

Programação em Lógica

Relatório do Trabalho Prático Nº 1



Grupo Stac_1

José Miguel Melo - ei12050

Rui Gomes - ei12038

11 de Novembro de 2014

Resumo

No âmbito da Unidade Curricular Programação em Lógica, foi proposta a implementação de do jogo de tabuleiro [Stac](#), no modos de Jogador vs. Jogador, Jogador vs. Computador e Computador vs. Computador.

O jogo consiste num tabuleiro 5x5, inicialmente preenchido por uma peça em cada posição do tabuleiro e cada jogador situado num dos cantos. O objetivo do jogo é fazer torres com 3 peças, sendo a vitória atribuída ao primeiro jogador a conseguir fazer três torres com três peças, seguindo um conjunto de regras que serão descritas e analisadas ao longo do relatório.

Utilizando PROLOG, o grupo implementou toda a lógica referente às regras de jogo, incluindo regras específicas relativamente aos movimentos, regras referente à vitória no jogo e implementou também um mecanismo de jogadas automáticas, de forma a ser possível jogar contra o computador, apenas com um nível de dificuldade.

Índice

[Resumo](#)

[Índice](#)

[Introdução](#)

[Stac](#)

[Regras do jogo](#)

[Lógica do jogo](#)

[Representação do estado do jogo](#)

[Visualização do tabuleiro](#)

[Lista de jogadas válidas](#)

[Execução de jogadas](#)

[Avaliação do tabuleiro e final do jogo](#)

[Jogada do computador](#)

[Interface com o utilizador](#)

[Conclusão](#)

Introdução

O objetivo do primeiro trabalho prático era a implementação do jogo de tabuleiro [Stac](#) na linguagem PROLOG, com três modos distintos: Jogador vs. Jogador, Jogador vs. Computador e Computador vs. Computador.

Ao longo do relatório será descrito o jogo, bem como as suas regras específicas, bem como a estratégia de implementação adotada pelo grupo, incluindo os mecanismos de visualização e cumprimento de regras específicas ao jogo.

Stac

O jogo Stac, originalmente publicado em 2014, é um jogo de tabuleiro para dois jogadores, desenhado pelo norte-americano Daniel Isom, tendo sido já premiado com o *2 Player Print and Play Best Abstract Game (2014)* e também o *2 Player Print and Play Best Language-Independent Game - 2nd Place (2014)*

Consiste num tabuleiro quadrado, 5x5, inicialmente com uma peça redonda, *disco*, em cada uma das casas, num total de 25 peças da mesma cor.

Os jogadores começam o jogo em cantos opostos do tabuleiro, e são representados por um peão, peça distinta dos discos e distintas também entre si, que pode ser usado para deslocar os discos, criando stacks de 3 discos que são identificadas como sendo de um jogador específico por uma peça de cor diferente.

O primeiro jogador a criar 4 torres de 3 discos é considerado o vencedor. Caso não haja mais movimentos possíveis, o jogador com mais *stacks* é então o vencedor.



Figura 1. Estado inicial do tabuleiro



Figura 2. Jogador branco vence

Regras do jogo

Apesar de partir de um conceito simples, o Stac é um jogo com alguma complexidade devido às suas regras, que podem ser encontradas online no seguinte endereço: <https://drive.google.com/file/d/0B1OsbF2YhZDrSnIOOTNvWjKxLWs/edit>.

As regras relativas ao movimento dos peões são:

1. Os peões só se podem movimentar na linha ou na coluna em que se encontram.
2. Um peão não se pode deslocar para a casa onde já esteja outro peão.
3. Os peões apenas podem passar por cima de outro peão caso não estejam a carregar um disco.

Relativamente ao movimento dos peões e construção de stacks, aplicam-se as seguintes regras:

1. Um peão pode deslocar o disco da casa onde se encontra, apenas se este for o único disco nessa mesma casa.
2. Um disco apenas pode ser deslocado para uma casa vazia, uma casa com um disco, ou uma casa com dois discos.
3. Um jogador não pode deslocar o mesmo disco dois turnos consecutivos.

As regras relativas às stacks são:

1. Assim que um disco seja deslocado para uma casa já com dois discos, o jogador deve marcar essa stack como sua, através de uma peça que identifica a sua cor (branco/preto).
2. Um jogador pode deslocar-se para uma stack desde que não esteja a carregar um disco.

De forma a terminar o jogo têm-se as seguintes regras:

1. O primeiro jogador a obter 4 stacks diferentes é declarado o vencedor, terminando assim o jogo.
2. Se não houver mais movimentos possíveis, o jogador com mais stacks é declarado o vencedor.
3. O jogo termina em empate caso nenhum dos jogadores consiga ganhar via um acordo.

