Jogo de Damas em Haskell

Relatório de Implementação

Desenvolvido por:
Maycon Douglas Henrique da Silva Gomes

1. Introdução

Este relatório tem como propósito documentar o desenvolvimento de um Trabalho Prático da disciplina Linguagens de Programação (DCC019) da Universidade Federal de Juiz de Fora (UFJF), cujo objetivo foi exercitar conceitos de programação funcional em Haskell estudados em sala de aula através da implementação do Jogo de Damas.

Este é um jogo de tabuleiro estratégico disputado entre dois jogadores, em um tabuleiro 8x8 com casas alternadas em preto e branco. Cada jogador controla doze peças posicionadas nas casas escuras das três primeiras fileiras do seu lado do tabuleiro. Vence o jogo quem capturar todas as peças do adversário. Foram implementados dois modos de jogo: Jogador vs Máquina e Máquina vs Máquina.

Nas seções seguintes, são descritas a arquitetura do sistema e as principais etapas do desenvolvimento, incluindo a geração do tabuleiro, a lógica de movimentação e as condições de término do jogo.

2. Objetivos

2.1. Objetivo Geral

Aplicar e aprofundar os conhecimentos em programação funcional por meio do desenvolvimento de uma aplicação em Haskell que simule o jogo de Damas, com foco na prática de conceitos fundamentais da linguagem, como imutabilidade, recursão, definição de tipos, pattern matching e composição de funções.

2.2. Objetivos Específicos

Para alcançar o objetivo geral proposto, este trabalho busca atingir os seguintes objetivos específicos:

- Implementar a estrutura do tabuleiro 8X8, utilizando somente caracteres ASCII para representar visualmente as casas e as peças, com identificação única de

cada quadra por meio da combinação entre letras (colunas A-H) e números (linhas 1-8).

- Desenvolver a lógica de posicionamento inicial das peças, de acordo com as regras formais do jogo de damas, dispondo os conjuntos J1 (Jogador 1) e J2 (Jogador 2) nas três primeiras e três últimas linhas, respectivamente.
- Implementar a lógica de movimentação das peças simples, permitindo apenas movimentos diagonais para frente em direção ao lado do oponente, respeitando as regras de ocupação das casas.
- Incorporar a lógica de promoção para dama, promovendo automaticamente uma peça simples ao alcançar a linha de coroação do lado oposto, e atribuindo a ela movimentos ampliados em qualquer direção diagonal.
- Desenvolver o sistema de capturas, tanto para peças simples quanto para damas, incluindo:
    - a verificação de capturas válidas (simples e múltiplas)
    - a regra de obrigatoriedade da captura
    - A priorização da jogada que captura o maior número de peças
    - o adiamento da remoção das peças capturadas até o fim da sequência de captura
- Criar um controle de turnos, alternando corretamente os jogadores após cada jogada válida.
- Rejeitar movimentos inválidos, informando ao jogador sobre o erro e requisitando um novo comando até que uma jogada válida seja fornecida.
- Permitir dois modos de jogo: Jogador vs Máquina e Máquina vs Máquina, com opção de escolha de quem inicia a partida.
- Implementar uma estratégia automatizada para a máquina, evitando decisões puramente aleatórias.
- Garantir a atualização contínua da interface textual, imprimindo o estado atualizado do tabuleiro a cada jogada.

3. Estrutura do Código

A estrutura do projeto está divida em:

- Main.hs

    - Contém o fluxo principal, menu inicial e loop do jogo.
- Posicao.hs
    - Contém funções relacionadas à leitura e conversão de linhas e colunas, de acordo com a entrada do jogador e com a estrutura interna do tabuleiro.
- Tabuleiro.hs
    - Contém a definição dos novos tipos de dados Peca, Casa e Jogador, sinônimos Linha (lista de Casas) e Tabuleiro (lista de Linhas), funções relacionadas aos tipos Peca, Casa e Jogador e funções de geração e exibição do Tabuleiro.
- Movimento.hs
    - Contém funções relacionadas a movimentação e captura das peças do jogo.
- Maquina.hs
    - Contém funções relacionadas às jogadas da Máquina, definindo a melhor estratégia de jogo de acordo com a situação da partida.

