Nome: Carlos Henrique Alencar Lima - RA: 11202021040
Nome: Guilherme de Sousa Santos - RA: 11201921175

Qual é o seu projeto?

O projeto é um jogo de Shogi, conhecido como Xadrez Japonês, desenvolvido em Haskell e projetado para ser executado no terminal. Para esta entrega final, as peças foram rotuladas em japonês, assim como é o jogo na realidade.

Como utilizar o seu código?

Para executar o projeto, basta utilizar o comando **stack run** no terminal. Esse comando inicializa o jogo, permitindo ao usuário começar a jogar Shogi diretamente na interface do terminal. Vale lembrar que, nesta versão, o jogador B (lado inferior do tabuleiro, ou de cor azul) começa o jogo.

Dificuldades, surpresas e destaques do seu código:

Para esta segunda entrega, a primeira dificuldade encontrada e superada foi a refatoração do código para aplicar os conceitos de Monad State. A motivação para essa mudança reside no fato dela permitir o gerenciamento do estado do jogo de maneira funcional e eficiente. Como no Shogi alguns elementos são modificados ao longo das rodadas (tabuleiro, peças capturadas e jogador da vez), o Monad State fornece uma abordagem limpa e de fácil compreensão para manipular essas mudanças no estado. Entretanto, como o jogo necessita usar operações de input (coletar entradas dos jogadores) e output (printar o estado do jogo na tela), o simples uso do Monad State não foi suficiente. Para isso, foi utilizado o Monad State Transformer (StateT), que permitiu a combinação da mudança de estado e efeitos de IO através do embrulho do StateT no Monad IO. Tal conhecimento foi obtido através de pesquisas no StackOverflow e Claude AI. Em seguida, foi feita toda a refatoração do código para torná-lo adaptado ao uso de Monad State.

O Monad State permitiu melhorar a escrita do código, removendo o encadeamento excessivo de estruturas ‘case of’ observado na entrega 1, e torná-lo mais legível e de fácil manutenção.

A maior dificuldade encontrada na refatoração das funções de check e checkmate foi a dependência da lógica de movimentação, que estava alterando o estado original do tabuleiro através da função modify. Na primeira entrega, como o tabuleiro não era modificado com o Monad State, foi complicado compreender a lógica necessária para realizar testes de movimentos e verificações de escape sem alterar o tabuleiro original na segunda entrega.

Em resumo, a aplicação de Monad State foi desafiadora visto a quantidade de código que precisou ser ajustado, mas realmente se fez necessária para permitir a mudança de estados eficiente e transparente.

Destaca-se que o projeto tem uma gama de validações o que permite que ele funcione sem qualquer evento inesperado durante as partidas, incluindo inputs no formato incorreto ou tentativas de burlar as regras do jogo.

Este vídeo contém uma demonstração um checkmate rápido no Shogi e, para simular os movimentos do vídeo no projeto em Haskell, basta seguir a seguinte sequência de inputs: 7h 6h, 3g 4g, 7c 6c, 3a 4a, 9d 8c, 1e 2f, 9g 8f, 2f 2g, 6h 5h, 1c 2d, 5h 4h, 3h 4h, 8h 4h, 2h 8b, 9c 8b, 3e 4e, repor, 1 3h, repor, 2 3g, 4h 4g, repor, 1 8h, 9f 9g, 8h 9h, 9g 9h, repor, 1 2f, 4g 6g, 2g 3h, 6g 6h, repor, 1 5h, 6h 5h, 3h 2i, repor, 1 6e, 2f 4g, 6e 4g.

Link do vídeo do projeto: https://youtu.be/bV52Wwes2kE