

Resumo

Artificial Intelligence – A Modern Approach, Russel and Norvig

Capítulos 1 e 2

Disciplina de Sistemas Multi-agentes

Manoel Campos da Silva Filho

O que é IA?

Definições de inteligência artificial são mostradas na figura 1.1, de acordo com oito livros texto. Estas definições variam ao longo de duas principais dimensões. Grosseiramente, as no topo são preocupadas com *processos de pensamento e raciocínio*, enquanto as de baixo endereçam a comportamento. As definições à esquerda medem sucesso em termos de fidelidade para a performance *humana*, enquanto as da direita medem contra um conceito *ideal* de inteligência, que chamaremos de **racionalidade**. Um sistema é racional se ele faz a “coisa certa”, dado o que ele sabe.

Sistemas que pensam como humanos	Sistemas que pensam racionalmente
“O excitante novo esforço para fazer computadores pensarem ... máquinas com mentes, no sentido literal e completo.” (Haugeland, 1985)	“O estudo de faculdades mentais por meio do uso de modelos computacionais.” (Charniak e McDermott, 1985)
“[A automação de] atividades que nós associamos com pensamento humano, atividades tais como tomada de decisão, resolução de problemas, aprendizagem...” (Bellman, 1978)	“O estudo de computações que fazem possível perceber, raciocinar e agir.” (Winston, 1992)
Sistemas que agem como humanos	Sistemas que agem racionalmente
“A arte de criar máquinas que realizam funções que requerem inteligência quando realizadas por pessoas.” (Kurzweil, 1990)	“Inteligência Computacional é o estudo do projeto de agentes inteligentes.” (Poole et al., 1998)
“O estudo de como fazer computadores realizarem coisas em que, no momento, pessoas são melhores.” (Rick e Knight, 1991)	“IA ...é preocupada comportamento inteligente em artefatos.” (Nilsson, 1998)
Figura 1.1 - Algumas definições de inteligência artificial, organizadas em quatro categorias	

Historicamente todas as quatro propostas têm sido seguidas. Como alguém pode esperar, uma tensão existe entre propostas centradas ao redor de humanos e propostas centradas ao redor de racionalidade (não estamos dizendo que humanos são irracionais no sentido de “emocionalmente instáveis” ou “insanos”, apenas que não somos perfeitos). Uma proposta centrada em humanos deve ser uma ciência empírica, envolvendo hipóteses e confirmações experimentais. Uma proposta racionalista envolve uma combinação de matemática e engenharia.

Agir humanamente: A proposta do Teste de Turing

O Teste de Turing, proposto por Alan Turing (1950), foi projetado para prover uma definição satisfatória de inteligência. Um computador passa no teste se um humano interrogador, depois de colocar algumas questões escritas, não puder dizer se as respostas escritas são de uma pessoa ou não. Um computador, para passar no teste, precisa ter as seguintes capacidades:

- **representação de conhecimento** para armazenar o que ele sabe ou ouve;
- **raciocínio automatizado** para usar a informação armazenada para responder questões e para tirar novas conclusões;
- **aprendizado de máquina** para se adaptar às novas circunstâncias e para detectar e extrapolar padrões.

O chamado **Teste Total de Turing** requer do computador outras capacidades como:

- **visão computacional** para perceber objetos, e
- **robótica** para manipular objetos.

Pensar humanamente: A proposta de modelagem cognitiva

Se formos dizer que um programa pensa como um humano, precisamos de alguma forma de determinar como humanos pensam. Precisamos estar por dentro do funcionamento da mente humana. Existem duas formas de fazer isto: através de introspecção – tentando pegar nossos próprios pensamentos a medida que eles surgem – e através de experimentos psicológicos. Uma vez que temos uma teoria precisa o suficiente sobre a mente, se torna possível expressar essa teoria como um programa de computador. Se as entradas e saídas do programa e o tempo dos comportamentos correspondem com comportamentos humanos, é evidente que alguns mecanismos de programas podem também estar operando em humanos. Por exemplo, Allen Newell e Herbert Simon, que desenvolveram o GPS, o General Problem Solver (1961) não estavam contentes que seu programa resolvesse problemas corretamente, eles queriam comparar os passos do seu raciocínio com os de como um humano resolve o mesmo problema.

Pensar racionalmente: A proposta das “leis do pensamento”

O filósofo grego Aristóteles foi um dos primeiros a tentar codificar o “pensamento correto”, que é, irrefutavelmente, o processo de raciocínio. Seu **silogismo** provê padrões para estruturas de argumentos que sempre produzem conclusões corretas quando dadas corretas premissas – por exemplo, “Sócrates é um homem”; todos os homens são mortais; logo, Sócrates é mortal”. Estas leis de pensamento eram supostas por governar a operação da mente; seu estudo iniciou o campo chamado **lógica**.

Agir racionalmente: A proposta de agente racional

Um **agente** é apenas algo que age (agente vem do latim *agere*, fazer). Mas agentes de computador são esperados por terem outros atributos que distinguem eles de meros programas, tal como operando sobre controle autônomo, percebendo seu ambiente, persistindo sobre um prologando período de tempo, se adaptando às mudanças e sendo capaz de pensar em outros

objetivos. Um **agente racional** é um que age de forma a alcançar o melhor resultado ou, quando em incerteza, o melhor resultado esperado.

Na proposta das “leis do pensamento a ênfase foi em inferências corretas. Fazendo corretas inferências é algumas vezes parte de ser um agente racional, devido uma forma de agir racionalmente é raciocinar logicamente para a conclusão que uma dada ação irá alcançar um objetivo e então agir na conclusão. Por outro lado, inferências corretas não são tudo de racionalidade, devido existirem frequentemente ações onde não há uma coisa provavelmente correta a fazer, mas alguma coisa deve ser feita. Existem também formas de agir racionalmente que não podem envolver inferência. Por exemplo, se afastar de um forno quente é uma ação de reflexo que tem normalmente mais sucesso que uma ação mais lenta tomada depois de cuidadosa deliberação.

As Fundações da Inteligência Artificial

A IA se fundamenta em diversas áreas do conhecimento, como mostrado a seguir.

Neurociência

Chips de computador podem executar instruções em nanossegundos, enquanto neurônios são milhões de vezes mais lentos. A lei de Moore diz que o número de transistores por polegada quadrada dobra a cada 1 ou 1,5 ano. A capacidade do cérebro humano dobra grosseiramente a cada 2 a 4 milhões de anos.

Filosofia

Kenneth Craik especificou os três passos chaves de um agente baseado em conhecimento (DEFINIR AGENTE BASEADO EM CONHECIMENTO): 1) o estímulo deve ser traduzido em representações internas, 2) a representação é manipulada pelo processo cognitivo (DEFINIR COGNITIVO) para derivar novas representações internas, e 3) estes são por sua vez, retraduzidos de volta em ação. Ele claramente explicou por que isso era um bom projeto para um agente:

Se o organismo transporta um “modelo de pequena escala” de realidade externa e de suas próprias possíveis ações dentro de sua cabeça, ele é capaz de tentar várias alternativas, concluir qual é a melhor delas, reagir a situações futuras antes que elas surjam, utilizar o conhecimento de eventos passados tratando com o presente e o futuro, e em cada forma de reagir de uma muito mais completa, segura e mais competente maneira à emergências com as quais ele encara.

Engenharia de Computação

O primeiro computador eletrônico, o ABC, foi montado por John Atanasoff e seu estudante Clifford Berry entre 1940 e 1942 na universidade estadual de Iowa. Sua pesquisa recebeu pouco suporte e reconhecimento; foi o ENIAC, desenvolvido como parte de um projeto militar secreto na universidade de Pennsylvania pelo time incluindo John Mauchly e John Eckert, que provou ser o mais influente precursor dos computadores modernos.

Na metade do século, desde então, cada geração de hardware de computador tem trazido um aumento na velocidade e capacidade e reduzido no preço. Performance dobra a cada 18 meses ou mais, com uma década ou duas nesta taxa de crescimento. Depois disto, nós precisaremos de engenharia molecular ou alguma outra nova tecnologia.

