2 AGENTES INTELIGENTES

Discutiremos aqui o que um agente inteligente faz, como ele está relacionado com seu ambiente, como ele evolui e como podemos construi-lo.

2.1 INTRODUÇÃO

Um **agente** é uma entidade que possa perceber seu ambiente através de **sensores** e **age** sobre esse ambiente através de "**efetuadores"**. Um agente humano possui olhos, ouvidos e outros órgãos como sensores, e mãos, pernas, boca e outras partes do corpo como efetuadores. Um agente robótico utiliza câmeras e raios infravermelhos como sensores e vários motores como efetuadores. Um agente de software codifica cadeias de bits tanto para suas percepções como para suas ações. Um agente genérico está diagramado na figura 2.1.

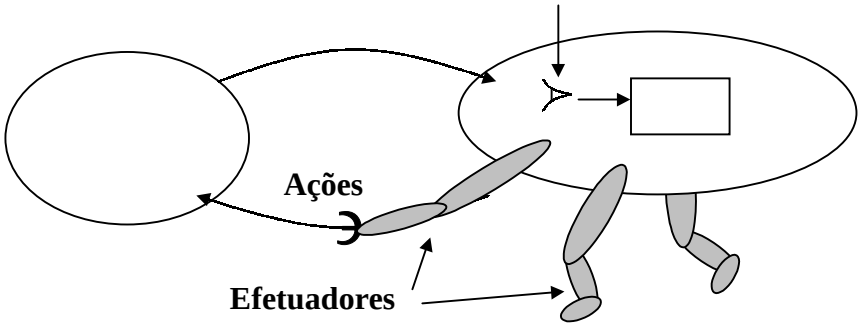

Figura 2.1 Agentes interagem com ambientes através de sensores e efetuadores

Nosso objetivo nesse livro é projetar agentes que façam um bom trabalho de atuação em seu ambiente. Primeiramente, teremos de ser um pouco mais precisos sobre o que queremos dizer com um bom trab**a**lho. Falaremos então sobre diferentes projetos para agentes de sucesso – preenchendo o ponto de interrogação na figura 2.1. Discutiremos alguns dos princípios gerais utilizados no projeto de agentes ao longo do livro, principalmente sobre o princípio de que os agentes deveriam *saber* coisas. Finalmente, mostraremos como casar um agente com um ambiente e descrever vários tipos de ambientes.

2.2 COMO AGENTES DEVEM AGIR

Um **agente racional** é aquele que faz as coisas corretamente. Obviamente, isto é melhor do que fazer as coisas de forma errada, mas o que isso significa? Como primeira aproximação, diremos que a ação correta é aquela que causará maior sucesso ao agente. Isso nos deixa com o problema de decidir *como* e *quando* avaliar o sucesso do agente.

Utilizamos o termo **medida de desempenho** para o *como* – o critério que determina o quanto ele obteve sucesso. Obviamente, não existe uma medida fixa adequada para todos os agentes. Poderíamos pedir ao agente uma opinião subjetiva de quão feliz está com seu próprio desempenho, mas alguns agentes estariam impossibilitados de responder, e outros se iludiriam. (Os agentes humanos são particularmente notório em "tirar o corpo fora", dizendo que eles não quiseram algo, depois que eles realmente proceder de uma determinada maneira após seu fracasso. Em outras palavras, como observadores externos, nós estabelecemos um padrão do que significa ter sucesso em um ambiente e utilizá-lo para medir a capacidade de um agente.

Como um exemplo, considere o caso de um agente que se supõe estar aspirando um chão sujo. Uma medida de desempenho plausível seria a quantidade de sujeira limpada em um turno de oito horas de trabalho. Uma medida de desempenho mais sofisticada tomaria como fator a quantidade de eletricidade consumida ou a quantidade de ruído gerado. Uma terceira medida de desempenho pode dar notas mais altas a um agente que não apenas limpa o chão silenciosa e eficientemente, mas também encontra tempo para surfar nos fins de semana.

O *quando* da avaliação do desempenho também é importante. Se medimos quanta sujeira o agente limpou na primeira hora do dia, estaríamos recompensando aqueles agentes que começaram primeiro (mesmo se eles fazem pouco ou nenhum trabalho mais tarde), e punindo aqueles que trabalham consistentemente. Daí, queremos medir o desempenho durante o tempo dedicado ao trabalho, seja ele um intervalo de oito horas ou o tempo de uma vida.

Precisamos ter cuidado ao distinguir racionalidade e onisciência. Um agente onisciente conhece o resultado real de suas ações, e pode agir de acordo; mas onisciência é impossível na realidade. Considere o exemplo seguinte. Estou caminhando ao longo da Champs Elysées um dia e eu avisto um velho amigo do outro lado da rua. Não existe tráfego nas proximidades e eu não estou apressado. Assim, sendo racional, eu começo a atravessar a rua. Enquanto isso, a 33.000 pés, a porta de carga de um avião cai, e antes eu chegue ao outro lado da rua, sou atingido. Fui irracional cruzar a rua? É improvável que no meu obituário leiam "tentativa idiota de atravessar a rua". Ao comtrário, isso mostra que racionalidade está preocupada com *sucesso esperado de acordo com o que foi percebido*. Cruzar a rua foi racional porque na maioria das vezes o cruzamento teria sucesso, e não havia nenhuma maneira de antever a queda da porta do avião. Note que outro agente equipado com radar para detectar portas caindo ou uma gaiola de aço forte o bastante para repelí-la teria mais sucesso, mas não seria em nada mais racional.

Em outras palavras, não podemos culpar um agente por falhar em levar em conta algo que não pôde perceber, ou por falhar em tomar uma ação (assim como repelir a porta de carga) que é incapaz de tomar. Mas, o relaxamento dos requisitos de perfeição não é apenas uma pergunta de ser justo para os agentes. O ponto é que nós especificamos que um agente inteligente deveria sempre realizar o que é *de fato* a coisa correta, será impossível projetar um agente que preencha todos os requisitos dessa especificação - a menos que melhoremos o desempenho das bolas de cristal.

Em suma, o que é racional em qualquer momento dado depende de quatro coisas:

- A medida de desempenho que define o grau de sucesso.
- Tudo o que o agente conseguiu perceber até então. Chamaremos esse histórico completo de percepção de **seqüência de percepção**.
- O que o agente sabe sobre o ambiente.
- As ações que o agente pode executar.

Isto nos leva à definição de um agente racional ideal: para cada possível seqüência de percepção, um agente racional ideal deve fazer qualquer ação que seja esperada para maximizar sua medida de desempenho, com base na evidência fornecida pela seqüência de percepção e qualquer conhecimento embutido que o agente possua.

Precisamos olhar esta definição cuidadosamente. À primeira vista, pode parecer como que permitisse a um agente incorrer em algumas atividades decididamente sub-inteligentes. Por exemplo, se um agente não olha ambos os lados antes de atravessar uma rua movimentada, então sua seqüência de percepção não lhe dirá que existe um grande caminhão que se aproxima em alta velocidade. A definição parece dizer que seria correto para ele atravessar a rua. De fato, esta interpretação está errada em dois

sentidos: primeiro, não seria racional atravessar a rua, pois o risco de atravessar sem olhar é muito grande. Segundo, um agente racional ideal teria escolhido a ação “olhar" antes de atravessar a rua, porque esta ação ajudaria a maximizar o desempenho esperado. Realizar ações *para se obter informações úteis* é uma parte importante da racionalidade e é coberta em profundidade no Capítulo 16.

