

Agentes inteligentes

Em que discutimos a natureza dos agentes, perfeitos ou não, a diversidade de ambientes e a consequente variedade de tipos de agentes.

O Capítulo 1 identificou o conceito de **agentes racionais** como questão central para nossa abordagem da inteligência artificial. Neste capítulo, tornaremos essa noção mais concreta.

Veremos que o conceito de racionalidade pode ser aplicado a uma ampla variedade de agentes que operam em qualquer ambiente imaginável. Nosso plano neste livro é usar esse conceito para desenvolver um pequeno conjunto de princípios de projeto com a finalidade de construir sistemas de agentes bem-sucedidos — sistemas que possam ser adequadamente chamados **inteligentes**.

Começaremos examinando agentes, ambientes e o acoplamento entre eles. A observação de que alguns agentes se comportam melhor que outros leva naturalmente à ideia de agente racional — um agente que se comporta tão bem quanto possível. A medida da qualidade do comportamento de um agente depende da natureza do ambiente; alguns ambientes são mais difíceis que outros. Apresentaremos uma divisão geral dos ambientes em categorias e mostraremos como as propriedades de um ambiente influenciam o projeto de agentes adequados para esse ambiente. Descreveremos vários “esqueletos” básicos de projetos de agentes que serão utilizados no restante do livro.

2.1 AGENTES E AMBIENTES

Um **agente** é tudo o que pode ser considerado capaz de perceber seu **ambiente** por meio de **sensores** e de agir sobre esse ambiente por intermédio de **atuadores**. Essa ideia simples é ilustrada na Figura 2.1. Um agente humano tem olhos, ouvidos e outros órgãos como sensores, e tem mãos, pernas, boca e outras partes do corpo que servem como atuadores. Um agente robótico pode ter câmeras e detectores da faixa de infravermelho funcionando como sensores e vários motores como atuadores. Um agente de software recebe sequências de teclas digitadas, conteúdo de arquivos e pacotes de rede como entradas sensórias e atua sobre o ambiente exibindo algo na tela, escrevendo em arquivos e enviando pacotes de rede.



Usamos o termo **percepção** para fazer referência às entradas perceptivas do agente em um dado instante. A **sequência de percepções** do agente é a história completa de tudo o que o agente já

percebeu. Em geral, a escolha de ação de um agente em qualquer instante dado pode depender da sequência inteira de percepções recebidas até o momento, mas não de percepções não recebidas. Se pudermos especificar a escolha de ação do agente para toda sequência de percepções possível, teremos dito quase tudo o que existe a dizer sobre o agente. Em termos matemáticos, afirmamos que o comportamento do agente é descrito pela **função do agente** que mapeia qualquer sequência de percepções específica para uma ação.

Podemos imaginar a *tabulação* da função do agente que descreve qualquer agente dado; para a maioria dos agentes, o resultado seria uma tabela muito grande — na verdade infinita, a menos que seja definido um limite sobre o comprimento das sequências de percepções que queremos considerar. Dado um agente para a realização de experimentos, podemos, em princípio, construir essa tabela tentando todas as sequências de percepções e registrando as ações que o agente executa em resposta.¹ É claro que a tabela é uma caracterização *externa* do agente. *Internamente*, a função do agente para um agente artificial será implementada pelo **programa do agente**. É importante manter essas duas ideias distintas. A função de agente é uma descrição matemática abstrata; o programa do agente é uma implementação concreta, executada em um sistema físico.

Para ilustrar essas ideias, usaremos um exemplo muito simples — o mundo de aspirador de pó ilustrado na Figura 2.2. Esse mundo é tão simples que podemos descrever tudo o que acontece; ele também é um mundo inventado e, portanto, podemos criar muitas variações. Esse mundo particular tem apenas dois locais: os quadrados *A* e *B*. O agente aspirador de pó percebe em que quadrado está e se existe sujeira no quadrado. Ele pode optar por mover-se para a esquerda, mover-se para a direita, aspirar a sujeira ou não fazer nada. Uma função do agente muito simples é: se o quadrado atual estiver sujo, então aspirar, caso contrário mover-se para o outro quadrado. Uma tabulação parcial da função desse agente é mostrada na Figura 2.3 e um programa do agente que o implementa aparece na Figura 2.8, página 43.

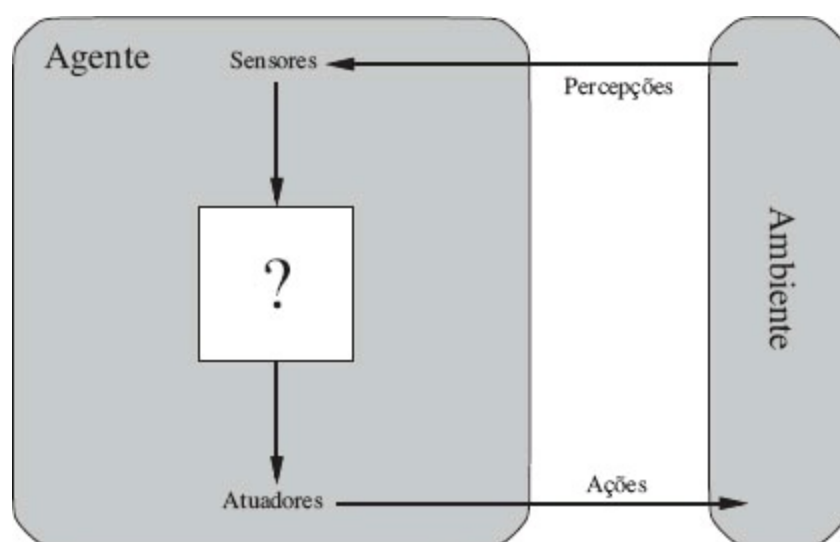


Figura 2.1 Agentes interagem com ambientes por meio de sensores e atuadores.

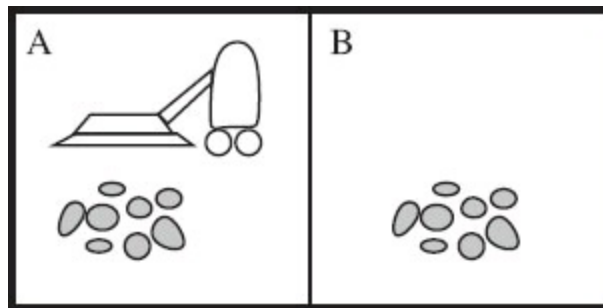


Figura 2.2 Um mundo de aspirador de pó com apenas dois locais.

Sequência de percepções	Ação
[A, Limpo]	
[A, Sujo]	
[B, Limpo]	
[B, Sujo]	
[A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Sujo]	Aspirar
.	Esquerda
.	Aspirar
.	Direita
[A, Limpo], [A, Limpo], [A, Limpo]	Aspirar
[A, Limpo], [A, Limpo], [A, Sujo]	Direita
.	Aspirar
.	
.	

Figura 2.3 Tabulação parcial de uma função de agente simples correspondente ao mundo de aspirador de pó mostrado na Figura 2.2.

Examinando a Figura 2.3, vemos diversos agentes do mundo de aspirador de pó que podem ser definidos simplesmente preenchendo-se de várias maneiras a coluna da direita. Então, a pergunta óbvia é: *Qual é a maneira correta de preencher a tabela?* Em outras palavras, o que torna um agente bom ou ruim, inteligente ou estúpido? Responderemos a essas perguntas na próxima seção.


Antes de fecharmos esta seção, enfatizaremos que a noção de um agente deve ser vista como uma ferramenta para analisar sistemas, não como uma caracterização absoluta que divide o mundo em agentes e não agentes. Poderíamos visualizar uma calculadora portátil como um agente que escolhe a ação de exibir “4” ao receber a sequência de percepções “2 + 2 = ”, mas tal análise dificilmente ajudaria nossa compreensão da calculadora. De certo modo, todas as áreas de engenharia podem ser vistas como projetar artefatos que interagem com o mundo; a IA opera no que os autores consideram ser o final mais interessante do espectro, onde os artefatos têm consideráveis recursos computacionais e o ambiente de tarefa requer uma tomada de decisão não trivial.

2.2 BOM COMPORTAMENTO: O CONCEITO DE RACIONALIDADE

Um **agente racional** é aquele que faz tudo certo — em termos conceituais, toda entrada na tabela correspondente à função do agente é preenchida de forma correta. É óbvio que fazer tudo certo é melhor do que fazer tudo errado; porém, o que significa fazer tudo certo?

Responderemos a essa antiga questão de uma forma antiquada: considerando as *consequências* do comportamento do agente. Quando um agente é colocado em um ambiente, gera uma sequência de ações de acordo com as percepções que recebe. Essa sequência de ações faz com que o ambiente passe por uma sequência de estados. Se a sequência for desejável, o agente teve bom desempenho. Essa noção de “desejável” é capturada por uma **medida de desempenho** que avalia qualquer sequência dada dos estados do ambiente.

Observe que dissemos estados do *ambiente*, não estados do *agente*. Se definirmos sucesso em termos da opinião do agente do seu próprio desempenho, um agente poderia alcançar a racionalidade perfeita simplesmente iludindo-se de que seu desempenho foi perfeito. Os agentes humanos em particular são notórios por ficar com “dor de cotovelo”, acreditando que realmente não queriam alguma coisa (por exemplo, um Prêmio Nobel) depois de não conseguir.

 Obviamente, não há uma medida de desempenho fixa para todas as tarefas e agentes; normalmente, um projetista vai desenvolver uma adequada às circunstâncias. Não é tão fácil como parece. Considere, por exemplo, o agente aspirador de pó da seção anterior. Poderíamos propor medir o desempenho pela quantidade de sujeira aspirada em um único turno de oito horas. É claro que, no caso de um agente racional, você obtém aquilo que solicita. Um agente racional pode maximizar essa medida de desempenho limpando a sujeira e, em seguida, despejando-a toda no chão, depois limpando novamente, e assim por diante. Uma medida de desempenho mais apropriada recompensaria o agente por deixar o chão limpo. Por exemplo, ele poderia ser recompensado por cada quadrado limpo em cada período (talvez com uma penalidade pela eletricidade consumida e pelo ruído gerado). *Como regra geral, é melhor projetar medidas de desempenho de acordo com o resultado realmente desejado no ambiente, em vez de criá-las de acordo com o comportamento esperado do agente.*

Mesmo que as armadilhas óbvias sejam evitadas, ainda existem algumas questões complexas para desembaraçar. Por exemplo, a noção de “chão limpo” no parágrafo anterior se baseia na limpeza média ao longo do tempo. Ainda assim, a mesma limpeza média pode ser alcançada por dois agentes diferentes, um dos quais faz o trabalho tedioso de limpeza o tempo todo, enquanto o outro limpa energicamente, mas faz longas pausas. A estratégia preferível pode parecer um detalhe secundário da ciência do trabalho doméstico, mas de fato é uma profunda questão filosófica com extensas implicações. O que é melhor: uma vida aventureira, cheia de altos e baixos, ou uma existência segura, porém monótona? O que é melhor: uma economia em que todos vivam em pobreza moderada ou aquela em que alguns vivem em plena riqueza enquanto outros são muito pobres? Deixaremos essas perguntas como exercício para o leitor.

2.2.1 Racionalidade

A definição do que é racional em qualquer instante dado depende de quatro fatores:

- A medida de desempenho que define o critério de sucesso.
- O conhecimento prévio que o agente tem do ambiente.
- As ações que o agente pode executar.
- A sequência de percepções do agente até o momento.

Isso conduz a uma **definição de um agente racional**:



Para cada sequência de percepções possível, um agente racional deve selecionar uma ação que se espera venha a maximizar sua medida de desempenho, dada a evidência fornecida pela sequência de percepções e por qualquer conhecimento interno do agente.

