Agentes Inteligentes: Estrutura e Tipos

Compreendendo a Arquitetura e Classificações de Agentes em IA

Márcio Nicolau

2025-08-28

Table of contents

List of Figures

Introdução

No universo da Inteligência Artificial, o conceito de "agente inteligente" é fundamental. Um agente é tudo aquilo que pode ser visto como percebendo seu ambiente através de **sensores** e agindo sobre esse ambiente através de **atuadores** (Russell; Norvig, 2004, p. 34). Esta aula se aprofundará na estrutura e nos diferentes tipos de agentes inteligentes, explorando como eles interagem com o mundo e como suas "mecanismos de pensamento" são projetados. Entender a arquitetura e as classificações de agentes é essencial para construir sistemas de IA eficazes e adaptáveis.

Objetivo de Aprendizagem

Ao final desta aula, você será capaz de:

- Definir o conceito de agente inteligente e seus componentes.
- Descrever um problema de IA usando o modelo PEAS (Performance, Environment, Actuators, Sensors).
- Distinguir entre os diferentes tipos de programas de agentes inteligentes: reativos simples, reativos baseados em modelo, baseados em metas, baseados em utilidade e de aprendizado.
- Identificar a aplicabilidade de cada tipo de agente a diferentes problemas e ambientes.

O que é um Agente Inteligente?

Um **agente** é uma entidade que executa ações. Um **agente inteligente** é aquele que age de forma autônoma em seu ambiente para alcançar seus objetivos, otimizando seu desempenho (Russell; Norvig, 2004, p. 34). A inteligência de um agente reside na sua capacidade de escolher a melhor ação a ser executada a cada instante, dadas suas percepções e seu conhecimento.

A característica central de um agente inteligente é a **racionalidade**. Um agente racional é aquele que faz a coisa certa, ou seja, a ação que é esperada para maximizar sua medida de desempenho, dada a evidência fornecida pela sequência de percepções e qualquer conhecimento embutido que o agente possua (Russell; Norvig, 2004, p. 36).

Descrevendo Agentes e Ambientes: O Modelo PEAS

Para projetar um agente inteligente, é crucial ter uma descrição clara do ambiente em que ele operará e dos objetivos que ele deve alcançar. O modelo **PEAS** (Performance, Environment, Actuators, Sensors) fornece uma estrutura para essa descrição (Russell; Norvig, 2004, p. 37).

- **P (Performance - Medida de Desempenho):** Quais são os critérios para avaliar o sucesso do agente? O que o agente está tentando maximizar ou minimizar? (e.g., para um carro autônomo: segurança, tempo de viagem, conforto, consumo de combustível).
- **E (Environment - Ambiente):** Onde o agente opera? Quais são as características do mundo em que ele vive? (e.g., para um carro autônomo: ruas, rodovias, outros veículos, pedestres, sinais de trânsito, clima).
- **A (Actuators - Atuadores):** Quais são os meios pelos quais o agente pode agir sobre o ambiente? (e.g., para um carro autônomo: volante, acelerador, freio, pisca-pisca, buzina).

- **S (Sensors - Sensores):** Como o agente percebe o ambiente? Quais são as entradas que ele recebe? (e.g., para um carro autônomo: câmeras, radar, lidar, GPS, velocímetro, odômetro).

Um exemplo do modelo PEAS para um carro autônomo:

Componente	Descrição
Performance	Maximizar segurança (evitar acidentes), maximizar tempo de chegada, maximizar cumprimento das leis d
Environment	Ruas, rodovias, zonas de pedestres, outros veículos, pedestres, sinais de trânsito, semáforos, condições clim
Actuators	Volante, acelerador, freio, pisca-pisca, buzina, luzes, display de informações.
Sensors	Câmeras (imagens de vídeo), radar (distância/velocidade de outros carros), lidar (mapa 3D de objetos), so

