Cage

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo Cage_1:
Rui Gonçalves - 201201775
Bruno Santos - 201402962

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

16 de Outubro de 2016

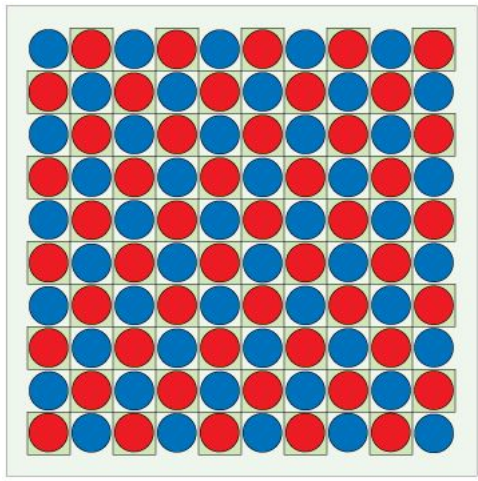
O Jogo Cage

História

Cage é um jogo de estratégia abstrato para dois jogadores. Criado por Mark Steere a Maio de 2010, cage é um jogo de tabuleiro finito e omnidirecional com uma duração média de 30 minutos e idade mínima recomendada de 12 anos.

Objectivo

O jogo é composto por um tabuleiro 10x10 e 100 peças (50 peças azuis; 50 peças vermelhas) de forma cilíndrica, como nas Damas. O objetivo do jogo é capturar todas as peças do adversário, de forma a que no tabuleiro apenas restem peças da nossa cor, evitando ao mesmo tempo que o adversário faça o mesmo, sendo por isso considerado um jogo de aniquilação. O jogo acaba quando um dos jogadores o fizer, sendo considerado o vencedor.



Regras

A imagem à direita representa o estado inicial do tabuleiro. O jogador Vermelho começa o jogo, movimentando uma das suas peças, seguido do jogador Azul, jogando assim por turnos. Se um jogador tiver uma jogada possível, ele tem de jogar, mas caso não tenha, este deve esperar até poder efectuar uma jogada, consoante as jogadas do adversário, sendo que garantidamente pelo menos um dos jogadores pode efetuar uma jogada.

Existem 4 tipos de movimentos possíveis no *Cage*, sendo que os jogadores apenas podem movimentar as suas peças de acordo com as regras impostas por esses movimentos. Esses movimentos são *restricted*, *centering*, *adjoining*, *jump*.

Restricted
No movimento *Restricted* existem duas restrições aos movimentos do jogador:

- Não se pode mover uma peça de modo a que esta fique ortogonal a outra peça da mesma cor, nem mesmo momentaneamente, aquando de uma jogada múltipla.
- Não se pode mover uma peça que tenha uma peça adversária ortogonal para uma casa onde não tenha, a não ser que esse movimento seja um Jump.

Centering
No movimento *Centering* uma peça pode ser movimentada para uma casa desocupada que seja adjacente (verticalmente, horizontalmente ou diagonalmente) que leve a que a peça movimentada fique mais perto do centro do tabuleiro, isto é, que a distância em linha reta entre a peça e o ponto central do tabuleiro diminua.

Adjoing
No movimento Adjoining uma peça pode, caso não tenha nenhuma peça adversária ortogonal, ir para uma casa adjacente em que passe a ter uma peça adversária ortogonal.

Jump
Por fim, no movimento *Jump* pode-se mover uma peça da nossa cor para capturar uma peça adversária se esta estiver ortogonal para uma casa ortogonal à peça adversária do lado oposto, deste que esta casa esteja desocupada. Assim que isto acontecer, a peça adversária deve ser retirada do tabuleiro.

É também possível efetuar este movimento para capturar uma peça adversária que esteja no limite do tabuleiro com uma peça da sua cor que esteja numa casa ortogonal à peça adversária do lado oposto ao limite do tabuleiro. Desta forma, ambas as peças são removidas do tabuleiro.

A jogada de capturar uma peça adversária com *Jump* não é obrigatória, a não ser que esta seja a única jogada possível naquele turno.

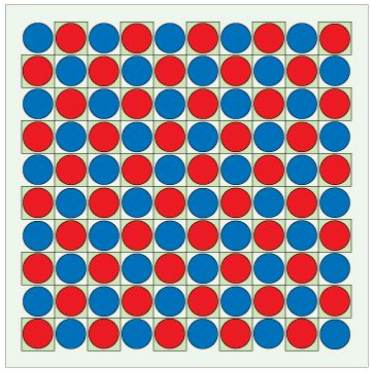
Se for possível usar a peça que usámos para efetuar um movimento Jump para realizar outro movimento Jump, somos obrigados a fazê-lo, as vezes que forem precisas, pelo que a jogada acaba quando não for possível efetuar mais jogadas *Jump*.

Se um jogador efetuar uma jogada *Jump* com a sua última peça para capturar a(s) última(s) peça(s) do adversário e é também removida do tabuleiro, ficando deste modo o tabuleiro sem peças, esse jogador ganha o jogo.

