

A Ludo Board Game for the AGTIVE 2007 Tool Contest

Moritz Kroll Rubino Geiß

July 15, 2007

1 Introduction

The goal of this case is to construct a deterministic¹ “Mensch ärgere dich nicht” game, a variant of the Ludo game. The following is adapted from Wikipedia:

“Mensch ärgere dich nicht” is a German board game, by Joseph Friedrich Schmidt (1907/1908). It is a Cross and Circle game, similar to the Indian game Pachisi, the American game Parcheesi, and the English game Ludo, though as with Ludo the circle is collapsed onto the cross.

The board consists of a directed 40 field ring in form of a cross (see Figure 1). Every 10th field serves as a starting field for a player. Directly in front of each starting field is a junction to four consecutive goal fields of the player according to the starting field. The rules are as follows (adapted from Wikipedia):

1. There are four players in a cyclic order: red, blue, yellow and green.
2. Every player owns four figures which are not in the game initially.
3. The players throw a six-sided dice one after the other.
4. If one of the following moves is possible, the player must choose one:
 - (a) The player may move any of his figures in the game forward by the *exact* number of dots on the dice.
 - (b) When the player has a six and still has figures outside the game, he may put one figure on his start field.

¹Normally the game “Mensch ärgere dich nicht” is not deterministic. We change the game to be so to get comparable runs between the tools.

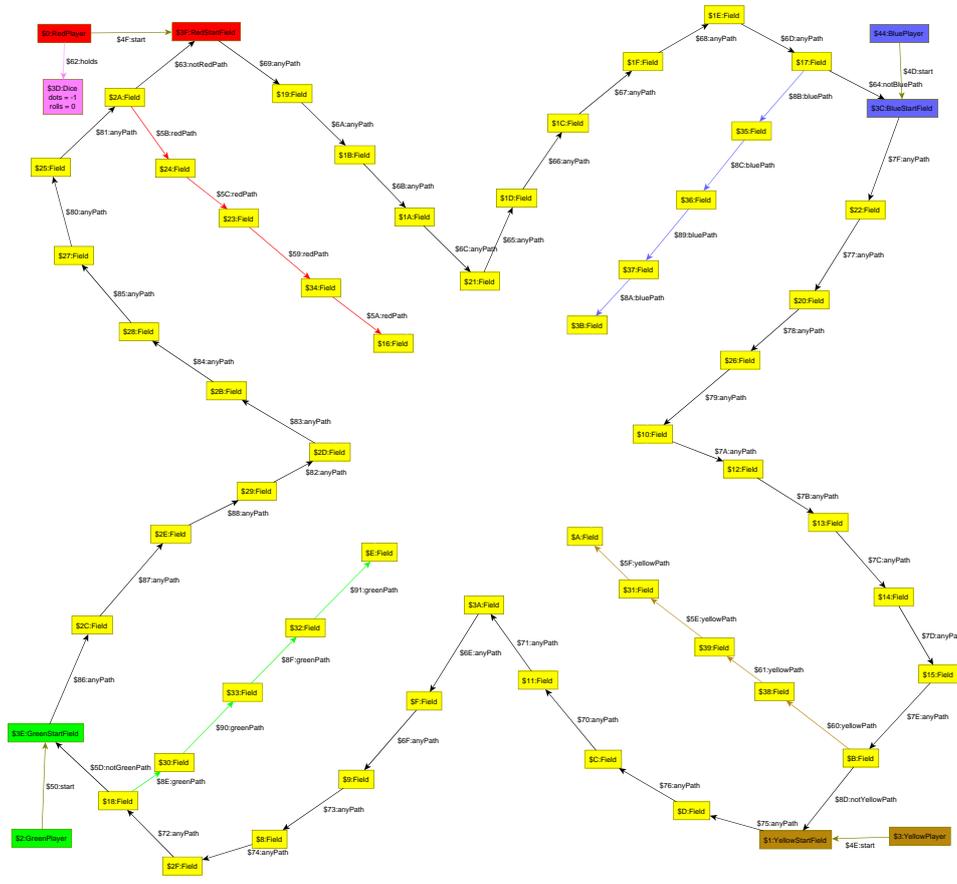


Figure 1: The initial "Mensch ärgere dich nicht" board

If the destination field of a move is occupied by a figure of its own, the move is not allowed. If it is occupied by an opponent's figure, the opponent's figure is taken off the game. The figures may neither visit nor pass the start field of the according player except from outside the game.

5. Having thrown a six, the player must play again, otherwise the next player must play.
6. After completing a round around the board a figure must enter the goal fields of the according player. The first player who fills all four goal fields wins.