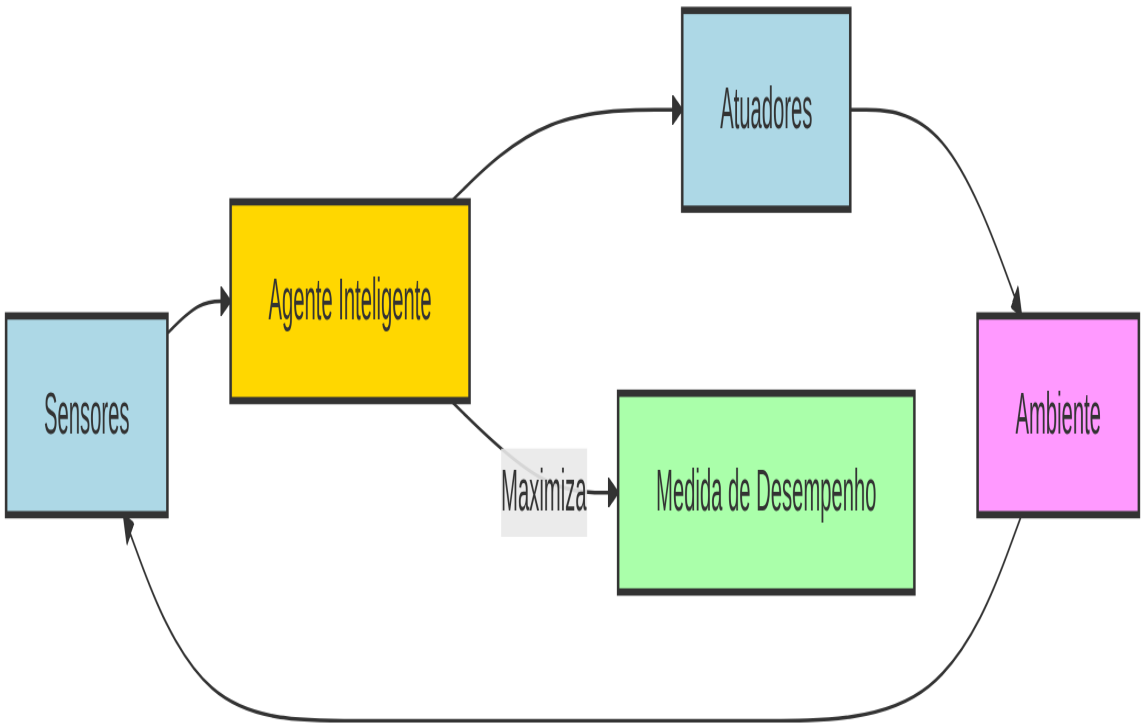

Figure 1: Diagrama do Modelo PEAS (Performance, Environment, Actuators, Sensors)

Estrutura de Agentes Inteligentes

Um agente inteligente é fundamentalmente composto por dois elementos principais: a **arquitetura** e o **programa do agente** (Russell; Norvig, 2004, p. 39).

- **Arquitetura:** É a plataforma física ou o ambiente de software onde o programa do agente é executado. Inclui os sensores e atuadores reais ou simulados, o hardware de computação, os sistemas operacionais, etc. É a "estrutura" do corpo do agente.
- **Programa do Agente:** É a implementação da função que mapeia a sequência de percepções recebidas pelos sensores para as ações a serem executadas pelos atuadores. Essa função define o comportamento do agente. Matematicamente, pode ser vista como: `f: Sequência de Percepções -> Ação`.

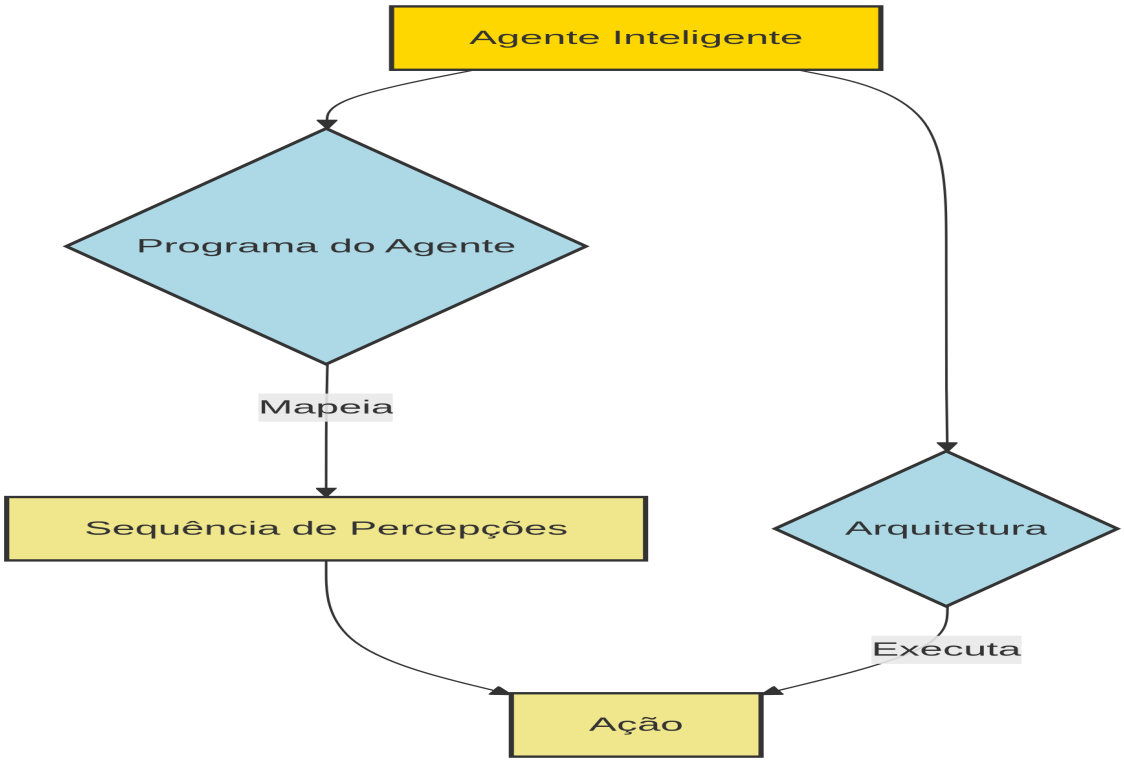

Figure 2: Estrutura de um Agente Inteligente

Nesta aula, nosso foco será principalmente nos diferentes tipos de **programas de agente**.