A máquina analítica de Charles Babbage (1792-1871) foi de longe a mais ambiciosa: ele incluía memória endereçável, programas armazenados e pulos condicionais, e foi a primeira capaz de realizar computação universal. A colega de Babbage, Ada Lovelace, foi talvez a primeira programadora do mundo (a linguagem de programação Ada foi nomeada depois dela).

A IA tem pioneirizado muitos trabalhos na área de ciência da computação, como time sharing, interpretadores interativos, computadores pessoais com janelas e mouse, ambientes de desenvolvimento rápido de aplicações (Rapid Application Development Tools – RAD Tools), tipos de dados de listas encadeadas, gerenciamento automático de armazenamento e conceitos-chaves de programação simbólica, funcional, dinâmica e orientada a objetos.

Teoria do controle e cibernética

Linguística moderna e IA, “nasceram” ao mesmo tempo, e cresceram juntas, tendo intersecção em um campo híbrido chamado **linguística computacional** ou **processamento de linguagem natural**. O problema de se entender linguagem em breve se tornou consideravelmente mais complexo do que parecia ser em 1957. Entender linguagem requer um entendimento do assunto e contexto, não apenas entender da estrutura das sentenças. Isto pode parecer óbvio, mas não era amplamente apreciado até a década de 1960. Muito dos trabalhos anteriores em representação do conhecimento (o estudo de como colocar o conhecimento em uma forma que o computador possa raciocinar sobre ele) foi ligado a linguagem e informado pesquisas em linguística, que foi conectado por sua vez, a décadas de trabalho de análise filosófica de linguagem.

A história da Inteligência Artificial

A gestação da inteligência artificial

O primeiro trabalho que é hoje geralmente reconhecido como IA foi feito por Warren McCulloch e Walter Pitts (1943). Eles usaram 3 fontes: conhecimento de filosofia básica e função dos neurônios no cérebro; uma análise formal da lógica proposicional devido a Russel e Whitehead; e a teoria da computação de Turing. Eles propõem um modelo de neurônios artificiais em que cada neurônio é caracterizado como estando “ligado” ou “desligado”, com a mudança para “ligado” ocorrendo em resposta a estimulação por um suficiente número de neurônios vizinhos. O estado do neurônio foi concebido como “realmente equivalente a proposição que propõe seu adequado estímulo”. Eles mostraram, por exemplo, que qualquer função computável poderia ser computada pela mesma rede de neurônios conectados, e que todos os conectivos lógicos (and, or, not, etc) poderiam ser implementados por simples estruturas de rede. Eles também sugeriram que redes definidas adequadamente também poderiam aprender.

Dois alunos de graduação no departamento de matemática de Princeton, Marvin Minsky e Dean Edmonds, construíram a primeira rede neural de computador em 1951. O SNARC, como foi chamada, usou 3000 tubos de vácuo para simular uma rede de 40 neurônios.

Existiram outros trabalhos anteriores que podem ser caracterizados como IA, mas foi Alan Turing quem primeiro articulou uma visão completa de IA em seu artigo “Computação de Máquina e Inteligência” em 1950. Lá ele introduziu o teste de Turing, aprendizado de máquina, algoritmos genéticos e reforço da aprendizagem.

O nascimento da IA (1956)

Princeton foi a casa de outra figura influente em IA, John McCarthy. Depois da graduação foi para o Dartmouth College, que se tornou o local oficial de nascimento da área de IA.

Entusiasmo anterior, grandes expectativas (1952-1969)

Sucessos anteriores de Newell e Herbert Simon foram seguidos do General Problem Solver ou GPS. Diferente da Lógica Teorista, este programa foi projetado no início para imitar os protocolos humanos de resolução de problemas. Dentro de uma limitada classe de jogos/quebra-cabeças que ele podia manipular, ele revelou que a ordem em que o programa considerava sub-objetivos e possíveis ações era semelhante àquela em que o homem abordava os mesmos problemas. Assim, o programa GPS foi provavelmente o primeiro a incorporar a proposta do “pensar humanamente”. O sucesso do GPS e programas subsequentes como modelos de cognição guiaram Newell e Simon (1976) para formular a famosa hipótese do **sistema de símbolos físicos**, que afirma que “um sistema de símbolo físico tem os meios necessários e suficientes para ações inteligentes gerais”. O que eles quiseram dizer é que qualquer sistema (homem ou máquina) exibindo inteligência devem operar pela manipulação de estruturas de dados compostas de símbolos.

John McCarthy foi de Dartmouth para o MIT e lá fez 3 cruciais contribuições em um ano histórico de 1958: definiu a linguagem de alto nível chamada Lisp, que se tornou a linguagem dominante para programação de IA. Lisp é a segunda mais velha linguagem de alto nível em uso, um ano mais nova que FORTRAN. Com Lisp, McCarthy tinha a ferramenta que precisava, mas acesso escasso e caro a recursos computacionais era um sério problema. Em resposta, ele e outros no MIT inventaram o time sharing. Também em 1958 ele publicou o artigo “Programs with Common Sense” em que ele descreveu o Advice Taker (Tomador de Conselhos), um programa hipotético que pode ser visto como o primeiro sistema de IA completo. Como a Teoria Lógica e o Provedor de Teoremas Geométricos, o programa de McCarthy foi projetado para usar conhecimento para encontrar soluções para problemas. Mas diferente de outros, ele incorporou o conhecimento geral do mundo. Por exemplo, ele mostrou como alguns axiomas (DEFINIR AXIOMAS) simples poderiam permitir a programas gerarem um plano para dirigir até o aeroporto. O programa foi projetado também para que possa aceitar novos axiomas durante o curso normal de operação, permitindo que ele alcance competência em novas áreas sem ser reprogramado. O Advice Taker então incorpora os princípios centrais de representação do conhecimento e raciocínio: que é útil ter uma representação explícita e formal do mundo, e da forma que as ações dos agentes afetam o mundo e ser capaz de manipular essas representações com processo dedutivos.

Uma dose de realidade

Desde o início, pesquisadores de IA não eram tímidos em fazer previsões de seus próximos sucessos. Termos como “futuro visível” podem ser interpretados de várias formas mas Simon também fez uma previsão mais concreta: que em 10 anos um computador poderia ser campeão mundial de xadrez.

Uma típica história ocorrida no início dos esforços de tradução de textos por máquina, que foram generosamente financiados pelo Conselho Nacional de Pesquisa dos EUA na tentativa de agilizar a tradução de artigos científicos em russo, no fraco lançamento do Sputnik (nome do programa de lançamento de satélites artificiais da antiga URSS- União das Repúblicas Socialistas

Soviéticas) em 1957. Foi pensado inicialmente que seriam simples transformações sintáticas baseadas nas gramáticas russa e inglesa, e substituição de palavras usando um dicionário eletrônico, bastaria para preservar o exato significado das sentenças. O fato é que a tradução requer conhecimento geral do assunto para resolver ambiguidades e estabelecer o conteúdo da sentença.

O segundo tipo de dificuldade foi a não intratabilidade de problemas que a AI estava tentando resolver. Muitos dos primeiros programas de IA resolveram problemas tentando diferentes combinações de passos até a solução ser encontrada. Esta estratégia funcionou inicialmente porque micro mundos continham pouquíssimos objetos e portanto, pouquíssimas ações possíveis e curtíssimas sequências de solução. Antes da teoria de complexidade computacional ser desenvolvida, era amplamente pensado que resolver problemas maiores era simplesmente uma questão de hardware mais rápido e mais memória.

Sistemas baseados em conhecimento: A chave do poder? (1969-1979)

A figura da resolução de problemas que surgiu durante a primeira década de pesquisa em IA foi de um mecanismo de pesquisa para fins gerais tentando juntar passos elementares de raciocínio para encontrar soluções completas. Tais propostas foram chamadas de **métodos fracos**, pois, apesar de gerais, elas não funcionam para problemas de maior escala. A alternativa para os métodos fracos é usar conhecimento de domínio específico mais poderosos que permita maiores passos de raciocínio e possa mais facilmente manusear tipicamente casos de ocorrência em estreitas áreas de expertise. Alguém pode dizer que para resolver um problema difícil, você precisa quase já conhecer a resposta.

