

MCS Project Part 1: Reversi (second chance)

Ingmar.Dasseville@cs.kuleuven.be
Bart.Bogaerts@cs.kuleuven.be

July 30, 2017

1 Introduction

In the MCS project, we will model a variant of the Reversi game¹.

The project consists of two parts:

- In this first part, you model the game in IDP. This should be done in the LTC formalism. We use this theory to verify some properties of the game.
- In the second part, you model this same game in Event-B and use CTL and LTL to verify other properties of the game.

In Section 2 we describe the game in general. Section 3.1 explains what you have to do for the first part of the assignment. The project for this course must be completed **alone**.

2 The game

For this year's project you are tasked with modelling **a variant of** the Reversi game. This is a board game, played on an unchecked board. The original game is played on an 8x8 board. However, our version is played on arbitrary boards. The game is played with game pieces called *discs* that are white on one side and black on the other side. These discs will be put on the board; when we refer to a *white disc*, we mean a disc placed on the board with the white side up (and similarly for black).

This is a two player game. Each player plays with one of the colors (white/black).

¹<https://en.wikipedia.org/wiki/Reversi>

2.1 Extra rules

Compared to Reversi as you know it, there are two new rules (for the second chance assignment):

- Whenever a player makes at least five discs change sides, he is allowed to play again.
- Instead of playing an extra disc, a player is also allowed to take away some of his own discs as an extra action.

2.2 The start of the game

We assume that some initial occupation of the board is given. Traditionally, the game starts with two white and two black discs placed in the center of the board, but your solution should work with any initial setup.

2.3 Trapped discs

We say that a disc d_1 of a given color is *trapped* between two discs d_2 and d_3 of the opposing color if the following hold:

- d_2 and d_3 are on the same horizontal, vertical or diagonal line on the board.
- d_1 is in between d_2 and d_3 on that line.
- all positions in between d_2 and d_3 are occupied by a disc the same color as d_1 .

For instance, in Figure 1, the discs on locations $b5, c5$ and $e5$ are trapped between the white discs on locations $a5$ and $f5$. The disc on location $c6$ is not trapped between any two discs.

2.4 Turns

The two players take turns. **However, sometimes (see later) a player can keep his turn.** A player can add a disc of their own color to the board on a given position, if there is at least one opposing disc trapped **between the newly placed disc and one disc already on the board.** For instance, in Figure 2, if it is blacks turn, he can *only* add a disc to the board on position $a7$, since that is the only location where a white disc (namely those one on positions $b6$ and $c5$) is trapped between the newly placed disk and another.

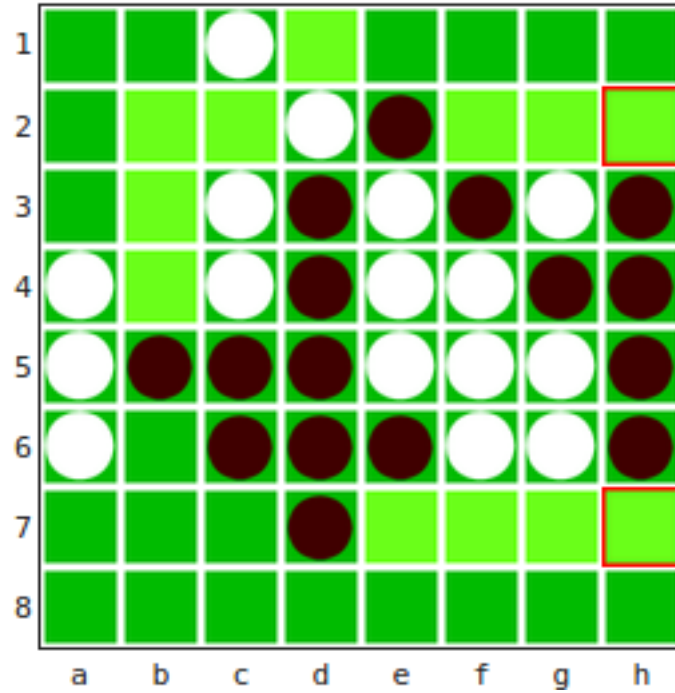


Figure 1: An example Reversi board.

If a player can play a disc, he *must* play a disc. If not, he can choose: either he *passes* or he **removes one of his own discs and all of his own discs connected (horizontally, vertically or diagonally) to that disc**. For instance, if white removes the disc on A5 in Figure 1, that same action also removes A4 and A6. If black removes the disc on D7 in that same figure, all of black's discs are removed.

When a player adds a disc, all opponent's discs that are trapped **between the newly placed disc and any other disc** change side (white ones become black and vice versa). We say that those discs are **attacked** by the players move. For instance, in Figure 3, if white plays a disc on location *d3*, he attacks all black discs. As a result, in the next turn all discs on the table are white.

If a player makes at least five discs change sides, then at the next time point, it is his turn again. Otherwise, it is the opponents turn.