Tipos de Agentes Inteligentes (Programas de Agente)

Existem várias formas de estruturar o programa de um agente, dependendo da complexidade do ambiente, das informações disponíveis e dos objetivos do agente. (Russell; Norvig, 2004, p. 44) descrevem cinco tipos básicos:

Agentes Reativos Simples (Simple Reflex Agents)

Estes são os agentes mais básicos. Eles selecionam ações com base apenas na **percepção atual**, ignorando o histórico de percepções. Possuem um conjunto de regras de condição-ação, onde cada regra associa uma

percepção a uma ação.

- **Funcionamento:** `IF (condição) THEN (ação)`.
- **Vantagens:** Simples de implementar, rápidos.
- **Desvantagens:** Podem ser ineficazes em ambientes parcialmente observáveis ou que exigem memória para tomar decisões ótimas. Podem entrar em loops infinitos se não houver aleatoriedade ou mecanismos de interrupção.
- **Exemplo:** Um termostato que liga o aquecimento `SE` a temperatura estiver abaixo de 20°C e desliga `SE` estiver acima. Um aspirador de pó simples que, `SE` encontrar sujeira, a aspira; `SENÃO`, se estiver em um canto, muda de direção.

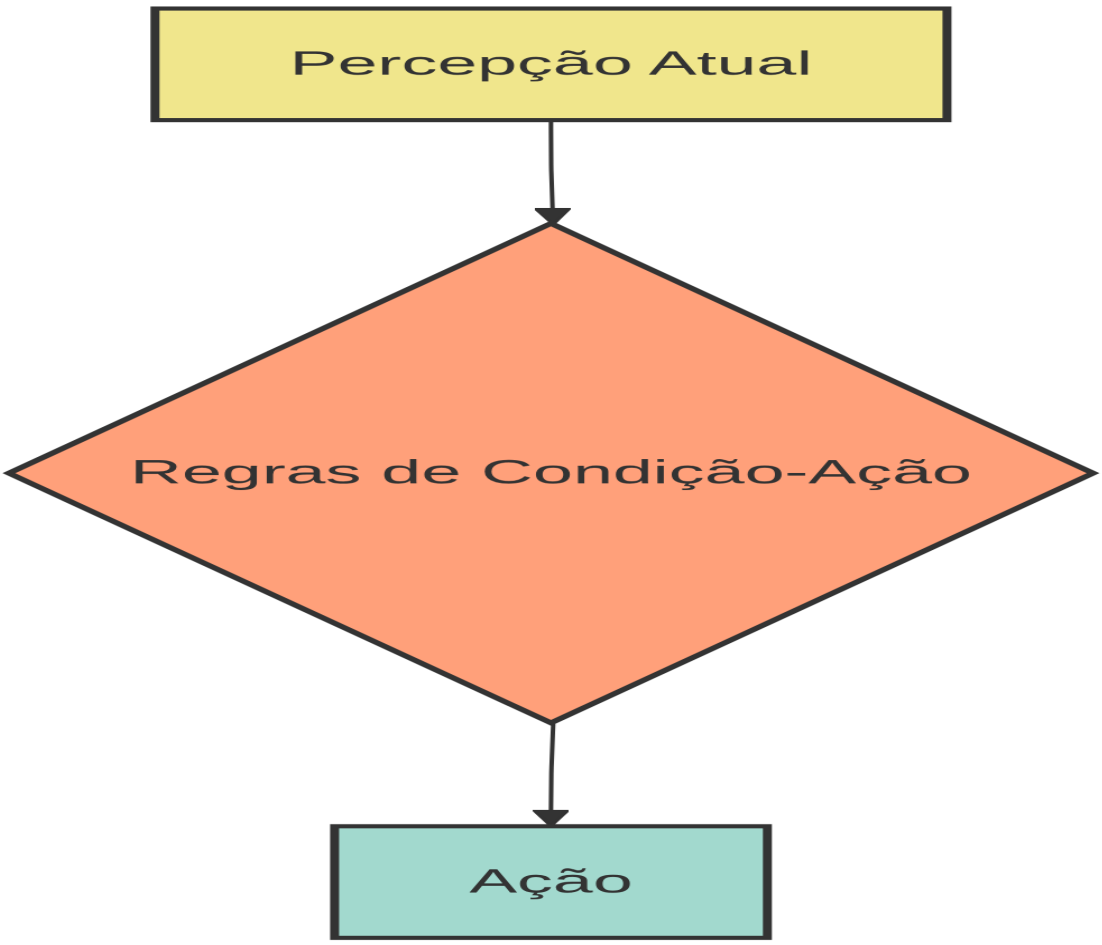

Figure 3: Agente Reflexivo Simples

Agente Reflexivo Simples em Python

```
def simple_reflex_agent(percepcao):
    if percepcao == "sujeira":
        return "aspirar"
    elif percepcao == "parede":
        return "virar"
    elif percepcao == "sem_sujeira_e_livre":
        return "mover_frente"
    else:
        return "nenhuma_acao"

# Teste
print(f"Ação para 'sujeira': {simple_reflex_agent('sujeira')}")
print(f"Ação para 'parede': {simple_reflex_agent('parede')}")```
```

Agentes Reativos Baseados em Modelo (Model-Based Reflex Agents)

Para ambientes parcialmente observáveis, um agente precisa manter um **estado interno** que capture o histórico de percepções e, assim, inferir as partes não observáveis do ambiente. Este estado interno é chamado de **modelo do mundo**.

- **Funcionamento:** Utiliza a percepção atual e o estado interno para atualizar o modelo do mundo, e então usa o modelo para decidir a ação. O modelo deve representar como o mundo evolui e como as ações do agente afetam o mundo.
- **Vantagens:** Pode operar em ambientes parcialmente observáveis, mais robusto que o agente simples.
- **Desvantagens:** Requer um modelo preciso do mundo, o que pode ser difícil de construir ou manter.
- **Exemplo:** Um aspirador de pó com sensores de sujeira, bateria e um mapa interno para registrar quais áreas já foram limpas e onde a sujeira foi encontrada.

Figure 4: Agente Reflexivo Baseado em Modelo