O programa DENDRAL foi um exemplo inicial desta proposta, que objetivava resolver o problema de inferência de estruturas moleculares a partir de informações providas por um espectrômetro de massa. Ele foi significativo pois foi o primeiro sistema de conhecimento intenso de sucesso: sua expertise derivava de um amplo número de regras de propósito especial. Sistemas posteriores também incorporaram o principal tema da proposta do programa Advice Taker de McCarthy – a clara separação do conhecimento (na forma de regras) do componente de raciocínio.

Com esta lição em mente, Feigenbaum e outros em Stanford iniciaram o Heuristic Programming Project (HPP), para investigar a onde a nova metodologia de **sistemas especialistas** poderia ser aplicada para outras áreas do conhecimento humano. O próximo maior esforço foi na área de diagnósticos médicos. Feigenbaum, Buchanan e o Dr Edward Shortliffe desenvolveram o MYCIN para diagnósticos de infecções sanguíneas. Com cerca de 450 regras, o MYCIN era capaz de trabalhar tão bem como um especialista, e consideravelmente melhor que um médico estagiário. Ele também continha duas maiores diferenças do DENDRAL. Primeiro, diferente das regras do DENDRAL, não existia nenhum modelo teórico geral a partir do qual as regras do MYCIN poderiam ser deduzidas. Elas tinham que ser adquiridas em extensas entrevistas com especialistas, que por sua vez adquiriam em livros, outros especialistas e experiência vividas. Segundo, as regras tinham que refletir as incertezas associadas com o conhecimento médico. O MYCIN incorporou um cálculo de incertezas chamado **fatores de certeza**, que pareciam (na época) se adequar bem na forma como os médicos avaliavam o impacto de evidências no diagnóstico.

A importância do domínio do conhecimento foi também evidente na área da compreensão de linguagem natural.

O crescimento generalizado de aplicações para problemas do mundo real causaram um aumento paralelo na demanda de exequíveis esquemas de representação do conhecimento. Um amplo número de diferentes representações e linguagens de raciocínio foram desenvolvidas. Algumas foram baseadas na lógica - por exemplo, a linguagem Prolog se tornou popular na Europa.

IA se torna um padrão de indústria (1980 até o presente)

Em toda parte, a indústria de IA estourou, de poucos milhões de dólares em 1980 para bilhões de dólares em 1988. Logo depois veio o período chamado “Inverno da IA”, em que muitas companhias sofreram por terem feito promessas extravagantes.

Embora a ciência da computação tenha amplamente abandonado o campo de redes neurais no final da década de 1970, o trabalho continuou em outros campos. Físicos como John Hopfield (1982) usou técnicas de máquinas estatísticas para analisar o armazenamento e propriedades de otimização de redes neurais, tratando coleções de nós como coleções de átomos. Psicólogos incluindo David Rumelhart e Geoff Hinton continuaram o estudo de modelos de memória de redes neurais. O real impulso veio em meados da década de 1980 quando pelo menos quatro diferentes grupos reinventaram o algoritmo de aprendizagem back-propagation (retro propagação) primeiro encontrado em 1969 por Bryson e Ho. O algoritmo foi aplicado para muitos problemas de aprendizado em ciência da computação e psicologia, e a generalizada disseminação dos resultados na coleção Parallel Distributed Processing (Rumelhart e McClelland, 1986) causou grande excitação.

AI se tornou uma ciência (1987 até o presente)

Usando metodologias melhoradas e frameworks teóricos, o campo chegou no entendimento de que redes neurais podem atualmente ser comparadas com correspondentes técnicas da estatística, reconhecimento de padrões, aprendizado de máquina e a mais promissora técnica pode ser aplicada para cada aplicação. Como resultado destes desenvolvimentos, a então chamada tecnologia de **data mining** tem dado novos vigorosos padrões de indústria.

O formalismo das **redes Bayesianas** foi inventado para permitir representação eficiente de um rigoroso raciocínio com conhecimento incerto. Esta proposta amplamente resultou muitos problemas de sistemas de raciocínio probabilístico de 1960 e 1970; ela agora domina as pesquisas de IA em raciocínio incerto e sistemas especialistas.

Resumo

- Diferentes pessoas pensam sobre IA diferentemente. Duas importantes questões para perguntar são: Você está preocupado com o conhecimento ou com o comportamento? Você quer modelar humanos ou trabalhar a partir de um padrão ideal?
- Neste livro, adotamos a visão de que inteligência se preocupa principalmente com ação racional. Idealmente, um agente inteligente executa a melhor ação possível em uma situação. Nós estudaremos o problema de construção de agentes que são inteligentes neste sentido.
- Filósofos (anteriores a 400 a.C.) conceberam IA considerando as idéias que a mente é de alguma forma como uma máquina, que ela opera sobre conhecimento codificado em alguma linguagem interna, e que o pensamento pode ser usado para escolher quais ações executar.
- Matemáticos proveram ferramentas para manipular declarações de certezas lógicas bem como incertezas e declarações probabilísticas. Eles também configuraram a base para compreender computação e raciocínio sobre algoritmos.
- Economistas formalizaram o problema de tomar decisões que maximizem os resultados esperados pelo administrador (decision-marker).

- Psicólogos adotaram a idéia de que humanos e animais podem ser considerados máquinas de processamento de informações. Linguistas mostraram qual uso de linguagem é adequado neste modelo.
- Engenheiros de computação proveram os artefatos que fazem as aplicações de IA possíveis. Programas de IA tendem a ser grandes, e eles podem não funcionar sem um grande avanço na velocidade e memória que a indústria de computadores tem provido.
- Teoria do controle trata com projeto de dispositivos que agem opcionalmente na base de feedbacks do ambiente. Inicialmente, as ferramentas matemáticas de teoria do controle eram muito diferentes da IA, mas os campos estão ficando mais próximos.
- A história da IA tem ciclos de sucesso, otimismo mal colocados, resultando cortes no entusiasmo e financiamento. Houveram também ciclos de introdução de novas propostas criativas e sistemático refinamento das melhores.
- IA tem avançado mais rapidamente na década passada devido ao grande uso de métodos científicos em experimentos e propostas de comparação.
- Progresso recente no entendimento da base teórica de inteligência tem sido disseminado, com melhorias nas capacidades reais do sistema. As sub-áreas da IA tem se tornado mais integradas, e a AI tem encontrado base comum com outras disciplinas.

2. Agentes Inteligentes

Agentes e Ambientes

Um **agente** é qualquer coisa que pode perceber seu ambiente através de **sensores** e agir nesse ambiente por meio de **atuadores**. Esta simples idéia é ilustrada na figura 2.1. Um agente humano tem olhos, orelhas e outros órgãos como sensores, e mãos, pernas, boca e outras partes do corpo como atuadores. Um agente robô pode ter câmeras e localizadores infravermelho como sensores, e vários motores como atuadores. Um agente de software recebe entradas do teclado, conteúdo de arquivos e pacotes de rede como sensores de entrada e age no ambiente mostrando resultados na tela, gravando em arquivos e enviando pacotes pela rede.

