Projeto 6: Agente Baseado em Conhecimento - Mundo de Wumpus Visual

Autor: Marcos Antonio Teles de Castilhos

Disciplina: FGA0221 - Inteligência Artificial

Professor: Fabiano Araujo Soares, Dr.

1. Introdução

Este projeto final implementa um **agente lógico baseado em conhecimento**, um dos pilares da Inteligência Artificial clássica (GOFAI). O agente opera no ambiente simulado do **Mundo de Wumpus**, um cenário clássico onde a informação é parcial e o raciocínio lógico é essencial para a sobrevivência e exploração.

O objetivo do agente é explorar o mundo de forma segura, utilizando suas percepções sensoriais (Brisa e Fedor) para construir uma **Base de Conhecimento (KB)** sobre o ambiente e realizar **inferências** lógicas para determinar quais casas são seguras para visitar.

O script inclui uma **visualização gráfica** (matplotlib) que mostra, passo a passo, o "mapa mental" do agente (o conteúdo de sua KB) sendo construído e comparado com o estado real do mundo.

2. O Problema: Mundo de Wumpus

O Mundo de Wumpus é um ambiente em grade (4x4 neste caso) contendo:

- Um **Agente:** Começa em (1,1).
- Um **Wumpus:** Monstro mortal.
- **Poços:** Mortais se o agente entrar.
- **Ouro:** (Não implementado neste agente, o foco é a exploração segura).

Percepções do Agente:

- **Fedor:** Sentido nas casas adjacentes (não diagonais) ao Wumpus.
- **Brisa:** Sentida nas casas adjacentes (não diagonais) a um Poço.

Objetivo do Agente: Explorar o máximo possível do mundo, movendo-se apenas para casas que ele possa **provar logicamente** que são seguras (livres de Poços e do Wumpus).

3. Arquitetura do Agente Lógico

O agente implementado segue os princípios de um agente baseado em conhecimento :

- **Base de Conhecimento (KB):**
    1. Implementada como um **set** Python, armazenando fatos como strings (Lógica Proposicional simplificada).
    2. **Fatos:**
        - **~P_x,y**: Não há Poço em (x,y).
        - **~W_x,y**: Não há Wumpus em (x,y).
        - **B_x,y**: Há Brisa em (x,y).
        - **S_x,y**: Há Fedor em (x,y).
        - OK_x,y: A casa (x,y) é segura (provado que ~P_x,y E ~W_x,y).
    3. **Operações:** tell(fato) adiciona um fato ao set; ask(query) verifica se um fato está no set.
- **Ciclo de Raciocínio (decidir_proxima_acao):**
    1. **Perceber:** Recebe as percepções da sua localização atual do ambiente simulado.
    2. **Raciocinar (Atualizar KB):**
        - Adiciona as percepções atuais à KB (tell).
        - **Inferência Principal (atualizar_kb_percepcoes):** Aplica a regra lógica mais importante: Se uma percepção **não** está presente (ex: sem Brisa), infere-se que o perigo correspondente (Poço) **não** existe em **nenhuma** das casas adjacentes. Fatos **~P e ~W** são adicionados à KB.
        - **Inferência Secundária (inferir_casas_seguras):** Percorre a KB para encontrar casas onde **~P_x,y** e **~W_x,y** são ambos verdadeiros, inferindo e adicionando o fato OK_x,y. Também inclui uma lógica simplificada para marcar casas como

potencialmente perigosas (**P?, W?**) com base nas percepções dos vizinhos.

3. **Agir:** Seleciona uma casa OK que ainda não foi visitada e se move para ela. Se nenhuma casa segura e não visitada for conhecida, a exploração termina.

4. Implementação e Visualização Gráfica

- **Classe AgenteWumpus:** Encapsula a KB, a posição atual, a lista de visitados e os métodos de raciocínio e decisão.
- **Classe MundoWumpus:** Simula o ambiente, posicionando Wumpus e Poços aleatoriamente e fornecendo as percepções corretas ao agente com base em sua posição. O agente **não** tem acesso direto às informações internas desta classe (localização real dos perigos).
- **Visualização (matplotlib):**
    - Cria uma janela com dois painéis:
        - **Painel Direito (ax_verdade):** Mostra o "Mapa da Verdade", com a localização real do Wumpus ('W') e dos Poços ('P'). Permanece estático.
        - **Painel Esquerdo (ax_agente):** Mostra o "Mapa Mental" do agente, atualizado a cada passo. Representa o conteúdo da KB:
            - **'A':** Posição atual do agente.
            - **Fundo Cinza:** Casas já visitadas.
            - **Texto 'Brisa'/'Fedor':** Percepções registradas nas casas visitadas.
            - **Fundo Verde com 'OK':** Casas inferidas como seguras, mas ainda não visitadas.
            - **Texto 'P?'/'W?':** Casas onde o agente suspeita (mas não tem certeza) que possa haver um perigo.
- **Animação:** O script pausa após cada movimento do agente, permitindo observar como as percepções levam a novas inferências e à expansão do mapa mental.