Agente Reflexivo Baseado em Modelo em Python

```python
class ModelBasedAgent:
    def __init__(self):
        self.estado_interno = {"localizacao": "A", "sujeira_em_A": False, "sujeira_em_B": False}

    def _atualizar_modelo(self, percepcao, ultima_acao):
        # Lógica para atualizar o estado interno com base na percepção e última ação
        if "sujeira" in percepcao:
            if self.estado_interno["localizacao"] == "A":
                self.estado_interno["sujeira_em_A"] = True
            elif self.estado_interno["localizacao"] == "B":
                self.estado_interno["sujeira_em_B"] = True

        if ultima_acao == "mover_para_B":
            self.estado_interno["localizacao"] = "B"
        elif ultima_acao == "mover_para_A":
            self.estado_interno["localizacao"] = "A"
        # etc.
        print(f"Modelo interno atualizado: {self.estado_interno}")

    def agent_program(self, percepcao, ultima_acao=None):
        self._atualizar_modelo(percepcao, ultima_acao)

        # Regras baseadas no estado interno
        if self.estado_interno["localizacao"] == "A":
            if self.estado_interno["sujeira_em_A"]:
                self.estado_interno["sujeira_em_A"] = False # Limpa a sujeira
                return "aspirar"
            elif self.estado_interno["sujeira_em_B"]: # Sabe que tem sujeira em B
                return "mover_para_B"
            else:
                return "nenhuma_acao" # Ou mover aleatoriamente
        elif self.estado_interno["localizacao"] == "B":
            if self.estado_interno["sujeira_em_B"]:
                self.estado_interno["sujeira_em_B"] = False
                return "aspirar"
            elif self.estado_interno["sujeira_em_A"]: # Sabe que tem sujeira em A
                return "mover_para_A"
            else:
                return "nenhuma_acao"

        return "nenhuma_acao"

# Teste
agente_modelo = ModelBasedAgent()
print(f"Ação: {agente_modelo.agent_program('percebi_sujeira', None)}") # Em A
print(f"Ação: {agente_modelo.agent_program('limpo', 'aspirar')}")     # Ação anterior foi aspirar
print(f"Ação: {agente_modelo.agent_program('limpo', None)}")         # Agora deve ir para B
print(f"Ação: {agente_modelo.agent_program('limpo', 'mover_para_B')}")# Em B, deve aspirar se houve su
```

Agentes Baseados em Metas (Goal-Based Agents)

Agentes baseados em metas não apenas conhecem o estado atual do mundo (via modelo), mas também o **estado objetivo** que desejam alcançar. Eles usam o planejamento e a busca para encontrar uma sequência de ações que os levem a esse objetivo.

