2022.Q2 - Paradigmas de Programação
Proposta de Projeto

Um Engine para Xadrez

Emilio Francesquini
e.francesquini@ufabc.edu.br

21 de julho de 2022

1 Introdução

Este texto descreve um possível projeto para a disciplina de Paradigmas de Programação, oferecimento do segundo quadrimestre de 2022. Note que o projeto é livre, esta descrição é apenas uma sugestão do que pode ser feito além de que notas pode-se esperar a depender da complexidade que for entregue.

O projeto pode ser feito em grupos de até 3 pessoas, contudo a complexidade esperada para projetos feitos em grupo é, naturalmente, maior do que aquela esperada de projetos individuais.

O prazo final para entrega do projeto é **às 23h59 do dia 28/08/2022**. Após este prazo, entregas com atraso serão aceitas mas receberão descontos segundo a tabela disponível em `https://haskell.pesquisa.ufabc.edu.br/22.q2.haskell/`.

2 Descrição do projeto

Neste projeto você (e sua eventual equipe) deverão desenvolver um *engine* de xadrez em Haskell. As regras e movimentações das peças devem seguir (e serão avaliadas) seguindo as regras oficiais disponíveis no site da Confederação Brasileira de Xadrez (`http://www.cbx.org.br/files/downloads/Lei_do_Xadrez.pdf`).

Ele deverá minimamente:

- Ter conhecimento do posicionamento das peças no tabuleiro, e como elas se movimentam pelo tabuleiro incluindo capturas de peças adversárias (não inclui roque, promoção ou *en passant*) (Artigos 2 e 3).
- Reconhecer que o jogo acabou (Artigo 5)
- A partir de uma posição no tabuleiro (estado) reconhecer todas as possíveis movimentações válidas, ou seja quais são os possíveis próximos estados a partir de um estado.
- Possuir alguma maneira de interação entre o usuário e o seu programa. Pode ser modo texto, pode ser modo gráfico, como preferir.

Existem alguns *milestones* para o desenvolvimento que, se atingidos na versão entregue, já dão uma ideia da nota que você pode esperar na sua avaliação. Claro, a nota pode variar a depender da qualidade do código e do grau de completude atingido em cada um dos *milestones* abaixo. Entre parênteses ao final de cada tópico vocês tem quantos pontos a funcionalidade vale (que pode sofrer deduções, conforme explicado) para projetos feitos de maneira individual/grupo:

1. Os 4 tópicos mínimos descritos acima. (3/2)
2. Ter conhecimento/saber lidar e avaliar as regras avançadas como: pequeno/grande roque (incluindo a avaliação das movimentações passadas e ameaças para determinar se o roque é válido), promoção (para qualquer peça), *en passant* (avaliando inclusive se a captura deixou de ser válida pois não é o lance imediatamente após a movimentação do peão). (3/2)
3. Interface onde seja possível que dois jogadores humanos joguem entre si (com uma visualização do tabuleiro). A interface pode ser modo texto (2/1) (por exemplo, usando caracteres Unicode para desenhar o tabuleiro e notação algébrica - veja Apêndice C - pra indicar movimentações) ou modo gráfico. (3/1)
4. Uma IA, ainda que precária (pode ser até um movimento aleatório, desde que válido), que permita que o usuário jogue contra o computador ou que permita que o computador jogue consigo mesmo. (2/1)
5. Uma IA um pouco mais elaborada que faça uma busca na árvore de soluções, que use heurísticas de busca, etc. Outra alternativa é juntar uma IA precária com um banco de dados de aberturas como o

do `https://github.com/pychess/chess_db`, `https://github.com/pychess/scoutfish` ou se você quer pirar mesmo, o Caïssabase com mais de 5 milhões de jogos (`http://caissabase.co.uk/`) (3/2)

6. Implementar a UCI (`http://wbec-ridderkerk.nl/html/UCIProtocol.html`). UCI é uma interface modo texto que pode ser usada para se fazer a comunicação entre *engines* de Xadrez e interfaces gráficas como o PyChess (`https://pychess.github.io/`) ou o Arena (`http://www.playwitharena.de/`). Estas duas interfaces em particular permitem a configuração de diversos *engines* permitindo inclusive colocar o seu engine para jogar contra o Stockfish (`https://stockfishchess.org/`) :). Especificamente, o Arena tem suporte a debug (basta apertar F4) para comunicação com UCI, então caso vá fazer este milestone eu sugiro esta escolha. Este tópico, caso implementado, torna o tópico 3 irrelevante, ou seja, faça apenas o 3 ou o 6. (6/5)

Como descrito acima, esta é apenas uma sugestão de projeto. O projeto é livre! Fique a vontade para adicionar ou remover funcionalidades. A lista de pontuação por funcionalidades acima pode servir, inclusive, para balizar quanto uma funcionalidade poderá valer durante a avaliação. Note, contudo, que o professor mantém a sua autonomia para decidir sobre a complexidade/qualidade do código e consequente pontuação durante a correção.