A noção de um agente é tomada como uma ferramenta para analisar sistemas, não uma caracterização absoluta que divide o mundo em agentes e não-agentes. Considere um relógio. Ele pode ser considerado como um objeto inanimado, ou pode ser considerado como um agente simples. Como um agente, a maioria dos relógios sempre faz a ação correta: mover os ponteiros (ou exibir dígitos) de maneira apropriada. Relógios são um tipo de agente degenerados em que a sucessão de percepção deles é vazia; não importa o que aconteça lá fora, a ação do relógio não deverá ser afetada.

Bem, isto não é bem verdade. Se o relógio e seu dono fazem uma viagem da Califórnia para a Austrália, a coisa certa para o relógio fazer seria atrasar-se seis horas. Não ficamos chateados porque nossos relógios não podem fazer isto. Percebemos que eles estão agindo racionalmente, dado a ausência de equipamento perceptivo.

O mapeamento ideal de seqüências de percepção para ações

Uma vez percebido que o comportamento de um agente depende apenas de sua seqüência de percepção até o estágio atual, podemos descrever então qualquer agente particular fazendo uma tabela das ações em reposta a cada seqüência de percepção possível. (Para a maioria dos agentes, esta seria uma lista muita longa - infinita, de fato, a menos que estabeleçamos um limite para o comprimento das seqüências de percepção que queremos considerar.) Tal lista é chamada de **mapeamento** das seqüências de percepção para ações. Podemos, em princípio, descobrir qual mapeamento descreve corretamente um agente testando todas as possibilidades de seqüências de percepção e registrando quais ações o agente executa como resposta. (Se o agente utiliza algum processo aleatório em suas computações, teríamos então que testar algumas seqüências de percepção várias vezes para termos uma boa idéia do comportamento médio de um agente.) E se os mapeamentos descrevem agentes, então **mapeamentos ideais** descrevem agentes ideais. *A especificação de qual ação um agente deve tomar em resposta a qualquer seqüência de percepção dada, fornece um projeto para um agente ideal*.

Isto não significa, certamente, que devemos criar um tabela explícita com entradas para cada seqüências de percepção possível. É possível definir a especificação do mapeamento sem enumerá-lo exaustivamente. Considere um agente muito simples: a função raiz quadrada numa calculadora. A seqüência de percepção para esse agente é uma seqüência de acionamentos de teclas representando um número, e a ação é a apresentação do número na tela. Um mapeamento ideal é aquele que quando a percepção é um número positivo x, a ação correta é mostrar um número positivo z, tal que $z^2 \cong x$, com aproximação de, digamos, 15 casas decimais. Esta especificação do mapeamento ideal não requer do projetista a construção de uma tabela de raízes quadradas. Nem a função de raiz quadrada necessita de uma tabela para funcionar corretamente: na Figura 2.2 é apresentado parte do mapeamento ideal e um programa simples que implementa o mapeamento usando o método de Newton.

O exemplo da raiz quadrada ilustra a relação entre o mapeamento ideal e um projeto de agente ideal para uma tarefa muito restrita. Embora a tabela seja muito grande, o agente é um programa bem compacto. Isto significa que podemos construir agentes compactos que implementem o mapeamento ideal para situações mais gerais: agentes que resolvem uma variedade ilimitada de tarefas em uma variedade ilimitada de ambientes. Antes de discutirmos como fazer isso, precisamos dar uma olhada em mais um requisito que um agente inteligente deve satisfazer.

Autonomia

Existe mais uma coisa para se tratar na definição de um agente racional ideal: a parte do “conhecimento embutido”. Se as ações de um agente são completamente baseadas em conhecimento embutido, de maneira que ele não precisa considerar suas seqüências de percepção, então dizemos que tal agente carece de **autonomia**. Por exemplo, se o fabricante do relógios tivesse preciosismo suficiente para saber que o dono do relógio iria para a Austrália em uma data particular, então um mecanismo poderia ser

construído para ajustar os ponteiros automaticamente em 6 horas no tempo correto. Este seria um comportamento de sucesso, mas a inteligência seria do fabricante do relógio e não do relógio em si.

Percepção x	Ação z
1.0	1.000000000000000
1.1	1.048808848170152
1.2	1.095445115010332
1.3	1.140175425099138
1.4	1.183215956619923
1.5	1.224744871391589
1.6	1.264911064067352
1.7	1.303840481040530
1.8	1.341640786499874
1.9	1.378404875209022
...	...

function SQRT(x)
  $z \leftarrow 1.0$
  repeat until $|z^2 - x| < 10^{-15}$
    $z \leftarrow z - (z^2 - x) / (2z)$
  end
return z

Figura 2.2 Parte do mapeamento ideal para o problema

Um comportamento de um agente pode ser baseado na sua própria experiência e no conhecimento embutido utilizado na construção do agente para o ambiente particular em que opera. *Um sistema é autônomo na medida em que seu comportamento é determinado por sua própria experiência*. Seria muito difícil, contudo, querer completa autonomia a partir do comando "execute!": quando um agente tem pouca ou nenhuma experiência, teria que reagir aleatoriamente, a menos que o projetista tenha lhe fornecido alguma assistência. Então, assim como a evolução fornece aos animais alguns reflexos embutidos para que eles sobrevivam o bastante para aprenderem sozinhos, seria razoável fornecer um conhecimento inicial a um agente com inteligência artificial, assim como a habilidade para aprender.

Autonomia não apenas condiz com nossa intuição, mas é também um exemplo de prática da engenharia legítima e completa. Um agente que opera com base em suposições embutidas só operará com sucesso quando aquelas suposições forem sustentadas, e então carece de flexibilidade. Considere por exemplo, a vespa. Depois de cavar para enterrar seus ovos, ela traz uma bola de esterco que esteja por perto para fechar a entrada; se a bola de esterco for removida do local, a vespa continua pensando que existe uma bola de esterco, não percebendo que ela foi retirada. A evolução embutiu uma suposição no comportamento da vespa, e quando esta é violada, comportamentos sem sucessos resultarão. Um verdadeiro agente inteligente autônomo deve ser capaz de operar com sucesso em uma grande variedade de ambientes, dado tempo suficiente para adaptação.

2.3 ESTRUTURA DE AGENTES INTELIGENTES

Até agora temos falado sobre agentes descrevendo seu comportamento – a ação que é realizada após qualquer seqüência de percepções dada. Agora, temos que arregaçar as mangas e conversar sobre a estrutura interna dos agentes. O trabalho da Inteligência Artificial é projetar um **programa agente**: uma função que implementa o mapeamento, do agente, das percepções para as ações. Assumimos que esse programa será executado em algum tipo de dispositivo de computação, que chamaremos de **arquitetura**. Obviamente, o programa que escolhemos deve ser tal que a arquitetura o aceitará e o executará. A arquitetura pode ser um computador simples, ou pode incluir um hardware de propósito especial para certas tarefas, tal como câmeras de processamento de imagens ou filtros de entrada de áudio. A arquitetura pode também incluir software que fornece um grau de isolamento entre o computador em si e o programa agente, de maneira que possamos programar em um nível mais alto. Em geral, a arquitetura realiza a percepção a partir de sensores disponíveis ao programa, executa o programa e alimenta as

escolhas de ação do programa para os efetuadores enquanto são gerados. A relação entre agentes, arquiteturas e programas pode ser definida como segue:

agente = arquitetura + programa

A maior parte deste livro é sobre o projeto de programas agentes, embora os capítulos 24 e 25 tratem diretamente da arquitetura.