4. Detalhes de Implementação das Funcionalidades

4.1. Fluxo do Jogo

O fluxo principal do jogo combina:

- Controle de turno dos jogadores
- Identificação se o jogador é humano ou máquina
- Leitura das posições de Origem e Destino
- Leitura das opções selecionadas pelo usuário
- Verificação se movimento é válido
- Verificação se peça escolhida pertence ao jogador atual
- Verificação das melhores jogadas possíveis
- Realização dos movimentos
- Atualização e exibição do tabuleiro no terminal

O loop é encerrado quando todas as peças de um jogador foram capturadas, decretando a vitória para o jogador que ainda possui peças no tabuleiro.

4.2. Tipos de Dados e Sinônimos

Novos tipos de dados e sinônimos foram criados para representar estados do jogo.

- Peca
  - Tipo de dado que representa a peça de um jogador.
  - Pode variar entre peça comum, Dama ou Semicapturada.
  - A Semicapturada é usada para representar uma peça que será capturada ao fim de uma sequência de capturas.
  - Deriva dos tipos Eq e Show:
    - Eq -> Derivação utilizada para que permita comparar duas peças.
    - Show -> Derivação utilizada para que permita exibir a peça no Terminal.

```
data Peca = PecaJogador1 | PecaJogador2 | DamaJogador1
| DamaJogador2 | Semicapturada Peca
    deriving (Eq, Show)
```

- Casa
  - Tipo de dado para representar uma casa do Tabuleiro.
  - Pode variar entre Vazia e Ocupada Peca
  - Deriva dos tipos Eq e Show:
    - Eq -> Derivação utilizada para que permita comparar duas casas.
    - Show -> Derivação utilizada para que permita exibir a casa no Terminal.

```
data Casa = Vazia | Ocupada Peca
    deriving (Eq, Show)
```

- Jogador
  - Tipo de dado para representar um jogador

- Pode variar entre Jogador1 e Jogador2
- Deriva dos tipos Eq e Show:
  - Eq -> Derivação utilizada para que permita comparar duas casas.
  - Show -> Derivação utilizada para que permita exibir a casa no Terminal.

```
data Jogador = Jogador1 | Jogador2
    deriving (Eq, Show)
```

- Linha
  - Sinônimo usado para representar uma Linha do Tabuleiro.
  - Significa que é uma Lista de Casas.

```
type Linha = [Casa]
```

- Tabuleiro
  - Sinônimo usado para representar um Tabuleiro.
  - Significa que é uma Lista de Linha, ou seja, uma Lista de Listas de Casas.

```
type Tabuleiro = [Linha]
```

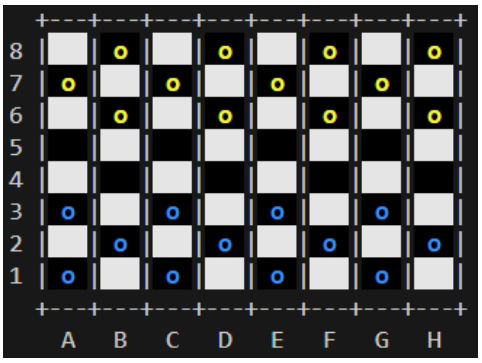
4.3. Representação do Tabuleiro internamente e no Terminal

O tabuleiro foi representado no terminal utilizando caracteres ASCII para desenhar suas linhas, limites e conteúdo. Além disso, foi utilizado o código ANSI para colorir alguns caracteres no terminal.

As principais funções utilizadas para exibição do Tabuleiro no Terminal são *mostrarTabuleiro*, *mostrarLinha* e *mostrarCasa*.

Internamente, o tabuleiro é constituído por uma lista de Linhas, ou seja, lista de listas de Casas. Inicialmente, a função *tabuleiroInicial* é utilizada para criar a primeira versão do tabuleiro, posicionando as peças de acordo com as regras estabelecidas. A cada jogada, o tabuleiro é atualizado de acordo com a movimentação realizada pelo usuário. Essa atualização ocorre através, principalmente, da função *atualizarCasa*.

Peças comuns são representadas pelo caractere 'o' e peças Dama são representadas pelo caractere 'D'. Peças do Jogador 1 são azuis, Peças do Jogador 2 são Amarelas. Peças Semicapturadas são exibidas com a cor Branca.



