Akinator sobre cientistas

Predicado principal

```
None
caracteristicas([realoufictício, nacionalidade, genero, area, periodomaioratividade, vivooufalecido, conquista, premio_nobel]).
```

Lista de Cientistas - base de dados inicial

1. Albert Einstein
2. Marie Curie
3. Charles Darwin
4. Stephen Hawking
5. Isaac Newton
6. Nikola Tesla
7. Ada Lovelace
8. Alan Turing
9. Sigmund Freud
10. Oswaldo Cruz
11. Carlos Chagas
12. Hedy Lamarr
13. Grace Hopper
14. Jane Goodall
15. Rosalind Franklin
16. Galileu Galilei
17. Marie Tharp
18. George Washington Carver
19. Katherine Johnson
20. Mae Jemison
21. Sheldon Cooper
22. Amy Farrah Fowler

Arquivos projeto

- **main.prolog:** o arquivo contém as regras principais, como startGame, perguntar, adivinhar.

- **inferencia.prolog:** contém a parte da lógica de inferência para o jogo

- **database.prolog:** arquivo que contém os cientistas da base de dados, os mesmos são declarados através do fato nomeado “entidade(nome cientistas)” e são relacionados com fatos como realoufictício(nome cientista, resposta), nacionalidade(nome

cientista, resposta), genero(nome cientista, resposta), area(nome cientista, resposta), periodomaioratividade(nome cientista, resposta), vivooufalecido(nome cientista, resposta), conquista(nome cientista, resposta), premio_nobel(nome cientista, resposta).

- **aprendizado.prolog:** contém a lógica para adicionar novos cientistas à base de dados. É chamado quando o usuário ganha.

Diretiva dynamic

Essa diretiva dynamic serve para declarar que os predicados entidade/1 e pergunta/2 podem ser modificados dinamicamente durante a execução do programa (por exemplo, com assert, retract, etc). Isso também se aplica nos dynamic em aprendizado.prolog. \1 significa que tem 1 argumento e \2 que tem 2 argumentos. Por exemplo, o predicado pergunta/2 tem dois argumentos porque ele relaciona uma característica com o valor da pergunta que será feita ao jogador.

Predicado findall

findall/3 é um predicado fundamental em Prolog que coleta todas as soluções para um objetivo específico e as armazena em uma lista. Ele é definido com três argumentos:

- Template: Este argumento especifica qual termo você deseja coletar para cada solução. Geralmente, é uma variável que representa o valor de interesse.
- Goal: Este é o objetivo (ou conjunto de objetivos) para o qual você deseja encontrar todas as soluções. O Prolog tentará satisfazer este Goal e, para cada vez que ele for bem-sucedido, o Template correspondente será adicionado à lista.
- List: Esta é a variável unificada com a lista de todas as soluções encontradas.

Predicado forall

forall/2 é um predicado que verifica se todas as soluções possíveis para uma condição (Cond) satisfazem uma certa ação ou verificação (Ação). Para todo Cientista que ainda está na base (entidade(Cientista)), Se ele não estiver na Lista, então ele é removido da base com retract(entidade(...)).

main.prolog

Predicado perguntar

1. O contador(Cont) recupera quantas perguntas já foram feitas até agora. Se Cont >= 20, o jogo atinge o limite máximo de perguntas:Informa o jogador. e Chama perguntar_e_adicionar_cientistapara aprender com o jogador quem era o cientista. Depois, reinicia o jogo com jogar_novamente.

2. O findall busca todos os candidatos que ainda são possíveis, caso não tenha sobrado nenhum cadidato avisa o jogador e chama perguntar_e_adicionar_cientista

3. Caso reste apenas um candidato chama o predicado adivinhar.

4. Caso ainda exista vários candidatos obtém a lista de possíveis características, busca todas a combinações possíveis com findall(...) que ainda não foram perguntadas (\+ pergunta(...)) e são válidas, ou seja, ocorrem em algum candidato e calcula o peso da pergunta, ou seja, que pergunta dividiria melhor a base de dados.

5. Se houver perguntas válidas, faça a melhor pergunta ordenando a opções por peso e escolhendo a primeira pergunta com sort(OpcoesNaoUnicas, [(MelhorPeso, Carac, Valor)|_]),

Predicado adivinhar
Se NumEntidades = 1, pega o único cientista da lista com pattern matching. Exibe a pergunta ao usuário: "O seu cientista chama X. Acertei?"

inferencia.prolog

Declara os predicados dinâmicos, ou seja, que podem ser modificados em tempo de execução (com assert, retract, etc).

- **Regra conta_caracteristica:** conta quantos cientistas ainda possíveis têm uma característica com um determinado valor. Ex: conta_caracteristica(genero, feminino, C - retorna em C quantos cientistas ainda são mulheres.
- **Regra peso_pergunta:** Essa regra calcula o "peso" (utilidade) de uma pergunta, com base em quantos candidatos ela conseguiria eliminar no pior cenário. O Peso é o menor entre os dois, pois uma pergunta boa é a que divide bem os candidatos. Ex: se 5 têm e 5 não têm, o peso é 5 (ideal) e se 9 têm e 1 não tem, o peso é 1 (quase inútil).
- **Regra processar resposta:** processa a resposta do usuário a uma pergunta. Se a resposta for sim, mantém apenas os cientistas que têm essa característica com esse valor chamado "**atualizar base**". Se a resposta for não remover todos os cientistas que têm essa característica com esse valor chamando "**remover da base**".
- **Definição caracteristicas:** define a lista de características que o sistema pode usar nas perguntas.

aprendizado.prolog

Com dynamic declara os predicados serão modificados em tempo de execução.
O predicado **aprender_novo_cientista** inicialmente pede o nome do cientista, converte para atomo e registra como uma nova entidade usando **assert(entidade(Nome))**. Chama **características(Lista)** para pegar a lista de características definidas em inferencia.prolog e chama **aprender_caracteristicas** para perguntar valores dessas características para o novo cientista. Por fim chama **salvar_novo_cientista** para salvar no arquivo database.prolog