Antes de projetarmos o agente de programa, devemos ter uma boa idéia das percepções e ações possíveis, que metas ou medidas de desempenho o agente é suposto capaz de realizar, e sob que tipos de ambientes ele irá operar. Estes vêm numa grande variedade. Na figura 2.3 são apresentados os elementos básicos para uma seleção de tipos de agentes.

Tipo de agente	Percepção	Ações	Metas	Ambiente
Sistema de diagnóstico médico	Sintomas, descobertas, respostas dos pacientes	Questionários, testes, tratamentos	Saúde do paciente, minimização dos custos	Paciente, hospital
Sistema de análise de imagens de satélite	Variação de intensidade dos pixels, cores	Imprimir a categorização de um cenário	Categorização correta	Imagens do satélite orbital
Robô coletor de peças	Variação da intensidade dos pixels	Recolhe peça e distribui entre recipientes	Coloca peça nos recipientes corretos	Esteira rolante com peças
Controlador de refinaria	Temperatura, leitura de pressão	Abre, fecha válvulas, ajusta temperatura	Maximiza pureza, produção, segurança	Refinaria
Tutor interativo de Inglês	Palavras digitadas	Imprime exercícios, sugestões, correções	Maximiza a nota dos alunos nos testes	Grupo de estudantes

Figura 2.3 Exemplos de tipos de agentes e suas páginas de descrição

Pode até ser uma surpresa para alguns leitores termos incluído na nossa lista de tipos de agentes alguns programas que parecem operar num ambiente totalmente artificial definido por uma entrada por teclado e uma saída de caracteres numa tela. ”Certamente” alguém pode dizer, “este 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 seqüência de percepções gerada pelo ambiente, e as metas que o agente é capaz de realizar. Alguns ambientes “reais” são efetivamente simples. Por exemplo, um robô projetado para inspecionar peças, na medida que passam pela esteira rolante, pode fazer uso de um número de suposições simplificadas: que o peso está ajustado de maneira que as únicas coisas sobre a esteira rolante serão peças de um certo tipo, e que existem apenas duas ações: aceitar a peça ou marcá-la como rejeitada.

Em contraste, alguns agentes de software (ou robôs de software ou softbots) existem em domínios ricos e ilimitados. Imagine um softbot projetado para pilotar um simulador de vôo de um 747. O simulador é um ambiente detalhado, complexo, e o agente de software deve escolher entre uma diversidade de ações em tempo real. Imagine ainda um softbot projetado para procurar novidades “online” e repassar os itens de interesse para seus clientes. Para fazê-lo bem, precisará de algumas habilidades de processamento de linguagem natural, precisará aprender em que cada cliente está interessado, e precisará mudar dinamicamente seus planos quando, por exemplo, se perde uma conexão com uma das fontes de notícias ou quando uma nova destas conexões é estabelecida.

Alguns ambientes obscurecem a distinção entre “real” e “artificial”. Nos ambientes vivos (Maes et al.,1994), é dado aos agentes de software como percepção, imagens de uma câmera digitalizadas instalada num cômodo onde um humano caminha. O agente processa a imagem da câmera e processa uma ação. O ambiente também apresenta a imagem da câmera num grande monitor que o humano pode assistir, e sobrepõe na imagem um renderização gráfica do agente de software. Uma dessas imagens é um desenho animado de um cachorro, que foi programado para mover-se em direção ao humano (a menos que ele acene para que o chachorro vá embora) e para mover as mãos ou pular ansioso quando o humano fizer certos gestos.

O mais famoso ambiente artificial é o ambiente do Teste de Turing, no qual o principal ponto é que agentes reais e artificiais estão num mesmo nível, mas o ambiente é desafiador de tal modo que fica muito difícil para o agente de software proceder tão bem quanto o humano. A Seção 2.4 descreve em maiores detalhes os fatores que fazem alguns ambientes mais exigentes do que outros.

Programas agentes

Estaremos construindo agentes inteligentes durante a leitura deste livro. Todos eles terão o mesmo esqueleto, isto é, aceitarão percepções de um ambiente e gerarão ações. As versões primárias dos programas agentes terão uma forma muito simples (Figura 2.4). Cada um utilizará alguma estrutura de dados interna que será atualizada à medida que chegarem novas percepções. Estas estruturas de dados são operadas pelos procedimentos de tomada de decisão do agente para gerar uma escolha de ação, que é então passada para a arquitetura para ser executada.

Existem duas coisas que devem ser notadas sobre esse esqueleto de programa. Primeiro, ainda que definamos o mapeamento do agente como uma função da seqüência de percepções para ações, o programa agente recebe apenas uma percepção simples como entrada. É responsabilidade do agente construir a seqüência de percepções na memória, se isto for sua de vontade. Em alguns ambientes, é possível se ter muito sucesso sem armazenar a seqüência de percepções, e em domínios complexos, é inviável armazenar a seqüência completa. Segundo, objetivo ou a medida de desempenho não é parte do esqueleto do programa. Isto porque a medida de desempenho é aplicada externamente para julgar o comportamento do agente, e é sempre possível alcançar um alto desempenho sem o conhecimento explicito da medida de desempenho. (Ver, e.g., o agente de raiz quadrada).

```
function AGENTE-ESQUELETO(percepção) returns ação
  static: memoria, a memoria do mundo do agente

  memoria ← ATUALIZA-MEMÓRIA(memoria,percepção)
  ação ← ESCOLHE-MELHOR-AÇÃO(memoria)
  memoria ← ATUALIZA-MEMÓRIA(memoria,ação)

  return ação
```

Figura 2.4 O agente esqueleto. Em cada invocação, a memória do agente é atualizada para refletir a nova percepção, a melhor ação é escolhida, e o fato de que a ação foi tomada e armazenada na memória. A memória persiste de uma invocação para outra.

Por que não observar apenas as respostas?

Comecemos com o modo mais simples possível que podemos pensar para escrever o programa agente – uma olhada na tabela. Na Figura 2.5 é apresentado o programa agente. Ele opera retendo na memória todas as suas seqüências de percepção, e as utiliza para indexar a tabela, que contém as ações apropriadas para todas as possíveis seqüências de percepção.

```
function AGENTE-DIRIGIDO-POR-TABELA(percepção) returns ação
  static: percepções, uma seqüência, inicialmente vazia
          tabela, uma tabela, indexada por seqüências de percepção, totalmente especificada desde
          o início

  concatenar percepção no fim de percepções
  ação ← PROCURA(percepção,tabela)

  return ação
```

Figura 2.5 Um agente baseado numa tabela de pesquisa pré-especificada. Ele armazena as seqüências de percepção e só procura o a melhor ação

É instrutivo considerar por que esta proposta está fadada ao fracasso:

1. A tabela necessária para algo tão simples como um agente que possa apenas jogar xadrez teria aproximadamente 35^{100} entradas.
2. O projetista levaria um tempo muito grande para construir a tabela.
3. O agente não tem nenhuma autonomia, porque o cálculo das melhores ações está completamente embutido. Então, se o ambiente muda de maneira inesperada, o agente se perderia.
4. Mesmo se dermos ao agente um mecanismo de aprendizagem, de maneira que ele pudesse ter algum grau de autonomia, ele tomaria a eternidade para aprender o valor correto para todas as entradas da tabela.

Apesar disso tudo, AGENTE-DIRIGIDO-POR-TABELAS faz o que nós queremos: implementa o mapeamento de agente desejado. Não é suficiente dizer: “Ele não pode ser inteligente”; a questão é entender por que um agente que raciocina (em oposição a procurar coisas na tabela) pode fazer melhor evitando as quatro desvantagens listadas aqui.