- **Funcionamento:** O agente considera as consequências de suas ações no futuro. Ele calcula qual sequência de ações o levará ao estado desejado.
- **Vantagens:** Mais flexíveis, podem alcançar metas complexas, capazes de planejar e replanejar.
- **Desvantagens:** Planejamento e busca podem ser computacionalmente caros, especialmente em espaços de estados grandes.
- **Exemplo:** Um robô que precisa navegar de um ponto A a um ponto B em um ambiente. Ele planeja o caminho otimizando a distância ou o tempo. Um sistema de planejamento de rotas em um GPS.

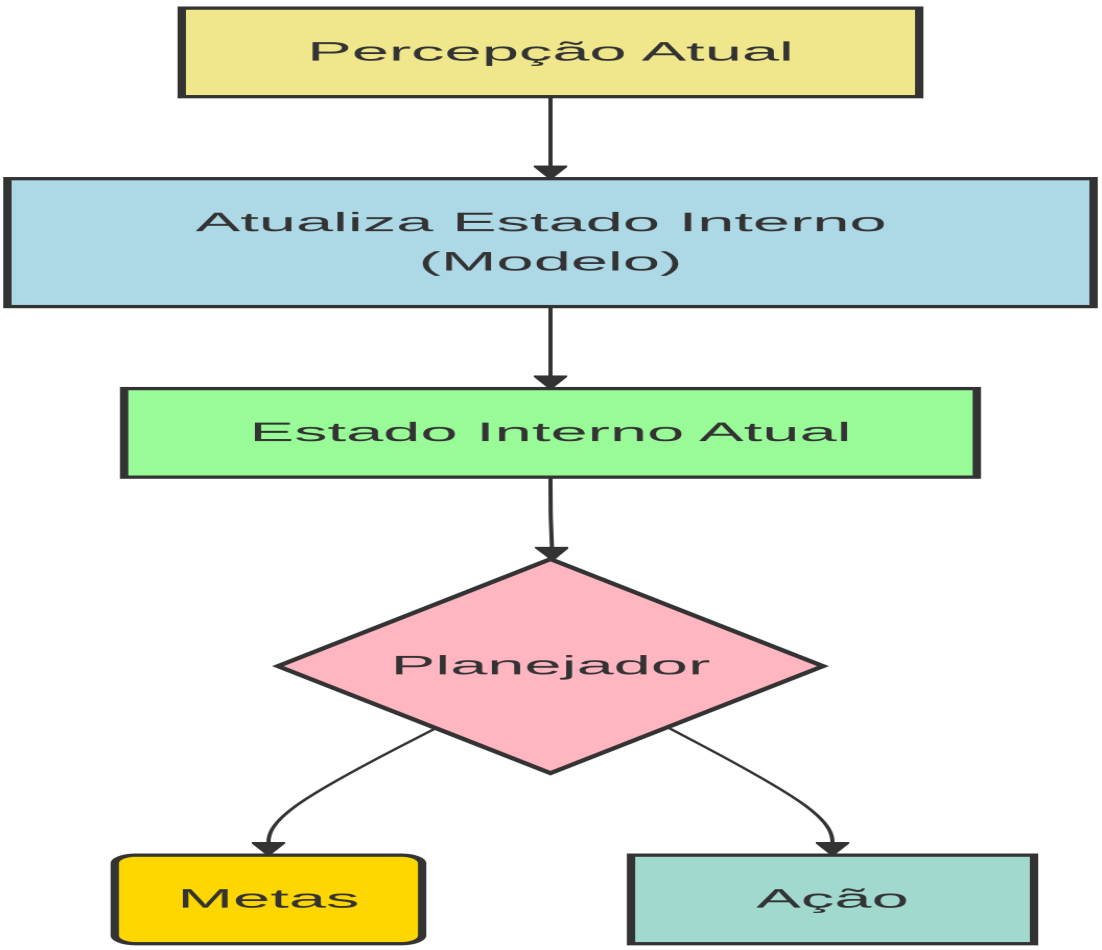

Figure 5: Agente Baseado em Metas

Agente Baseado em Metas em Python

```python
# Suponha que temos uma função de busca (e.g., BFS ou DFS do módulo anterior)
# def bfs(graph, start, goal): ...

class GoalBasedAgent:
    def __init__(self, environment_map, start_node, goal_node):
        self.environment_map = environment_map
        self.current_location = start_node
        self.goal = goal_node
        self.plan = [] # Sequência de ações para atingir a meta

    def _plan_route(self):
        # Aqui usaríamos um algoritmo de busca, como BFS, para encontrar o caminho
        # Para simplificar, vamos simular um plano fixo ou usar uma busca genérica

        # Exemplo simplificado de planejamento usando BFS (conceitual)
        # from collections import deque # Assumindo que bfs está disponível
        # queue = deque([(self.current_location, [self.current_location])])
        # visited = set()
        # while queue:
        #     node, path = queue.popleft()
        #     if node == self.goal:
        #         self.plan = path[1:] # Exclui o nó inicial, guarda as "próximas" cidades
        #         print(f"Plano gerado: {self.plan}")
        #         return
        #     if node not in visited:
        #         visited.add(node)
        #         for neighbor in self.environment_map.get(node, []):
        #             queue.append((neighbor, path + [neighbor]))

        # Substituindo por um plano manual para ilustração
        if self.current_location == 'Arad' and self.goal == 'Bucuresti':
            self.plan = ['Sibiu', 'Fagaras', 'Bucuresti'] # Um caminho possível
            print(f"Plano inicial de Arad para Bucuresti: {self.plan}")
        else:
             print("Nenhum plano predefinido ou busca complexa para esta meta.")

    def agent_program(self, percepcao=None):
        if not self.plan:
            self._plan_route() # Gerar plano se não houver um

        if self.current_location == self.goal:
            return "meta_atingida"

        if self.plan:
            next_step = self.plan.pop(0) # Pega a próxima cidade no plano
            print(f"Agente em {self.current_location}. Próximo passo no plano: {next_step}")
            self.current_location = next_step # Simula o movimento
            return f"mover_para_{next_step}"
        else:
            return "aguardar"

# Teste
mapa_romenia = {
