COM2001: Advanced Programming Topics

Assignment 3

**Design**

**Data Types**

In this assignment, one new data type is introduced.

**1. Tactic:** Each tactic represents a strategy to the game

type Tactic = Hand -> DomBoard -> Player -> Scores -> Maybe (Domino, End)

**Framework for Tactics**

A framework is designed so that the tactics can be easily added, removed or changed.

tactics = [Tactic]

tactics = [firstDropTactic,

aggressiveTactic,

comboToWinTactic,

matchPointTactic,

luckyWinTactic,

blockWinTactic,

drawGameTactic]

The skillPlayer will be given the list of tactics and return a **DomsPlayer**,

skillPlayer :: [Tactic] -> DomsPlayer

Therefore, when calling the function **domsMatch** in terminal, the skillPlayer will require a list of tactics to be provided in order to run,

\*Dominoes> domsMatch (skillPlayer tactics) hsdPlayer 1000 431

If an empty list of tactics is provided, then the skillPlayer will just become a hsdPlayer.

**Implementation of skillPlayer**

****

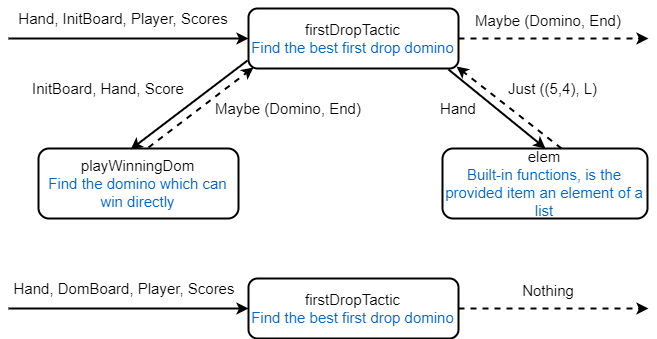
**Figure 1. Design of skillPlayer**

A good dominoes player (except hsdPlayer and simplePlayer) will be playing the game based on the given tactics. The framework is designed in a way that the player will try the tactics according to their order in the list. The player will decide his moves as below:

1. Each tactic in the tactics list will be given the **Hand**, **DomBoard**, **Player** and **Scores** in order.
2. Since the return type of **Tactic** is **Maybe**, therefore, a tactic will return **Nothing** if no suitable domino can be played using this tactic, or return **Just (dom, end)** if the tactic finds the best domino to be played at current game state.
3. Lastly, the player will choose the first **Just (dom, end)** in the list. This is because sometimes there will be more than one tactics that will return a **Just** result. Hence, the order of the tactics given to the player is important and will affect the results.
4. If all the tactics return **Nothing**, then the player will just become a hsdPlayer, and play the highest scoring domino which will not bust

**Tactics**

**1. firstDropTactic:** If the provided board is not an empty board **(InitBoard)**, return **Nothing**



**Figure 2. Design of firstDropTactic**

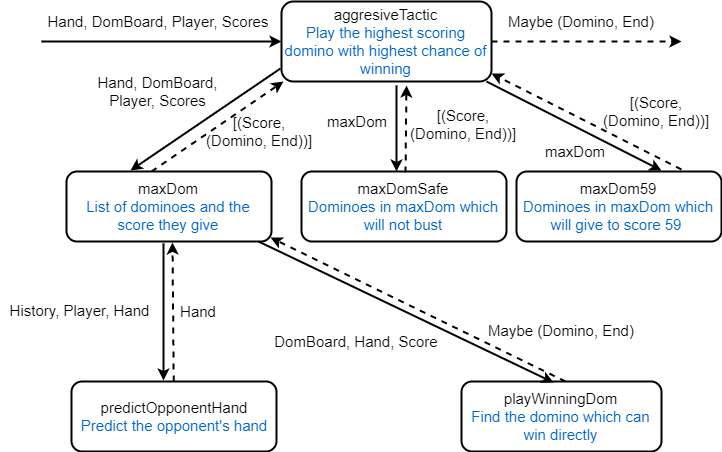
**Steps:** 1. It will check if the given **DomBoard** is not **InitBoard**, then return **Nothing**

2. Call **playWinningDom**, this is for the case where both the players have score near 61, and they start a new game with their scores carried over.

3. If **playWinningDom** returns **Nothing**, then check if the hand has **(5,4)**. If it has **(5,4)**, then play it.

4. The tactic returns **Nothing** if all the steps above fail.

**2. aggresiveTactic**



**Figure 3. Design of aggressiveTactic**

**Steps:** 1. Use list comprehension to create **maxDom**, where it contains a list of dominoes with their score in the form **(score, (dom, end)).** The dominoes in the list will be filtered out calling **predictOpponentHand** first, and then remove the domino that will give the opponent a chance to win in the next round.