Um exemplo

Neste ponto, será útil considerar um ambiente particular, de maneira que nossas discussões possam tornar-se mais concretas. Principalmente devido as suas familiaridades, e porque envolve uma larga faixa de possibilidades, observaremos o trabalho de projeto de automação de um motorista de táxi. Deveríamos mostrar, antes do leitor ficar alarmado, que tal sistema é atualmente algo que está além da capacidade da tecnologia existente, embora a maioria dos componentes estejam disponíveis de alguma forma. A tarefa de dirigir plenamente é extremamente aberta – não há um limite para o novelo de combinações das circunstâncias que podem surgir (que é uma outra razão pela qual escolhemos este como um foco para discussão).

Devemos primeiro pensar sobre as percepções, as ações, as metas e o ambiente do taxi. Eles estão resumidos na figure 2.6 e discutidos a seguir.

Tipo do agente	percepção	ação	meta	ambiente
Motorista de taxi	Câmeras, GPS, velocímetro, sonar, microfone	Dirigir, acelerar, frear, falar com o passageiro	Segurança, rapidez, legalidade, máximo profissionalismo, viagem confortável	Estrada, pedestres, outros tráficos, clientes

Figura 2.6 Agente do tipo Motorista de Taxi

O táxi precisará saber onde está, o que mais está na rua, e quão rápido está indo. Esta informação pode ser obtida das percepções fornecidas por uma ou mais câmeras de TV controladas e pelo velocímetro. Para controlar o veículo corretamente, especialmente nas curvas, ele deveria ter um medidor

da aceleração; também precisará saber o estado mecânico do veículo, de forma que também precisará dos usuais sensores do motor e dos sistemas elétricos. Poderá ter instrumentos que não estão disponíveis para o motorista humano: um sistema satélite de posicionamento global (GPS) para dar a informação precisa da posição referente a um mapa eletrônico; ou infravermelho ou sensores sonares para detectar distâncias de outros carros e obstáculos. Finalmente, precisará de um microfone ou teclado para os passageiros dizerem seu destino.

As **ações** disponíveis para o motorista de taxi serão mais ou menos as mesmas disponíveis para um motorista humano: controle do motor através de um pedal e controle sobre a direção e os freios. Além disso, precisará de uma saída para uma tela ou um sintetizador de voz para responder ao passageiros, e talvez algum modo de se comunicar com outros veículos.

Quais **medidas de desempenho** gostaríamos que nosso motorista automático aspirasse? Qualidades desejáveis incluem atingir o destino correto; minimizar o consumo de combustível; minimizar o tempo e/ou custo da viagem; minimizar violações das leis de trânsito e distúrbios a outros motoristas; maximizar a segurança e o conforto dos passageiros; maximizar lucros. Obviamente, alguns destes objetivos são conflitantes, fazendo-se necessário negociações.

Finalmente, onde isto é um projeto real, precisaremos decidir que tipo de **ambiente** de condução o táxi irá encontrar. Ele deve operar em estradas locais, ou também em rodovias? Ele estará no sul da Califórnia onde a neve é raramente um problema, ou no Alaska onde a neve não é tão rara? Ele vai sempre estar dirigindo na direita ou talvez queiramos que ele seja flexível o suficiente para dirigir na esquerda, no caso de se querer operar com o táxi na Grã-Bretanha ou Japão? Obviamente, quanto mais restrito o ambiente, mais fácil o problema.

Agora temos decidir como construir um programa real para implementar o mapeamento das percepções em ações. Descobriremos que diferentes aspectos de condução sugerem diferentes tipos de programas agentes. Consideraremos quatro tipos de programas agentes:

- Agentes de reflexos simples
- Agentes que mantêm registros do mundo
- Agentes baseados em metas
- Agentes baseados em utilidades

Agente de reflexo simples

A opção de construir uma tabela explícita de referência está fora de questão. A entrada visual de uma simples câmera chega a uma taxa de 50 megabytes por segundo (25 quadros por segundo, 1000 × 1000 pixels com 8 bits para cor e 8 bits para para informação sobre intensidade). Então, a tabela de referência para uma hora teria $2^{60\times60\times50M}$ entradas.

No entanto, podemos resumir porções da tabela anotando certas ocorrências comuns de associações de entrada/saída. Por exemplo, se o carro da frente freia, e suas luzes de freio ascendem, então o motorista deve notar isso e começar a freiar. Em outras palavras, algum processamento está sendo feito na entrada visual para estabelecer a condição que chamamos de "O carro da frente está freiando"; isto ativa então algumas conexões estabelecidas no programa agente para a ação "ativar frenagem". Chamamos esta conexão de **regra de condição-ação**[7], escrita da seguinte forma:

SE *carro-da-frente-está-freiando* **ENTÃO** *ativar frenagem*

Humanos também têm muitas dessas conexões, algumas das quais são respostas aprendidas (como para dirigir) e algumas delas são reflexos involuntários (como ao piscar quando algo se aproxima dos olhos). No decorrer do livro, veremos várias maneiras pelas quais tais conexões podem ser aprendidas e implementadas.

[7] Também chamadas **regras de situação-ação, produção,** ou **regras se-então**. O último termo é também usado por alguns autores para implicações lógicas, então nós iremos evitá-los.

Na Figura 2.7 é mostrada a estrutura de um agente de reflexo simples de forma esquemática, mostrando como regras de condição-ação permitem ao agente fazer a conexão entre percepção e ação (não se preocupe se isto parecer trivial; se tornará mais interessante brevemente). Usamos retângulos para denotar o estado interno corrente do processo de decisão do agente, e ovais para representar as informações secundárias usadas no processo. O programa agente, que também é muito simples, é mostrado na Figura 2.8. A função INTERPRETA-ENTRADA gera uma descrição abstrata do estado corrente da percepção, e a função CASA-REGRA retorna a primeira regra no conjunto de regras que casam com a descrição de estado dada. Apesar desses agentes poderem ser implementados muito eficientemente (ver Capítulo 10), seu campo de aplicação é muito restrito, com veremos.

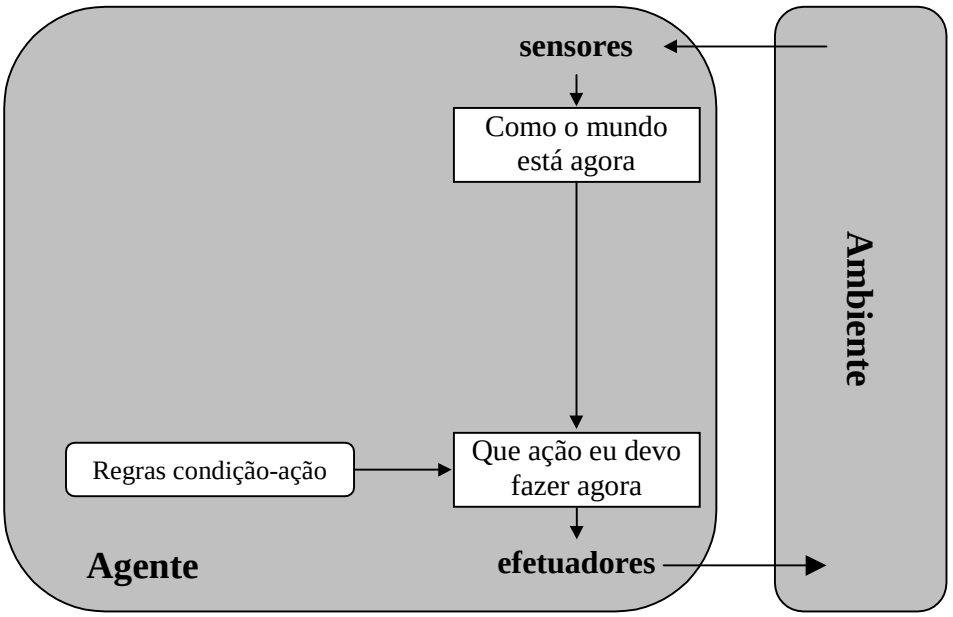

Figura 2.7 Diagrama esquemático de um agente de reflexo simples

```
function AGENTE-DE-REFLEXO-SIMPLES(percepção) returns ação
  static: regras, um conjunto de regras condição-ação

  estado ← INTERPRETA-ENTRADA(percepção)
  regra ← CASA-REGRAS(estado,regras)
  ação ← REGRA-AÇÃO[regra]

  return ação
```