Lógica do jogo

Representação do estado do jogo

De forma a representar o tabuleiro do Stac na linguagem PROLOG, foi necessária a definição de alguns símbolos:

- vv - simboliza uma casa do tabuleiro vazia;
- p1 - simboliza uma casa do tabuleiro com um disco;
- p2 - simboliza uma casa do tabuleiro com dois discos;
- a0 - simboliza o jogador A, numa casa sem qualquer disco;
- a1 - simboliza o jogador A, numa casa com um disco;
- a2 - simboliza o jogador A, numa casa com dois discos;
- a3 - simboliza o jogador A, numa casa com três discos;
- b0 - simboliza o jogador B, numa casa sem qualquer disco;
- b1 - simboliza o jogador B, numa casa com um disco;
- b2 - simboliza o jogador B, numa casa com dois discos;
- b3 - simboliza o jogador B, numa casa com três discos;
- a4 - simboliza uma stack pertencente ao jogador A;
- b4 - simboliza uma stack pertencente ao jogador B;

Visualização do tabuleiro

De forma a apresentar o tabuleiro de jogo na consola, foi desenvolvida o predicado `showBoard(Board)`, que tem como argumento uma lista de listas, isto é, a representação em lista do tabuleiro.

```
p1 | p1 | p1 | p1 | a1 |  
p1 | p1 | p1 | p1 | p1 |  
p1 | p1 | p1 | p1 | p1 |  
p1 | p1 | p1 | p1 | p1 |  
b1 | p1 | p1 | p1 | p1 |
```

Figura 3. Representação do tabuleiro na consola

Lista de jogadas válidas

Foi criado o predicado `availableMoves(Board, Player, AvailMoves, MovesList)`, que aceita o argumento `Board`, representação em listas do tabuleiro, e o `Player`, jogador que se pretende saber a lista de jogadas válidas.

O predicado irá atribuir o número de jogadas válidas na variável *AvailMoves* e a respetiva lista de jogadas válidas na variável *MovesList*.

O resultado deste predicado é utilizado para verificar se existe uma situação de empate.

Execução de jogadas

De forma a mover os peões, foram desenvolvidas algumas funções que são utilizadas para escolher o movimento a executar, validar esse mesmo movimento e por fim executá-lo.

`choose_move_player(Board, Player, X, Y, Carry)` - este predicado aceita um `Board`, tabuleiro, o jogador atual, e define nas variáveis `X` e `Y` o movimento que pretende efetuar e na variável `Carry` define se o jogador pretende, ou não, deslocar um disco.

`carry(Player, CarryPlayer, Carry)` - define na variável `CarryPlayer` se o peão está, ou não, a carregar um disco, consoante o valor de `Carry`.

`remove_spawn(Board, [], FinalBoard, CarryPlayer)` - remove da casa inicial do tabuleiro o peão e um disco, caso o jogador pretenda deslocá-lo.

`move(FinalBoard, X, Y, [], FinalBoard1, CarryPlayer)` - efetua a jogada no quadro, mudando o peão de localização e adicionando um disco se for necessário.

`next_player(Player, NextPlayer)` - muda o jogador do atual para o próximo (A para B, ou B para A).

Avaliação do tabuleiro e final do jogo

O estado do tabuleiro é avaliado através do predicado `end_game()`, que define a variável `TowerNumbers`, que representa o número de stacks do jogador em questão. Quanto mais perto estiver de ter quatro stacks completas, mais perto está o jogador de ganhar o jogo.

O predicado, definido por `end_game(FinalBoard1, Player, 0, TowerNumber)`, verifica se o jogador atual venceu o jogo e define em `TowerNumber` o número de stacks que o jogador tem.

Jogada do computador

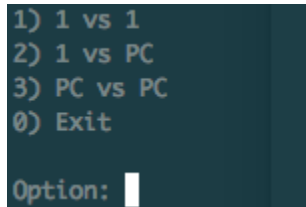
Para implementar a jogada do computador, foram desenvolvidos os seguintes predicados:

`pc_play(Board, Player, FinalBoard)` - o predicado gera 3 números aleatórios, para definir um movimento horizontal ou vertical, a casa para que pretende jogar e se pretende carregar a peça atual ou não.

Interface com o utilizador

Toda a interface foi feita com recurso às funções `write()` e `read()`.

O primeiro menu visualizado pelo utilizador é o seguinte:



```
1) 1 vs 1
2) 1 vs PC
3) PC vs PC
0) Exit
Option: |
```

Figura 4. Main Menu

Neste menu, o utilizador pode escolher entre jogar Jogador vs. Jogador, Jogador vs. Computador ou até mesmo Computador vs. Computador.

Foi implementada uma opção extra para sair para uma melhor usabilidade do programa.

Depois de iniciar o jogo, em todas as jogadas é pedida a casa de destino para a qual o jogador se quer deslocar, e é imprimida uma série de informações, como o jogador atual e o próprio quadro.

No caso de se estar num jogo Computador vs. Computador, é necessário fornecer um input ao programa, de forma a poder analisar todas as jogadas, passo a passo.

Conclusão

O trabalho realizado serviu como uma excelente aplicação prática da teoria que foi sendo aprendido ao longo do semestre, dando-nos uma perspetiva diferente sobre as vantagens do PROLOG face a outras linguagens já lecionadas no curso.

Infelizmente não nos foi possível implementar um nível de dificuldade superior, tal como seria esperado, pelo que essa seria a melhoria que seguramente implementaríamos caso tivéssemos tempo adicional.