2. Create **maxDomSafe**, which is a list of dominoes from **maxDom** but with more dominoes removed if the score they give will result in a bust.

3. Create **maxDom59**, which is a list of dominoes from **maxDom** but only contains dominoes that will score to 59

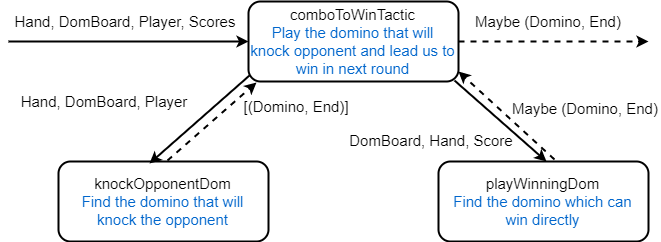
4. If the opponent score is less 53, then play the highest scoring domino in **maxDom**

5. If the player score is more than or equal to 53, then play the highest scoring domino in **maxDomSafe**

6. If **step 4** and **step 5** fail, then play the highest scoring domino in **maxDom59**

7. Otherwise, return **Nothing**

**3. comboToWinTactic**



**Figure 4. Design of comboToWinTactic**

**Steps:** 1. Call **knockOpponentDom**, which will return a list dominoes that can be played to knock the opponent

2. Call **playWinningDom**, the **DomBoard** provided to it will be the board after playing the dominoes in **knockOpponentDom.** The list will now be filtered out so only the dominoes that will provide an opportunity to win after playing it will remain.

**4. matchPointTactic**

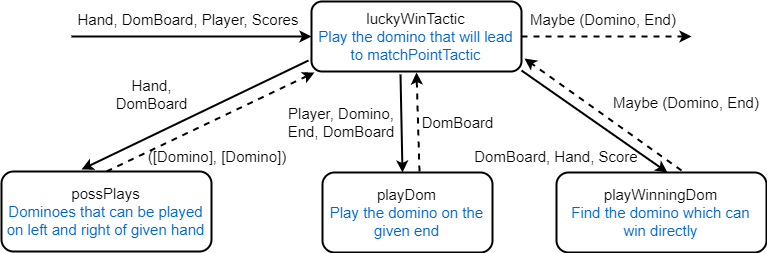
****

**Figure 5. Design of matchPointTactic**

**Steps:** 1. Call **playWinningDom**, which will return a domino to play at which end, and it will result in a win directly

2. If there is no such domino which will result in a win directly, return **Nothing**

**5. luckyWinTactic**



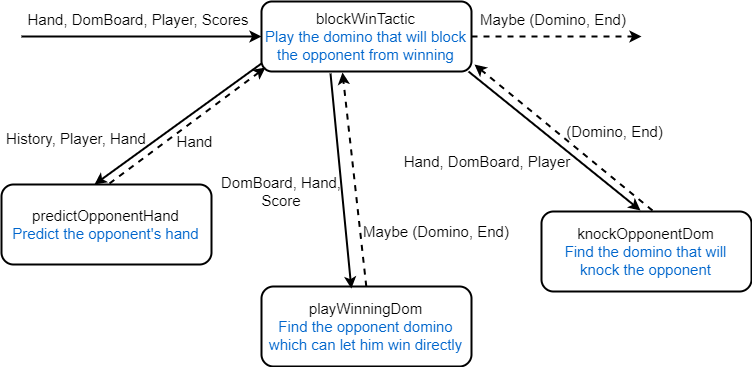
**Figure 6. Design of luckyWinTactic**

**Steps:** 1. Call **possPlays** to get a tuple of list of dominoes that can be played on left and right

2. Call **playDom** on those dominoes, return all boards after playing each of the dominoes

3. Call **playWinningDom**, which will filter and only keep those dominoes which might give an opportunity after playing it

**6. blockWinTactic**

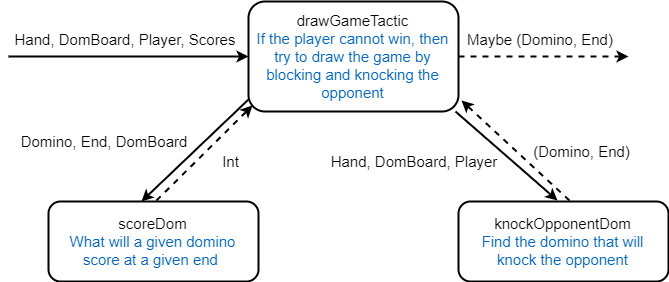
****

**Figure 7. Design of blockWinTactic**

**Steps:** 1. Call **playWinningDom**, which will return a domino to play at which end, and it will result in a win directly