Figura 2.8 Um agente de reflexo simples. Ele trabalha encontrando uma regra cuja condição se casa com a situação corrente (como definido pela percepção) e executando então a ação associada com tal regra

Agentes que mantêm informações sobre o mundo

O agente de reflexos simples que descrevemos anteriormente vai funcionar apenas se a decisão correta puder ser feita com base na percepção corrente. Se o carro da frente for um modelo recente, e tem uma luz de freio centralizada, que é agora exigida nos Estados Unidos, então será possível dizer se ele está freiando a partir de uma única imagem. Infelizmente, modelos mais antigos têm configurações diferentes de luz traseira, luzes de freio e luzes laterais, e nem sempre é possível dizer se o carro está freiando. Assim, mesmo para a simples regra de frenagem nosso motorista vai ter que manter algum tipo de estado interno para escolher a ação. Aqui, o estado interno não é muito extenso – ele necessita apenas

de um quadro anterior da câmera para detectar quando as duas luzes vermelhas na traseira do veículo ligam e desligam simultaneamente.

Considere os seguintes casos mais óbvios: de vez em quando, o motorista olha pelo espelho retrovisor para checar a localização dos veículos próximos. Quando o motorista não está olhando no espelho, os veículos na faixa ao lado não estão visíveis (os estados nos quais eles estão presentes ou ausentes são indistingüíveis); mas de maneira a decidir por uma manobra de mudança de faixa, é necessário que o motorista saiba se eles estão ou não lá.

O problema ilustrado nesse exemplo surge porque os sensores não fornecem acesso ao estado completo do mundo. Em tais casos, o agente pode precisar manter alguma informação de estado interna de maneira que possa distinguir entre estados do mundo que geram a mesma entrada perceptiva, mas que, apesar disso, sejam significativamente diferentes. Aqui, "significativamente diferente" significa que ações diferentes são apropriadas nos dois estados.

A atualização da informação de estado interno, com o passar do tempo, exige que dois tipos de conhecimento sejam codificados no programa agente. Primeiro, precisamos de alguma informação de como o mundo evolui independentemente dos agentes – por exemplo, a informação de que um carro que aparece de repente vai estar mais próximo de você do que estava num momento anterior. Segundo, precisamos de alguma informação sobre como as próprias ações dos agentes influenciam o mundo – por exemplo, a informação de que quando o agente se muda para a faixa direita, então existe um espaço (pelo menos temporariamente) na faixa em que ele estava anteriormente, ou que após dirigir por cinco minutos, no sentido norte da rodovia, ele vai estar a mais ou menos cinco milhas ao norte de onde ele estava cinco minutos atrás.

Na figura 2.9 é apresentada a estrutura do agente de reflexos, mostrando como a percepção corrente é combinada com os antigos estados internos para gerar a descrição atual do estado corrente. O programa agente é mostrado na figura 2.10. A parte interessante é a função ATUALIZA-ESTADO, que é responsável pela criação das descrições de novos estados internos. Assim, como interpretação da nova percepção, à luz do conhecimento existente sobre o estado, ele utiliza informação sobre como o mundo evolui para manter o registro de partes ocultas do mundo, e deve também saber como as ações do agente influenciam o mundo. Exemplos detalhados aparecem nos capítulos 7 e 17.

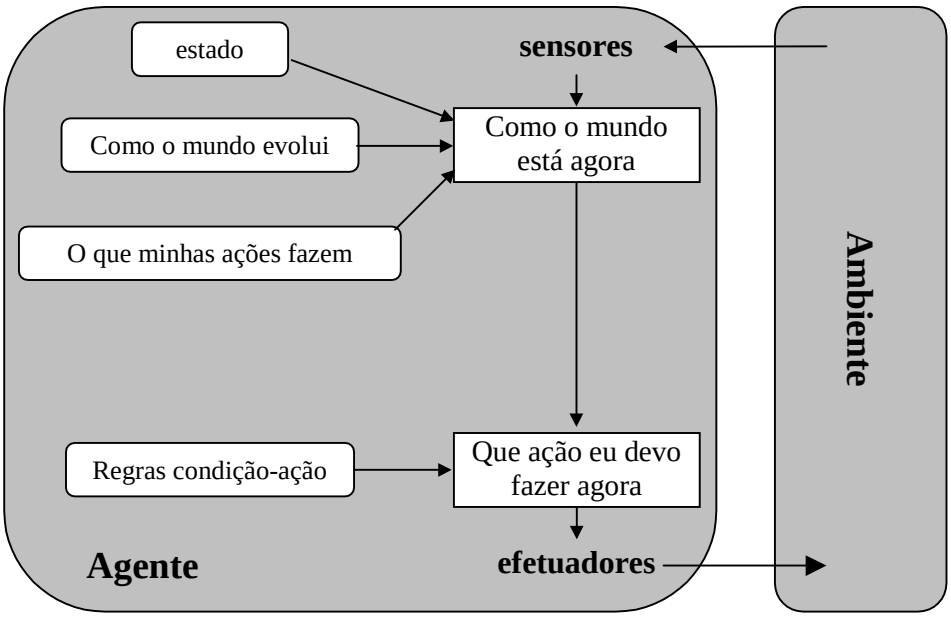

Figura 2.9 Um agente de reflexo com estado interno

```
function AGENTE-DE-REFLEXO-COM-ESTADO(percepção) returns ação
  static: estado, uma descrição do estado do mundo corrente
          regras, um conjunto de regras condição-ação

  estado ← ATUALIZA-ESTADO(estado,percepção)
  regra ← CASA-REGRA(estado,regras)
  ação ← REGRA-AÇÃO [regra]
  estado ← ATUALIZA-ESTADO(estado,ação)

  return ação
```

Figura 2.10 Um agente de reflexo com estado interno. Ele funciona encontrando uma regra cuja condição se casa com a situação corrente (como definido pela percepção pelo estado interno armazenado) e executando então a ação associada com a regra

Agentes baseados em metas

O conhecimento sobre o estado atual do ambiente nem sempre é suficiente para se decidir o que fazer. Por exemplo, num cruzamento de rodovias, o taxi pode virar à direita, à esquerda ou seguir em frente. A decisão correta depende de onde o táxi deseja chegar. Em outras palavras, assim como a descrição do estado corrente, o agente precisa de alguma informação-**meta**, que descreve situações que são desejáveis – por exemplo, estar no destino do passageiro. O programa agente pode combinar isso com a informação sobre os resultados das ações possíveis (a mesma informação como foi utilizada para atualizar o estado interno do agente de reflexo) de maneira a escolher ações que atinja a meta. Algumas vezes, isso será simples, quando a meta é atingida a partir da aplicação imediata de uma ação simples. Algumas vezes, não será tão simples, quando o agente tiver que considerar longas seqüências de mudanças e voltas para encontrar o resultado esperado. **Busca** (capítulos 3 a 5) e **planejamento** (capítulos 11 a 13) são as sub-áreas da IA voltadas para se encontrar seqüências de ações que atinjam as metas do agente.

