Jogo da Velha em Prolog

Cândido Leandro de Queiroga Bisneto, José Matheus Marques Vinagre.

Resumo:

O jogo da velha é um jogo simples e mundialmente conhecido, no qual, é comum os desenvolvedores de jogos colocarem em prática suas habilidades criando esse jogo de forma virtual em diversas linguagens de programação. O objetivo do trabalho é desenvolver o jogo da velha, através da linguagem de programação Prolog. É possível observar algumas limitações na linguagem para a criação de algumas regras do jogo, devido à falta de funções específicas comum nas linguagens atuais, como if e switch que encurtam o código. Entretanto, apesar dessas limitações, o SWI-Prolog é capaz de cumprir as regras e criar a interface do jogo em seu terminal.

Palavras-chave

Jogo — Virtual — Prolog

Centro de Informática, Universidade Federal da Paraíba, João Pessoa, Brasil.

Contato: {jose.vinagre, clqb} @academico.ufpb.br.

Sumário

Introdução

A partir dos conhecimentos adquiridos em sala de aula, foi realizado um Jogo da Velha utilizando os conceitos de proposição, conectivos e premissas, predicados e demais conceitos da lógica proposicional aplicada a computação em uma linguagem de propagação baseada em paradigmas chamada Prolog.

A qual, é uma linguagem de programação que se enquadra no paradigma de Programação em Lógica Matemática. É uma linguagem de uso geral que é especialmente associada com a inteligência artificial e linguística computacional.

1. Lógica Implementada

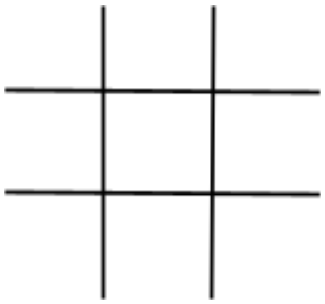
Para verificar a vitória, derrota ou empate, foi utilizado regras de inferência previamente definidas dentro da lógica determinando as posições das marcações

necessárias para a vitória, seja ela, horizontal, vertical ou diagonal. Além disso, também foi implementado um sistema que verifica a legalidade da jogada, caso o usuário faça um comando irregular ou tente preencher uma casa já preenchida, ele perde a vez.

Assim como no jogo original, também foi definida uma lógica de empate, ou "velha", caso o jogador não consiga preencher as casas necessárias para a sua vitória, e nem o 'bot', o qual está como adversário no jogo.

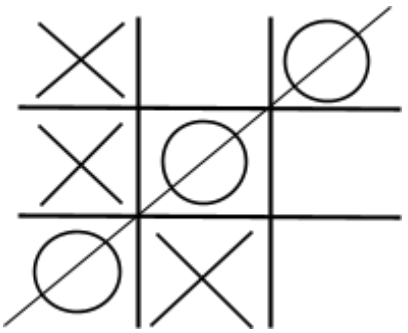
1.1 Jogo da Velha

O jogo da velha consiste em um tabuleiro de nove casas, que formam três linhas e três colunas:



Tabuleiro 1. Jogo da velha.

O objetivo do jogo é fazer uma sequência de três símbolos iguais, seja em linha vertical, horizontal ou diagonal, enquanto tenta impedir que seu adversário faça o mesmo. Quando um dos participantes faz uma linha, ganha o jogo.



Tabuleiro 2. Círculo vencedor.

Além de ser divertido e dinâmico, o jogo desenvolve algumas competências, como: desenvolvimento do raciocínio lógico, aperfeiçoamento de capacidades espacial e visual, interação e competitividade saudável, e bem estar psicológico geral.

1.2 Ferramenta

O tabuleiro do jogo é formado no terminal pela regra 'disp', no qual gera o tabuleiro com 9 casas: "A,B,C,D,E,F,G,H,I".

```
disp([A,B,C,D,E,F,G,H,I]) :-
        write('|'),
        write([A,B,C]),write('|'),nl,
        write('|'),
        write([D,E,F]),write('|'),nl,   write('|'),
        write([G,H,I]),write('|'),nl,nl.
```

Figura 1. IDE.

