Foreword

This assignment involved extending the Dominoes framework and Haskell even further to build intelligent players. These aimed to deal effectively with a variety of situations in a similar manner to a competitive fives-and-threes player. Here, I have tested the players' various tactics and the helper methods that comprise them.

Testing

`forTheWin`

`intelligentPlayer3` will play the winning `Dom` if it is within range and has a `Dom` with the exact score necessary to hit 61 and win.

`forTheWin` in action

As observed here in the final move of `domsMatch hsdPlayer intelligentPlayer3 100 9`, `intelligentPlayer3` recognises that it is within a potential 3-point move for the win. It has 58 points, and to place `(6,5)` at the left of this newly-reset board will win it the game.

***Key**: TACTIC: Scores | Board Left End Right End History | Chosen Move*

```
FTW: (52,58) Board (6,4) (6,4) [((6,4),P1,1)] ((6,5),L)
```

`denyAllMoves`

If `intelligentPlayer3` can attack on not only the opponent's knocks, but the entire set of unplayed `Doms` too, then it will choose to broaden its attack in this way.

`denyAllMoves` in action

As observed in these trace logs, `intelligentPlayer3` uses `denyAllMoves` when it knows for sure that it can deny any unplayed `Dom`, whether the opponent is knocking or the `Dom` is resting. The scores here are not shown for conciseness, but you can observe that the previous turn was taken by `intelligentPlayer3`, player 2 (turn 14 at the right). This shows that `intelligentPlayer3`'s opponent was successfully denied a move.

```
DNA:
Doms in play: [(4,3),(4,2),(3,3),(2,2)]
Chosen move: ((1,1),R)
DNY: Board (6,6) (1,1) [((6,6),P2,11),((6,4),P2,9),((4,1),P2,7),
                        ((1,5),P1,6),((5,5),P1,1),((5,4),P2,2),
                        ((4,4),P1,3),((4,0),P2,4),((0,5),P1,5),
                        ((5,6),P1,8),((6,0),P1,10),((0,0),P1,12),
                        ((0,1),P1,13),((1,1),P2,14)] ((3,1),R)
```

target59

As `intelligentPlayer3` closes in on 61, it uses the move that will take it closest to 59 if it cannot win directly.

target59 in action

As observed in these trace logs, `intelligentPlayer3` breaks into a `target59` strategy as soon as its score hits 53. This follows nicely into `forTheWin`, which takes priority and is used as soon as possible.

An additional predicate sits just below `forTheWin` which, after implementation, gave five additional wins when running `domsMatch intelligentPlayer3 hsdPlayer 100 94`.

```
DNY: (49,45) ((5,5),R)
TAR: (49,53) ((6,0),R)
TAR: (50,54) ((6,6),L)
TAR: (54,58) ((6,3),L)
TAR: (54,59) ((1,0),R)
TAR: (56,60) ((1,1),R)
FTW: (58,60) ((5,0),R)
DNY: (0,0) ((6,3),L)
```

denyMoves

If `intelligentPlayer3` is able to deny its opponent a move, using the `DomBoard History` to determine what they're knocking on, then it will. `intelligentPlayer3` will select a `Dom` that the opponent is knocking on to buy itself an extra turn.

denyMoves in action

In the logs below, `intelligentPlayer3` recognises that its opponent is knocking on

```
DNY: [3,2]Board (3,4) (2,0) [((3,4),P1,7),((4,0),P2,6),((0,6),P1,5),
                             ((6,6),P1,1),((6,1),P2,2),((1,3),P1,3),
                             ((3,3),P2,4),((3,2),P2,8),((2,1),P1,9),
                             ((1,4),P2,10),((4,5),P1,11),((5,2),P2,12),
                             ((2,0),P2,13)] ((5,0),R)
DNY: [3,2]Board (3,4) (0,5) [((3,4),P1,7),((4,0),P2,6),((0,6),P1,5),
                             ((6,6),P1,1),((6,1),P2,2),((1,3),P1,3),
                             ((3,3),P2,4),((3,2),P2,8),((2,1),P1,9),
                             ((1,4),P2,10),((4,5),P1,11),((5,2),P2,12),
```

```
                                ((2,0),P2,13),((0,5),P1,14)]
((5,1),R)
HSD: Board (3,4) (5,1) [((3,4),P1,7),((4,0),P2,6),((0,6),P1,5),((6,6),P1,1),((6,1),P2,2),((1
```

Aside: on `denyMoves`

What I originally considered a bug in `denyMoves` may actually be a feature. `canDenyMoves` is still `True` if the other `Player` isn't knocking on anything. `denyMoves` reduces to `hsdPlayer` in this case, as it simply chooses the highest scoring `Dom` from an unfiltered `Hand`. Testing with 100 games each on the following five seeds gave these results:

- **Seed 94 - -1**
- **Seed 95 - -4**
- **Seed 96 - +1**
- **Seed 97 - +1**
- **Seed 98 - -1**

Particularly with regards to the majority of negative results, I decided to keep this little foresight in, as it actually seems to pay off.