Note que a tomada de decisão desse tipo é fundamentalmente diferente das regras condição-ação descritas anteriormente, pois envolve consideração do futuro – do tipo “o que vai acontecer se eu fizer isto e isto?”, ou “isto me satisfará?”. Nos projetos de agentes de reflexo, esta informação não é usada explicitamente, porque o projetista pré-computou a ação correta para vários casos. O agente de reflexo freia quando vê luzes de freio. Um agente baseado em metas, em princípio, pode raciocinar que se o carro em frente tiver suas luzes de freio acesas, ele diminuirá a velocidade. Da forma como mundo normalmente evolui, a única ação que vai alcançar a meta de não bater no outro carro é freiar. Apesar dos agentes baseados em metas parecerem menos eficientes, são bem mais flexíveis. Se começa a chover, o agente pode atualizar seu conhecimento de quão eficientemente seus freios irão operar. Isso automaticamente fará com que todos os comportamentos relevantes sejam alterados para ajustar-se às novas condições. Para o agente de reflexo, por outro lado, teríamos que reescrever um grande número de regras condição-ação. Certamente, o agente baseado em metas também é mais flexível com respeito a alcançar destinos diferentes. Simplesmente pela especificação de um novo destino, podemos fazer com que o agente baseado em metas apresente um novo comportamento. A regra de agente de reflexo para quando virar e quando seguir em frente apenas funcionará para um destino simples; todas elas deves ser substituídas para se chegar a um novo lugar.

Na figura 2.11 é apresentada a estrutura do agente baseado em metas. O capítulo 13 contém programas agentes detalhados para agentes baseados em metas.

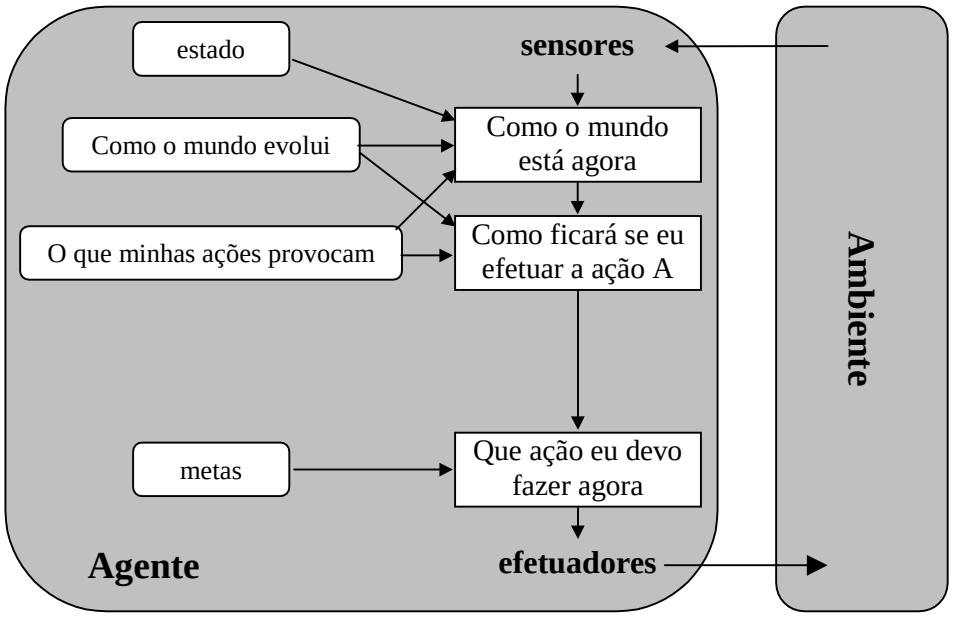

Figura 2.11 Um agente com metas explícitas

Agentes baseados em utilidade

As metas, por si só, não são realmente suficientes para gerar comportamento de alta qualidade. Por exemplo, existem muitas seqüências de ações que vão levar o taxi ao seu destino, alcançando então a meta, mas algumas são mais seguras, outras mais rápidas, confiáveis, baratas, etc. Metas fornecem apenas uma distinção crua entre os estados "satisfeito" e "insatisfeito", enquanto uma medida de desempenho mais geral deve permitir uma comparação entre diferentes estados (ou seqüências de estados) do mundo, de acordo com o grau de satisfação do agente se tivesse alcançado sua meta. Como "satisfeito" não parece muito científico, a terminologia tida como apropriada é dizer que se um estado do mundo tem preferência sobre outro, então maior **utilidade** para o agente.

Utilidade é, entretanto, uma função que mapeia um estado[1] em um número real, que descreve o grau de "satisfação" associado. Uma especificação completa da função de utilidade permite decisões racionais em dois tipos de casos onde as metas apresentam problemas. Primeiro, quando existem metas conflitantes, apenas algumas delas podem ser alcançadas (por exemplo, velocidade e segurança), a função de utilidade especifica a negociação apropriada. Segundo, quando existem várias metas que o agente pode almejar, e quando nenhuma delas pode ser alcançada com certeza, a utilidade fornece uma maneira pela qual o agente, diante desse fato, possa vir a se satisfazer com o que foi possível alcançar.

No capítulo 16, mostramos que qualquer agente racional pode ser descrito como possuidor de uma função de utilidade. Um agente que possui uma função de utilidade *explícita* e que então pode tomar decisões racionais, mas terá que comparar as utilidades alcançadas por diferentes cursos de ações. Metas, embora mais cruas, habilitam o agente a tomar imediatamente uma opção se ela satisfizer a meta. Em alguns casos, a função utilidade pode ser traduzida em um conjunto de metas, tal que as decisões tomadas por um agente baseado em meta sejam idênticas àquelas tomadas por um agente baseado em utilidade.

A estrutura do agente baseado em utilidade aparece na Figura 2.12. Programas agentes baseados em utilidade aparecerão no capítulo 5, onde examinamos programas que jogam e que devem distinguir bem entre várias posições possíveis no tabuleiro; e no capítulo 17, quando atacamos o problema geral do projeto de agentes de tomada de decisão.

[1] Ou seqüência de estados, se estamos medindo a utilidade de um agente sobre uma longo período.