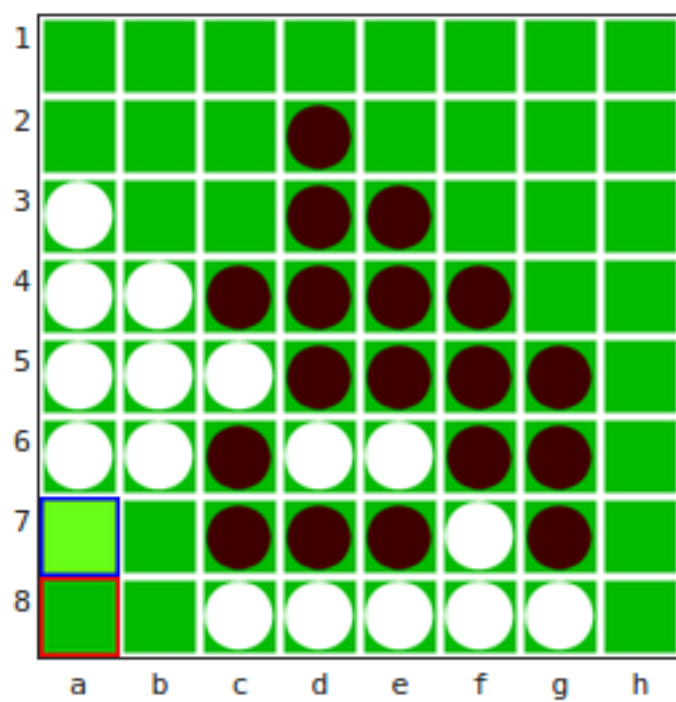


Figure 2: An example Reversi board.

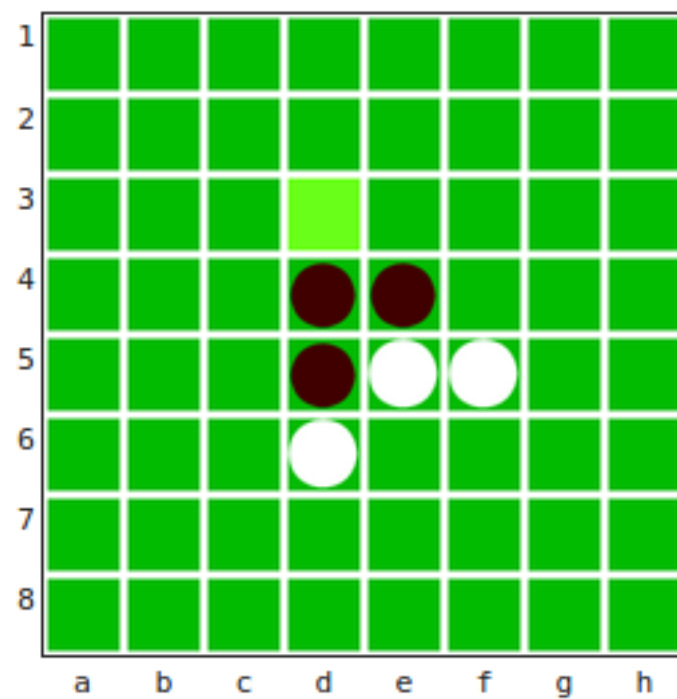


Figure 3: An example Reversi board.

2.5 End of the game

The game ends when two players subsequently *pass* (i.e., there are no more possible moves for any of the players). In this case, the player with the highest number of discs wins the game.

3 Part 1: Modelling the game in IDP

3.1 Build an LTC specification.

You are **obliged** to model this assignment in the LTC formalism! A structure of your LTC theory specifies a process of playing the game. There should be a one to one correspondence between the models of your LTC theory, and correct ways to play the game.

Do not add redundant constraints: each constraint should cut away a class of impossible worlds. For example, if f is a total function, the constraints $\forall x : \exists y : f(x) = y$ and $\forall x : \#\{y : f(x) = y\} \leq 1$ are useless. They follow from the fact that f is a total function. Adding useless constraints to your specification often makes it less readable, and less readable means less points.

3.2 Vocabulary

In order to model this, you use an extension of the following vocabulary:

```
LTCvocabulary reversiVoc {  
  type Time isa nat  
  Start : Time  
  partial Next(Time) : Time  
  
  type xco isa int  
  type yco isa int  
  type position constructed from {pos(xco,yco)}  
  
  type direction constructed from {U,UR,R,DR,D,DL,L,UL}  
  
  type player_color constructed from {black,white}  
  
  at(Time, position, player_color)  
  
  init_at(position, player_color)  
  c_at(Time, position, player_color)  
  cn_at(Time, position, player_color)
```

```

    turn(Time, player_color)

    play(Time, position, player_color)
    remove(Time, position, player_color)

    winner(Time, player_color)

    possible(Time, position)
    attacked(Time, position)
    gameOver(Time)
}

```

It is permitted, and advised, to extend this vocabulary with new predicates or functions! You **cannot** change the names, arity, types, ... of the given predicates, types and functions, since our automatic verifications are based on these properties.