Considere o agente aspirador de pó simples que limpa um quadrado se ele estiver sujo e passa para o outro quadrado se o primeiro não estiver sujo; essa é a função do agente tabulada na Figura 2.3. Esse é um agente racional? Depende! Primeiro, precisamos dizer o que é a medida de desempenho, o que se conhece sobre o ambiente e quais são os sensores e atuadores que o agente tem. Vamos supor que:

- A medida de desempenho ofereça o prêmio de um ponto para cada quadrado limpo em cada período de tempo, ao longo de um “tempo de vida” de 1.000 passos de tempo.
- A “geografia” do ambiente seja conhecida *a priori* (Figura 2.2), mas a distribuição da sujeira e a posição inicial do agente não sejam previamente conhecidas. Quadrados limpos permanecem limpos, e a aspiração limpa o quadrado atual. As ações *Esquerda* e *Direita* movem o agente para a esquerda e para a direita, exceto quando isso leva o agente para fora do ambiente; nesse caso, o agente permanece onde está.
- As únicas ações disponíveis são *Esquerda*, *Direita* e *Aspirar*.
- O agente percebe corretamente sua posição e se essa posição contém sujeira.

Afirmamos que, *sob essas circunstâncias*, o agente é de fato racional; espera-se que seu desempenho seja pelo menos tão alto quanto o de qualquer outro agente. O Exercício 2.2 lhe pede para provar esse fato.

Podemos ver facilmente que o mesmo agente seria irracional sob circunstâncias diferentes. Por exemplo, uma vez que toda a sujeira seja limpa, o agente oscila desnecessariamente de um lado para outro; se a medida de desempenho incluir uma penalidade de um ponto para cada movimento à esquerda ou à direita, o agente ficará em má situação. Um agente melhor para esse caso não faria nada se tivesse certeza de que todos os quadrados estão limpos. Se quadrados limpos puderem ficar sujos novamente, o agente deve ocasionalmente verificar e voltar a limpá-los, se necessário. Se a geografia do ambiente for desconhecida, o agente precisará explorá-la, em vez de se fixar nos quadrados *A* e *B*. O Exercício 2.2 pede para projetar agentes para esses casos.

2.2.2 Onisciência, aprendizado e autonomia

Precisamos ter o cuidado de distinguir entre racionalidade e **onisciência**. Um agente onisciente

sabe o resultado *real* de suas ações e pode agir de acordo com ele; porém, a onisciência é impossível na realidade. Considere o exemplo a seguir: estou caminhando nos Champs Elysées e de repente vejo um velho amigo do outro lado da rua. Não existe nenhum tráfego perto e não tenho nenhum outro compromisso; assim, sendo racional, começo a atravessar a rua. Enquanto isso, a 10.000 metros de altura, a porta do compartimento de carga se solta de um avião² e, antes de chegar ao outro lado da rua, sou atingido. Foi irracional atravessar a rua? É improvável que a notícia de minha morte fosse “idiota tenta cruzar rua”.

Esse exemplo mostra que racionalidade não é o mesmo que perfeição. A racionalidade maximiza o desempenho *esperado*, enquanto a perfeição maximiza o desempenho *real*. Fugir à exigência de perfeição não é apenas uma questão de ser justo com os agentes. Se esperarmos que um agente realize aquela que virá a ser a melhor ação após o fato, será impossível projetar um agente para satisfazer essa especificação, a menos que melhoremos o desempenho de bolas de cristal ou máquinas do tempo.

Portanto, nossa definição de racionalidade não exige onisciência porque a escolha racional só depende da sequência de percepções *até o momento*. Também devemos assegurar que não permitimos que o agente se engaje sem querer em atividades decididamente pouco inteligentes. Por exemplo, se um agente não olhar para os dois lados antes de atravessar uma estrada movimentada, sua sequência de percepções não o informará de que existe um grande caminhão se aproximando em alta velocidade. Nossa definição de racionalidade afirmaria que agora é correto atravessar a estrada? Longe disso! Primeiro, não seria racional atravessar a estrada dada essa sequência de percepções pouco informativa: o risco de acidente resultante de atravessar a estrada sem olhar para os lados é muito grande. Em segundo lugar, um agente racional deveria escolher a ação “olhar” antes de iniciar a travessia porque olhar ajuda a maximizar o desempenho esperado. A realização de ações *com a finalidade de modificar percepções futuras* — às vezes chamada **coleta de informações** — é uma parte importante da racionalidade e é abordada em profundidade no Capítulo 16. Um segundo exemplo de coleta de informações é dado pela **exploração** que tem de ser empreendida por um agente aspirador de pó em um ambiente inicialmente desconhecido.

Nossa definição exige um agente racional não apenas para coletar informações, mas também para **aprender** tanto quanto possível a partir do que ele percebe. A configuração inicial do agente poderia refletir algum conhecimento prévio do ambiente, mas, à medida que o agente ganha experiência, isso pode ser modificado e ampliado. Existem casos extremos em que o ambiente é completamente conhecido *a priori*. Em tais casos, o agente não precisa perceber ou aprender; ele simplesmente age de forma correta. É claro que tais agentes são muito frágeis. Considere o humilde besouro de esterco. Depois de cavar seu ninho e depositar os ovos, ele busca uma bola de esterco em um monte próximo para fechar a entrada. Se, *durante o percurso*, a bola de esterco for removida de suas garras, o besouro seguirá em frente e imitará o fechamento do ninho com a bola de esterco inexistente, sem notar que ela foi retirada. A evolução construiu uma suposição sobre o comportamento do besouro e, quando essa hipótese é violada, resulta um comportamento malsucedido. A vespa *Sphex* é um pouco mais inteligente. A fêmea da *Sphex* cava uma cova, sai, pica uma lagarta e a arrasta até a borda da cova, entra novamente na cova para verificar se tudo está bem, arrasta a lagarta para dentro e deposita seus ovos. A lagarta servirá como alimento quando os ovos eclodirem. Até aqui tudo bem, mas se um entomologista afastar a lagarta algumas polegadas enquanto a fêmea estiver fazendo a

verificação, ela voltará à etapa de “arrastar” de seu plano e continuará o plano sem modificação, mesmo depois de dezenas de intervenções de afastamento de lagartas. A Sphex é incapaz de aprender que seu plano inato está falhando e, portanto, não o modificará.

Quando um agente se baseia no conhecimento anterior de seu projetista e não em suas próprias percepções, dizemos que o agente não tem **autonomia**. Um agente racional deve ser autônomo — ele deve aprender o que puder para compensar um conhecimento prévio parcial ou incorreto. Por exemplo, um agente aspirador de pó que aprende a prever onde e quando aparecerá mais sujeira funcionará melhor que um agente incapaz de fazer essa previsão. Na prática, raramente se exige autonomia completa desde o início: quando o agente tem pouca ou nenhuma experiência, ele deve agir ao acaso, a menos que o projetista tenha dado a ele alguma assistência. Então, da mesma forma que a evolução fornece aos animais reflexos internos suficientes para que eles possam sobreviver pelo tempo necessário para aprenderem por si mesmos, seria razoável fornecer a um agente de inteligência artificial algum conhecimento inicial, bem como habilidade para aprender.

Depois de adquirir experiência suficiente sobre seu ambiente, o comportamento de um agente racional pode se tornar efetivamente *independente* de seu conhecimento anterior. Em consequência disso, a incorporação do aprendizado permite projetar um único agente racional que terá sucesso em ampla variedade de ambientes.

2.3 A NATUREZA DOS AMBIENTES

Agora que temos uma definição de racionalidade, estamos quase prontos para pensar em construir agentes racionais. Porém, primeiro devemos pensar em **ambientes de tarefas**, que são essencialmente os “problemas” para os quais os agentes racionais são as “soluções”. Começamos mostrando como especificar um ambiente de tarefa ilustrando o processo com vários exemplos. Em seguida, mostramos que há vários tipos de ambientes de tarefas. O tipo de ambiente de tarefa afeta diretamente o projeto apropriado para o programa do agente.

2.3.1 Especificando o ambiente de tarefa

Em nossa discussão sobre a racionalidade do agente aspirador de pó simples, tivemos de especificar a medida de desempenho, o ambiente e os atuadores e sensores do agente. Agruparemos todos esses itens sob o título **ambiente da tarefa**. Para os leitores que gostam de acrônimos, chamaremos essa descrição de **PEAS** (**P**erformance, **E**nvironment, **A**ctuators, **S**ensors — desempenho, ambiente, atuadores, sensores). Ao projetar um agente, a primeira etapa deve ser sempre especificar o ambiente de tarefa de forma tão completa quanto possível.

O mundo do aspirador de pó foi um exemplo simples; vamos considerar um problema mais complexo: um motorista de táxi automatizado. Utilizaremos esse exemplo em todo o restante do capítulo. Devemos destacar, antes que o leitor fique alarmado, que um táxi totalmente automatizado no momento está um pouco além da capacidade da tecnologia atual (veja, na página 28, uma descrição de um robô motorista). A tarefa completa de dirigir é extremamente *aberta*. Não existe

nenhum limite para as novas combinações de circunstâncias que podem surgir — outra razão para termos escolhido essa tarefa como foco de discussão. A Figura 2.4 resume a descrição PEAS para o ambiente de tarefa do táxi. Descreveremos cada elemento com mais detalhes nos próximos parágrafos.

Tipo de agente	Medida de desempenho	Ambiente	Atuadores	Sensores
Motorista de táxi	Viagem segura, rápida, dentro da lei, confortável, maximizar lucros	Estradas, outros tipos de tráfego, pedestres, clientes	Direção, acelerador, freio, sinal, buzina, visor	Câmeras, sonar, velocímetro, GPS, hodômetro, acelerômetro, sensores do motor, teclado

Figura 2.4 Descrição de PEAS do ambiente de tarefa para um táxi automatizado.

Primeiro, que **medida de desempenho** gostaríamos que nosso motorista automatizado tivesse como objetivo? As qualidades desejáveis incluem chegar ao destino correto, minimizar o consumo de combustível e desgaste, minimizar o tempo e/ou o custo de viagem, minimizar as violações às leis de trânsito e as perturbações a outros motoristas, maximizar a segurança e o conforto dos passageiros e maximizar os lucros. É óbvio que alguns desses objetivos serão conflitantes; então será necessário fazer uma escolha.

Em seguida, qual é o **ambiente** de direção que o táxi enfrentará? Qualquer motorista de táxi deve lidar com diversos tipos de estradas, variando desde estradas rurais e avenidas urbanas até rodovias com 12 pistas. As estradas contêm outros tipos de tráfego, pedestres, animais perdidos, trabalhadores na pista, policiamento, poças e buracos. O táxi também deve interagir com passageiros potenciais e reais. Existem ainda algumas escolhas opcionais. O táxi poderia precisar operar no sul da Califórnia, onde a neve raramente é um problema, ou no Alasca, onde ela normalmente é um problema. Ele sempre poderia estar dirigindo no lado direito da pista ou talvez quiséssemos que ele fosse flexível o bastante para dirigir no lado esquerdo quando estivesse na Inglaterra ou no Japão. É óbvio que, quanto mais restrito o ambiente, mais fácil se torna o problema de projetar.

Os **atuadores** para um táxi automatizado incluem aqueles disponíveis para um motorista humano: controle sobre o motor através do acelerador e controle sobre a direção e a frenagem. Além disso, ele precisará da saída para uma tela de exibição ou um sintetizador de voz para se comunicar com os passageiros e, talvez, de algum meio para se comunicar com outros veículos, de forma educada ou não.

Os **sensores** básicos do táxi vão incluir uma ou mais câmeras de TV controláveis para que possa observar a estrada, que podem ser potencializadas com infravermelho ou sensor sonar para detectar distâncias de outros carros e obstáculos. Para evitar multas por excesso de velocidade, o táxi deverá possuir velocímetro, e, para controlar o veículo de forma correta, especialmente em curvas, deverá ter um acelerômetro. Para conhecer o estado mecânico do veículo, será necessário o conjunto habitual de sensores do motor, combustível e sistema elétrico. Como muitos motoristas humanos, pode querer um sistema de posicionamento global por satélite (GPS) para não se perder. Finalmente,

ele precisará de um teclado ou microfone para que o passageiro possa solicitar um destino.