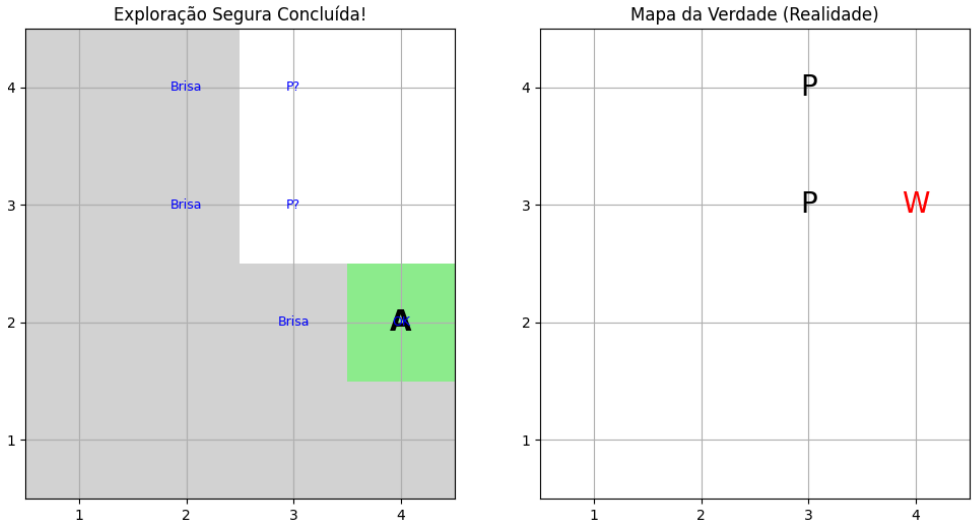
5. Como Usar o Programa

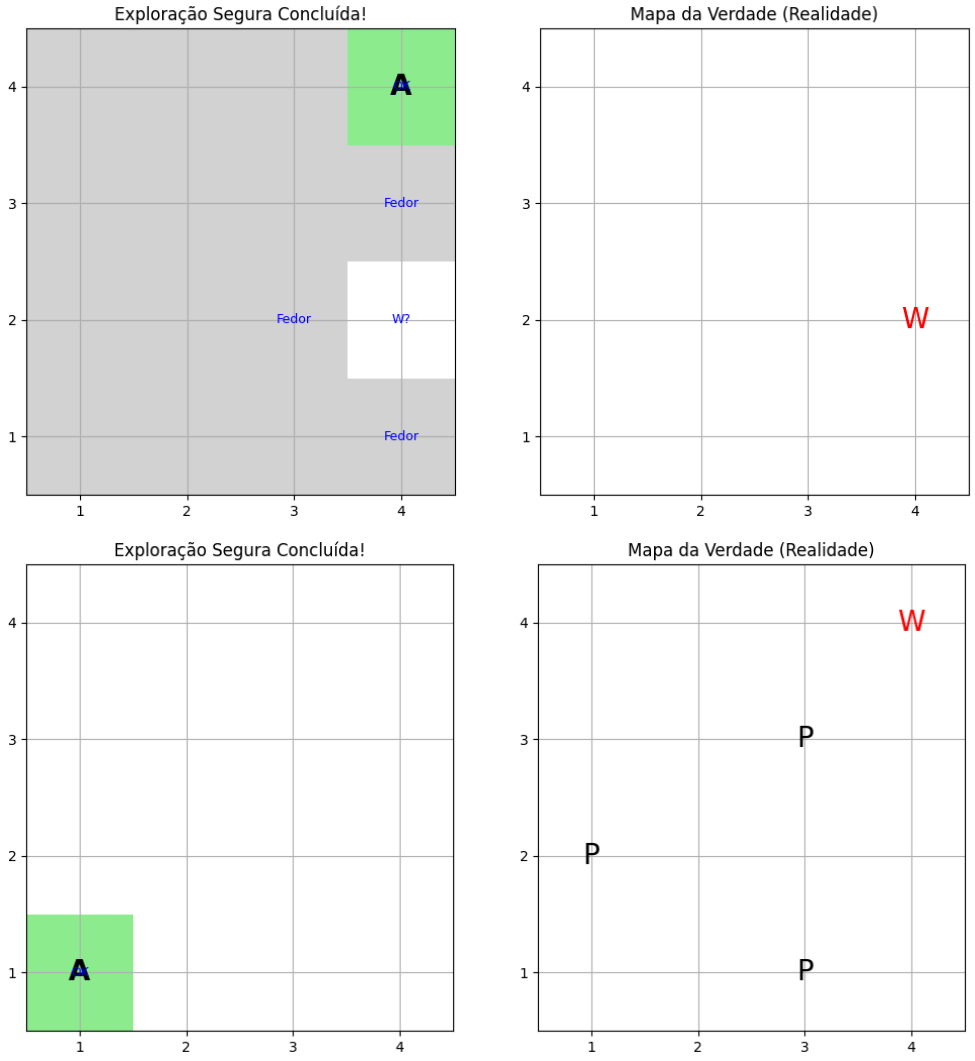
1. **Pré-requisitos:** Python 3 e matplotlib. Instale com pip install matplotlib.
2. **Execução:**

   python base-conhecimento.py

3. **Visualização:** Uma janela gráfica será aberta. Observe o painel esquerdo (mapa do agente) evoluir a cada passo, comparando-o com o painel direito (realidade).
4. **Resultado:** A animação continua até o agente explorar todas as casas seguras que consegue identificar, morrer (raro), ou atingir o limite de passos. O resultado final da exploração (e se o agente sobreviveu) é mostrado no título da janela e no terminal.

6. Imagens





7. Conclusão

Este projeto demonstra com sucesso a implementação de um agente lógico simples capaz de raciocinar sobre um ambiente parcialmente observável. Utilizando uma Base de Conhecimento e regras de inferência básicas (principalmente a partir da ausência de percepções), o agente constrói um modelo do mundo e toma decisões seguras para exploração. A visualização comparativa entre o "mapa mental" do agente e a realidade do mundo ilustra claramente o processo de aquisição de conhecimento e raciocínio em ação.