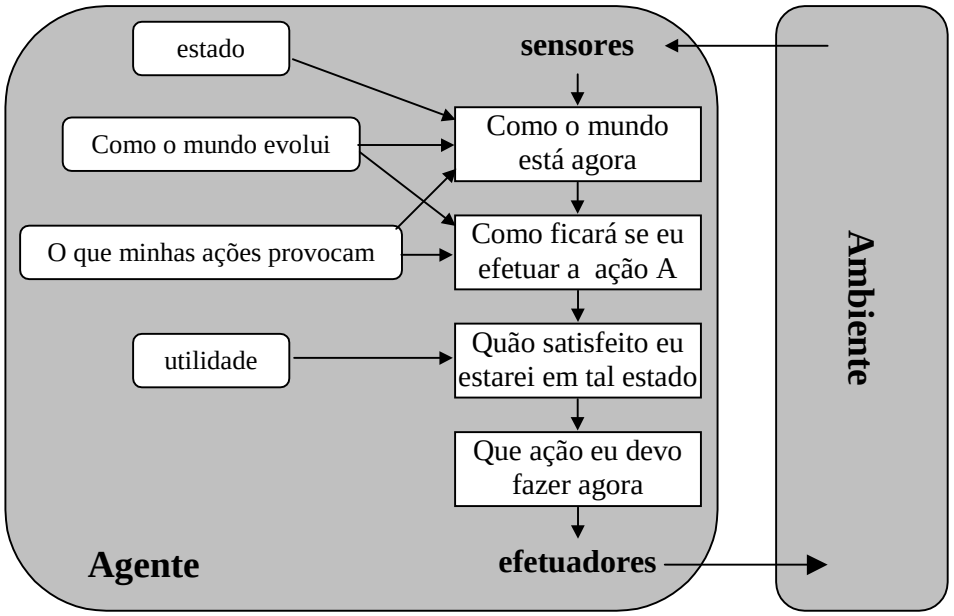

Figura 2.12 Um agente completo baseado em utilidade

2.4 AMBIENTES

Nessa seção, e no exercício no final do capítulo, você vai ver como casar um agente a um ambiente. A seção 2.3 introduziu diferentes tipos de agentes e ambientes. Em todos os casos, entretanto, a natureza da conexão entre eles é a mesma: ações são realizadas pelo agente sobre o ambiente, que fornece estímulos em turnos aos agentes. Primeiro, vamos descrever os vários tipos de ambientes e como eles afetam o desenvolvimento de agentes. Então, vamos descrever programas ambientes que podem ser usados como bancadas de teste para programas agentes.

Propriedades dos ambientes

Ambientes aparecem em várias formas. As principais distinções a serem feitas são as seguintes:

- **Acessável vs. Inacessável** – Se o aparato sensorial de um agente lhe dá acesso ao estado completo do ambiente, dizemos então que este ambiente é acessível ao agente. Um ambiente é efetivamente acessível se os sensores detectam todos os aspectos que são relevantes à escolha da ação. Um ambiente acessível é conveniente porque o agente não precisa manter qualquer estado interno para manter informação sobre o mundo.
- **Determinístico vs**. **Não-determinístico –** Se o próximo estado do ambiente é determinado completamente pelo estado atual e pelas ações selecionadas pelos agentes, dizemos então que o ambiente é determinístico. Em princípio, o agente não precisa se preocupar com a incerteza em um ambiente acessível e determinístico. Se o ambiente é inacessível, entretanto, ele pode *aparentar* ser não-determinístico. Isto é particularmente verdadeiro se o ambiente é complexo, tornando-se difícil controlar todos os aspectos não acessíveis. Assim, é sempre melhor pensar num ambiente como determinístico ou não-determinístico *do ponto de vista do agente*.
- **Episódico vs. Não-episódico –** Num ambiente episódico, a experiência do agente é dividida em "episódios". Cada episódio consiste da percepção e conseqüente ação do agente. A qualidade de sua ação depende apenas do próprio episódio, porque os episódios subseqüentes não dependem de que ações ocorram nos episódios anteriores. Ambientes episódicos são muito mais simples porque os agentes não precisam pensar adiante.

- **Estático vs. Dinâmico –** Se o ambiente pode mudar enquanto o agente estiver deliberando, dizemos então que o ambiente é dinâmico para o agente, do contrário ele é estático. Ambientes estáticos são fáceis de se tratar porque o agente não precisa ficar observando o mundo enquanto ele decide por uma ação, nem precisa se preocupar com a passagem do tempo. Se o ambiente não muda com o passar do tempo, mas a medida de performance do agente sim, dizemos que este ambiente é semi-dinâmico.
- **Discreto vs. Contínuo –** Se existe um número distinto, claramente definido, de percepções e ações, dizemos que o ambiente é discreto. O xadrez é discreto – existe um número fixo de movimentos possíveis a cada turno. Dirigir um táxi é contínuo – a velocidade e a localização do taxi e dos outros veículos variam dentro de uma faixa de valores contínuos[2].

Veremos que diferentes tipos de ambiente requerem programas agentes de certa forma diferentes para lidar eficientemente com eles. Assim, como você pode imaginar, o caso mais difícil é o de um ambiente inacessível, não-episódico, dinâmico e contínuo. Ele também nos mostrará que a maioria das situações reais são tão complexas que mesmo sendo realmente determinísticas, para propósitos práticos elas devem ser tratadas como não-determinísitcas.

Ambiente	Acessível	Determinístico	Episódico	Estático	Discreto
Xadrez com relógio	Sim	Sim	Não	Semi	Sim
Xadrez sem relógio	Sim	Sim	Não	Sim	Sim
Poker	Não	Não	Não	Sim	Sim
Gamão	Sim	Não	Não	Sim	Sim
Motorista de taxi	Não	Não	Não	Não	Não
Sistema de diagnóstico médico	Não	Não	Não	Não	Não
Sistema de análise de imagem	Sim	Sim	Sim	Semi	Não
Robô com garras	Não	Não	Sim	Não	Não
Controle de refinaria	Não	Não	Não	Não	Não
Tutor interativo de Inglês	Não	Não	Não	Não	Sim

Figura 2.13 Exemplos de ambientes e suas características

A figura 2.13 lista as propriedades de um número de ambientes familiares. Note que as respostas podem mudar dependendo de como você conceitualiza os agentes e ambientes. Por exemplo, poker é determinístico se o agente pode se manter informado da ordem das cartas na mesa, mas é não-determinístico caso contrário. Além disso, vários ambientes são episódicos a um nível mais alto do que as ações individuais do agente. Por exemplo, um torneio de xadrez consiste em uma seqüência de jogos; cada jogo é um episódio, pois a contribuição dos movimentos em um jogo para o desempenho geral do agente não é afetada pelos movimentos no próximo jogo. Por outro lado, movimentos dentro de um único jogo certamente interagem, de maneira que o agente precisa prever vários movimentos.

Programas ambientes

O programa ambiente genérico na figura 2.14 ilustra a relação básica entre agentes e ambientes. Neste livro, veremos que é conveniente para muitos dos exemplos e exercícios utilizar um simulador de ambientes que siga essa estrutura de programa. O simulador toma um ou mais agentes como entrada e se organiza para repetidamente dar a cada agente a percepção correta e para recebe de volta uma ação. O simulador então atualiza o ambiente baseado nas ações, e possivelmente outros processos dinâmicos no ambiente que não são considerados como agentes (a chuva, por exemplo). O ambiente é entretanto definido pelo estado inicial e pela função de transição. Certamente, um agente que trabalha em um

[2] Num bom nível de granularidade, mesmo o ambiente de dirigir um taxi é discreto, porque a imagem da câmera é digitalizada de forma que resulte num número discreto de pixels. Mas qualquer programa agente sensível teria de abstrair acima desse nível, até um nível de granularidade que seja contínuo.

simulador deve também trabalhar em um ambiente real que forneça o mesmo tipo de percepção e aceite os mesmos tipos de ações.

```
function EXECUTA-AMBIENTE(estado, ATUALIZA-FN, agentes, terminação)
 entradas: estado, o estado inicial do ambiente
           ATUALIZA-FN, função para modificar o ambiente
           Agentes, um conjunto de agentes
           Terminação, um predicado para testar quando

 for each agente in agentes do
  PERCEPÇÃO[agente] ← RECEBE-PERCEPÇÃO(agente,estado)
 end
 for each agente in agentes do
  AÇÃO[agente] ← PROGRAMA[agente](PERCEPÇÃO[agente])
 end
 regra ← CASA-REGRAS(estado,regras)

 return terminação(estado)
```