I find it equally strange that prioritising it over `target59` where getting `59` itself is not actually possible also results in higher scores, but this could be because `target59`'s predicate is much more general compared to that of `denyMoves` and may prevent it from playing more offensively against the other player. When the other player *is* knocking, `denyMoves` can attack in close-call situations where both players are in endgame and win `intelligentPlayer3` the game.

`clearOut`

When `intelligentPlayer3` has the lead in middlegame, it uses `clearOut`. `clearOut` uses the strongest `Dom` that has the most frequent number of spots in the hand on one or both sides.

`clearOut` in action

The `DomBoard` was reset to the `InitBoard` type before this move was made, and as such, the player could not determine the opponent's `Hand` from the `DomBoard History`. Hence, `intelligentPlayer3` resorted to `clearOut`, which made the highest-scoring move with the domino that had the most frequent number of spots, that being 6.

```
CLR: [(2,1),(3,0),(1,1),(5,3),(6,3),(5,0),(6,5),(6,6),(2,2)]
(35,45) ((6,6),L)
```

Aside: on `denyMoves` and `clearOut`

Upon testing, I discovered that when `checkKnocking` returns an empty list - that is, the opponent is believed to have `Dom`s of all spot counts - `denyMoves` reduces to `hsdPlayer`. I then added an extra conditional to `canDenyMoves` to ensure the player's `Hand` and the filtered `Hand` were not equal, in which case `canDenyMoves` would return False. In cases where the player was losing at that time, it would fall back to `clearOut`. By eliminating `clearOut` from `intelligentPlayer3` entirely, the performance of `intelligentPlayer3` *improved*, despite this being recommended as a path to take when developing the player - `clearOut` was clearly making moves that would lose the game.

For the reasons described above, `clearOut` and its associated functions are present but not used in `intelligentPlayer3`. Its previous implementation within `intelligentPlayer3` is commented out. However, upon further experimentation after making `target59` chase 59 rather than 60, it seems that `clearOut` is still a setback, hence it remains as described above, but the number of games its use loses for `intelligentPlayer3` varies.

Aside: on `denyMoves` and `target59`

It seems that in select cases, prioritising offensive plays against forcefully aiming for the necessary score can win singular games. For example, playing `domsMatch hsdPlayer intelligentPlayer3 100 85` with `denyMoves` prioritised over `target59` can win just one more game over `hsdPlayer`.

`hsdPlayer`

This is `intelligentPlayer3`'s final fallback when it is losing and cannot make any strong endgame moves. It is inherited from the original `domsMatch` code, and as such is not tested here.

Players

Description

intelligentPlayer

A more individual, unaware player that focuses on clearing out its hand and only plays offensively as a fallback. This player turned out somewhat weak - at first, I focused on creating just one intelligent player that was as strong as I could make it, so this was more of a stepping stone on that path that I'm keeping for comparison's sake.

intelligentPlayer2

A more offensive player that still tries to "clear out" dominoes it has a lot of if it has the lead. This helps it beat its predecessor, but interestingly, using `clearOut` when it has the lead has a questionable effect on play - in seed 64, it wins one game less, in seed 98, it ties, and in seed 196, it pulls an extra game from its superior.

It's clear that `clearOut` is something of a questionable strategy. Whether (theoretically) implementing it by playing only one number of spots where possible, or by playing the most frequent number of spots in the hand as I have here, `clearOut` is likely to leave its player knocking by concentrating or spreading the spot distribution across `Dom`s respectively.

intelligentPlayer3

intelligentPlayer3 makes use of the `Board` and its `History` a major priority, resorting to attacking as soon as it can't play a winning or endgame move, and only playing the highest scoring `Dom` if it can't do either. This reactive play makes it the strongest I have made.

Results

100 games were played between each of the three `domsPlayers`. Here are the number of victories each obtained in their matchups.

Seed 64

H/A	iP1	iP2	iP3
iP	NA	46	38
iP2	54	NA	37
iP3	62	63	NA

Seed 98

H/A	iP1	iP2	iP3
iP	NA	48	33
iP2	52	NA	33
iP3	67	67	NA

Seed 196

H/A	iP1	iP2	iP3
iP	NA	41	39
iP2	59	NA	40
iP3	61	60	NA