A regra 'disp' que permite definir uma relação entre as coordenadas contidas nas listas 'xmove' e 'omove', e caso o usuário selecione uma cordenada ja usada,

ele é repreendido pela regra de mesmo nome da lista ‘xmove’, a qual imprime uma mensagem no terminal “Movimento ilegal!”. Além disso, o jogador perde a sua vez para o ‘bot’ do jogo.

```
omove([a,B,C,D,E,F,G,H,I], Jogador, [Jogador,B,C,D,E,F,G,H,I]).
omove([A,a,C,D,E,F,G,H,I], Jogador, [A,Jogador,C,D,E,F,G,H,I]).
omove([A,B,a,D,E,F,G,H,I], Jogador, [A,B,Jogador,D,E,F,G,H,I]).
omove([A,B,C,a,E,F,G,H,I], Jogador, [A,B,C,Jogador,E,F,G,H,I]).
omove([A,B,C,D,a,F,G,H,I], Jogador, [A,B,C,D,Jogador,F,G,H,I]).
omove([A,B,C,D,E,a,G,H,I], Jogador, [A,B,C,D,E,Jogador,G,H,I]).
omove([A,B,C,D,E,F,a,H,I], Jogador, [A,B,C,D,E,F,Jogador,H,I]).
omove([A,B,C,D,E,F,G,a,I], Jogador, [A,B,C,D,E,F,G,Jogador,I]).
omove([A,B,C,D,E,F,G,H,a], Jogador, [A,B,C,D,E,F,G,H,Jogador]).
```

Figura 2. IDE.

```
xmove([a,B,C,D,E,F,G,H,I], 1, [x,B,C,D,E,F,G,H,I]).
xmove([A,a,C,D,E,F,G,H,I], 2, [A,x,C,D,E,F,G,H,I]).
xmove([A,B,a,D,E,F,G,H,I], 3, [A,B,x,D,E,F,G,H,I]).
xmove([A,B,C,a,E,F,G,H,I], 4, [A,B,C,x,E,F,G,H,I]).
xmove([A,B,C,D,a,F,G,H,I], 5, [A,B,C,D,x,F,G,H,I]).
xmove([A,B,C,D,E,a,G,H,I], 6, [A,B,C,D,E,x,G,H,I]).
xmove([A,B,C,D,E,F,a,H,I], 7, [A,B,C,D,E,F,x,H,I]).
xmove([A,B,C,D,E,F,G,a,I], 8, [A,B,C,D,E,F,G,x,I]).
xmove([A,B,C,D,E,F,G,H,a], 9, [A,B,C,D,E,F,G,H,x]).
xmove(Tabuleiro, _, Tabuleiro) :-
        write('movimento ilegal!'), nl.
```

Figura 3. IDE

A regra ‘jogada’ exibe ao usuário que ele é o jogador X, e explica ao usuário como funciona a marcação das coordenadas.

```
jogada :-
  write('você é o jogador X.'),
  nl,
  disp([1,2,3,4,5,6,7,8,9]).
```

Figura 4. IDE.

O ‘bot’, inicialmente, segue uma lógica aritmética, que marca as casas iniciais “1,2,3…” , essa "ingenuidade" da máquina prossegue até ela ter uma chance de formar uma linha, ou impedir o usuário de formá-la. A máquina possui esse “ponto fraco” para dar uma chance ao usuário de vencer, pois o jogo da velha é um jogo tecnicamente empatado devido à necessidade de um deslize do adversário para vencer.

```
vez(Tabuleiro,NovoTabuleiro) :-
  omove(Tabuleiro, o, NovoTabuleiro),
  vitoria(NovoTabuleiro, o),!.
vez(Tabuleiro,NovoTabuleiro) :-
  omove(Tabuleiro, o, NovoTabuleiro),
  not(x_vitoria(NovoTabuleiro)).
vez(Tabuleiro,NovoTabuleiro) :-
  omove(Tabuleiro, o, NovoTabuleiro).
vez(Tabuleiro, NovoTabuleiro) :-
  not(member(a,Tabuleiro)),!,
  disp(NovoTabuleiro).
```

Figura 5. IDE

O movimento de vitória é dado pelas seguintes regras: ‘linha_vitoria’, ‘coluna_vitoria’, ‘diagonal_vitoria’ e por fim pela regra ‘vitoria’, a qual tem a função de detectar o tipo de vitória alcançado pelo jogador, se foi pela formação de uma linha horizontal, vertical ou pelas diagonais do tabuleiro.

```
vitoria(Tabuleiro, Jogador) :-
                linha_vitoria(Tabuleiro, Jogador);
                coluna_vitoria(Tabuleiro, Jogador);
                diagonal_vitoria(Tabuleiro, Jogador).
%linha
linha_vitoria(Tabuleiro, Jogador) :-
                 Tabuleiro = [Jogador,Jogador,Jogador,_,
                 Tabuleiro = [_,_,_,Jogador,Jogador,Joga
                 Tabuleiro = [_,_,_,_,_,_,Jogador,Jogadc

%coluna
coluna_vitoria(Tabuleiro, Jogador) :-
                 Tabuleiro = [Jogador,_,_,Jogador,_,_,Jc
                 Tabuleiro = [_,Jogador,_,_,Jogador,_,_,
                 Tabuleiro = [_,_,Jogador,_,_,Jogador,_,
%diagonal
diagonal_vitoria(Tabuleiro, Jogador) :-
                 Tabuleiro = [Jogador,_,_,_,Jogador,_,_,
                 Tabuleiro = [_,_,Jogador,_,Jogador,_,Jc
```

Figura 6. IDE.