N a Figura 2.5, esboçamos os elementos básicos do PEAS para diversos tipos de agentes. Exemplos adicionais aparecem no Exercício 2.4. Talvez seja surpresa para alguns leitores que a nossa lista de tipos de agentes inclua alguns programas que operam no ambiente completamente artificial definido pela entrada no teclado e pela saída de caracteres em uma tela. Alguém poderia dizer: “Certamente, esse não é um ambiente real, é?” De fato, o que importa não é a distinção entre ambientes “reais” e “artificiais”, mas a complexidade do relacionamento entre o comportamento do agente, a sequência de percepções gerada pelo ambiente e a medida de desempenho. Alguns ambientes “reais” na realidade são bastante simples. Por exemplo, um robô projetado para inspecionar peças à medida que elas chegam em uma correia transportadora pode fazer uso de uma série de suposições simplificadoras: que a iluminação será sempre perfeita, que os únicos itens na correia transportadora serão peças de um tipo que ele conhece e que apenas duas ações serão possíveis (aceitar ou rejeitar).

Tipo de agente	Medida de desempenho	Ambiente	Atuadores	Sensores
Sistema de diagnóstico médico	Paciente saudável, minimizar custos	Paciente, hospital, equipe	Exibir perguntas, testes, diagnósticos, tratamentos, indicações	Entrada pelo teclado para sintomas, descobertas, respostas do paciente
Sistema de análise de imagens de satélite	Definição correta da categoria da imagem	Link de transmissão de satélite em órbita	Exibir a categorização da cena	Arrays de pixels em cores
Robô de seleção de peças	Porcentagem de peças em bandejas corretas	Correia transportadora com peças; bandejas	Braço e mão articulados	Câmera, sensores angulares articulados
Controlador de refinaria	Maximizar pureza, rendimento, segurança	Refinaria, operadores	Válvulas, bombas, aquecedores, mostradores	Sensores de temperatura, pressão, produtos químicos
Instrutor de inglês interativo	Maximizar nota de aluno em teste	Conjunto de alunos, ambiente de testes	Exibir exercícios, sugestões, correções	Entrada pelo teclado

Figura 2.5 Exemplos de tipos de agentes e suas descrições PEAS.

Em contraste, existem alguns **agentes de software** (ou robôs de software ou, ainda, **softbots**) em ambientes ricos e ilimitados. Imagine um softbot operador de website, projetado para vasculhar fontes de notícias da Internet e mostrar os itens interessantes a seus clientes, enquanto vende espaço

de publicidade para gerar renda. Para funcionar bem, ele precisará de algumas habilidades de processamento de linguagem natural, precisará aprender o que interessa a cada usuário e investidor e terá de mudar seus planos dinamicamente — por exemplo, quando a conexão para uma fonte de notícias cair ou quando uma nova fonte estiver on-line. A Internet é um ambiente cuja complexidade rivaliza com a do mundo físico e cujos habitantes incluem muitos agentes artificiais e humanos.

2.3.2 Propriedades de ambientes de tarefas

A variedade de ambientes de tarefas que podem surgir em IA é sem dúvida vasta. Entretanto, podemos identificar um número bastante reduzido de dimensões ao longo das quais os ambientes de tarefas podem ser divididos em categorias. Em grande parte, essas dimensões determinam o projeto apropriado de agentes e a aplicabilidade de cada uma das principais famílias de técnicas de implementação de agentes. Primeiro, listamos as dimensões, depois analisamos vários ambientes de tarefas para ilustrar as ideias. Aqui, as definições são informais; os capítulos posteriores fornecerão declarações e exemplos mais precisos de cada tipo de ambiente.

Completamente observável versus parcialmente observável: Se os sensores de um agente permitem acesso ao estado completo do ambiente em cada instante, dizemos que o ambiente de tarefa é completamente observável. Um ambiente de tarefa é de fato completamente observável se os sensores detectam todos os aspectos que são *relevantes* para a escolha da ação; por sua vez, a relevância depende da medida de desempenho. Ambientes completamente observáveis são convenientes porque o agente não precisa manter qualquer estado interno para acompanhar as mudanças do mundo. Um ambiente poderia ser parcialmente observável devido ao ruído e a sensores imprecisos ou porque partes do estado estão simplesmente ausentes nos dados do sensor — por exemplo, um agente aspirador de pó com apenas um sensor de sujeira local não pode saber se há sujeira em outros quadrados, e um táxi automatizado não pode saber o que outros motoristas estão pensando. Se o agente não tiver sensores, o ambiente será **inobservável**. Alguém poderia pensar que, nesses casos, a situação do agente fica desesperadora, mas, como discutiremos no Capítulo 4, os objetivos do agente ainda poderão ser alcançáveis, e em alguns casos, com certeza.

Agente único versus multiagente: A distinção entre ambientes de agente único e de multiagente pode parecer bastante simples. Por exemplo, um agente que resolve um jogo de palavras cruzadas sozinho está claramente em um ambiente de agente único, enquanto um agente que joga xadrez está em um ambiente de dois agentes. Porém, existem algumas questões sutis. Primeiro, descrevemos como uma entidade *pode* ser visualizada como um agente, mas não explicamos que entidades *devem* ser visualizadas como agentes. Um agente *A* (por exemplo, o motorista de táxi) tem de tratar um objeto *B* (outro veículo) como um agente ou ele pode ser tratado apenas como um objeto comportando-se de acordo com as leis da física, análogo às ondas do mar ou às folhas espalhadas pelo vento? A distinção fundamental é saber se o comportamento de *B* é ou não melhor descrito como a maximização de uma medida de desempenho cujo valor depende do comportamento do agente *A*. Por exemplo, em xadrez, a entidade oponente *B* está tentando maximizar sua medida de desempenho que, pelas regras de xadrez, minimiza a medida de desempenho do agente *A*. Desse modo, o jogo de xadrez é um ambiente de multiagente **competitivo**. Por outro lado, no ambiente de direção de um táxi,

evitar colisões maximiza a medida de desempenho de todos os agentes; assim, esse é um ambiente de multiagente parcialmente **cooperativo**. Ele também é parcialmente competitivo porque, por exemplo, apenas um carro pode ocupar um espaço no estacionamento. Os problemas de projeto de agentes que surgem em ambientes de multiagentes muitas vezes são bem diferentes dos que surgem em ambientes de um único agente; por exemplo, a **comunicação** com frequência emerge como um comportamento racional em ambientes de multiagentes; em alguns ambientes competitivos parcialmente observáveis, o **comportamento aleatório** é racional porque evita as armadilhas da previsibilidade.

Determinístico versus estocástico: Se o próximo estado do ambiente é completamente determinado pelo estado atual e pela ação executada pelo agente, dizemos que o ambiente é determinístico; caso contrário, ele é estocástico. Em princípio, um agente não precisa se preocupar com a incerteza em um ambiente completamente observável e determinístico. (Na nossa definição, ignoramos a incerteza que surge exclusivamente das ações de outros agentes em um ambiente multiagente; assim, um jogo pode ser determinístico, mesmo sendo cada agente incapaz de prever as ações dos outros.) Porém, se o ambiente for parcialmente observável, ele poderá *parecer* estocástico. A maioria das situações reais é tão complexa que é impossível acompanhar todos os aspectos não observados; para finalidades práticas devem ser tratados como estocásticos. O motorista de táxi é claramente estocástico nesse sentido porque nunca se pode prever o comportamento do tráfego com exatidão; além disso, pode ocorrer o estouro de um pneu e a falha de um motor sem aviso prévio. O mundo do aspirador de pó que descrevemos é determinístico, mas as variações podem incluir elementos estocásticos, como o aparecimento de sujeira ao acaso e um mecanismo de sucção não confiável (Exercício 2.13). Dizemos que um ambiente é incerto se não for totalmente observável ou determinístico. Observação final: o nosso uso da palavra “estocástico” geralmente implica que a incerteza sobre os resultados é quantificada em termos de probabilidades; um ambiente **não determinístico** é aquele em que as ações são caracterizadas por seus resultados *possíveis*, sem probabilidade associada a ele. As descrições do ambiente não determinístico são normalmente associadas às medidas de desempenho que exigem que o agente tenha sucesso em *todos* os resultados *possíveis* de suas ações.

Episódico versus sequencial: Em um ambiente de tarefa episódico, a experiência do agente é dividida em episódios atômicos. Em cada episódio, o agente recebe uma percepção e em seguida executa uma única ação. É crucial que o episódio seguinte não dependa das ações executadas em episódios anteriores. Em ambientes episódicos, a escolha da ação em cada episódio só depende do próprio episódio. Muitas tarefas de classificação são episódicas. Por exemplo, um agente que tem de localizar peças defeituosas em uma linha de montagem baseia cada decisão na peça atual, independentemente das decisões anteriores; além disso, a decisão atual não afeta o fato da próxima peça estar ou não com defeito. Por outro lado, em ambientes sequenciais, a decisão atual poderia afetar todas as decisões futuras.³ Jogar xadrez e dirigir um táxi são sequenciais: em ambos os casos, ações em curto prazo podem ter consequências a longo prazo. Ambientes episódicos são muito mais simples que ambientes sequenciais porque o agente não precisa pensar à frente.

Estático versus dinâmico: Se o ambiente puder se alterar enquanto um agente está deliberando, dizemos que o ambiente é dinâmico para esse agente; caso contrário, ele é estático. Ambientes estáticos são fáceis de manipular porque o agente não precisa continuar a observar o mundo enquanto está decidindo sobre a realização de uma ação nem precisa se preocupar com a passagem do tempo.

Por outro lado, ambientes dinâmicos estão continuamente perguntando ao agente o que ele deseja fazer; se ele ainda não tiver se decidido, isso será considerado a decisão de não fazer nada. Se o próprio ambiente não mudar com a passagem do tempo, mas o nível de desempenho do agente se alterar, diremos que o ambiente é **semidinâmico**. O ambiente em que se dirige um táxi é claramente dinâmico: os outros carros e o próprio táxi continuam a se mover enquanto o algoritmo de direção hesita sobre o que fazer em seguida. O jogo de xadrez, quando jogado com a contagem do tempo, é semidinâmico. O jogo de palavras cruzadas é estático.

Discreto versus contínuo: A distinção entre discreto e contínuo aplica-se ao *estado* do ambiente, ao modo como o *tempo* é tratado, e ainda às *percepções* e *ações* do agente. Por exemplo, um ambiente de jogo de xadrez tem um número finito de estados distintos (excluindo o relógio). O xadrez também tem um conjunto discreto de percepções e ações. Dirigir um táxi é um problema de estado contínuo e tempo contínuo: a velocidade e a posição do táxi e dos outros veículos passam por um intervalo de valores contínuos e fazem isso suavemente ao longo do tempo. As ações de dirigir um táxi também são contínuas (ângulos de rotação do volante etc.). A entrada proveniente de câmeras digitais é discreta, em termos estritos, mas em geral é tratada como a representação de intensidades e posições que variam continuamente.

Conhecido versus desconhecido: Estritamente falando, essa distinção não se refere ao ambiente em si, mas ao estado de conhecimento do agente (ou do projetista) sobre as “leis da física” no meio ambiente. Em um ambiente conhecido, são fornecidas as saídas (ou probabilidades das saídas se o ambiente for estocástico) para todas as ações. Obviamente, se o ambiente for desconhecido, o agente terá de aprender como funciona, a fim de tomar boas decisões. Observe que a distinção entre os ambientes conhecido e desconhecido não é a mesma que entre ambientes totalmente e parcialmente observáveis. É perfeitamente possível para um ambiente *conhecido* ser *parcialmente* observável — por exemplo, em jogos de cartas solitários, eu conheço as regras, mas sou incapaz de ver as cartas que ainda não foram viradas. Por outro lado, um ambiente *desconhecido* pode ser *totalmente* observável — em um novo videogame, a tela pode mostrar o estado inteiro do jogo, mas eu ainda não sei o que os botões fazem até experimentá-los.

Como se poderia esperar, o caso mais difícil é *parcialmente observável, multiagente, estocástico, sequencial, dinâmico, contínuo e desconhecido*. Dirigir um táxi é difícil em todos esses sentidos, exceto que para a maioria dos motoristas o ambiente é conhecido. Dirigir um carro alugado em um país desconhecido, com a geografia e de leis de trânsito desconhecidas, é muito mais emocionante.