    'Arad': ['Zerind', 'Timisoara', 'Sibiu'],
    'Sibiu': ['Arad', 'Oradea', 'Fagaras', 'RimnicuVilcea'],
```

Agentes Baseados em Utilidade (Utility-Based Agents)

Agentes baseados em utilidade são mais sofisticados que os baseados em metas. Eles não apenas buscam atingir uma meta, mas também tentam maximizar sua **função de utilidade**, que é uma medida de quão "feliz" o agente está em um determinado estado ou com um determinado resultado (Russell; Norvig, 2004, p. 48). Isso é crucial quando há múltiplos objetivos que podem ser conflitantes ou quando há incerteza sobre os resultados das ações.

- **Funcionamento:** O agente considera todas as consequências possíveis das ações, seus custos e recompensas associadas, e escolhe a ação que maximiza a utilidade esperada.
- **Vantagens:** Permitem um comportamento mais racional e otimizado em ambientes complexos, lidando com trade-offs e incerteza.
- **Desvantagens:** Definir e computar funções de utilidade pode ser extremamente complexo.
- **Exemplo:** Um carro autônomo que não apenas quer chegar ao destino (meta), mas também quer fazer isso com segurança, conforto e minimizando o tempo e o consumo de combustível (utilidades conflitantes que precisam ser ponderadas).

Agentes de Aprendizado (Learning Agents)

Qualquer um dos tipos de agentes acima pode se tornar um agente de aprendizado, que é capaz de melhorar seu desempenho ao longo do tempo. Um agente de aprendizado pode ser dividido em quatro componentes conceituais (Russell; Norvig, 2004, p. 50):

1. **Elemento de Aprendizado:** Responsável por fazer melhorias. Ele pega o feedback do Crítico e determina como os componentes do agente devem ser modificados para melhor desempenho futuro.
2. **Crítico:** Recebe uma medida de desempenho (como o agente está se saindo) e fornece feedback ao Elemento de Aprendizado sobre como o agente está agindo e quão bem ele está se saindo.
3. **Gerador de Problemas:** Sugere ações que levarão a novas experiências e, consequentemente, a novos conhecimentos para o Elemento de Aprendizado. Isso incentiva a exploração em vez de apenas a exploração do conhecimento existente.
4. **Elemento de Desempenho:** É o programa do agente "normal" (reflexivo, baseado em modelo, baseado em meta ou utilidade) que seleciona as ações externas do agente.

- **Funcionamento:** O agente aprende a partir da experiência (feedback do crítico, sugestões do gerador de problemas) e atualiza seu elemento de desempenho para melhorar a tomada de decisão futura.
- **Vantagens:** Adaptabilidade, autonomia, robustez a ambientes em mudança.
- **Desvantagens:** Requer dados de treinamento, pode ser difícil de projetar o sistema de feedback e o gerador de problemas.
- **Exemplo:** Um sistema de recomendação de filmes que aprende as preferências do usuário com base nas avaliações e sugere filmes que o usuário provavelmente gostará. Um carro autônomo que aprende a se comportar em novas condições de estrada.