2. If there is no such domino which will result in a win directly, return **Nothing**

**7. drawGameTactic**



**Figure 8. Design of drawGameTactic**

**Steps:** 1. Call **scoreDom** and create a list of the scores of each domino in the hand.

2. Call **knockOpponentDom** to get a list of dominoes that can knock the opponent.

3. If the minimum scoring domino will give a score that will bust, and the list from **knockOpponentDom** is not empty, then try to knock the opponent and to draw the game.

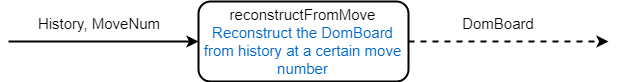
**Tactics Helper Functions**

**1. reconstructDomBoard:** Uses list comprehension to reconstruct the board by keeping the domino and removing the player and move number

****

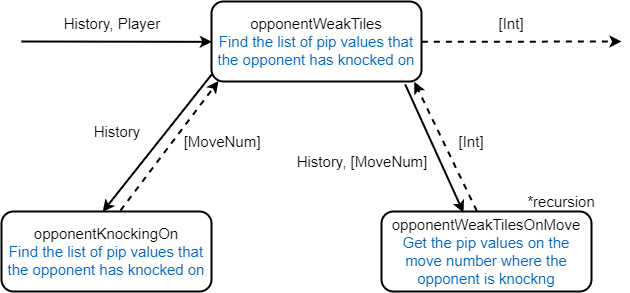
**Figure 9. Design of reconstructDomBoard**

**2. reconstructFromMove:** Uses list comprehension too but reconstruct it at a certain move

****

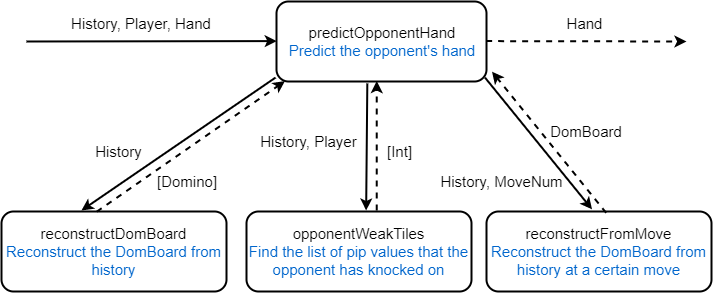
**Figure 10. Design of reconstructFromMove**

**3. opponentWeakTiles**

****

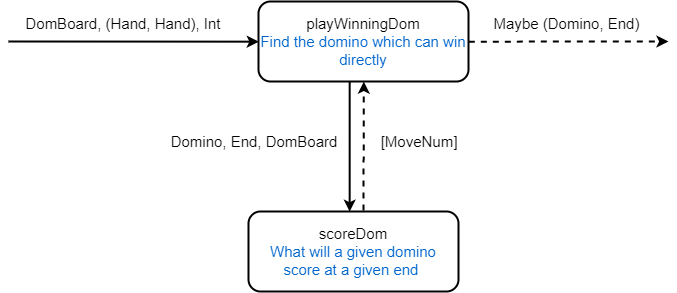
**Figure 11. Design of opponentWeakTiles**

**4. predictOpponentHand**

****

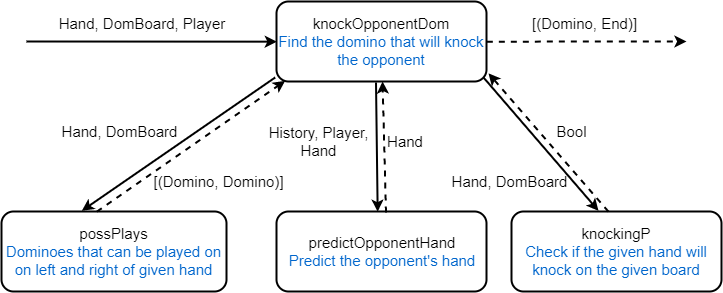
**Figure 12. Design of predictOpponentHand**

**5. playWinningDom**

****

**Figure 13. Design of playWinningDom**

**6. knockOpponentDom**

****

**Figure 14. Design of knockOpponentDom**