A Figura 2.6 lista as propriedades de vários ambientes familiares. Observe que as respostas nem sempre são definitivas. Por exemplo, descrevemos o robô de seleção de peças como episódico porque ele normalmente considera cada peça isoladamente. Mas, se um dia houver um grande lote de peças defeituosas, o robô deverá aprender através de várias observações que a distribuição de defeitos mudou e deverá modificar o seu comportamento para as peças subsequentes. A coluna “conhecido/desconhecido” não foi incluída porque, como explicado anteriormente, ela não é estritamente uma propriedade do ambiente. Em alguns ambientes, tais como xadrez e pôquer, é muito fácil suprir o agente com pleno conhecimento das regras, mas não deixa de ser interessante considerar como um agente poderá aprender a jogar esses jogos sem tal conhecimento.

Ambiente de tarefa	Observável	Agentes	Determinístico	Episódico	Estático	Discreto
Jogo de palavras cruzadas Xadrez com um relógio	Completamente Completamente	Único Multi	Determinístico Determinístico	Sequencial Sequencial	Estático Semi	Discreto Discreto
Pôquer Gamão	Parcialmente Completamente	Multi Multi	Estocástico Estocástico	Sequencial Sequencial	Estático Estático	Discreto Discreto
Direção de táxi Diagnóstico médico	Parcialmente Parcialmente	Multi Único	Estocástico Estocástico	Sequencial Sequencial	Dinâmico Dinâmico	Contínuo Contínuo
Análise de imagens Robô de seleção de peças	Completamente Parcialmente	Único Único	Determinístico Estocástico	Episódico Episódico	Semi Dinâmico	Contínuo Contínuo
Controlador de refinaria Instrutor interativo de inglês	Parcialmente Parcialmente	Único Multi	Estocástico Estocástico	Sequencial Sequencial	Dinâmico Dinâmico	Contínuo Discreto

Figura 2.6 Exemplos de ambientes de tarefas e suas características.

Muitas das respostas na tabela dependem da forma como o ambiente de tarefa é definido. Listamos a tarefa de diagnóstico médico como uma tarefa de agente único porque o processo de doença em um paciente não poderia ser modelado de modo proveitoso como um agente; porém, um sistema de diagnóstico médico também poderia ter de lidar com pacientes obstinados e funcionários céticos e, assim, o ambiente poderia ter um aspecto multiagente. Além disso, o diagnóstico médico é episódico se a tarefa for concebida como a seleção de um diagnóstico dada uma lista de sintomas; o diagnóstico será sequencial se a tarefa puder incluir a proposição de uma série de testes, a avaliação do progresso durante o tratamento, e assim por diante. Também há muitos ambientes episódicos em níveis mais altos que as ações individuais do agente. Por exemplo, um torneio de xadrez consiste em uma sequência de jogos; cada jogo é um episódio porque (em geral) a contribuição dos movimentos em um jogo para o desempenho global do agente não é afetada pelos movimentos de seu jogo anterior. Por outro lado, a tomada de decisões em um único jogo certamente é sequencial.

O repositório de código associado a este livro (aima.cs.berkeley.edu) inclui implementações de vários ambientes, juntamente com um simulador de ambiente de uso geral que coloca um ou mais agentes em um ambiente simulado, observa seu comportamento com o passar do tempo e os avalia de acordo com determinada medida de desempenho. Com frequência, tais experimentos são executados não para um único ambiente, mas para muitos ambientes extraídos de uma **classe de ambientes**. Por

exemplo, avaliar um motorista de táxi em tráfego simulado requer a execução de muitas simulações com diferentes condições de tráfego, iluminação e condições meteorológicas. Se projetássemos o agente para um único cenário, poderíamos tirar proveito de propriedades específicas do caso particular, mas não poderíamos criar um bom projeto para dirigir de maneira geral. Por essa razão, o repositório de código também inclui um **gerador de ambientes** para cada classe de ambientes que seleciona ambientes específicos (com certas variações aleatórias) nos quais seria possível executar o agente. Por exemplo, o gerador de ambientes de aspirador de pó inicializa o padrão de sujeira e a posição do agente de forma aleatória. Então, estamos interessados no desempenho médio do agente sobre a classe de ambientes. Um agente racional para dada classe de ambientes maximiza seu desempenho médio. Os Exercícios 2.8 a 2.13 conduzem o leitor pelo processo de desenvolver uma classe de ambientes e de avaliar diversos agentes dentro dessa classe.

2.4 A ESTRUTURA DE AGENTES

Até agora fizemos referência aos agentes descrevendo o *comportamento* — a ação executada após qualquer sequência de percepções específica. Agora, teremos de seguir em frente e descrever o funcionamento interno desses agentes. O trabalho da IA é projetar o **programa do agente** que implementa a função do agente — que mapeia percepções em ações. Supomos que esse programa será executado em algum tipo de dispositivo de computação com sensores e atuadores físicos — chamamos esse conjunto de **arquitetura**:

$$\textit{agente} = \textit{arquitetura} + \textit{programa}.$$

É óbvio que o programa que escolhermos tem de ser apropriado para a arquitetura. Se o programa recomendar ações como *Caminhar*, é melhor que a arquitetura tenha pernas. A arquitetura pode ser apenas um PC comum ou talvez um carro robótico com diversos computadores, câmeras e outros sensores a bordo. Em geral, a arquitetura torna as percepções dos sensores disponíveis para o programa, executa o programa e fornece as escolhas de ação do programa para os atuadores à medida que elas são geradas. A maior parte deste livro trata do projeto de programas de agentes, embora os Capítulos 24 e 25 lidem diretamente com os sensores e atuadores.

2.4.1 Programas de agentes

Os programas de agentes que projetaremos neste livro têm todos a mesma estrutura básica: eles recebem a percepção atual como entrada dos sensores e devolvem uma ação para os atuadores.⁴ Note a diferença entre o programa do agente, que toma a percepção atual como entrada, e a função do agente, que recebe o histórico de percepções completo. O programa do agente recebe apenas a percepção atual como entrada, uma vez que nada mais está disponível do ambiente; se as ações do agente dependem da sequência de percepções inteira, o agente terá de memorizar as percepções.

Descreveremos os programas de agentes por meio da linguagem de pseudocódigo simples definida no Apêndice B (o repositório de código on-line contém implementações em linguagens de

programação reais). Por exemplo, a Figura 2.7 mostra um programa de agente bastante trivial que acompanha a sequência de percepções e depois a utiliza para realizar a indexação em uma tabela de ações, a fim de decidir o que fazer. A tabela — cujo exemplo foi dado para o mundo do aspirador de pó na Figura 2.3 — representa explicitamente a função do agente que o programa do agente incorpora. Para construir um agente racional desse modo, devemos construir uma tabela que contenha a ação apropriada para todas as sequências de percepções possíveis.

<p>função AGENTE-DIRIGIDO-POR-TABELA(<i>percepção</i>) retorna uma ação</p> <p>variáveis estáticas: <i>percepções</i>, uma sequência, inicialmente vazia</p> <p style="padding-left: 40px;"><i>tabela</i>, uma tabela de ações, indexada por sequências de percepções, inicialmente completamente especificada</p> <p><i>anexar percepção</i> ao fim de <i>percepções</i></p> <p><i>ação</i> \leftarrow ACESSAR(<i>percepções</i>, <i>tabela</i>)</p> <p>retornar <i>ação</i></p>
--

Figura 2.7 O programa AGENTE-DIRIGIDO-POR-TABELA é invocado para cada nova percepção e retorna uma ação de cada vez. Ele mantém a sequência de percepções completas na memória.

É instrutivo considerar por que a abordagem orientada a tabelas para construção de agentes está condenada ao fracasso. Seja \mathcal{P} o conjunto de percepções possíveis e seja T o tempo de duração do agente (o número total de percepções que ele receberá). A tabela de pesquisa conterá $\sum_{t=1}^T |\mathcal{P}|^t$ entradas. Considere o táxi automatizado: a entrada visual de uma única câmera chega à velocidade de aproximadamente 27 megabytes por segundo (30 quadros por segundo, 640×480 pixels com 24 bits de informações de cores). Isso nos dá uma tabela de pesquisa com mais de $10^{250.000.000.000}$ entradas para uma hora de direção. Até mesmo a tabela de pesquisa para o xadrez — um minúsculo e bem-comportado fragmento do mundo real — teria pelo menos 10^{150} entradas. O tamanho assustador dessas tabelas (o número de átomos no universo observável é menor que 10^{80}) significa que (a) nenhum agente físico nesse universo terá espaço para armazenar a tabela, (b) o projetista não teria tempo para criar a tabela, (c) nenhum agente poderia sequer apreender todas as entradas de tabelas corretas a partir de sua experiência e (d) mesmo que o ambiente seja simples o bastante para gerar uma tabela de tamanho viável, o projetista ainda não terá nenhuma orientação sobre como inserir as entradas da tabela.

Apesar de tudo isso, o AGENTE-DIRIGIDO-POR-TABELA *faz* o que queremos: implementa a função de agente desejada. O desafio fundamental da IA é descobrir como escrever programas que, na medida do possível, produzam um comportamento racional a partir de um pequeno programa em vez de uma grande tabela. Temos muitos exemplos mostrando que isso pode ser feito com sucesso em outras áreas: por exemplo, as enormes tabelas de raízes quadradas usadas por engenheiros e por estudantes antes da década de 1970 foram substituídas por um programa de cinco linhas que corresponde ao método de Newton e é executado em calculadoras eletrônicas. A pergunta é: a IA pode fazer pelo comportamento inteligente em geral o que Newton fez para as raízes quadradas? Acreditamos que a resposta seja sim.

No restante desta seção, descreveremos quatro tipos básicos de programas de agentes que

incorporam os princípios subjacentes a quase todos os sistemas inteligentes:

- Agentes reativos simples.
- Agentes reativos baseados em modelo.
- Agentes baseados em objetivos.
- Agentes baseados na utilidade.

Cada tipo de programa de agente combina componentes específicos de maneiras específicas para gerar ações. A Seção 2.4.6 explica, em termos gerais, como converter todos esses agentes em *agentes de aprendizagem* que podem melhorar o desempenho de seus componentes de modo a gerar melhores ações. Finalmente, a Seção 2.4.7 descreve uma variedade de maneiras de como os próprios componentes podem ser representados dentro do agente. Essa variedade proporciona um princípio organizador fundamental para o campo e para o próprio livro.

2.4.2 Agentes reativos simples

O tipo mais simples de agente é o **agente reativo simples**. Esses agentes selecionam ações com base na percepção *atual*, ignorando o restante do histórico de percepções. Por exemplo, o agente aspirador de pó cuja função do agente é tabulada na Figura 2.3 é um agente reativo simples porque sua decisão se baseia apenas na posição atual e no fato de essa posição conter ou não sujeira. Um programa para esse agente é mostrado na Figura 2.8.

Note que o programa do agente aspirador de pó na realidade é muito pequeno em comparação com a tabela correspondente. A redução mais óbvia vem de se ignorar o histórico de percepções, o que reduz o número de possibilidades de 4^T para apenas 4. Uma pequena redução adicional vem do fato de que, quando o quadrado atual está sujo, a ação não depende da posição em que o agente esteja.

Comportamentos reativos simples ocorrem mesmo em ambientes mais complexos. Imagine-se como o motorista do táxi automatizado. Se o carro da frente frear e suas luzes de freio se acenderem, você deve notar esse fato e começar a frear. Em outras palavras, algum processamento é realizado de acordo com a entrada visual para estabelecer a condição que chamamos de “O carro da frente está freando”. Então, isso ativa alguma conexão estabelecida no programa do agente para a ação “começar a frear”. Chamaremos tal conexão de **regra condição-ação**,⁵ escrita como:

se carro-da-frente-está-freando então começar-a-frear.

Os seres humanos também têm muitas dessas conexões, algumas das quais são respostas aprendidas (como dirigir) e outras são reflexos inatos (como piscar quando algo se aproxima de seu olho). No decorrer do livro, veremos várias maneiras diferentes de aprender e implementar tais conexões.