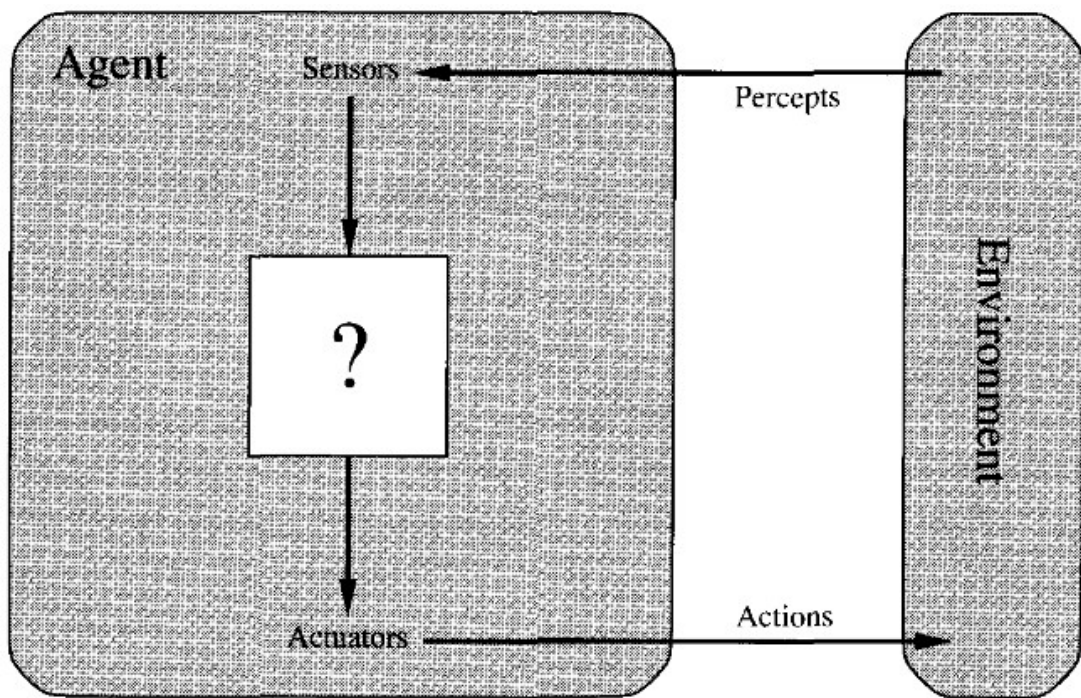


Figure 2.1 Agents interact with environments through sensors and actuators.

Em geral, a ação de escolha de um agente a qualquer instante pode depender de todo o histórico de percepções até o momento atual.

Matematicamente falando, nós dizemos que um comportamento de agente é descrito por uma **função de agente** que mapeia qualquer dada sequência de percepções para uma ação.

Podemos imaginar tabular a função de agente que descreve qualquer agente; para a maioria dos agentes, esta poderia ser uma tabela muito grande, até infinita, de fato, ao menos que coloquemos uma fronteira no tamanho das sequências de percepções que queremos considerar. Dado um agente como experimento, podemos, em princípio, construir sua tabela tentando todas as possíveis sequências de percepções e gravando cada ação que o agente faz em resposta. A tabela é, obviamente, uma caracterização externa do agente. Internamente, a função de agente para um agente artificial será implementada por um **programa de agente**. É importante distinguir essas duas idéias. A função de agente é uma descrição matemática abstrata; o programa do agente é uma implementação concreta, rodando na arquitetura do agente.

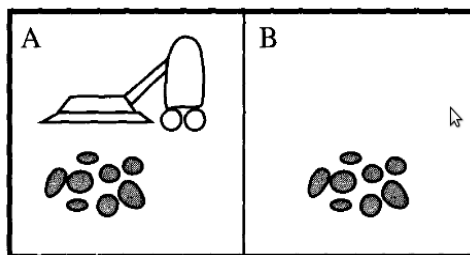


Figure 2.2 A vacuum-cleaner world with just two locations.

Para ilustrar essas idéias, usaremos um exemplo simples do mundo do aspirador de pó, mostrado na figura 2.2, que é bem simples e podemos descrever tudo que acontece nele. Ele tem apenas dois locais, quadrado A e quadrado B. O agente aspirador de pó percebe em que quadrado ele está e se há sujeira lá. Ele pode se mover para direita ou esquerda, aspirar a sujeira ou não fazer nada. Uma muito simples função de agente é: se o quadrado atual está sujo, aspire a sujeira, senão,

vá para o outro quadrado. Uma tabulação parcial desta função de agente é mostrada na figura 2.3. Um simples programa de agente para esta função de agente é dado posteriormente na figura 2.8.

Percept sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck
⋮	⋮
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

Bom comportamento: O conceito de racionalidade

Um **agente racional** é um que faz a coisa certa – conceitualmente falando, cada entrada na tabela para a função de agente é preenchida corretamente. Obviamente, fazer a coisa certa é melhor que fazer a errada, mas o que significa fazer a coisa certa? Como uma primeira aproximação, diremos que a ação correta é aquela que permite ao agente ter mais sucesso.

Medidas de performance

A **medida de performance** incorpora o critério de sucesso do comportamento de um agente. Quando um agente é atirado em um ambiente, ele gera uma sequência de ações de acordo com as percepções que ele recebe. Se a sequência é desejável, então o agente tem desempenhado seu trabalho bem. Obviamente, não existe nenhuma medida fixa adequada para todos os agentes.

Podemos propor como uma medida de performance, a quantidade de sujeira limpa em um único turno de 8 horas. Um agente racional pode maximizar esta medida de performance, limpando a sujeira e jogando ela no chão novamente, então limpando outra vez, e assim consecutivamente. Uma medida de performance mais adequada poderia premiar o agente por ter limpado o chão. Por exemplo, um ponto pode ser conseguido por cada quadrado limpo a cada passo (talvez com uma penalidade por consumo de energia e geração de ruído).

A seleção de uma medida de performance não é sempre fácil. Por exemplo, a noção de “chão limpo” no parágrafo anterior é baseada na média de limpeza sobre o tempo. Ainda que a mesma média de limpeza possa ser alcançada por dois diferentes agentes, um pode fazer um trabalho medíocre todo o tempo enquanto o outro limpa energeticamente mas faz longas paradas.

Racionalidade

O que é racional em um dado intervalo depende de 4 coisas:

- A medida de performance que define os critérios de sucesso
- O conhecimento prévio do agente sobre o ambiente
- As ações que o agente pode realizar
- A sequência de percepções do agente até o momento atual

Isto guia para uma **definição de agente racional**:

Para cada sequência de percepções possíveis, um agente racional deve selecionar uma ação que é esperada para maximizar suas medida de performance, dada a evidência provida pela sequência de percepções, e ignorar o conhecimento embutido que o agente tem.

Considerando o agente aspirador de pó, ele pode ser considerado racional, pois ele realiza as ações visando maximizar sua performance, a geografia do ambiente é conhecida, mas a sua distribuição da sujeira e localização inicial não, as ações de ir para esquerda ou direita são executadas de modo que o agente não saia dos dois quadrados, o agente percebe sua localização e se existe sujeira nela.

Alguém pode ver facilmente que um mesmo agente poderia ser irracional sobre diferentes circunstâncias. Por exemplo, uma vez que toda a sujeira é limpa, ele irá ficar se movendo desnecessariamente; se a medida de performance inclui uma penalidade de um ponto para cada movimento, o agente irá ter um desempenho baixíssimo. Um agente melhor para este caso poderia não fazer nada, uma vez que ele está certo de que todos os quadrados estão limpos. Se o quadrado pode se tornar sujo novamente, o agente poderia ocasionalmente checar e limpar novamente se necessário.

Onisciência, aprendizagem e autonomia

Precisamos ser cuidadosos em distinguir entre racionalidade e **onisciência**. Um agente onisciente conhece o resultado atual de suas ações e pode agir de acordo; mas onisciência é impossível na realidade. Considere o exemplo seguinte: Eu estou caminhando em direção à Champs Elysées e vejo um velho amigo cruzar a rua. Não há tráfego próximo e eu não estou ocupado, então, sendo racional, eu inicio a travessia da rua. Enquanto isso, a 10.000 metros de altitude, uma carga cai de um avião que estava passando, e antes que eu chegue ao outro lado da rua, sou achatado pela carga. Então, eu fui irracional ao atravessar a rua?

Este exemplo mostra que racionalidade não é o mesmo que perfeição. Racionalidade maximiza a performance esperada, enquanto perfeição maximiza a performance atual.

Nossa definição de racionalidade não requer onisciência, devido a escolha racional depender somente da sequência de percepções até o momento atual. Nós também asseguramos que não temos inadvertidamente permitido ao agente aderir em atividades decididamente não inteligentes. Por exemplo, se um agente não olha para ambos os lados antes de atravessar uma rua movimentada, então sua sequência de percepções não dirá a ele que há um grande caminhão se aproximando em alta velocidade. Nossa definição de racionalidade diz que está tudo OK para cruzar a rua? Longe disso. Primeiro, poderia não ser racional cruzar a rua dado essa sequência de percepções não informativas: o risco de acidente ao cruzar uma rua sem olhar para ambos os lados é muito grande. Segundo, um agente racional deveria escolher a ação de “olhar” antes de atravessar a rua, porque

olhar ajuda a maximizar a performance esperada. Fazendo ações para modificar percepções futuras – algumas vezes chamada de **obtenção de informações** – é uma importante parte da racionalidade. Um segundo exemplo de obtenção de informações é provido pela **exploração** que deve ser responsabilidade do agente aspirador de pó em um ambiente inicialmente desconhecido.