Considerações Finais

A escolha do tipo de agente a ser construído depende em grande parte das características do problema e do ambiente em que o agente deve operar. Agentes mais simples são suficientes para ambientes totalmente observáveis e determinísticos, enquanto ambientes complexos, incertos ou dinâmicos exigem agentes mais

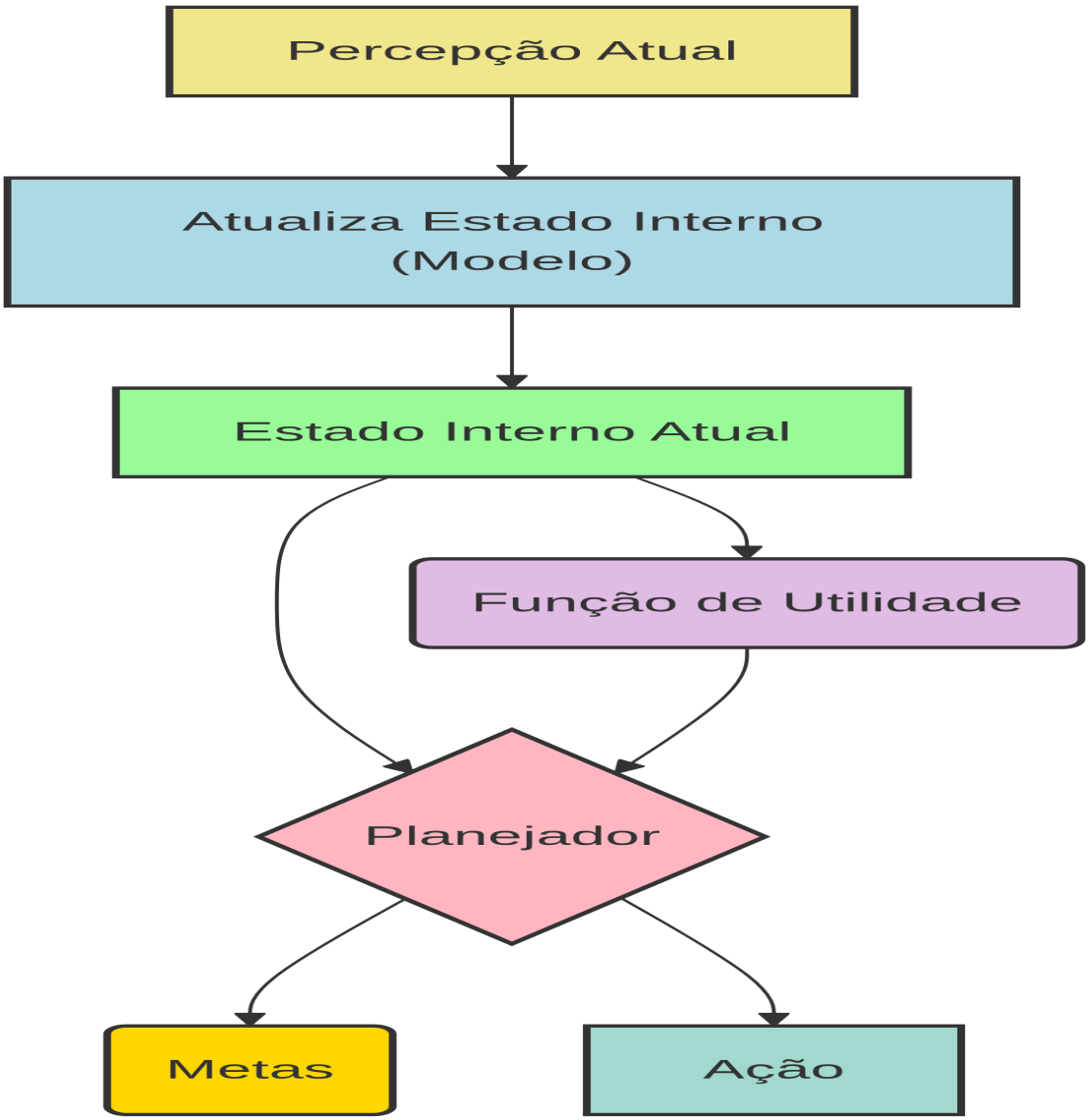

Figure 6: Agente Baseado em Utilidade

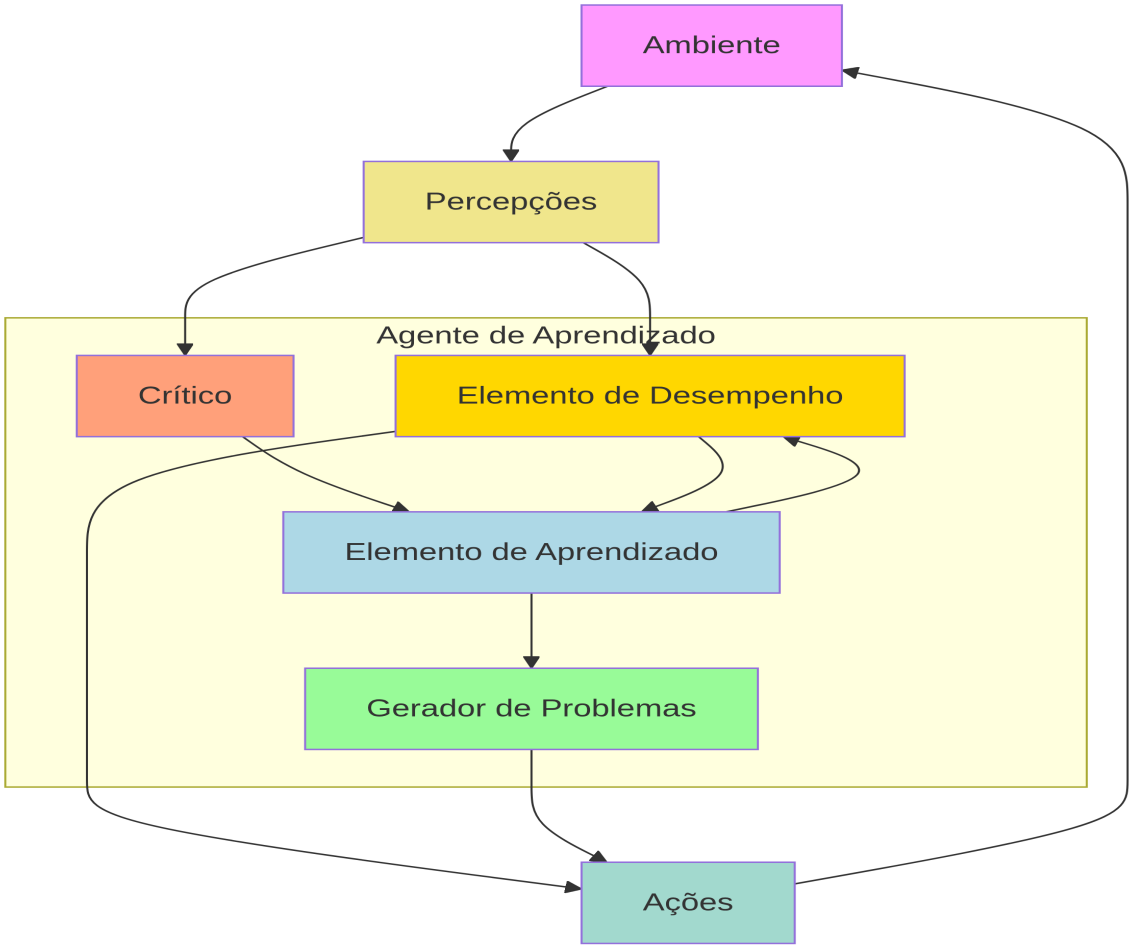

Figure 7: Estrutura de um Agente de Aprendizado

sofisticados, com modelos internos, metas, funções de utilidade e, idealmente, a capacidade de aprender e se adaptar.

A construção de agentes inteligentes é um campo ativo de pesquisa e desenvolvimento, e a compreensão desses fundamentos é a base para explorar tópicos mais avançados em IA.

Verificação de Aprendizagem

Responda às seguintes questões para solidificar seu entendimento sobre os agentes inteligentes.

1. **Definição e Componentes:**
    a) Com suas próprias palavras, defina o que é um "agente inteligente".
    b) Quais são os dois componentes principais que constituem a estrutura de um agente inteligente? Explique brevemente a função de cada um.
2. **Modelo PEAS:** Considere um **agente de recomendação de músicas** (como em um aplicativo de streaming). Descreva este agente usando o modelo PEAS:
    - **P**erformance (Medida de Desempenho):
    - **E**nvironment (Ambiente):
    - **A**ctuators (Atuadores):
    - **S**ensors (Sensores):