4.4. Movimentação das peças comuns

A movimentação das peças comuns é configurada para que seja permitido somente movimentos diagonais para frente (direção rumo às fileiras iniciais do oponente), respeitando as regras de ocupação das casas. Portanto, se tiver uma peça impedindo o movimento, a única forma de prosseguir é realizando a captura, mas para isso é necessário que a outra peça seja do oponente e a casa de destino seja livre e a seguinte, logo após a peça do oponente, naquela mesma diagonal.

As principais funções referentes a movimentação das peças comuns são *movimentoSimplesVálido*, *moverPeçaComum*, *movimentoCapturaVálido* e *capturarPecaSimples*.

4.5. Promoção para Dama

A promoção para Dama só pode ocorrer se ao final de uma jogada a peça do jogador se encontra na linha de coroação válida (última linha do lado oposto). A partir do momento que uma peça comum é coroada, ela passará a ter seu valor atualizado para que permita seguir as regras da Dama durante as próximas rodadas, que são:

- movimentar livremente nas diagonais referentes a casa atual da peça
- realizar capturas pulando casas vazias até alcançar uma peça do oponente

- é necessário que exista uma casa vazia naquela mesma direção após a peça do oponente
- não pode ter peças do jogador nesse trajeto

A avaliação de coroação de uma peça é realizada pela função avaliarPromocaoParaDama e é chamada ao final de jogadas, como por exemplo nas funções *moverPecaComum*, *capturarPecaSimples* e *fazerCapturas*. Se for o momento da coroação, a função *promoverParaDama* é executada.

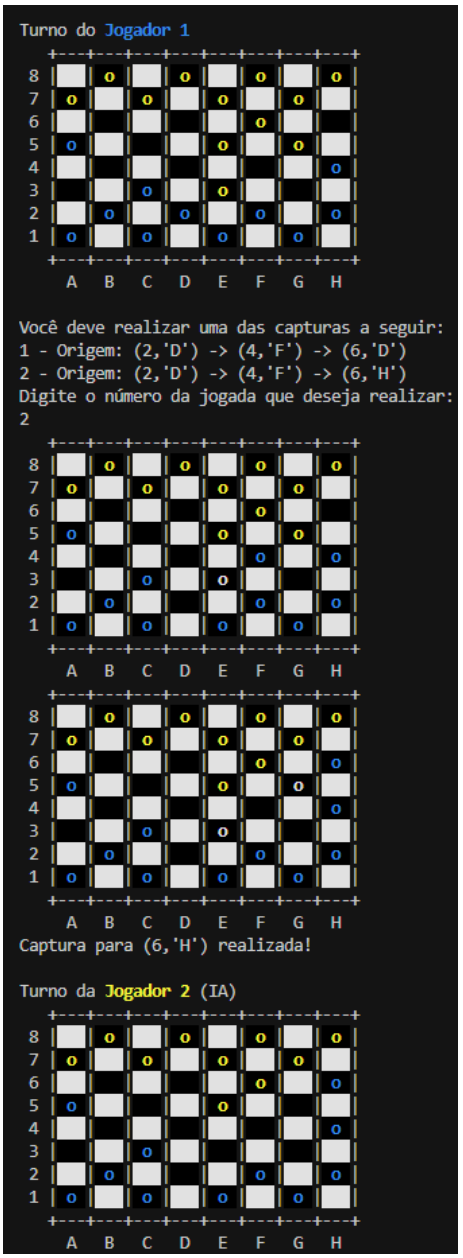
Ocorre apenas ao final de jogadas pois se uma peça passar pela linha de coroação durante uma sequência de capturas não deve ser coroada, exceto se a posição final após a sequência for na linha de coroação.

4.6. Sequência de Capturas

O usuário deve sempre optar por uma jogada onde realiza o maior número de capturas possível para aquela rodada. Portanto, o sistema realiza as possibilidades e retorna ao usuário as opções de sequências que cumprem essa requisição, permitindo a ele que selecione a opção que mais desejar (em caso de mais de uma opção).

O cálculo dessas possibilidades ocorre na função *melhoresCapturas* e o algoritmo consiste em coletar todas as peças do jogador atual com a função posicoesDoJogador e a partir de cada um delas gerar uma lista de sequências de capturas possíveis, usando a função *sequenciasCaptura*, que implementa uma busca em profundidade. Uma vez finalizada, essa lista passa por um filtro para que crie uma nova lista composta somente por sequências do maior tamanho encontrado na busca.