Nossa definição de agente racional não é somente no sentido de obter informações, mas também de aprender tanto quanto possível a partir de suas percepções. O agente pode ter algum conhecimento prévio do ambiente. Existem casos extremos em que o ambiente é totalmente conhecido a priori, nestes casos, o agente não precisa perceber ou aprender; ele simplesmente age corretamente. Obviamente, tais agentes são muito frágeis. Ver exemplo do besouro e da vespa na página 37.

A natureza dos ambientes

Especificando o ambiente de tarefas

Em nossas discussões de racionalidade do simples agente aspirador de pó, nós tivemos que especificar a medida de performance, o ambiente, os atuadores e sensores do agente. Iremos agrupar tudo isso sobre o título de **ambiente de tarefas**, onde definiremos a sigla PEAS (Performance, Environment, Actuators, Sensors, na primeira versão do livro era denominado PAGE – Percepts, Actions, Goals, Environment) como descrição do ambiente. Em projeto de agentes, o primeiro passo é sempre especificar o ambiente tão completo quanto possível.

O mundo do aspirador foi um exemplo muito simples; vamos considerar um problema mais complexo: um motorista automático de táxi. A figura 2.4 resume a descrição PEAS para o ambiente do táxi.

Tipo de Agente	Medidas de Performance (Performance Measure)	Ambiente (Environment)	Atuadores (Actuators)	Sensores (Sensors)
Táxi automatizado	Seguro, rápido, legal, viagem confortável, maximizar lucros	Ruas, tráfego, pedestre, clientes	Direção, acelerador, freio, setas, buzina, painel	Câmeras, sonar, velocímetro, GPS, odômetro, medidor de aceleração, sensores no motor, teclado

Figura 2.4 PEAS para o ambiente de tarefa de um táxi automatizado

Primeiro, qual é a **medida de performance** que gostaríamos que nosso táxi aspire? Qualidades desejáveis incluem pegar o destino correto; minimizar consumo de combustível e desgaste; minimizar tempo de viagem e/ou custo; minimizar violações de leis de trânsito e perturbações a outros motoristas; maximizar a segurança e conforto do passageiro; maximizar os lucros. Obviamente, alguns desses objetivos conflitam, logo, existirão trocas envolvidas.

Em seguida, qual é o ambiente que o táxi irá encarar? Qualquer motorista de táxi deve tratar com uma variedade de ruas, desde vielas rurais e becos urbanos a auto-estradas. As ruas contêm outros tráfegos, pedestres, animais deambulando, ruas em obras, carros de polícia, poças e buracos. O táxi deve também interagir com potenciais e atuais passageiros. Existem também algumas escolhas opcionais. O táxi pode precisar operar na Califórnia do Sul, onde neve é um problema raríssimo, ou no Alasca, onde constantemente é. Ele pode sempre dirigir pela direita, mas pode ser flexível para dirigir pela esquerda quando estiver no Japão ou Grã-Bretanha. Obviamente, quanto mais restrito o ambiente, mais fácil de projetar uma solução.

Os **atuadores** disponíveis em um táxi automatizado serão quase os mesmos dos disponíveis para um motorista humano: controle sobre o motor através do acelerador e controle sobre a direção e frenagem. Adicionalmente, ele necessitará de uma saída para uma tela ou sintetizador de voz para falar com os passageiros, e talvez alguma forma de se comunicar com outros veículos,

educadamente.

Para alcançar seus objetivos no ambiente, o táxi necessitará conhecer onde ele está, o que mais há na rua, e a velocidade que ele está indo. Seus **sensores** básicos podem então incluir uma ou mais câmeras de TV controláveis, um velocímetro e um odômetro. Ele pode ter um GPS para dar informações precisas de seu posicionamento e sensores de infra-vermelho ou sonares para detectar a distância de outros carros ou obstáculos. Finalmente, ele precisa de um microfone ou teclado para os passageiros requisitarem um destino.

Na figura 2.5 temos um esboço dos elementos PEAS básicos para diversos outros tipos de agentes. De fato, o que importa é a complexidade do relacionamento entre o comportamento do agente, a sequência de percepções geradas pelo ambiente e a medida de performance. Alguns ambientes reais são muito simples. Por exemplo, um robô projetado para inspecionar peças vindas por uma esteira rolante pode fazer algumas suposições simples: que as únicas coisas vindas pela esteira são peças que ele conhece e que existem apenas duas ações (aceitar ou rejeitar).

Tipo de Agente	Medidas de Performance (Performance Measure)	Ambiente (Environment)	Atuadores (Actuators)	Sensores (Sensors)
Sistema de diagnóstico médico	Saúde do paciente, minimizar custos, causas	Paciente, hospital, equipe médica	Mostrar questões, testes, diagnósticos, tratamentos, orientação	Entrada dos sintomas por teclado, respostas do paciente, pesquisa
Sistema de análise de imagem de satélite	Categorização correta da imagem	Downlink a partir do satélite em órbita	Mostrar categorização da cena	Arrays de pixels de cores
Robô coletor de partes	Percentual de partes na caixa correta	Esteira rolante com partes; caixas	Braço articulado e mão	Câmera, sensores de ângulo articulado
Controlador de refinaria	Maximizar pureza, produção, segurança	Refinaria, operadores	Válvulas, bombas de água, aquecedores, telas	Temperatura, pressão, sensores químicos
Tutor de inglês interativo	Maximizar a pontuação de estudantes em testes	Conjunto de estudantes, agência de testes	Mostrar exercícios, sugestões, correções	Entrada pelo teclado

Figura 2.5 Exemplos de tipos de agentes e suas descrições PEAS

Em contraste, alguns **agentes de software** (softwares robôs ou **sofbots**) existem em domínios ricos e ilimitados. Imagine um softbot projetado para voar em um simulador de voo. O simulador é muito detalhado, o ambiente complexo inclui outros aviões e operações em terra, e o agente de software deve escolher uma ação, a partir de uma ampla variedade de ações, em tempo real. Ou imagine um softbot projetado para scanear fontes de notícias na internet e mostrar itens de interesse para seus clientes. Para fazer isso bem, ele necessitará de habilidades de processamento de linguagem natural, aprender os interesses de cada cliente e mudar os planos dinamicamente – por exemplo, quando a conexão para um fonte de notícias cai ou quando outra se torna online.

Propriedades dos ambientes de tarefa

A faixa de ambientes que podem surgir em IA é obviamente vasta. Nós podemos, todavia, identificar um suficiente pequeno número de dimensões em que o ambiente pode ser categorizado.

- **Totalmente observável x parcialmente observável (acessível x inacessível):** se o sensor

do agente tem acesso completo ao estado do ambiente o tempo todo, conseguindo observar todos os aspectos relevantes para escolher uma ação a executar, assim, o ambiente é completamente observável, sendo que relevância depende das medidas de performance. Um ambiente pode ser parcialmente observável devido a ruído ou sensores não acurados ou parte do estado do ambiente que estejam faltando - por exemplo, o agente aspirador de pó com apenas um sensor de sujeira local não pode dizer se há sujeira em outros quadrados, e um táxi automatizado não pode saber o que outros motoristas estão pensando.