3. **Tipos de Agentes:**
    a) Qual a principal diferença entre um **Agente Reativo Simples** e um **Agente Reativo Baseado em Modelo**? Quando você escolheria um sobre o outro?
    b) Um **Agente Baseado em Metas** e um **Agente Baseado em Utilidade** podem ambos usar um modelo do mundo e planejamento. Qual é a principal funcionalidade extra que o agente baseado em utilidade adiciona e por que ela é importante?
    c) Explique como um **Agente de Aprendizado** melhora seu desempenho. Quais são os quatro componentes conceituais de um agente de aprendizado, e qual a função de cada um?
4. **Aplicação de Tipos de Agentes:** Para cada um dos cenários abaixo, qual tipo de agente inteligente (entre os 5 discutidos) seria o mais adequado para a tarefa e por quê?
    a) Um sistema de controle de temperatura em um datacenter que precisa manter a temperatura dentro de uma faixa específica, ligando e desligando os condicionadores de ar.
    b) Um robô de resgate que deve navegar em um ambiente desconhecido e parcialmente observável para encontrar vítimas, considerando que o tempo é um fator crítico e há várias prioridades (encontrar vítimas vs. evitar perigos).
    c) Um programa de IA que joga um jogo de tabuleiro complexo como Go, aprendendo a estratégia através de milhões de jogos contra si mesmo.

Referências Bibliográficas

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência Artificial: Um Enfoque Moderno**. 2. ed. Rio de Janeiro: Prentice Hall, 2004.