Technical University of Munich
Chair for Logic and Verification
Prof. Tobias Nipkow, Ph.D.
J. Rädle, L. Stevens, K. Kappelmann, M. C. M. Eberl

WS 2019/20
20.12.2019
Deadline: 13.1.2020, 23:59

Functional Programming and Verification

Sheet 10

Homework

You need to collect 4 out of 5 Christmas stars (⋆) to pass this sheet.

Exercise H10.1 So This Is Christmas [1–3: ⋆, 4+5: ⋆, 6: ⋆⋆, 7: ⋆]

The Übungsleitung is drinking mulled wine, it is snowing on the submission system, in other words: it is Christmas. And just like every Christmas, the MC Jr is having a Christmas board game evening with his family. However, he got terribly *board* of playing "Settlers of Catan" and "Malefiz" every year and hence decided to introduce his lucky family to a new game this year: Domineering.

Domineering is a two person game played on an $n \times n$-board. The players have a collection of dominoes which they place on the grid in turn, covering up squares. One player places tiles vertically, while the other places them horizontally. Of course, one is not allowed to stack dominoes. The first player who cannot place another domino looses the game.

So, "why is he telling me all that?" you are asking? Well, the MC Jr has an ingenious plan that will make him the Christmas board game night champion of 2019, but he needs your help! He convinced his family that playing board games the traditional way is "uncool". Instead, the Christmas board game night of this year will be held on the MC Jr's new Linux-powered laptop running Haskell. What his family does not know is that the MC Jr will have some help of a very special Domineering-AI written by you.

To make this plan come reality, the MC Jr started writing a Domineering framework. However, just like every year, he forgot to buy Christmas presents on time and hence has to organise them *auf den letzten Drücker*. As a Christmas present to you, he copied his code to the template file so that you do not have to start from scratch.

1. Write a function `prettyShowBoard :: Board -> String` that creates a pretty string of a given board as exemplified by the following:

```
prettyShowBoard [] = ""
prettyShowBoard [[P H, P H],[E, E]] = "HH\n++\n"
prettyShowBoard [[P V, E],[P V, E]] = "V+\nV+\n"
prettyShowBoard [[E,P H,P H],[P V,E,E],[P V,E,E]]
  = "+HH\nV++\nV++\n"
```

   Note: You can then use `putStr $ prettyShowBoard b` to print a board `b` in ghci.

2. As a convention, we encode moves using a position (r, c) in the following way:

- If the horizontal player chooses (r, c), then positions (r, c) and $(r, c + 1)$ should be covered.
- If the vertical player chooses (r, c), then positions (r, c) and $(r+1, c)$ should be covered.

Write a function `isValidMove :: Game -> Pos -> Bool` that checks whether a move (as encoded by the position and current player) is valid. In particular, the move must not cover any already occupied spaces.

3. Write a function `canMove :: Game -> Bool` that checks whether the current player can play another move.
4. Write a function `updateBoard :: Board -> Pos -> Field -> Board` that updates the board at the given position with the given field value. You can assume that the function receives only valid positions.
5. Write a function `playMove :: Game -> Pos -> Game` that plays a move (as encoded by the position and current player) and returns the updated game. You can assume that the function only receives valid moves.
6. (**Competition**) Now let's turn to the most important exericse: write a Domineering-strategy `christmasAI :: Strategy` that chooses the next move for the current player. You can assume that your strategy is only used for 12×12 boards and is only called if the current player can play at least one more move. The strategy also receives an infinite list `rs :: [Double]` of random numbers between 0 and 1 as an input that you can use for probabilistic methods.

   Make sure your strategy is capable of beating the MC Jr's family, leading him to sweet victory! But beware: the MC Jr's family is really impatient and will simply stop playing but still claim victory if your strategy takes longer than one second per move. Moreover, for the competition, you will also have to compete against your fellow students.

   All submissions passing the exercise and containing the `{-WETT-}...{-TTEW-}` tags will automatically be simulated in regular intervals[1]. The most recent results (including fancy, animated gifs) will be available on our Domineering championship website that will be published in the upcoming days.

   Here are some possible methods to get started with a competitive strategy: Minimax, Alpha-beta pruning, Monte Carlo Tree Search, go complete crazy and use some Machine Learning techniques, or come up with your own home-brewed heuristics.

   As usual, the MCs would be very grateful if you leave some explanatory words for your strategy. If you have issues making your strategy work on the submission server, e.g. if you want to submit a (large) model for your machine learning approach, let us know on Piazza.

> **Important:** If you submit a competition exercise, you agree that we are allowed to publish your name as part of the competition on our website. If you just want

[1] depending on the number of submissions, this interval will be longer or shorter

to submit a competition exercise as part of your homework without taking part in the competition, you can just remove the {-WETT-}...{-TTEW-} comments of your submission.

7. Write a function

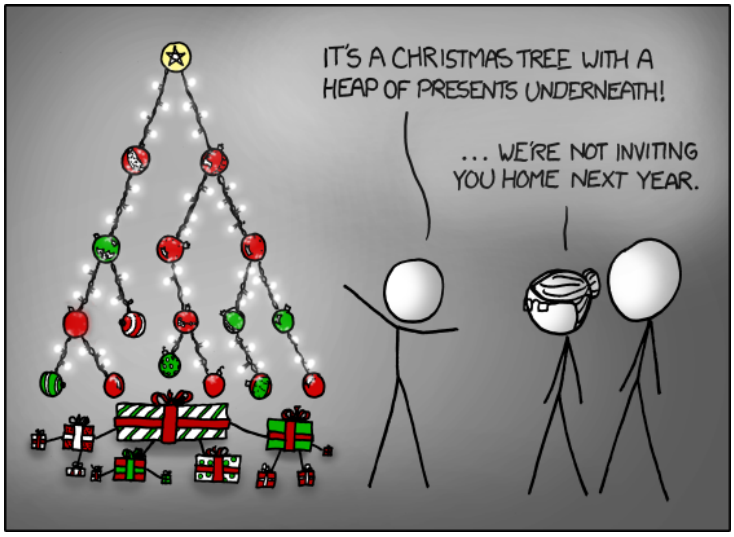
```
play :: [[Double]] -> Int -> Strategy -> Strategy ->
        ([Board],Player)
```

that receives a list of lists with random numbers `rss`, a dimension `dim`, and two strategies `sv, sh` and plays a game using strategy `sv` for the vertical and `sh` for the horizontal player starting from an empty `dim` × `dim` board. In each round, the first list of `rss` should be passed to the current player. You can assume that `rss` contains enough lists for the whole game. The function should return a list of all board states (start to end) and the winner of the game. By convention, the vertical player will start the game. A player looses the game if no valid move can be played or the player chooses an invalid move.

****

We all wish you a merry Christmas and a happy new year!

****



Source: https://www.xkcd.com/835/