентов: теория игр

Данная глава в основном посвящена теме принятия решений в неопределенных вариантах среды. А как обстоят дела в том случае, если неопределенность связана с наличием других агентов и осуществлением ими решений, которые они принимают? И что будет, если на решения этих агентов, в свою очередь, влияют решения нашего агента? Эти вопросы уже рассматривались в настоящей книге при описании игр в главе 6. Но в этой главе речь в основном шла об играх с полной информацией, в которых игроки делают ходы по очереди: в подобных играх для определения оптимальных ходов может использоваться минимаксный поиск. А в данном разделе рассматриваются некоторые идеи **теории игр**, которые могут применяться при анализе игр с одновременно выполняемыми ходами. Для упрощения изложения вначале рассмотрим игры, которые продолжаются только в течение одного хода. На первый взгляд может показаться, что слово "игра" не совсем подходит для обозначения такого упрощения, которое сводится к одному ходу, но в действительности теория игр используется в очень серьезных ситуациях принятия решений, включая ведение дел о банкротстве, организацию аукционов по распределению спектра радиочастот, принятие решений по разработке промышленной продукции и назначению на нее цен, а также национальную оборону. В таких ситуациях речь часто идет о миллиардах долларов и сотнях тысяч человеческих жизней. Теория игр может использоваться по меньшей мере в двух описанных ниже направлениях.

1. **Проектирование агента**. Теория игр позволяет анализировать решения агента и вычислять ожидаемую полезность для каждого решения (с учетом предположения, что другие агенты действуют оптимальным образом согласно теории игр). Например, в игре **в чет и нечет на двух пальцах** (эту игру на пальцах называют также *morra*, от итальянского слова *camorra* — группа) два игрока, `O` (Odd — нечетный) и `E` (Even — четный), одновременно показывают один или два пальца. Допустим, что общее количество показанных пальцев равно `f`. Если число `f` является нечетным, игрок `O` получает `f` долларов от игрока `E`,