O programa da Figura 2.8 é específico para um determinado ambiente do aspirador de pó. Uma abordagem mais geral e flexível consiste em primeiro construir um interpretador de uso geral para regras condição-ação e depois criar conjuntos de regras para ambientes de tarefas específicos. A Figura 2.9 fornece a estrutura desse programa geral em forma esquemática, mostrando como as regras

condição-ação permitem ao agente fazer a conexão entre percepção e ação (não se preocupe com o fato de esse assunto parecer trivial; ele ficará mais interessante em breve). Utilizamos retângulos para denotar o estado interno atual do processo de decisão do agente e elipses para representar as informações suplementares usadas no processo. O programa do agente, que também é muito simples, é mostrado na Figura 2.10. A função INTERPRETAR-ENTRADA gera uma descrição abstrata do estado atual a partir da percepção, e a função REGRA-CORRESPONDENTE retorna a primeira regra no conjunto de regras que corresponde à descrição de estado dada. Observe que a descrição em termos de “regras” e “correspondência” é puramente conceitual; as implementações reais podem ser tão simples quanto uma coleção de portas lógicas que implementam um circuito booleano.

Os agentes reativos simples têm a admirável propriedade de serem simples, mas se caracterizam por ter inteligência limitada. O agente da Figura 2.10 funcionará *somente se a decisão correta puder ser tomada com base apenas na percepção atual, ou seja, apenas se o ambiente for completamente observável*. Até mesmo uma pequena impossibilidade de observação pode causar sérias dificuldades. Por exemplo, a regra de frenagem apresentada anteriormente pressupõe que a condição *carro-da-frente-está-freando* pode ser determinada a partir da percepção atual — um único quadro de vídeo. Funciona se o carro tiver na frente uma luz de freio central. Infelizmente, modelos mais antigos têm configurações diferentes de lanternas, luzes de freio e luzes de setas, e nem sempre é possível saber por uma única imagem se o carro está freando. Um agente reativo simples que dirigisse atrás de um carro desse tipo frearia contínua e desnecessariamente ou, pior ainda, nunca frearia.

<p>função AGENTE-ASPIRADOR-DE-PÓ-REATIVO (<i>[posição, situação]</i>) retorna uma ação*</p> <p>se <i>situação = Sujo</i> então retorna <i>Aspirar</i></p> <p>senão se <i>posição = A</i> então retorna <i>Direita</i></p> <p>senão se <i>posição = B</i> então retorna <i>Esquerda</i></p>
--

Figura 2.8 Programa do agente para um agente reativo simples no ambiente de aspirador de pó de dois estados. Esse programa implementa a função do agente tabulada na Figura 2.3.

* *Nota do revisor técnico:* Como aqui se trata de “pseudocódigo”, é possível traduzir os comandos para facilitar a leitura. Apenas não se traduz quando se trata de uma linguagem de programação real.

Podemos ver um problema semelhante surgindo no mundo de aspirador de pó. Suponha que um agente aspirador de pó reativo simples seja destituído de seu sensor de posição e tenha apenas um sensor de sujeira. Tal agente tem apenas duas percepções possíveis: [*Sujo*] e [*Limpo*]. Ele pode *Aspirar* em resposta a [*Sujo*]; o que deve fazer em resposta a [*Limpo*]? Mover-se para a *Esquerda* falhará (sempre) se ele começar no quadrado *A*, e mover-se para a *Direita* falhará (sempre) se ele começar no quadrado *B*. Com frequência, laços de repetição infinitos são inevitáveis no caso de agentes reativos simples operando em ambientes parcialmente observáveis.

É possível escapar de laços de repetição infinitos se o agente puder tornar suas ações **aleatórias**. Por exemplo, se o agente aspirador de pó perceber [*Limpo*], ele pode jogar uma moeda para escolher entre *Esquerda* e *Direita*. É fácil mostrar que o agente alcançará o outro quadrado usando duas etapas em média. Em seguida, se esse quadrado estiver sujo, ele limpará a sujeira e a tarefa de limpeza será concluída. Consequentemente, um agente reativo simples aleatório poderia superar um

agente reativo simples determinístico.

Mencionamos na Seção 2.3 que um comportamento aleatório do tipo correto pode ser racional em alguns ambientes multiagentes. Em ambientes de um único agente, em geral a aleatoriedade *não* é racional. Ela é um artifício útil que ajuda um agente reativo simples em algumas situações, mas, na maioria dos casos, podemos fazer muito melhor com agentes determinísticos mais sofisticados.

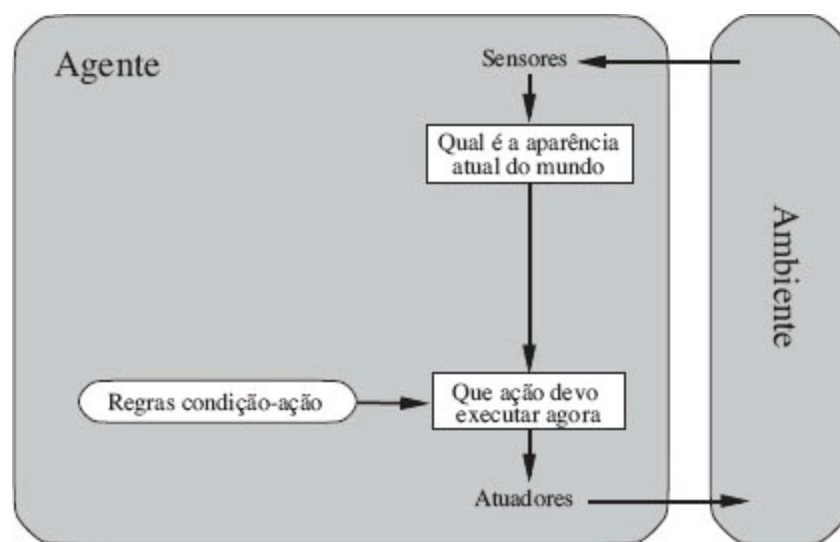


Figura 2.9 Diagrama esquemático de um agente reativo simples.

função AGENTE-REATIVO-SIMPLES (*percepção*) **retorna** uma ação

variáveis estáticas: *regras*, um conjunto de regras condição-ação

estado ← INTERPRETAR-ENTRADA (*percepção*)

regra ← REGRA-CORRESPONDENTE (*estado*, *regras*)

ação ← AÇÃO-DA-REGRA [*regra*]

retornar *ação*

Figura 2.10 Um agente reativo simples. Ele age de acordo com uma regra cuja condição corresponde ao estado atual definido pela percepção.

2.4.3 Agentes reativos baseados em modelos

O modo mais efetivo de lidar com a possibilidade de observação parcial é o agente *monitorar a parte do mundo que ele não pode ver agora*. Isto é, o agente deve manter algum tipo de **estado interno** que dependa do histórico de percepções e assim reflita pelo menos alguns dos aspectos não observados do estado atual. Para o problema do freio, o estado interno não é muito extenso — apenas o quadro anterior da câmera, que permite ao agente detectar quando duas luzes vermelhas na borda do veículo acendem ou apagam ao mesmo tempo. No caso de outras tarefas de direção, como trocar de pista, o agente precisa monitorar onde os outros carros estão, se não puder vê-los todos de uma vez. E, para que qualquer direção seja possível, o agente precisa saber onde as chaves estão.

A atualização dessas informações internas de estado à medida que o tempo passa exige que dois

tipos de conhecimento sejam codificados no programa do agente. Primeiro, precisamos de algumas informações sobre o modo como o mundo evolui independentemente do agente — por exemplo, que um carro que estiver ultrapassando em geral estará mais próximo do que estava um momento antes. Em segundo lugar, precisamos de algumas informações sobre como as ações do próprio agente afetam o mundo — de que, por exemplo, quando o agente girar o volante à direita, o carro irá virar para a direita ou de que, depois de dirigir por cinco minutos na direção norte da autoestrada, em geral ficamos cinco quilômetros ao norte de onde nos encontrávamos cinco minutos antes. Esse conhecimento de “como o mundo funciona” — seja ele implementado em circuitos booleanos simples ou em teorias científicas completas — é chamado de **modelo** do mundo. Um agente que usa tal modelo denomina-se **agente baseado em modelo**.

A Figura 2.11 fornece a estrutura do agente reativo baseado em modelo com seu estado interno, mostrando como a percepção atual é combinada com o estado interno antigo para gerar a descrição atualizada do estado atual, baseado no modelo do agente de como o mundo funciona.

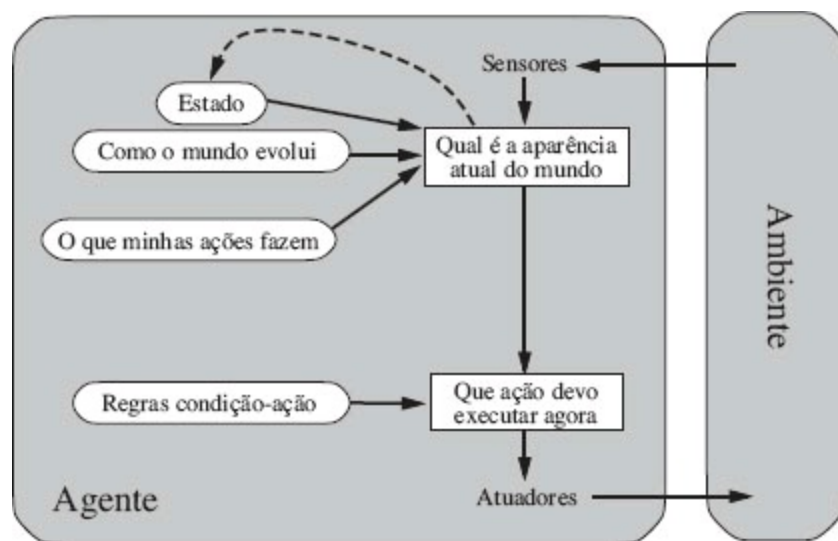


Figura 2.11 Agente reativo baseado em modelo.

O programa de agente é mostrado na Figura 2.12. A parte interessante é a função ATUALIZAR-ESTADO, responsável pela criação da descrição do novo estado interno. Os detalhes de como modelos e estados são representados variam amplamente dependendo do tipo de ambiente e da tecnologia em particular usada no projeto do agente. Nos Capítulos 4, 12, 11, 15, 17 e 25 aparecem exemplos detalhados de modelos e atualização de algoritmos.

função AGENTE-REATIVO-BASEADO-EM-MODELOS (*percepção*) **retorna** uma ação
persistente: *estado*, a concepção do agente do estado atual do mundo
modelo, uma descrição de como o próximo estado depende do estado atual e da ação
regras, um conjunto de regras condição-ação
ação, a ação mais recente, inicialmente nenhuma

estado ← ATUALIZAR-ESTADO (*estado*, *ação*, *percepção*, *modelo*)
regra ← REGRA-CORRESPONDENTE (*estado*, *regras*)

Figura 2.12 Um agente reativo baseado em modelo. Ele mantém o estado atual do mundo usando um modelo interno. Em seguida, ele escolhe uma ação da mesma maneira que o agente reativo simples.

Independentemente do tipo de representação utilizada, raramente é possível para o agente determinar *exatamente* o estado atual de um ambiente parcialmente observável. Em vez disso, a caixa rotulada “como o mundo se parece agora” (Figura 2.11) representa o “melhor palpite” do agente (ou, às vezes, os melhores palpites). Por exemplo, um táxi automatizado pode não ser capaz de enxergar através de um grande caminhão que parou na sua frente e talvez tenha apenas um palpite do que causou o bloqueio. Assim, a incerteza sobre o estado atual pode ser inevitável, mas o agente ainda terá que tomar uma decisão.

Um ponto talvez menos óbvio sobre o “estado” interno mantido por um agente baseado em modelo é que ele não tem que descrever “como o mundo se parece agora” em sentido literal. Por exemplo, o táxi pode estar dirigindo de volta para casa e pode ser que exista uma regra para colocar gasolina no caminho de casa se o tanque estiver pelo menos pela metade. Apesar de que “dirigir de volta para casa” *parece* um aspecto do estado do mundo, o fato do *destino* do táxi é na verdade um aspecto do estado interno do agente. Se você achar isso intrigante, considere que o táxi possa estar exatamente no mesmo lugar ao mesmo tempo, mas com a intenção de chegar a um destino diferente.