A regra ‘strt’ verifica se o regra ‘vitória’ deu retorno, caso sim, verifica se foi o jogador “o” ou “x”, e mostra a situação do tabuleiro e declara o fim da partida; caso contrário o código continua rodando a regra ‘vez’ e o próprio ‘strt’.

```
strt(Tabuleiro) :-
        vitoria(Tabuleiro, x), write('FIM DE J(
strt(Tabuleiro) :-
        vitoria(Tabuleiro, o), write('FIM DE J(
strt(Tabuleiro) :- read(N),
  xmove(Tabuleiro, N, NovoTabuleiro),
  disp(NovoTabuleiro),
  vez(NovoTabuleiro, NovoNovoTabuleiro),
  disp(NovoNovoTabuleiro),
  strt(NovoNovoTabuleiro).
```

Figura 7. IDE.

A regra ‘go’ é a responsável por iniciar o código, no qual executa a regra ‘jogada’, e dá início a regra ‘strt’, que funciona em looping e lê o comando dado pelo usuário para que seja feita a jogada.

```
go :- jogada, strt([a,a,a,a,a,a,a,a,a]).
```

Figura 8. IDE.

2. Método

A construção da aplicação foi feita através de uma IDE, denotando-se um conjunto de regras para o funcionamento do jogo.

A princípio, o jogo foi pensado para ser jogado por dois jogadores, mas, pelas limitações oferecidas pela linguagem, foi-se necessário optar por uma abordagem *single player,* a qual, ainda permitiu todo um desenvolvimento prático dos conhecimentos adquiridos em sala de aula somado a outros conhecimentos de programação adquiridos em outras cadeiras durante o curso.

Outrossim, foi usado demais recursos de conhecimento, como os encontrados na documentação do SWI-Prolog e materiais relacionados. Desse modo, possibilitando toda a implementação desenvolvida no código fonte.

A escolha para o usuário ser o jogador X aconteceu devido à necessidade do usuário começar a partida para haver um início de jogo aleatório, pois normalmente no jogo da velha o X inicia o jogo.

Por fim, após a consulta do código no terminal, foi realizado um teste de funcionamento da aplicação de modo que também foi testado usando um cenário em que o usuário realiza uma jogada errada e assim perdendo sua vez, e por consequência, perde o jogo também.

```
?- go.
você é o jogador X.
|[1,2,3]|
|[4,5,6]|
|[7,8,9]|

|: 3.
|[a,a,x]|
|[a,a,a]|
|[a,a,a]|

|[o,a,x]|
|[a,a,a]|
|[a,a,a]|

|: 5.
|[o,a,x]|
|[a,x,a]|
|[a,a,a]|

|[o,a,x]|
|[a,x,a]|
|[o,a,a]|

|: 7.
movimento ilegal!
|[o,a,x]|
|[a,x,a]|
|[o,a,a]|

|[o,a,x]|
|[o,x,a]|
|[o,a,a]|

FIM DE JOGO!
true .
```

Figura 9. Terminal.

3. Conclusão

A criação de jogos virtuais é um símbolo da inovação tecnológica, que cresce dia após dia. Com essa ideia em mente, percebemos que é possível a criação de um jogo da velha em Prolog. Entretanto, a criação de um looping para que o código permanecesse aberto e com uma "memória" das jogadas passadas fez com que a finalização por empate exibisse o número do endereço das coordenadas das casas, ao invés de exibir o tabuleiro preenchido. Algo corrigível em outras linguagens de programação.

Durante a criação, houve um conhecimento considerável adquirido sobre a linguagem e também como trabalhar usando apenas o terminal como display. Além disso, foi explorado a criatividade em como ‘bot’ iria funcionar durante a partida, no qual, não poderia ser algo muito difícil ou muito fácil.

4. Referências

[1] PEREIRA, Claudia. **Jogo da Velha**. Educamais, 25 mar. 2020. Disponível em: https://educamais.com/jogo-velha

[2] Ferreira, J. **Prolog**. Instituto Federal de Minas Gerais (IFMG): https://www.youtube.com/playlist?list=PLZ-Bk6jzsb-OScKa7vhpcQXoU2uxYGaFx

[3] Documentação do SWI-Prolog via **swi-prolog.org**: https://www.swi-prolog.org/pldoc/doc_for?object=manual

[4] Lago, S. **Introdução a Linguagem Prolog**. Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP): https://www.ime.usp.br/~slago/slago-prolog.pdf