Figura 2.14 Programa simulador do ambiente básico. Ele fornece a cada agente sua percepção, recebe uma ação de cada agente e então atualiza o ambiente

O procedimento EXECUTA-AMBIENTE exercita corretamente os agentes em um ambiente. Para alguns tipos de agentes, tais como os que conseguem manter diálogos em linguagem natural, pode ser suficiente apenas observar seu comportamento. Para conseguir informações mais detalhada sobre o desempenho do agente, inserimos algum código medidor de desempenho. A função EXECUTA-AMBIENTE-DE-AVALIAÇÃO, mostrada na figura 2.15, faz isso, ela aplica uma medida de desempenho para cada agente e retorna uma lista de resultados obtidos. As variáveis de resultado mantêm informação sobre o resultado de cada agente.

```
function EXECUTA-AVALIAÇÃO-DE-AMBIENTE(estado,ATUALIZA-FN,agentes,terminação,
DESEMPENHO-FN) returns scores
 local variables: scores, um vetor do mesmo tamanho que agentes, todos 0

 repeat
  for each agente in agentes do
   PERCEPÇÃO[agente] ← RECEBE-PERCEPÇÃO(agente,estado)
  End
  for each agente in agentes do
   AÇÃO[agente] ← PROGRAMA[agente](PERCEPÇÃO[agente])
  End
  estado ← ATUALIZA-FN (ações,agentes,estado)
  scores ← DESEMPENHO-FN (scores,agentes,estado)
 until terminação(estado)

 return scores
```

Figura 2.15 Um programa simulador do ambiente que trata da medida de desempenho para cada agente

No geral, a medida de desempenho pode depender da seqüência inteira de estados do ambiente gerada durante a operação do programa. Normalmente, entretanto, a medida de desempenho trabalha por acumulação simples, utilizando ou a soma, a média ou tomando o máximo. Por exemplo, se a medida de desempenho para um agente limpador a vácuo é a quantidade total de sujeira limpada em uma execução, *resultados* vai simplesmente manter informação sobre a sujeira já limpa até então.

O ambiente de avaliação do programa retorna a medida de desempenho para um ambiente simples, definido por um estado inicial simples e uma função de atualização particular. Usualmente, um agente é designado para trabalhar em uma **classe ambiente**, todo um diferente conjunto de ambientes. Por exemplo, designamos um programa de xadrez para jogar contra qualquer um de uma vasta coleção de oponentes humanos e máquinas. Se o designássemos para um oponente simples, poderíamos estar aptos a tirar proveito de uma fraqueza específica daquele oponente, mas isto não iria resultar em um bom programa para partidas contra adversários em geral. Falando estritamente, de maneira a medir o desempenho de um agente, precisamos ter um ambiente gerador que selecione ambientes particulares (com certas semelhanças) no qual devemos executar o agente. Estamos então interessados no desempenho médio do agente sobre a classe ambiente. Isto é fácil de ser implementado em um ambiente simulado, e os exercícios 2.5 ao 2.11 levará você através do total desenvolvimento de um ambiente e o processo de medição associado.

Uma possível confusão surge entre a variável de estado no ambiente simulador e a variável de estado no próprio agente (veja *agente reflexo com estado*). Assim como um programador implementando tanto o ambiente simulador como o agente, é tentador permitir ao agente dar uma espiada na variável de estado do ambiente simulador. Deve-se resistir a esta tentação a todo custo! A versão do estado do agente deve ser construída somente pela sua própria percepção, sem nenhum acesso à informação completa do estado.

2.5 RESUMO

Este capítulo é algo parecido a um passeio pelo redemoinho da IA, a qual conceituamo-la como sendo a ciência que projeta agentes. Os pontos principais a serem lembrados são os seguintes:

- Um **agente** é algo que percebe e age em um ambiente. Dividimos um agente em uma arquitetura e um programa agente.
- Um **agente ideal** é aquele que sempre toma a ação que é esperada para maximizar sua mediada de desempenho, dada a seqüência de compreensão que ele tem visto até então.
- Um agente é **autônomo** na extensão de que sua ações de escolha dependam da sua própria experiência, mais do que do conhecimento do ambiente que foi embutido pelo programador.
- Um **programa agente** mapeia de uma percepção para uma ação, enquanto atualiza um estado interno.
- Existe uma variedade de programas agentes básicos, dependendo do tipo de informação produzida explicitamente e usada no processo de decisão. Tais projetos variam em eficiência, compacidade e flexibilidade. O projeto apropriado de um programa agente depende das percepções, ações, metas e ambiente.
- **Agentes reflexos** respondem imediatamente às percepções, **agentes baseados em objetivo** agem de modo a alcançarem as suas metas, e **agentes baseados em utilidade** tentam maximizar seu próprio grau de satisfação.
- O processo de tomada de decisões pelo raciocínio com conhecimento é central à IA e ao projeto bem sucedidos de agentes. Isto significa que a representação do conhecimento é importante.
- Alguns ambientes demandam mais que outros. Ambientes que são inacessíveis, não-determinísticos, não-episódicos, dinâmicos e contínuos são os mais desafiantes.

NOTAS BIBLIOGRÁFICAS E HISTÓRICAS

A análise de agente racional como um mapeamento das seqüências de percepção para ações enraizam-se ultimamente nos esforços para identificar comportamentos racionais na área de economia e outras formas de raciocínio sob incerteza (citadas nos capítulos subseqüentes) e dos esforços dos psicológicos comportamentalistas como Skinner (1953) para reduzir a psicologia de organismos estritamente para mapeamento entrada/sadia ou estimulo/resposta. O avanço do comportamentalismo para o funcionalismo na psicologia, que foi pelo menos em parte dirigida pela aplicação da metáfora computacional de agentes (Putnam, 1960; Lewis, 1966), introduziu o estado interno de agentes nesse âmbito de estudo. O filosofo Daniel Dennett (1969; 1978b) ajudou a sintetizar esses pontos de vista em uma instância intencional para agentes. Um perspectiva abstrata de alto nível sobre agência é também tomada dentro do mundo da IA em (McCarthy e Hayes, 1969). Jon Doyle (1983) propôs que os projetos de agentes racionais são o núcleo da IA e permaneceriam como sua missão, enquanto os outros tópicos da IA se desenrolariam em novas disciplinas. Horvitz et al. (1988) sugere especificamente o uso de racionalidade concebida como a maximização de utilidade esperada como uma base para IA.

O pesquisador de IA e ganhador do prêmio Nobel, o economista Herbert Simon, esboçou uma clara distinção entre racionalidade sob limitações de recursoa (racionalidade procedural) e racionalidade como escolha racional objetiva (racionalidade substantiva) (Simon, 1958). Cherniak (1986) explora o nível mínimo de racionalidade necessária para qualificar uma entidade como um agente. Russell e Wefald (1991) lidam explicitamente com a possibilidade de utilizar uma variedade de arquiteturas de agentes. *Dung Beetle Ecology* (Hanski e Cambefort, 1991) fornece uma gama de interessante informações sobre o comportamento do besouro *rola-bosta*.

REFERÊNCIAS BIBLIOGRÁFICAS

Cherniak (1986)

Dennete (1969)

Dennete (1978b)

Hanski e Cambefort (1991)

Horvitz et al. (1988)

Jon Doyle (1983)

Lewis (1966)

Maes et al. (1994)

McCarthy e Hayes (1969)

Putnam (1960)

Russell e Wefald (1991)

Simon (1958)

Skinner (1953)