Representação do Estado do Jogo

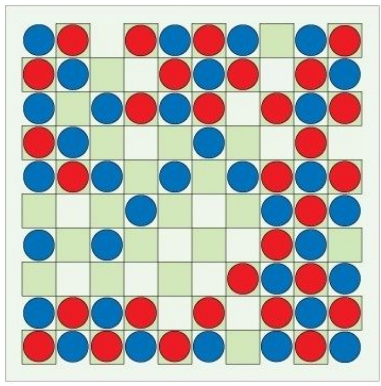
Inicialmente o tabuleiro terá todas as peças dispostas como se pode ver na figura anterior. Em qualquer estado do jogo, as casas apenas poderão ter uma peça azul, uma peça vermelha ou não ter peça nenhuma (casa vazia). Sendo assim, a representação inicial do tabuleiro em Prolog, utilizando uma lista de listas, será:

```
board(B) :- B=[[2,1,2,1,2,1,2,1,2,1],
              [1,2,1,2,1,2,1,2,1,2],
              [2,1,2,1,2,1,2,1,2,1],
              [1,2,1,2,1,2,1,2,1,2],
              [2,1,2,1,2,1,2,1,2,1],
              [1,2,1,2,1,2,1,2,1,2],
              [2,1,2,1,2,1,2,1,2,1],
              [1,2,1,2,1,2,1,2,1,2],
              [2,1,2,1,2,1,2,1,2,1],
              [1,2,1,2,1,2,1,2,1,2]].
```



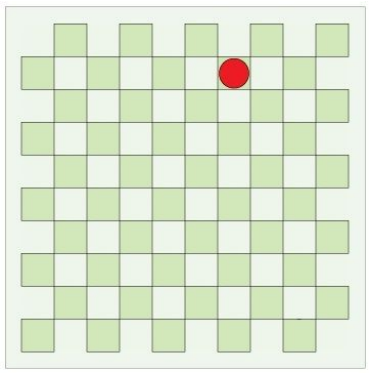
Uma representação intermédia do tabuleiro, tendo sempre em conta a legalidade das jogadas, poderá ser:

```
board(B) :- B=[[2,1,0,1,2,1,2,0,2,1],
              [1,2,0,0,1,2,1,0,1,2],
              [2,0,2,1,2,1,0,1,2,1],
              [1,2,0,0,0,2,0,0,1,0],
              [2,1,2,0,2,0,2,1,2,1],
              [0,0,0,2,0,0,0,2,1,2],
              [2,0,2,0,0,0,0,1,2,0],
              [0,0,0,0,0,0,1,2,1,2],
              [2,1,2,1,0,1,0,1,2,1],
              [1,2,1,2,1,2,0,2,1,2]].
```



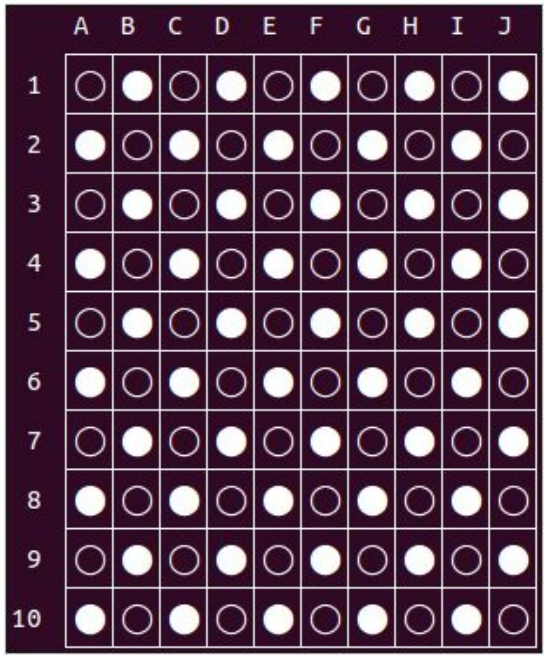
Por fim, uma representação final do tabuleiro, logo após a última jogada, poderá ser:

```
board(B) :- B=[[0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,1,0,0,0],
              [0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0,0,0]].
```



Visualização do tabuleiro

O tabuleiro de jogo será criado usando Unicode (através do predicado put_code) para representar tanto as casas como as peças. Recorre-se ao uso de letras e número para a representação de colunas e linhas, respectivamente. Exemplo de tabuleiro no princípio do jogo:



Para a construção do tabuleiro usou-se o predicado *print_board* (que permite a criação de tabuleiros com tamanho variável), e que recorre aos predicados *print_top_row*, *print_middle* e *print_bottom_row* para a impressão da estrutura do tabuleiro; e o predicado *print_house* para a impressão das casas, e que, posteriormente, ficará encarregue de apresentar no ecrã as peças no sítios certos.

Movimentos

Durante o jogo, os jogadores poderão recorrer aos 4 tipos de movimentos disponíveis (*restricted*, *centering*, *adjoining*, *jump*) para fazerem as suas jogadas. Assim, usaremos um predicado para representar cada um desses movimentos, com dois argumentos para representar as coordenadas para onde a peça será deslocada e um argumento que identificará o jogador a fazer esse movimento:

- restricted(Player,X,Y).
- centering(Player,X,Y).
- adjoining(Player,X,Y).
- jump(Player,X,Y).