Além disso, a remoção das peças capturadas durante uma sequência deve ser realizada somente ao final da execução da sequência. Por isso, o tabuleiro é exibido no terminal a cada captura realizada, destacando a peça capturada com a cor branca e alterando seu valor para Semicapturada. Dessa forma, ao final da execução, é usada a função *removerSemicapturadas*.



4.7. Movimentos inválidos

O jogo não deve permitir movimentos para fora do tabuleiro, movimentos para casas brancas, movimentos que sobrepõe outra peça e movimentos com peças que não pertencem ao jogador. Portanto, ao longo da execução da função loopJogo verificações são realizadas para garantir que essas exigências sejam cumpridas e em caso negativo, mensagens são exibidas ao usuário para que corrija a sua entrada.

As entradas do usuário podem se dar de duas formas:

- movimento sem captura: número da linha + letra da coluna
    - exemplos válidos:

        - 6b
        - 6B
    - exemplos inválidos:
        - 6 b
        - 6 + B
- Movimento com captura: Opção de captura
    - exemplos:
        - 1
        - 2

Isso ocorre para que obrigue o usuário a capturar peças quando possível e também que capture o maior número de peças naquele momento.

4.8. Modos de jogo

O jogo permite ao usuário escolher dois modos, um em que ele joga contra a máquina e outro em que ele assiste a partida entre duas máquinas. Essa escolha é realizada no menu inicial. Outra escolha realizada no início do jogo é a de qual jogador começa a partida, através da função *escolherInicio*.

As mesmas regras que são aplicadas ao usuário são aplicadas à máquina, entretanto, algumas das principais funções da máquina foram registradas separadamente, para que uma estratégia automatizada pudesse ser aplicada às jogadas da máquina.

No início do loop do jogo, é verificado se o jogador atual é uma máquina, caso seja, a função *jogadaIA* será executada. Caso contrário, é requisitado ao usuário ações para executar sua jogada.

4.9. Estratégia automatizada para Máquina

Foi adotado um conjunto de estratégias baseadas em regras heurísticas e simulação de jogadas. O comportamento foi projetado para respeitar as regras do jogo, tomar decisões inteligentes e evitar movimentos prejudiciais.

No ato da captura de peças, quando ocorre um empate no número de peças capturadas, é avaliado se o destino final da peça ficará ameaçado pelo oponente e também se o destino final promoverá a peça comum em uma Dama. A partir da melhor pontuação, caso tenha empate, escolhe um aleatoriamente para evitar previsibilidade.

Quando não há capturas possíveis, cada movimento é avaliado por uma função heurística que leva em conta se a peça ficará sob ameaça após o movimento (usando a função *destinoAmeacado*), e chance de promoção a dama . Essas estratégias são aplicadas na função

avaliarMovimento. Em caso de empate na pontuação dos movimentos, a escolha é feita aleatoriamente entre os melhores movimentos.

As previsões de capturas futuras e ameaça de ser capturada são realizadas graças a simulações de tabuleiro que são realizadas através das funções *simularSequenciaCaptura*, *simularCapturaSimples* e *simularCapturaDama*.

Para a aplicação de escolhas aleatórias diante de uma lista de melhores movimentos sem e com captura, foi necessária a instalação da biblioteca *System.Random*, permitindo o uso da função *randomRIO*.

5. Conclusão

O desenvolvimento do jogo de Damas em Haskell proporcionou experiência prática na aplicação de conceitos de programação funcional. Ao longo do projeto, foram explorados aspectos importantes da linguagem Haskell, como imutabilidade, recursão e manipulação de tipos personalizados.

A implementação das regras do jogo exigiu atenção especial às capturas obrigatórias e sequenciais, movimentos válidos e promoção correta das peças para damas. Além disso, destaca-se o desenvolvimento de estratégias de jogadas para a máquina, através da avaliação de risco, simulação de resultados e priorização de promoções. Apesar de simples, a máquina apresenta um comportamento convincente durante uma partida.

O projeto atingiu seus objetivos e demonstrou a viabilidade de se implementar um jogo de tabuleiro completo em Haskell, reforçando o potencial da linguagem funcional.