2.4.4 Agentes baseados em objetivos

Conhecer algo sobre o estado atual do ambiente nem sempre é suficiente para decidir o que fazer. Por exemplo, em um entroncamento de estradas, o táxi pode virar à esquerda, virar à direita ou seguir em frente. A decisão correta depende de onde o táxi está tentando chegar. Em outras palavras, da mesma forma que o agente precisa de uma descrição do estado atual, ele também precisa de alguma espécie de informação sobre **objetivos** que descreva situações desejáveis — por exemplo, estar no destino do passageiro. O programa de agente pode combinar isso com o modelo (as mesmas informações que foram usadas no agente reativo baseado em modelo), a fim de escolher ações que alcancem o objetivo. A Figura 2.13 mostra a estrutura do agente baseado em objetivos.

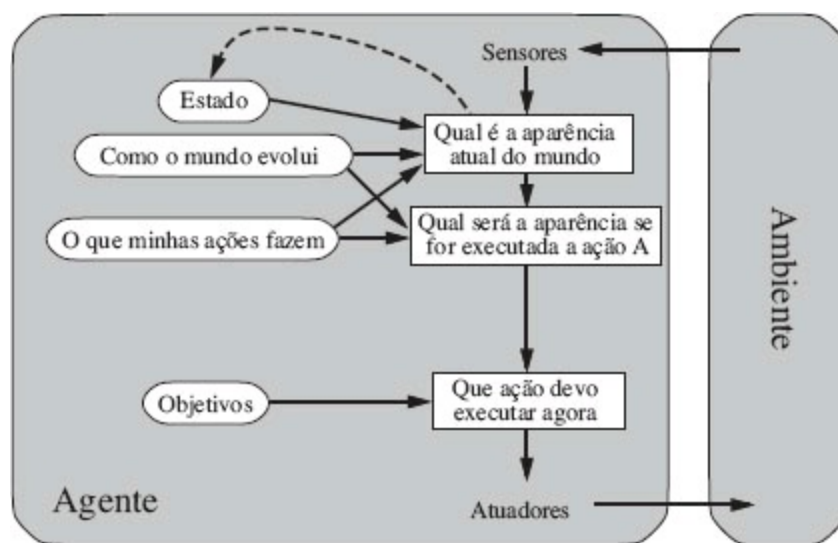


Figura 2.13 Um agente baseado em modelos e orientado pelos objetivos. Ele monitora o estado do mundo, bem como um conjunto de objetivos que está tentando atingir e escolhe uma ação que (no final) levará à realização de seus objetivos.

Às vezes, a seleção da ação baseada em objetivos é direta — por exemplo, quando a satisfação do objetivo resulta de imediato de uma única ação. Outras vezes ela será mais complicada — por exemplo, quando o agente tiver de considerar longas sequências de ações até encontrar um meio de atingir o objetivo. **Busca** (Capítulos 3 a 5) e **planejamento** (Capítulos 10 e 11) são as subáreas da IA dedicados a encontrar sequências de ações que alcançam os objetivos do agente.

Note que a tomada de decisões desse tipo é fundamentalmente distinta das regras condição-ação descritas anteriormente, pelo fato de envolver consideração do futuro, tanto de “O que acontecerá se eu fizer isso e aquilo?” e “Isso me fará feliz?”. Nos projetos de agentes reativos, essas informações não são representadas de forma explícita porque as regras internas fazem o mapeamento direto de percepções para ações. O agente reativo freia quando vê luzes de freio. Em princípio, um agente baseado em objetivos poderia raciocinar que, se o carro da frente tem suas luzes de freio acesas, ele diminuirá a velocidade. Dada a forma como o mundo costuma evoluir, a única ação que alcançará o objetivo de não atingir outros carros é frear.

Embora o agente baseado em objetivos pareça menos eficiente, ele é mais flexível porque o conhecimento que apoia suas decisões é representado de maneira explícita e pode ser modificado. Se começar a chover, o agente poderá atualizar seu conhecimento de como seus freios vão operar de modo eficiente; isso fará todos os comportamentos relevantes serem alterados automaticamente para atender às novas condições. Por outro lado, para o agente reativo, teríamos de reescrever muitas regras condição-ação. O comportamento do agente baseado em objetivos pode ser alterado com facilidade para ir a um destino diferente, simplesmente especificando o destino como objetivo. As regras do agente reativo sobre quando fazer curvas e quando seguir em frente só funcionarão para um único destino; todas elas terão de ser substituídas se for preciso ir para algum outro lugar.

2.4.5 Agentes baseados na utilidade

Sozinhos, os objetivos não são realmente suficientes para gerar um comportamento de alta

qualidade na maioria dos ambientes. Por exemplo, existem muitas sequências de ações que levarão o táxi até seu destino (alcançando assim o objetivo), mas algumas são mais rápidas, mais seguras, mais confiáveis ou mais econômicas que outras. Os objetivos simplesmente permitem uma distinção binária crua entre “estados felizes” e “infelizes”, enquanto uma medida de desempenho mais geral deve permitir uma comparação entre diferentes estados do mundo, de acordo com o grau exato de felicidade que proporcionariam ao agente. Tendo em vista que “feliz” não soa muito científico, em vez disso, economistas e cientistas da computação usam o termo **utilidade**.⁶

Nós já vimos que uma medida de desempenho atribui uma pontuação para qualquer sequência de estados do ambiente, e assim ela pode distinguir facilmente entre formas mais e menos desejáveis de chegar ao destino do táxi. A função **utilidade do agente** é essencialmente uma internalização da medida de desempenho. Se a função utilidade interna e a medida externa de desempenho estiverem em acordo, um agente que escolhe ações que maximizem a sua utilidade será racional de acordo com a medida de desempenho externa.

Vamos enfatizar novamente que essa não é a *única* maneira de ser racional — já vimos um programa de agente racional para o mundo do aspirador de pó (Figura 2.8) que não tem ideia de qual seja sua função utilidade, mas, como os agentes baseados em objetivos, um agente baseado em utilidade tem muitas vantagens em termos de flexibilidade e de aprendizagem. Além disso, em dois tipos de casos, os objetivos são inadequados, mas um agente baseado em utilidade ainda pode tomar decisões racionais. Primeiro, quando houver objetivos conflitantes, apenas alguns dos quais podem ser alcançados (por exemplo, velocidade e segurança), a função utilidade especifica a escolha apropriada. Segundo, quando há vários objetivos que o agente pode visar e nenhum dos quais pode ser alcançado com certeza, a utilidade proporciona uma maneira em que a probabilidade de sucesso pode ser pesada em relação à importância dos objetivos. Observabilidade parcial e estocasticidade são onipresentes no mundo real e assim, portanto, a tomada de decisão sob incerteza. Tecnicamente falando, um agente racional baseado em utilidade escolhe a ação que maximiza a **utilidade esperada** dos resultados da ação, isto é, a utilidade que o agente espera obter, em média, dadas as probabilidades e as utilidades de cada resultado (o Apêndice A define expectativa mais precisamente). No Capítulo 16, mostraremos que qualquer agente racional deve se comportar *como se* possuísse uma função utilidade cujo valor esperado ele tenta maximizar. Um agente que possui uma função utilidade *explícita* pode tomar decisões racionais por meio de um algoritmo de uso geral que não depende da função utilidade específica que está sendo maximizada. Desse modo, a definição “global” de racionalidade — designando-se como racionais as funções de agentes que têm o melhor desempenho — é transformada em uma restrição “local” sobre projetos de agentes racionais que podem ser expressos em um programa simples.

A estrutura de agente baseado na utilidade aparece na Figura 2.14. Os programas de agentes baseados na utilidade são examinados na Parte IV, em que projetamos agentes de tomada de decisões que devem lidar com a incerteza inerente aos ambientes estocásticos ou parcialmente observáveis.

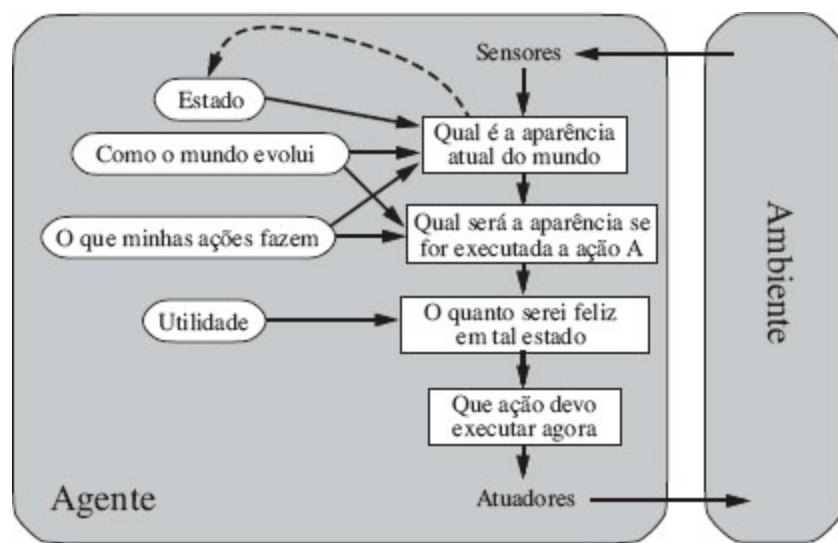


Figura 2.14 Um agente baseado em modelo e orientado para a utilidade. Ele usa um modelo do mundo juntamente com uma função utilidade que mede suas preferências entre estados do mundo. Em seguida, ele escolhe a ação que leva à melhor utilidade esperada, na qual a utilidade esperada é calculada pela média entre todos os estados resultantes possíveis, ponderados pela probabilidade do resultado.

Neste ponto, o leitor pode estar se perguntando: “É assim tão simples? Construimos agentes que maximizam a utilidade esperada e pronto?” É verdade que tais agentes são inteligentes, mas não é simples. Um agente baseado em utilidade tem de modelar e monitorar seu ambiente, tarefas que envolvem grande quantidade de pesquisas sobre percepção, representação, raciocínio e aprendizagem. Os resultados dessa pesquisa preencheram muitos capítulos deste livro. A escolha da maximização de utilidade do curso da ação também é uma tarefa difícil, exigindo algoritmos engenhosos que preencheram outros vários capítulos. Mesmo com esses algoritmos, geralmente o raciocínio perfeito é inatingível na prática por causa da complexidade computacional, mencionada no Capítulo 1.

2.4.6 Agentes com aprendizagem

Descrevemos programas de agentes com vários métodos para selecionar ações. Porém, até agora não explicamos como os programas de agentes *passam a existir*. Em seu famoso ensaio inicial, Turing (1950) considera a ideia de realmente programar suas máquinas inteligentes à mão. Ele estima quanto trabalho isso poderia exigir e conclui que “algum método mais eficiente parece desejável”. O método que ele propõe é construir máquinas com aprendizagem e depois ensiná-las. Em muitas áreas de IA, esse é agora o método preferencial para se criar sistemas do estado da arte. O aprendizado tem outra vantagem, como observamos antes: ele permite ao agente operar em ambientes inicialmente desconhecidos e se tornar mais competente do que seu conhecimento inicial sozinho poderia permitir. Nesta seção, introduzimos rapidamente as principais ideias de agentes com aprendizagem. Do começo ao fim do livro, faremos comentários sobre oportunidades e métodos de aprendizado em tipos específicos de agentes. A Parte V estuda com muito maior profundidade os diversos algoritmos de aprendizado propriamente ditos.

Um agente de aprendizado pode ser dividido em quatro componentes conceituais, como mostra a Figura 2.15. A distinção mais importante se dá entre o **elemento de aprendizado**, responsável pela execução de aperfeiçoamentos, e o **elemento de desempenho**, responsável pela seleção de ações externas. O elemento de desempenho é o que antes consideramos como sendo o agente completo: ele recebe percepções e decide sobre ações. O elemento de aprendizado utiliza realimentação do **crítico** sobre como o agente está funcionando e determina de que maneira o elemento de desempenho deve ser modificado para funcionar melhor no futuro.