The intended interpretation of the given symbols is as follows

Time represents the time in your domain;

Start is the initial time point;

Next(t) is the successor time of t ;

xco is the set of x-coordinates; these are integers; we use a standard mathematical axis system: bigger x coordinates are more to the right;

yco is the set of y-coordinates; these are integers; we use a standard mathematical axis system: bigger y coordinates are higher;

position is the set of all positions (tuples of x and y coordinates) in the game;

direction is a set of eight directions (left, right, up, down, left-down, left-up, right-down and right-up); these are probably useful to define the set of attacked discs;

player_color is a set consisting of the two colors (black and white) in the game (these colors are also used to refer to the two players);

at(t,p,c) is true if and only if there is a disc with color c on position p at time t ;

init_at, **c_at** and **cn_at** are standard LTC predicates to encode the initial state of at and the causal effects;

turn(t,c) holds if it is the player with color c 's turn at time t ;

play(t,p,c) holds if c plays a disc on position p at time t ;

remove(t,p,c) holds if c removes the disc on position p at time t ; automatically, all connected discs of the same color are removed as well (this action can hold for at most one position at each time point);

winner(t,c) holds if the player with color c won the game on time t or earlier; we want you to implement this predicate in such a way that there is a winner at time t if and only if the game is over at time t (see description of `gameOver` below);

possible(t,p) holds if it is possible to play a disc on position p at time t (i.e., playing on that location would mean attacking at least one disc); note: this is a defined fluent: it is defined in terms of the `at` predicate; you do not need to model inertia etcetera for this predicate;

attacked(t,p) holds iff the stone on position p is attacked by the move that happens on time t (if a move happens); note: this is a defined fluent: it is defined in terms of the `play` predicate; you do not need to model inertia etcetera for this predicate;

gameOver(t) holds iff the game is over at time t ; we want you to implement this **exactly** as follows: if one player has to pass at time point t and there are no possible moves for the other player at time point $t + 1$, then the game is “over” at time points $t + 1$ and later.

The IDP IDE supports autocompletion (CTRL+SPACE), use this to save yourself some time.

3.3 Verifications

Please, verify the following claims about your theory in the context of a given finite structure with finite time (use the verification procedures in the skeleton for this).

1. It is **possible** that at two consecutive points in time, the number of white discs increases.
2. At any point in time, any disc on the board is connected (horizontally, vertically or diagonally) to at least one other disc.
3. At any point in time, the set of all discs on the board is a connected set (in other words: any disc on the board is reachable, using horizontal, vertical or diagonal steps, from any other disc on the board using only positions occupied by discs).
4. It is impossible to play twice on the same position.
5. It is possible to arrive in a situation in which all the stones on the board are white.

6. Black can win the game.

If any of the claims are untrue explain why this is the case.

3.4 Visualisation

A function to visualise one run of the game (a `reversiVoc`-structure) is provided. In order to run it, you need to use the `idp-ide` (cfr exercise sessions). The visualisation can be found in the file `visualisation.idp`. In order to visualise a `reversiVoc` structure S , you use `include ‘‘visualisation.idp’’` and run `visualise(S)`.

Some explanation:

- We use grey instead of white (to make a better distinction with the background color)
- Grey/Black circles represent white/black discs
- Green squares represent empty positions where the current player can play (possible positions)
- Red squares represent empty positions where the current player cannot play (impossible positions)
- A grey or black square represents the place where the current player plays.

4 Practical

The output of this part of the project consists of

1. A *concise* report in which you discuss design decisions.
 - For each predicate symbol p and function symbol f you introduce, the report should contain a detailed description of its intended interpretation in the following format (which we also used above):
 - $p(x,y)$ is true if and only if $\langle \text{some condition on } x \text{ and } y \text{ here} \rangle$,
 - $f(x,y,z)$ is $\langle \text{some description here} \rangle$.
 - The report should also contain the time you spent on this part of the project. The expected time needed is 10 to 15 hours.
2. A file “`solution.idp`”, containing your IDP specification.
3. About the code:

- You are obliged to start from our skeletons, and you are **not allowed** to change the given symbols or their interpretation in the structures. Remember that you **are allowed** (and advised) to add new symbols to the vocabulary.
 - Use well-chosen names for introduced symbols and variables.
 - Make sure there are no warnings, except for the *Verifying and/or autocompleting structure* warnings!
 - Choose your ontology wisely. An important part of our quotation consists of checking whether your theory is a good and readable representation of the domain knowledge.
 - Write comments! Clearly specify the meaning of every line in your theories!
 - Use good indentation.
4. This assignment is graded mostly using automated tests (to see whether your specification matches the rules described in this document. Some points are given for readability of the code and automatic verifications.

You are allowed to discuss this project with other people, but are not allowed to copy any code from other students. This is not an open source project, i.e., you are also not allowed to put your code openly available on the web. If you want to use git, do not use public GitHub repositories (we will find them).

For any questions, do not hesitate to contact one of the assistants.

You submit the result on Toledo before **1/9/2017, 23.59**.

Good luck!