- **Determinístico x estocástico (determinístico x não determinístico):** Se o próximo estado do ambiente é completamente determinado pelo estado atual e ações executadas pelo agente, então podemos dizer que o ambiente é determinístico, ou seja, previsível; senão, é estocástico. Em princípio, um agente não precisa se preocupar sobre incertezas em um ambiente determinístico e completamente observável. Se o ambiente é parcialmente observável, todavia, ele pode parecer ser estocástico. Motorista de táxi é claramente estocástico, porque ele não pode predizer o comportamento do tráfego, além do mais, um pneu pode estourar. O mundo do aspirador de pó, como descrevemos, é determinístico, mas variações podem incluir elementos estocásticos como aparecimento randômico de sujeira e um mecanismo de sucção não confiável. Se o ambiente é determinístico, exceto pelas ações de outros agentes, dizemos que o ambiente é **estratégico**.
- **Episódico x sequencial (episódico x não episódico):** em um ambiente episódico, a experiência do agente é dividida em episódios atômicos. Cada episódio consiste da percepção do agente e realização de uma única ação. **Crucialmente, o próximo episódio não depende de ações realizadas em episódios anteriores.** Em um ambiente episódico, a escolha de uma ação depende somente do próprio episódio. Um agente para apontar peças defeituosas em uma linha de montagem baseia cada decisão somente na peça atual, sem se preocupar com decisões anteriores, e a decisão atual não afeta decisões futuras. **Em ambientes sequenciais, a decisão atual pode afetar todas as decisões futuras.** Xadrez e um motorista de táxi são sequenciais: em ambos os casos, ações de curto prazo podem ter consequências de longo prazo. **Ambientes episódicos são mais simples porque o agente não tem que pensar a frente, ou seja, não tem que pensar nas consequências futuras de suas ações, apenas nas consequências da ação atual.**
- **Estático x dinâmico:** Se o ambiente pode mudar enquanto o agente está deliberando, podemos dizer que ele é dinâmico, senão, é estático. Ambientes estáticos são mais fáceis de lidar pois o agente não precisa ficar observando o mundo enquanto está decidindo uma ação a executar, nem precisa se preocupar com a passagem do tempo. Ambientes dinâmicos, por outro lado, estão continuamente perguntando ao agente o que ele quer fazer; se ele não decidiu ainda, isto conta como se tivesse decidido por fazer nada. **Se o ambiente não muda com a passagem do tempo mas a performance do agente sim, então dizemos que ele é semi-dinâmico.** Dirigir um táxi é claramente dinâmico. Xadrez, quando jogado com um relógio, é semi-dinâmico. Jogos de palavras cruzadas são estáticos.
- **Discreto x contínuo:** Um ambiente discreto, como o jogo de xadrez, tem um conjunto finito de estados, além de um discreto(finito) conjunto de percepções e ações. Dirigir um táxi é um problema de estados, ações e tempo contínuos.
- **Agente único x multi agente:** Um jogo de palavras cruzadas é claramente de apenas um agente, já o xadrez é um ambiente de 2 agentes. Xadrez é um ambiente multi agente competitivo. No ambiente do táxi, evitar colisões maximiza a performance de todos os agentes, logo ele é um ambiente multi agente parcialmente cooperativo, além de parcialmente competitivo, por exemplo, competindo por vagas de estacionamento e passageiros.

Como podemos esperar, o mais difícil caso é parcialmente observável, estocástico,

sequencial, dinâmico, contínuo e multi agente.

A figura 2.6 lista algumas propriedades de diversos ambientes. Algumas podem variar de acordo com a forma que o ambiente foi definido e a especificidade com a qual deseja-se implementar a solução. Por exemplo, muitos ambientes são episódicos em um nível mais alto do que as ações individuais do agente, como em um torneio de xadrez, que consiste de uma sequência de jogos, onde cada jogo é um episódio, assim, a contribuição de um movimento em um jogo para a performance geral do agente não é afetada pelos movimentos no jogo anterior. Por outro lado, decisões feitas em um único jogo são certamente sequenciais.

Ambiente	Observável	Determinístico	Episódico	Estático	Discreto	Agentes
Palavras cruzadas	Totalmente	Determinístico	Sequencial	Estático	Discreto	Single
Xadrez com relógio	Totalmente	Estratégico	Sequencial	Semi-Dinâmico	Discreto	Multi
Dirigir táxi	Parcialmente	Estocástico	Sequencial	Dinâmico	Contínuo	Multi
Diagnóstico médico	Parcialmente	Estocástico	Sequencial	Dinâmico	Contínuo	Single
Análise de imagens	Totalmente	Determinístico	Episódico	Semi-Dinâmico	Contínuo	Single
Robô coletor de partes	Parcialmente	Estocástico	Episódico	Dinâmico	Contínuo	Single
Controlador de Refinaria	Parcialmente	Estocástico	Sequencial	Dinâmico	Contínuo	Single
Tutor de inglês interativo	Parcialmente	Estocástico	Sequencial	Dinâmico	Discreto	Multi

Figura 2.6 Exemplos de ambientes e suas características

A estrutura de agentes

O trabalho da IA é projetar o **programa do agente** que implementa a função de agente mapeando percepções para ações. Nós assumimos que este programa rodará em algum tipo de dispositivo computacional com sensores físicos e atuadores, o que chamamos de **arquitetura**:
 agente = arquitetura + programa.

Obviamente, o programa que escolhermos deve ser apropriado para a arquitetura, que pode ser um PC comum ou um carro robô com muitas cpu's, câmeras e outros sensores. Em geral, a arquitetura faz as percepções para o programa a partir dos sensores, roda o programa e alimenta as escolhas de ações do mesmo para os atuadores.

Programas de agente

O programa do agente que projetaremos terá o mesmo skeleton: ele pega as percepções atuais como entradas a partir de sensores e retorna uma ação para os atuadores. Note a diferença entre o programa do agente, que toma a percepção atual como entrada, e a função de agente, que toma todo o histórico de percepções. O programa do agente toma apenas a percepção atual como entrada pois nada mais é disponibilizado pelo ambiente; se as ações do agente dependem de toda a sequência de percepções, ele deve armazenar as percepções.

Por exemplo, a figura 2.7 mostra um programa de agente realmente trivial, que fica de olho na sequência de percepções e então as usa para indexar uma tabela de ações para decidir o que fazer. A tabela representa explicitamente a função de agente que o programa do agente incorpora. Para construir um agente racional desta forma, precisamos construir uma tabela que contenha as ações apropriadas para cada sequência de percepções possível.

```

function TABLE-DRIVEN-AGENT(percept) returns an action
  static: percepts, a sequence, initially empty
           table, a table of actions, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action

```

Figure 2.7 The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It keeps track of the percept sequence using its own private data structure.

É instrutivo considerar porque uma proposta baseada em tabelas para construção de agentes é condenada a falhar. Considere um táxi automático: a entrada visual a partir de uma única câmera vem na faixa de grosseiramente, 27Mbps (30 frames/s, 640x480 pixels com 24 bits de informação de cor). Isto dá uma tabela com cerca de $10^{250.000.000.000}$ entradas para uma hora de direção. Até uma tabela para xadrez poderia ter 10^{150} entradas. O número de átomos em um universo observável é menos que 10^{80} .

Apesar disto, um agente dirigido por tabelas faz o que queremos: implementa a função desejável de agente. O desafio chave para IA é encontrar como escrever programas que, para uma extensão possível, produzam comportamento racional a partir de uma quantidade pequena de código no lugar de um grande número de entradas de tabela.

```

function REFLEX-VACUUM-AGENT([location, status]) returns an action

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

```

Figure 2.8 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

Os tipos básicos de programas de agentes, que incorporam os princípios subjacentes de quase todos os sistemas inteligentes são:

- **Agentes reflexivos simples:** O tipo mais simples de agente, que selecionam ações com base na percepção atual, ignorando o resto do histórico de percepções. Por exemplo, o agente aspirador de pó, pois suas decisões são baseadas somente na localização atual e se lá há sujeira. Um programa para esse agente é mostrado na figura 2.8. Imagine você dirigindo um carro. Se a luz de freio do carro da frente acende, você nota isto e começa a frear. Em outras palavras, alguns processamentos são feitos em entradas visuais para estabelecer a condição que chamamos “O carro na frente está parando”. Então, isto dispara alguma conexão estabelecida no programa do agente para a ação “iniciar frenagem”. Nós chamamos tal conexão de **regra de condição-ação**, escrita como:
if car-in-front-is-braking **then** initiate-braking.

Humanos tem muitas conexões, algumas das quais são aprendidas (como dirigir) e outras

que são reflexos natos (como piscar). O programa da figura 2.8 é específico para um ambiente particular. Uma proposta mais genérica e flexível é primeiro construir um interpretador de propósito geral para regras de condição-ação e então criar um conjunto de regras para um ambiente específico. A figura 2.9 dá a estrutura desse programa geral em uma forma esquemática, mostrando como as regras de condição-ação permitem ao agente fazer conexões de percepções para ações. O programa do agente é mostrado na figura 2.10.