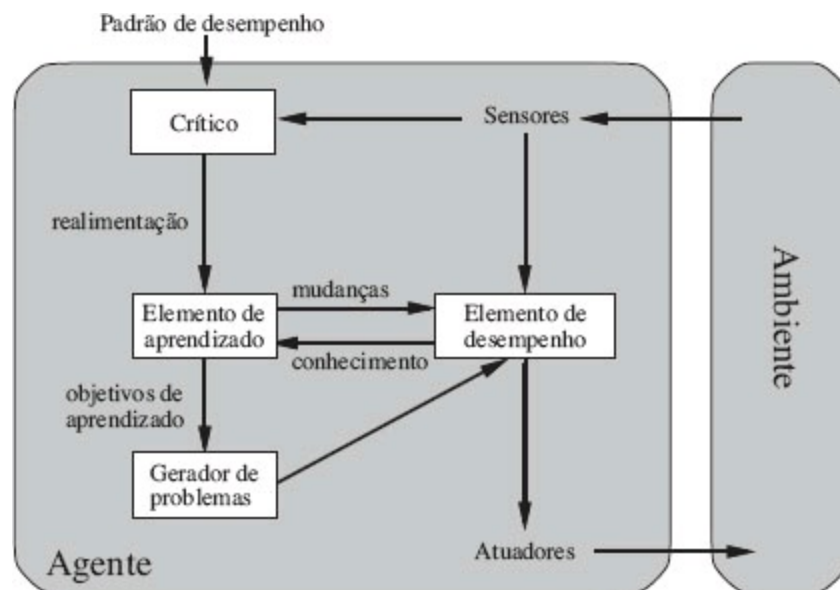


Figura 2.15 Um modelo geral de agentes com aprendizagem.

O projeto do elemento de aprendizado depende muito do projeto do elemento de desempenho. Quando se tenta projetar um agente que aprende certa capacidade, a primeira pergunta não é “Como farei com que ele aprenda isso?”, mas “Que tipo de elemento de desempenho meu agente precisará ter para fazer isso depois de ter aprendido como fazê-lo?”. Dado um projeto de agente, podem ser construídos mecanismos de aprendizado para otimizar cada parte do agente.

O crítico informa ao elemento de aprendizado como o agente está se comportando em relação a um padrão fixo de desempenho. O crítico é necessário porque as próprias percepções não oferecem nenhuma indicação do sucesso do agente. Por exemplo, um programa de xadrez poderia receber uma percepção indicando que aplicou um xeque-mate em seu oponente, mas o programa precisa de um padrão de desempenho para saber que isso é algo bom; a percepção em si não diz nada sobre isso. É importante que o padrão de desempenho seja fixo. Conceitualmente, deveríamos pensar nele como algo que está totalmente fora do agente porque o agente não deve modificá-lo para ajustá-lo a seu próprio comportamento.

O último componente do agente com aprendizagem é o **gerador de problemas**. Ele é responsável por sugerir ações que levarão a experiências novas e informativas. A questão é que, se o elemento de desempenho tivesse a possibilidade, ele continuaria a realizar as melhores ações, dadas as informações que possui. Porém, se o agente estivesse disposto a realizar uma pequena exploração e executar algumas ações que talvez não fossem ótimas a curto prazo, ele poderia descobrir ações muito melhores a longo prazo. A tarefa do gerador de problemas é sugerir essas ações exploratórias. É isso que os cientistas fazem quando realizam experiências. Galileu não pensava que soltar pedras do alto de uma torre em Pisa teria algum valor em si. Ele não estava tentando quebrar as pedras nem

modificar o cérebro dos pedestres desafortunados. Seu objetivo era modificar seu próprio cérebro, identificando uma teoria melhor sobre o movimento dos objetos.

Para tornar o projeto global mais concreto, vamos voltar ao exemplo do táxi automatizado. O elemento de desempenho consiste em qualquer coleção de conhecimento e procedimentos que o táxi tem para selecionar suas ações de dirigir. O táxi vai para a estrada e dirige, usando esse elemento de desempenho. O crítico observa o mundo e repassa informações ao elemento de aprendizado. Por exemplo, depois que o táxi faz uma rápida mudança para a esquerda cruzando três faixas de tráfego, o crítico observa a linguagem chocante utilizada por outros motoristas. A partir dessa experiência, o elemento de aprendizado é capaz de formular uma regra afirmando que essa foi uma ação ruim, e o elemento de desempenho é modificado pela instalação da nova regra. O gerador de problemas pode identificar certas áreas de comportamento que necessitam de melhorias e sugerir experimentos, como testar os freios em diferentes superfícies de rodagem sob condições distintas.

O elemento de aprendizado pode fazer mudanças em qualquer dos componentes de “conhecimento” mostrados nos diagramas de agentes (Figuras 2.9, 2.11, 2.13 e 2.14). Os casos mais simples envolvem o aprendizado direto a partir da sequência de percepções. A observação de pares de estados sucessivos do ambiente pode permitir ao agente aprender “Como o mundo evolui”, e a observação dos resultados de suas ações pode permitir que ele aprenda “O que minhas ações fazem”. Por exemplo, se o táxi exercer certa pressão nos freios ao dirigir em uma estrada molhada, ele logo descobrirá qual é a desaceleração realmente alcançada. É claro que essas duas tarefas de aprendizado serão mais difíceis se o ambiente for apenas parcialmente observável.

As formas de aprendizado no parágrafo anterior não precisam ter acesso ao padrão de desempenho externo — de certo modo, o padrão universal é fazer previsões que concordem com a experiência. A situação é um pouco mais complexa no caso de um agente baseado em utilidade que deseja aprender informações de utilidade. Por exemplo, suponha que o agente de direção de táxi não receba dos passageiros nenhuma gorjeta porque o táxi sacolejou muito durante a viagem. O padrão de desempenho externo deve informar ao agente que a falta de gorjetas é uma contribuição negativa para seu desempenho global; desse modo, o agente talvez fosse capaz de aprender que manobras violentas não contribuem para sua própria utilidade. De certo modo, o padrão de desempenho distingue parte da percepção de entrada como uma **recompensa** (ou **penalidade**) que fornece realimentação direta sobre a qualidade do comportamento do agente. Os padrões de desempenho internos como dor e fome em animais podem ser entendidos desse modo. Essa questão é discutida com maior profundidade no Capítulo 21.

Em resumo, os agentes têm uma variedade de componentes, e esses componentes podem ser representados de muitas formas dentro do programa do agente; dessa forma, parece haver grande variedade de métodos de aprendizado. No entanto, existe um único tema unificador. O aprendizado em agentes inteligentes pode ser resumido como um processo de modificação de cada componente do agente, a fim de levar os componentes a um acordo mais íntimo com as informações de realimentação disponíveis, melhorando assim o desempenho global do agente.

2.4.7 Como funcionam os componentes do programa de agente

Descrevemos os programas de agente (em termos de muito alto nível) consistindo de vários componentes cuja função é responder a perguntas como: “Como o mundo está agora?”, “Que ações devo tomar?”, “O que minhas ações realizam?”. A próxima pergunta para um estudante de IA é: “Como funcionam esses componentes?” Levará cerca de mil páginas para começar a responder a essas perguntas apropriadamente, mas queremos chamar a atenção do leitor para algumas distinções básicas entre as várias maneiras como os componentes podem representar o ambiente que o agente habita.

Grosso modo, podemos colocar as representações ao longo de um eixo de complexidade crescente e poder de expressividade — **atômico**, **fatorado** e **estruturado**. Para ilustrar essas ideias, consideremos um componente do agente em particular, como aquele que lida com “O que minhas ações realizam”. Esse componente descreve as alterações que podem ocorrer no ambiente como resultado de executar uma ação, e a Figura 2.16 fornece descrições esquemáticas de como as transições devem ser representadas.

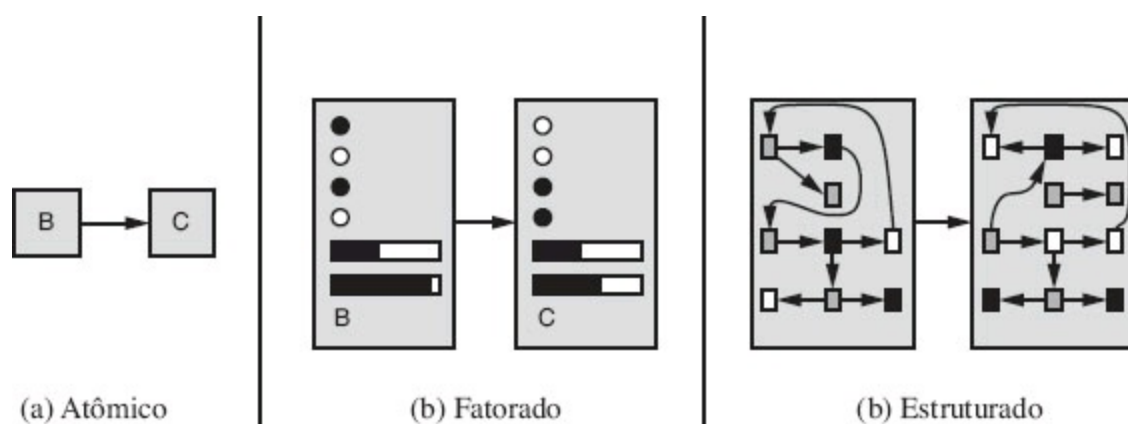


Figura 2.16 Três maneiras de representar os estados e as transições entre eles. (a) Representação atômica: um estado (como B ou C) é uma caixa preta, sem estrutura interna. (b) Representação fatorada: um estado consiste em um vetor de valores de atributos; os valores podem ser booleanos, valores reais ou um conjunto fixo de símbolos. (c) Representação estruturada: um estado inclui objetos; cada um deles pode ter atributos próprios, bem como relacionamentos com outros objetos.

Em uma **representação atômica**, cada estado do mundo é indivisível — não tem estrutura interna. Considere o problema de encontrar um caminho de uma extremidade à outra do país através de alguma sequência de cidades (tratamos esse problema na Figura 3.2). Para resolver esse problema, pode ser suficiente reduzir o estado do mundo apenas para o nome da cidade em que estamos — um único átomo de conhecimento; uma “caixa preta”, cuja única propriedade discernível é a de ser idêntico ou diferente de outra caixa preta. Os algoritmos que formam a base de **busca** e de **jogos** (Capítulos 3-5), **Modelos Ocultos de Markov** (Capítulo 15) e os **processos de decisão de Markov** (Capítulo 17) trabalham com representações atômicas ou, pelo menos, tratam as representações *como se* fossem atômicas.

Agora considere uma descrição de maior fidelidade para o mesmo problema, em que precisamos nos preocupar com mais do que apenas a localização atômica em uma cidade ou outra, talvez sendo necessário prestar atenção em quanta gasolina há no tanque, nas coordenadas atuais do GPS, se a luz de advertência do óleo está funcionando, quanto temos de troco para o pedágio, que estação está tocando na rádio, e assim por diante. Uma **representação fatorada** divide cada estado em um

conjunto fixo de **variáveis** ou **atributos**, cada um dos quais pode ter um **valor**. Enquanto dois estados atômicos diferentes não têm nada em comum — são apenas caixas pretas diferentes —, dois estados fatorados diferentes podem compartilhar alguns atributos (como estar em alguma localização GPS específica) e não compartilhar outros (como ter muita ou nenhuma gasolina), o que torna muito mais fácil planejar como ir de um estado para o outro. Com representações fatoradas, também se pode representar a *incerteza* — por exemplo, a ignorância sobre a quantidade de combustível no tanque pode ser representada deixando o atributo em branco. Muitas áreas importantes da IA são baseadas em representações fatoradas, incluindo algoritmos de **satisfação de restrição** (Capítulo 6), **lógica proposicional** (Capítulo 7), **planejamento** (Capítulos 10 e 11), **redes bayesianas** (Capítulos 13-16) e algoritmos de **aprendizado de máquina** nos Capítulos 18, 20 e 21. Para muitos propósitos, é necessário entender que o mundo tem *coisas* que estão *relacionadas* umas com as outras, não apenas variáveis com valores. Por exemplo, podemos notar que um grande caminhão à nossa frente está se movendo em sentido contrário na entrada de carros de uma fazenda de gado leiteiro, mas uma vaca soltou-se e está bloqueando o caminho do caminhão. É pouco provável que uma representação fatorada esteja pré-definida com o atributo *CaminhãoMovendoParaTrásFrenteEntradaFazendaLeiteiraBloqueadoVacaPerdida* com valor *verdadeiro* ou *falso*. Em vez disso, precisaríamos de uma **representação estruturada**, em que objetos, como vacas e caminhões e seus relacionamentos diversos e variados, possam ser explicitamente descritos (ver a Figura 2.16c). Representações estruturadas são base de **bancos de dados relacionais** e **lógica de primeira ordem** (Capítulos 8, 9 e 12), **modelos de probabilidade de primeira ordem** (Capítulo 14), de **aprendizagem baseada em conhecimento** (Capítulo 19) e grande parte da **compreensão da linguagem natural** (Capítulos 22 e 23). Na verdade, quase tudo o que os seres humanos expressam em linguagem natural diz respeito aos objetos e seus relacionamentos.

