Trabalho Linguagem de Programação:
O Jogo dos Dados em Haskell

Celso Zacarias da Silva Junior - 202076003
Wellington Pereira Silva - 201935041

04 de Agosto de 2024

1 - Introdução

Neste trabalho, foi desenvolvido um Jogo de Dados utilizando Haskell. Para o empacotamento do projeto, foi utilizado o Haskell Cabal, uma ferramenta que facilita o *building* e execução do projeto assim como de suas dependências, simplificando assim a implementação do mesmo e aumentando a portabilidade.

2 - Implementação

A decisão de projeto para a máquina de estados do jogo dos dados e da mesa envolve a definição de tipos e funções para gerenciar e manipular dados em um ambiente de jogo. A estrutura principal do projeto é baseada nos tipos **Dado** (Type Dado = [Int]) e **Mesa** (Type Mesa = [Dado]), e inclui funções para manipular e exibir esses dados.

O tipo **Dado** representa um dado como uma lista de inteiros, onde a cabeça da lista indica a face superior do dado e o restante representa os lados das faces disponíveis para rotação.

A **Mesa** é uma máquina finita de estados, onde cada estado é representado por um **Dado**, cuja condição de saída é ausência de Dados, o que significa o fim do jogo.

Características da nossa state machine e seus loops

Funções Principais

1. **Geração e Rotação de Dados**:
    - `geraDado`: Cria um dado com uma face superior específica;
    - `rodaDado`: Cria um dado com uma face superior específica que seja válido com as restrições do jogo;
    - `rodaDadoAleatorio`: Cria um dado com uma face superior aleatória que seja válido com as restrições do jogo.
2. **Manipulação da Mesa**:
    - `removeDadoDaMesa`: Remove um dado da mesa pelo seu índice;
    - `rotacionaDadoDaMesa`: Dada uma mesa, rotaciona um dado específico na mesa para uma nova face.
3. **Jogabilidade**:

    - *jogadaJogador*: Função para que o jogador realize uma jogada na rodada;
    - *jogadaComputadorFacil* e *jogadaComputadorDificil*: Função para que o computador realize uma jogada na rodada, na dificuldade fácil e difícil, respectivamente.
4. **Condições de Vitória e Derrota**:
    - *eConfiguracaoVencedora1Dado* e *eConfiguracaoVencedora2Dados*: Determinam se uma configuração de dados é vencedora ou perdedora para um ou dois dados, respectivamente. Conforme especificação;
    - *eConfiguracaoPerdedora*: Dada uma mesa, verifica se a configuração dos dados é perdedora;
    - *podeFormarParesPerdedores*: Reconfigura a mesa em pares de dados, para verificar se a configuração é perdedora, conforme especificação.

3 - Compilação

Para o desenvolvimento deste projeto, foi utilizado o GHC na versão 9.4.7. O empacotador e gerenciador de dependências utilizado foi o Cabal nas versões 3.8.1.0 e 3.10.3.
Dependências: System.Random

Para executar o programa basta entrar no diretório raiz da aplicação e rodar pela linha de comando:

$ cabal run