Programação Declarativa: Trabalho Prático

7 Janeiro 2017

Hiago Oliveira - 29248 ; Gil Catarino - 32378

Introdução

O objectivo do trabalho é a implementação de um validador de jogos de xadrez dando uso às ferramentas que aprendemos a usar nas aulas. O programa aceita as jogadas na notação **Postal**, também conhecida como notação *Epistolar* ou *ICCF numeric notation*.
O programa faz a validação das jogadas uma a uma, sem qualquer decoração incluída, da forma "5254" (para avançar o peão do rei branco 2 casas, por exemplo). O analisador continua a pedir o input do user até que encontre o EOF, ou uma jogada inválida (fácilmente se altera para o programa voltar a pedir o input neste caso). Aquando da escolha de terminar a inserção de jogadas, é mostrado o estado do tabuleiro após todas as jogadas analisadas e validadas. O utilizador pode ainda escolher continuar a introduzir jogadas através do comando **play.**.

Desenvolvimento

Representação das Jogadas

As jogadas no trabalho são representadas por um predicado com 4 argumentos, guardado na forma **getplay(Ocol, Olin, Dcol, Dlin)**. Esta notação é a usada no resto do programa para conseguir aceder à jogada feita naquele turno. **Ocol** e **Olin** são a coluna e linha da origem da jogada, e por sua vez, **Dcol** e **Dlin** a coluna e linha de destino.

Representação do Tabuleiro

O tabuleiro é representado também através de um conjunto de predicados. Cada peça está representada com um predicado na forma **pos(Peça, Cor, Coluna, Linha)**, como por exemplo **pos(rei, branco, 5, 1)**. Desta forma é possível saber que peça está numa certa posição (pos(X, C, 5, 1)), onde está uma certa peça (pos(rei, branco, X, Y)), entre outras coisas (todas as peças numa certa linha, ou coluna, ou a localização dos bispos pretos, etc...). O conjunto de todas as peças forma o tabuleiro. A actualização do tabuleiro é feita através de retracts e asserts.

Codificação de jogadas permitidas

Na validação das jogadas é necessário verificar diversos requisitos para que esta seja aceite. Estes são:

Jogada no formato correcto

É necessário que a jogada seja feita através de 4 números, ou 5, no caso especial da promoção de peões. Todos os outros casos, a jogada é inválida, quando este passo é validado, é inserida a jogada na base de dados.

Jogada dentro do tabuleiro

Um passo óbvio porém que necessita de ser verificado na mesma. Uma jogada tem que ser feita dentro do tabuleiro 8x8. Jogar na posição (0,0) ou (9,9) é impossível.

Capturas amigas

É necessário também verificar que na origem e no destino da jogada estejam peças de equipas diferentes, caso contrário dá origem a um jogo completamente diferente...

Movimento permitido da peça

Outro passo muito importante é verificar que a peça escolhida pode ser movida para o destino desejado, e cada peça tem as suas regras. No caso da torre e do bispo, esta verificação é feita passo a passo (recursivamente) entre a origem e o destino, pois é necessário que não haja peças pelo caminho. A verificação da Rainha é feita como uma jogada de uma torre ou de um bispo, a que for válida.

Check?

No fim da jogada, tem de se ter a certeza que essa jogada não causa um check ao seu próprio rei.
Caso a jogada verifique todas estas condições, é aceite como uma jogada válida e é feita a jogada no tabuleiro, actualizando o estado deste.

Organização do Código

O código está organizado por predicados que fazem todas as verificações precisas de uma jogada, mencionadas na secção anterior. Existem 2 predicados principais que controlam o jogo, o **play**, que é responsável por chamar os predicados para obter uma jogada e validar, validação esta que é feita pelo outro predicado principal, o **handleplay**, que faz as verificações necessárias da jogada, e caso falhe, desfaz as alterações feitas na base de dados.

Pontos Fortes e Fracos

Um dos pontos fortes do trabalho, mencionado em secções anteriores, é o facto de ser possível aceder facilmente a todas as peças e/ou grupos de peças que sejam necessários para fazer algo. Tivesse optado pelo uso de um array bi-dimensional ou algo do género seria necessário percorrer este cada vez que fosse preciso aceder a uma peça.
Um dos pontos fracos do programa é este não ter implementado a captura *en passant*.

Conclusão

A realização deste trabalho ajudou muito na maneira de perceber melhor como funciona a programação em lógica e as suas vantagens. É uma maneira de programar diferente da que tinhamos aprendido até agora (programação imperativa, por objectos...). Com mais tempo seria possível implementar a captura *en passant* e melhorar a forma como é feita a verificação do rei em check, pois a forma implementada é uma forma naïve de verificar, basicamente vê-se se alguma das peças da equipa adversária consegue atacar o rei, e isto é feito sempre. Seria possível fazer a verificação apenas uma vez e guardar tudo num buffer, que apenas seria recalculado para a peça que se mexe e para as peças que mudavam de posição e que influenciavam o caminho de outras (um peão que se mova que consiga abrir caminho para a rainha fazer check, por exemplo).