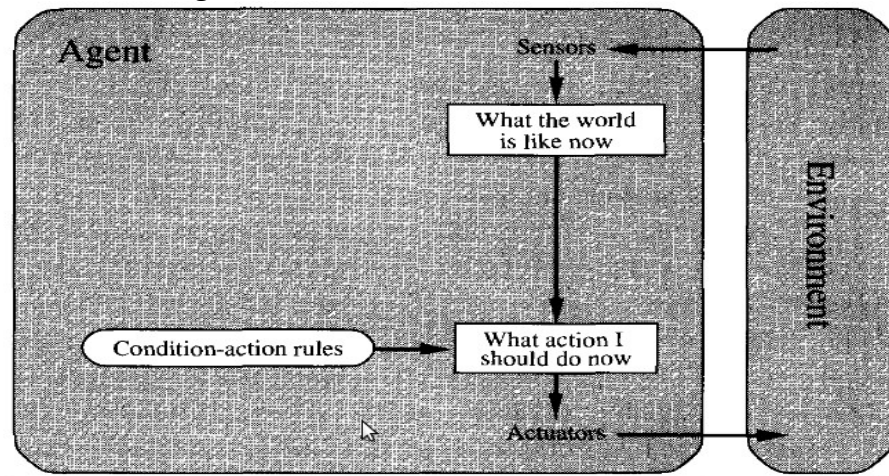


Figure 2.9 Schematic diagram of a simple reflex agent.

function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action

static: *rules*, a set of condition–action rules

state ← INTERPRET-INPUT(*percept*)

rule ← RULE-MATCH(*state*, *rules*)

action ← RULE-ACTION[*rule*]

return *action*

Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

- **Agentes reflexivos baseados em modelos:** A forma mais efetiva de gerenciar ambientes parcialmente observáveis é manter estados internos que depende do histórico de percepções, e com isso, tem-se informações de estado da área parcialmente observável do ambiente. Atualizar esta informação com o tempo requer que dois tipos de conhecimento sejam codificados no programa do agente. Primeiro, precisamos de informações de como o mundo evolui, independentemente do agente. Segundo, precisamos de algumas informações sobre como as ações do agente afetam o mundo. Este conhecimento do mundo é chamado de **modelo** do mundo. Um agente que usa tal modelo é chamado de agente baseado em modelo. A figura 2.11 dá a estrutura de um agente reflexivo com estado interno, mostrando como a percepção atual é combinado com um estado interno antigo para gerar a descrição atualizada do estado atual. O programa do agente é mostrado na figura 2.12.

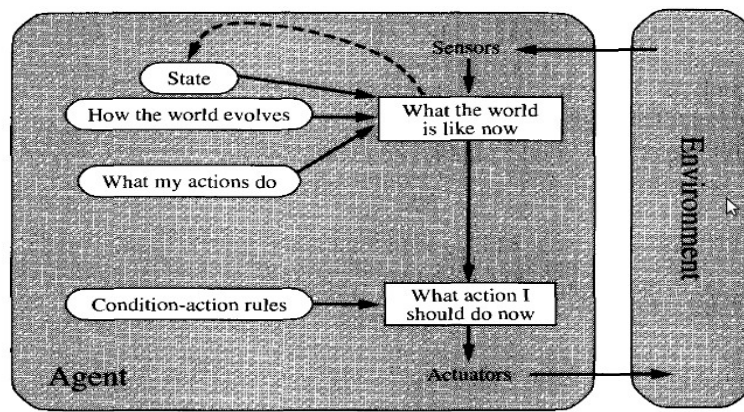


Figure 2.11 A model-based reflex agent.

function REFLEX-AGENT-WITH-STATE(*percept*) **returns** an action

static: *state*, a description of the current world state

rules, a set of condition–action rules

action, the most recent action, initially none

state ← UPDATE-STATE(*state*, *action*, *percept*)

rule ← RULE-MATCH(*state*, *rules*)

action ← RULE-ACTION[*rule*]

return *action*

Figure 2.12 A model-based reflex agent. It keeps track of the current state of the world using an internal model. It then chooses an action in the same way as the reflex agent.

- **Agentes baseados em objetivo:** Conhecer o estado atual do ambiente não é sempre suficiente para decidir o que fazer. Por exemplo, em uma junção de ruas, o táxi pode virar à esquerda, à direita ou ir em frente. A decisão correta depende de onde ele está tentando chegar, ou seja, além da descrição do estado atual, o agente precisa de algum tipo de informação de objetivo que descreva situações que são desejáveis – por exemplo, o destino do passageiro. O agente pode combinar isso com o resultado de possíveis ações para poder escolher ações que alcancem seus objetivos. A figura 2.13 mostra a estrutura deste tipo de agente.

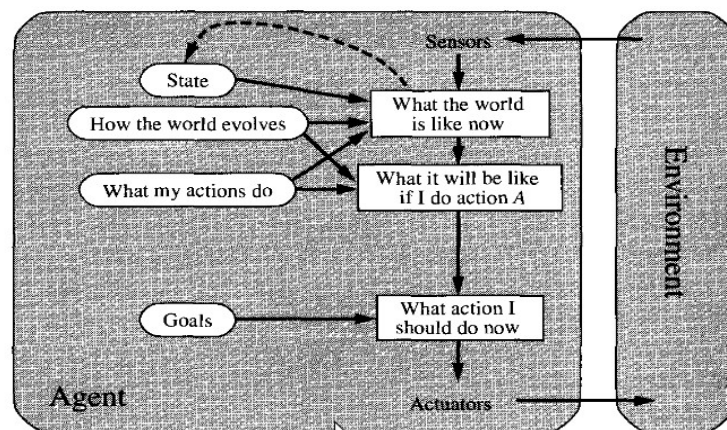


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Note que tomar decisões desse tipo é fundamentalmente diferente das regras de *condição-ação* descritas anteriormente, pois envolvem considerações sobre o futuro como “O que acontecerá se eu fizer isto e isto?” e “Isto me fará feliz?”. No projeto de agentes reflexivos, esta informação não é explicitamente representada, porque as regras embutidas

mapeiam diretamente percepções para ações.

Embora um agente baseado em objetivo pareça menos eficiente, ele é mais flexível pois o conhecimento que suporta suas decisões é representado explicitamente e pode ser modificado. Se começar a chover, o agente pode atualizar seu conhecimento de como efetivamente seu freio irá operar; isto automaticamente fará com que todos os comportamentos relevantes sejam alterados para se adequarem às novas condições. Para um agente reflexivo, por outro lado, nós podemos ter que reescrever muitas regras de **condição-ação**. O comportamento do agente baseado em objetivo pode facilmente ser alterado para ir a um local diferente. As regras do agente reflexivo para quando virar e quando ir em frente funcionam somente para um único destino; elas devem ser todas substituídas para ir a qualquer lugar novo.

- **Agentes baseados em utilidade (no sentido de ser útil):** Objetivos sozinhos não são realmente suficientes para gerar comportamento de alta qualidade na maioria dos ambientes. Por exemplo, existem muitas sequências de ações que levariam o táxi ao seu destino (desta forma, alcançando seu objetivo) mas algumas são mais rápidas, seguras e confiáveis, ou mais baratas que outras. Objetivos apenas proveem uma distinção binária crua entre os estados “feliz” e “infeliz”, enquanto uma medida de performance mais geral pode permitir uma comparação de diferentes estados do mundo de acordo com exatamente quão feliz eles podem fazer o agente e se eles podem ser alcançados. Devido “feliz” não soar muito científico, uma terminologia costumeira é dizer que um estado é preferível do que outro, então ele tem maior **utilidade** para o agente. A figura 2.14 mostra a estrutura deste tipo de agente.