2 Making the Game Deterministic

The game contains two points of indeterminism:

- The result of throwing the dice
- The choice of the move to take

2.1 The Deterministic Dice

To make the dice deterministic we use a linear congruential generator as a pseudorandom number generator. Using a given seed as the initial value for `rand` the number of dots on the dice `dots` is calculated in the following way using signed 32-bit integers:

```
do {  
    rand = 1664525 * rand + 1013904223;  
} while(rand < 0 || (rand >> 8) >= 0x7ffffe);  
dots = (rand >> 8) % 6 + 1;
```

2.2 Deterministic Choices

As for the players, we define a cyclic order on the figures of each player. When a player has thrown the dice, we check for each figure in order whether it can move, starting from the figure after the figure which moved last, and take the first move possible.

3 The Board as a Graph

As hinted at in the introduction, the fields should be modeled as nodes connected by directed edges. The figures should also be nodes being connected to the according field nodes, if they are in the game. Figure 2 shows the end of a game, where the blue player has won, because his goal fields are completely occupied by his figures. Additionally the graph contains eight invisible edges to improve the layout and nodes for the players and the dice. This is not necessary for an implementation of this case.

4 Goals of the Case

This case tests the ability of a tool to control the application of rules in a simple programmed manner. Moreover, tools allowing general formulations of rules reduce the number of rules needed to get the job done. Here, tools with advanced features such as iterated patterns or optional pattern elements profit extra, because similar rules or rules with a regular pattern can be merged to single rules, simplifying the development. The size of the host

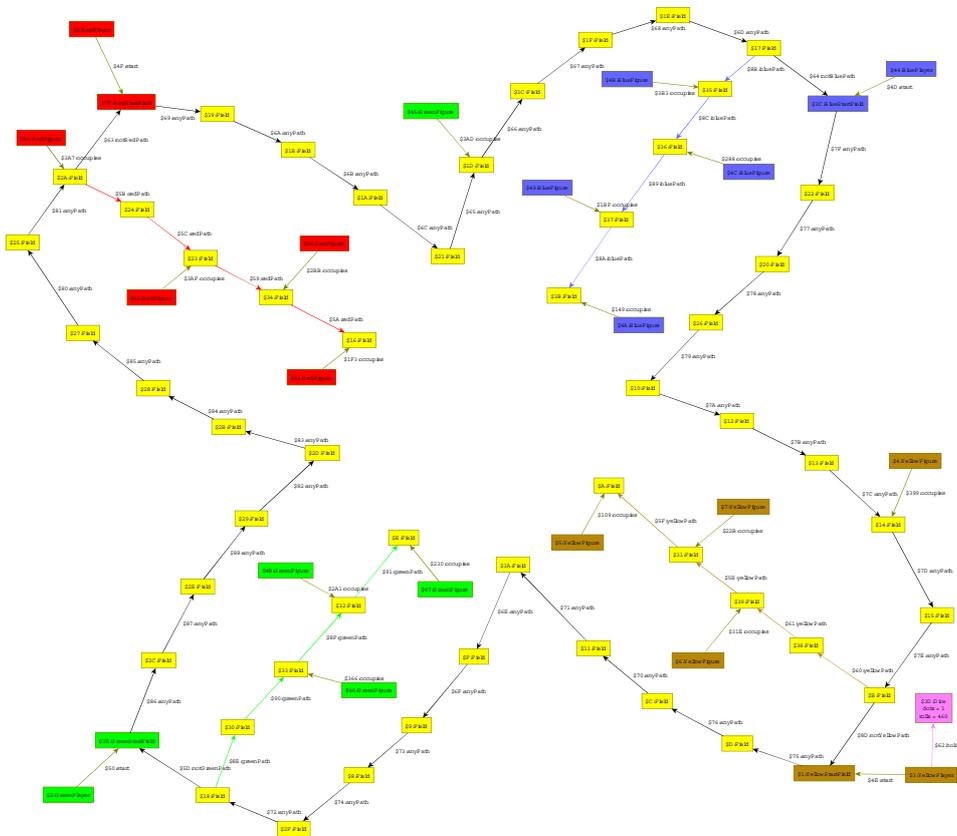


Figure 2: End placement with initial seed 98754321, where blue wins

graph is small and almost constant during all rule applications. The rules itself are of small to medium size.

To have a reference for testing your implementation, Table 1 shows the number of dice rolls (i.e. assignments to dots) until the game is won for different initial seeds of the pseudorandom number generator. Moreover, with the same initial seeds we can compare the running times of different tools.

Table 1: Initial seed and number of dice rolls until the end of the game

seed	# rolls
42	462
4937812	358
98754321	460
123456789	338
987654321	379