Como mencionamos anteriormente, o eixo ao longo do qual estão as representações atômica, fatorada e estruturada é o eixo de **expressividade** crescente. Grosseiramente falando, uma representação mais expressiva pode capturar, pelo menos de forma mais concisa, tudo o que algo menos expressivo pode capturar e ainda um pouco mais. Muitas vezes, a linguagem mais expressiva é *muito* mais concisa; por exemplo, as regras do xadrez podem ser escritas em uma página ou duas de uma linguagem com representação estruturada, como lógica de primeira ordem, mas requer milhares de páginas, quando escritas em uma linguagem com representação fatorada, como a lógica proposicional. Por outro lado, o raciocínio e a aprendizagem se tornam mais complexos à medida que aumenta o poder expressivo da representação. Para obter os benefícios das representações expressivas, evitando as suas limitações, pode ser que para o mundo real os sistemas inteligentes necessitem operar simultaneamente em todos os pontos ao longo do eixo de expressividade.

2.5 RESUMO

Este capítulo foi uma espécie de excursão vertiginosa pela IA, que concebemos como a ciência de projeto de agentes. Aqui estão os pontos importantes a serem lembrados:

- Um **agente** é algo que percebe e age em um ambiente. A **função do agente** especifica a ação executada pelo agente em resposta a qualquer sequência de percepções.

- A **medida de desempenho** avalia o comportamento do agente em um ambiente. Um **agente racional** age para maximizar o valor esperado da medida de desempenho, dada a sequência de percepções recebida até o momento.
- Uma especificação de **ambiente de tarefa** inclui a medida de desempenho, o ambiente externo, os atuadores e os sensores. Ao se projetar um agente, o primeiro passo sempre deve ser especificar o ambiente de tarefa de maneira tão completa quanto possível.
- Os ambientes de tarefas variam ao longo de diversas dimensões significativas. Eles podem ser completa ou parcialmente observáveis, agente único ou multiagente, determinísticos ou estocásticos, episódicos ou sequenciais, estáticos ou dinâmicos, discretos ou contínuos e conhecidos ou desconhecidos.
- O **programa do agente** implementa a função do agente. Existe uma variedade de projetos básicos de programas de agentes, refletindo o tipo de informação explicitada e usada no processo de decisão. Os projetos variam em eficiência, síntese e flexibilidade. O projeto apropriado do programa do agente depende da natureza do ambiente.
- Os **agentes reativos simples** respondem diretamente a percepções, enquanto os **agentes reativos baseados em modelos** mantêm o estado interno para controlar aspectos do mundo que não estão evidentes na percepção atual. Os **agentes baseados em objetivos** agem para alcançar seus objetivos, e os **agentes baseados em utilidade** tentam maximizar sua própria “felicidade” esperada.
- Todos os agentes podem melhorar seu desempenho por meio do **aprendizado**.

NOTAS BIBLIOGRÁFICAS E HISTÓRICAS

O papel central da ação na inteligência — a noção de raciocínio prático — remonta pelo menos à época da *Ética a Nicômaco* de Aristóteles. O raciocínio prático também foi o assunto do importante artigo de McCarthy (1958), “Programs with Common Sense”. Os campos da robótica e da teoria de controle, por sua própria natureza, se preocupam principalmente com a elaboração de agentes físicos. O conceito de **controlador** em teoria de controle é idêntico ao de um agente em IA. Talvez seja surpreendente o fato de a IA ter se concentrado, durante a maior parte de sua história, em componentes de agentes isolados — sistemas de resposta a perguntas, provadores de teoremas, sistemas de visão, e assim por diante — em lugar de agentes completos. A discussão de agentes no texto de Genesereth e Nilsson (1987) foi uma exceção importante. A visão do agente como um todo é agora extensamente aceita no campo e é tema central em textos recentes (Poole *et al.*, 1998; Nilsson, 1998); Padgham e Winikoff, 2004; Jones, 2007).

O Capítulo 1 identificou as raízes do conceito de racionalidade na filosofia e na economia. Em IA, o conceito era de interesse periférico até meados da década de 1980, quando começou a suscitar muitas discussões sobre os próprios fundamentos técnicos do campo. Um artigo de Jon Doyle (1983) previu que o projeto de agentes racionais viria a ser a missão central da IA, enquanto outros tópicos populares acabariam por constituir novas disciplinas.

A atenção cuidadosa às propriedades do ambiente e as suas consequências para o projeto de agentes racionais é mais presente na teoria de controle tradicional — por exemplo, sistemas de

controle clássicos (Dorf e Bishop, 2004; Kirk, 2004) lidam com ambientes completamente observáveis e determinísticos; o controle ótimo estocástico (Kumar e Varaiya, 1986; Bertsekas e Shreve, 2007) trata de ambientes estocásticos parcialmente observáveis, e o controle híbrido (Henzinger e Sastry, 1998; Cassandras e Lygeros, 2006) lida com ambientes que contêm elementos discretos e elementos contínuos. A distinção entre ambientes completa e parcialmente observáveis também é central na literatura de **programação dinâmica** desenvolvida na área de pesquisa operacional (Puterman, 1994), que discutiremos no Capítulo 17.

Os agentes reativos constituíram o modelo fundamental para psicólogos comportamentais como Skinner (1953), que tentou reduzir a psicologia de organismos a mapeamentos de entrada/saída ou estímulo/resposta. O avanço desde o behaviorismo até o funcionalismo em psicologia, que foi pelo menos em parte orientado pela aplicação da metáfora de computadores a agentes (Putnam, 1960; Lewis, 1966), inseriu no cenário o estado interno do agente. A maioria dos trabalhos relacionados à IA vê a ideia de agentes reativos puros com estado como algo demasiado simples para proporcionar grande avanço, mas o trabalho de Rosenschein (1985) e Brooks (1986) questionou essa suposição (ver o Capítulo 25). Nos últimos anos, muito trabalho tem sido dedicado à busca de algoritmos eficientes para controlar ambientes complexos (Hamscher *et al.*, 1992; Simon, 2006). O programa Remote Agent (descrito na página 28) que controlava a espaçonave Deep Space One (descrita na página 28) é um exemplo particularmente impressionante (Muscettola *et al.*, 1998; Jonsson *et al.*, 2000).

Os agentes baseados em objetivos são previstos em tudo desde a visão de Aristóteles do raciocínio prático até os primeiros artigos de McCarthy sobre a IA baseada em lógica. Shakey the Robot (Fikes e Nilsson, 1971; Nilsson, 1984) foi a primeira materialização robótica de um agente lógico baseado em objetivos. Uma análise lógica completa sobre agentes baseados em objetivos foi apresentada em Genesereth e Nilsson (1987), e uma metodologia de programação baseada em objetivos, denominada programação orientada a agentes, foi desenvolvida por Shoham (1993). A abordagem baseada em agente é hoje extremamente popular na engenharia de software (Ciancarini e Wooldridge, 2001). Também se infiltrou na área de sistemas operacionais, onde a **computação autônoma** refere-se a sistemas computacionais e redes que se monitoram e se controlam com um laço de percepção-ação e métodos de aprendizado de máquina (Kephart e Chess, 2003). Observando que uma coleção de programas de agente projetada para trabalhar juntos em um verdadeiro ambiente multiagente, necessariamente apresenta modularidade — os programas não compartilham o estado interno e se comunicam uns com os outros somente através do ambiente —, é comum na área de **sistemas multiagentes** projetar o programa de agente, de um único agente, como uma coleção de subagentes autônomos. Em alguns casos, pode até mesmo ser provado que o sistema resultante devolve as mesmas soluções ótimas que um projeto monolítico.

A visão de agentes baseada em objetivos também domina a tradição da psicologia cognitiva na área de resolução de problemas, começando com o influente trabalho *Human Problem Solving* (Newell e Simon, 1972) e passando por todo o trabalho mais recente de Newell (Newell, 1990). Os objetivos, analisados com maior profundidade como *desejos* (gerais) e *intenções* (atuais), são centrais para a teoria de agentes desenvolvida por Bratman (1987). Essa teoria tem sido influente tanto para a compreensão da linguagem natural quanto para sistemas multiagentes.

Horvitz *et al.* (1988) sugerem especificamente o uso da racionalidade concebida como a

maximização da utilidade esperada, como base para a IA. O texto de Pearl (1988) foi o primeiro em IA a abordar em profundidade a teoria de probabilidade e utilidade; sua exposição de métodos práticos para raciocínio e tomada de decisões sob incerteza talvez tenha sido o maior fator para a rápida mudança em direção a agentes baseados em utilidade nos anos 1990 (ver a Parte IV).

O projeto geral de agentes com aprendizagem representado na Figura 2.15 é clássico na literatura de aprendizagem de máquinas (Buchanan *et al.*, 1978; Mitchell, 1997). Exemplos do projeto, materializados em programas, remontam no mínimo ao programa de aprendizado de Arthur Samuel (1959, 1967) para jogar damas. Os agentes com aprendizagem são descritos em profundidade na Parte V.

O interesse em agentes e em projeto de agentes cresceu com rapidez nos últimos anos, em parte devido ao crescimento da Internet e à necessidade percebida de se desenvolver **softbots** automatizados e móveis (Etzioni e Weld, 1994). Documentos relevantes estão reunidos em *Readings in Agents* (Huhns e Singh, 1998), e em *Foundations of Rational Agency* (Wooldridge e Rao, 1999). Textos sobre sistemas multiagentes normalmente fornecem uma boa introdução a muitos aspectos do projeto de agente (Weiss, 2000a; Wooldridge, 2002). Uma série de conferências dedicadas aos agentes começou nos anos 1990, incluindo o Internacional Workshop on Agent Theories, Architectures and Languages (ATAL), a International Conference on Autonomous Agents (AGENTS) e a International Conference on Multi-Agents Systems (ICMAS). Em 2002, essas três se fundiram para formar a International Conference on Autonomous Agents and Multiagent Systems (AAMAS). O periódico *Autonomous Agents and Multi-Agent Systems* foi fundado em 1998. Finalmente, *Dung Beetle Ecology* (Hanski e Cambefort, 1991) fornece grande quantidade de informações interessantes sobre o comportamento de besouros de esterco. O YouTube traz gravações de vídeo inspiradas em suas atividades.

EXERCÍCIOS

2.1 Suponha que a medida de desempenho preocupa-se apenas com os T primeiros passos de tempo do ambiente e ignora tudo a partir de então. Mostre que a ação de um agente racional depende não apenas do estado do ambiente, mas também do passo de tempo que ele alcançou.

2.2 Vamos examinar a racionalidade de várias funções do agente aspirador de pó.

- Mostre que a função do agente aspirador de pó simples descrito na Figura 2.3 é realmente racional, conforme as suposições listadas na página 38.
- Descreva uma função de agente racional para o caso em que cada movimento custa um ponto. O programa de agente correspondente exige estado interno?
- Descreva possíveis projetos de agentes para os casos em que quadrados limpos podem ficar sujos e a geografia do ambiente é desconhecida. Faz sentido para o agente aprender a partir de sua experiência nessas situações? Em caso afirmativo, o que ele deve aprender? Se não, por quê?

2.3 Para cada uma das seguintes afirmações, diga se é verdadeiro ou falso e justifique com exemplos a sua resposta ou com contraexemplos se for o caso.

- Um agente que detecta apenas informações parciais sobre o estado não pode ser perfeitamente