A função de utilidade mapeia estados (ou uma sequência de estados) em números reais, que descrevem o associado grau de felicidade. Uma completa especificação da função de utilidade permite decisões racionais de dois tipos de casos onde os objetivos são inadequados. Primeiro, quando existem objetivos conflitantes, somente um dos quais pode ser alcançado (por exemplo, velocidade e segurança), a função de utilidade especifica a troca apropriada. Segundo, quando existem muitos objetivos que o agente pode mirar, e nenhum dos quais pode ser alcançado com certeza, a utilidade provê uma forma em que a probabilidade de sucesso pode ser ponderada contra a importância dos objetivos. Um agente que possui uma explícita função de utilidade pode tomar decisões racionais.

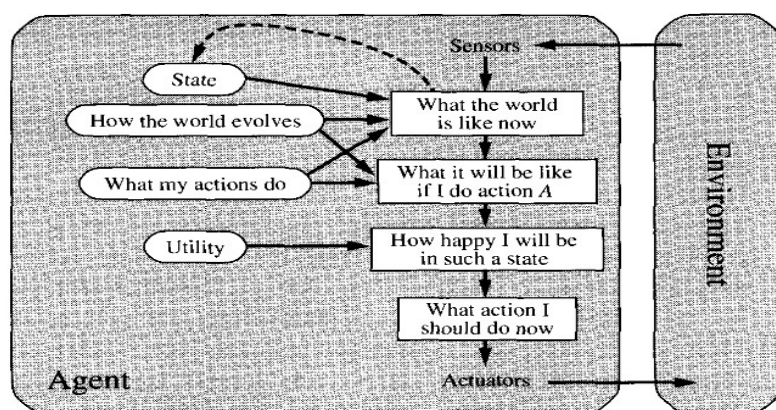


Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

Aprendizado de agentes

Em um famoso paper anterior, Turing (1950) considerou a idéia de programar sua máquina inteligente à mão. O método que ele propôs é para construir máquinas de aprendizagem e então ensiná-las. Em muitas áreas de IA, isto é atualmente o método preferido para criação de sistemas no estado da arte. Aprendizado tem outra vantagem: permite ao agente operar em um ambiente inicialmente desconhecido e se tornar mais competente que seu conhecimento inicial sozinho poderia permitir.

Um aprendizado de agente pode ser dividido em quatro componentes conceituais, como mostra a figura 2.15. A mais importante distinção é entre o **elemento de aprendizagem**, que é responsável por fazer melhorias, e o **elemento de performance**, que é responsável por selecionar ações externas. O elemento de performance é o que nós temos previamente considerado ser o agente inteiro: ele toma as percepções e decide em ações. O elemento de aprendizagem usa feedbacks a partir da **crítica** de como o agente está agindo e determina como o elemento de performance deve ser modificado para operar melhor no futuro.

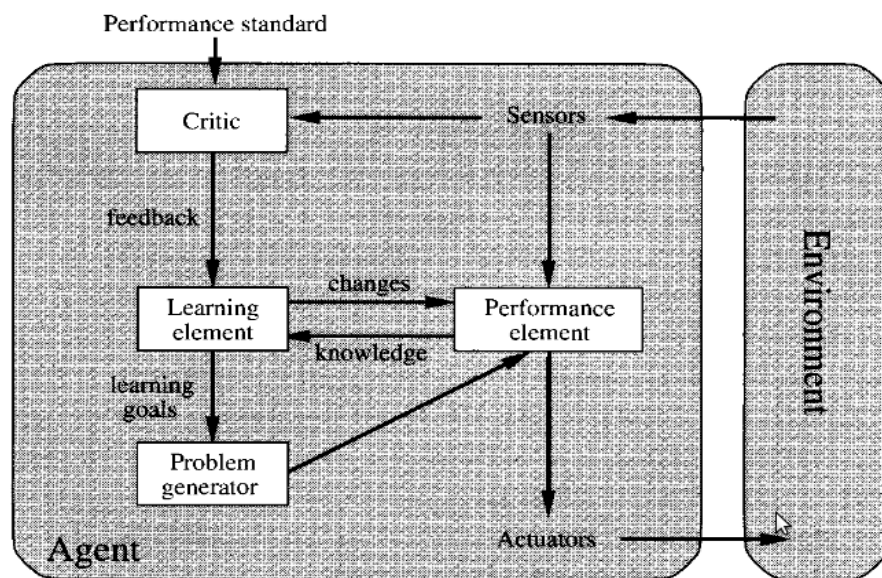


Figure 2.15 A general model of learning agents.

O projeto do elemento de aprendizagem depende muito do projeto do elemento de performance. Quando tentamos projetar um agente que aprende uma certa capacidade, a primeira questão não é “Como vou colocá-lo para aprender isto?” mas “Que tipo de elemento de performance meu agente necessitará para fazer isto uma vez e aprender como fazer?”. Dado um projeto de agente, mecanismos de aprendizagem podem ser construídos para melhorar cada parte do agente.

O elemento “**crítica**” diz ao elemento de aprendizagem quão bem o agente está operando com respeito a um padrão de performance fixo. A crítica é necessária porque as percepções por elas mesmas não proveem indicação do sucesso do agente.

O último componente do aprendizado do agente é o **gerador de problemas**. Ele é responsável por sugerir ações que irão guiar para novas e informativas experiências. O ponto é que, se o elemento de performance tem sua forma, ele pode continuar fazendo as ações que são melhores, dado o que ele conhece. Mas se o agente está desejando explorar um pouco, e fazer talvez algumas ações opcionais num curto prazo, ele pode descobrir muito melhores ações a longo prazo. O trabalho do gerador de problemas é sugerir estas ações exploratórias.

Para tornar todo o projeto mais concreto, vamos retornar ao exemplo do táxi automatizado. O elemento de performance consiste de ignorar coleções de conhecimento e procedimentos que o táxi tem para selecionar suas ações de direção. O táxi vai a rua e dirige, usando seu elemento de performance. A crítica observa o mundo e passa informações para o elemento de aprendizagem. Por exemplo, depois que o táxi faz uma virada rápida à esquerda entre três ruas de grande tráfego, a crítica observa a linguagem chocante dos outros motoristas. A partir desta experiência, o elemento de aprendizagem é capaz de formular uma regra dizendo que aquilo foi uma ação ruim, e o elemento de performance é modificado instalando a nova regra. O gerador de problemas pode identificar certas áreas de comportamento necessitando de melhoria e sugerir experimentos, tal como tentar parar em ruas de diferentes superfícies, sobre diferentes condições de tempo.

O elemento de aprendizagem pode fazer mudanças para qualquer componente de “conhecimento” mostrado no diagrama do agente (figuras 2.9, 2.11, 2.13 e 2.14). O caso mais simples envolve aprender diretamente a partir de sequências de percepções. Observações de pares de sucessivos estados do ambiente podem permitir que o agente aprenda “como o mundo evolui” e observações dos resultados de suas ações podem permitir ao agente aprender “o que minhas ações fazem”. Por exemplo, se um táxi exerce uma certa pressão de frenagem enquanto dirigindo em uma rua molhada, então ele irá em breve encontrar quanto de desaceleração ele irá alcançar.

Resumo

- Um **agente** é alguma coisa que percebe e age no ambiente. A **função de agente** para um agente especifica a ação tomada por um agente em resposta a qualquer sequência de percepções.
- A **medida de performance** avalia o comportamento do agente no ambiente. Um **agente racional** age para maximizar o valor esperado da medida de performance, dada a sequência de percepções que ele vê.
- Uma especificação de **ambiente** inclui uma medida de performance, o ambiente externo, os atuadores e os sensores. Projetando um agente, o primeiro passo deve sempre ser especificar o ambiente tão completamente quanto possível.
- Ambientes variam ao longo de várias dimensões significantes. Eles podem ser completamente ou parcialmente observáveis, determinísticos ou estocásticos, episódicos ou sequenciais, estáticos ou dinâmicos, discretos ou contínuos e de único agente ou multi agente.
- O **programa do agente** implementa a função do agente.
- **Agentes reflexivos simples** respondem diretamente a percepções, enquanto um **agente reflexivo baseado em modelo** mantém um estado interno para rastrear aspectos do mundo que não são evidentes na percepção atual. **Agentes baseados em objetivo** agem para alcançar seus objetivos, e **agentes baseados em utilidade** tentam maximizar suas próprias expectativas de “felicidade”.
- Todos os agentes podem melhorar suas performance através de **aprendizado**.